ConSol Software GmbH

ConSol CM Administrator Manual

Version 6.11.1.14

Contents

A - Introduction

This section provides general information about the content and structure of this manual as well as an introduction to ConSol CM.

This chapter discusses the following:

A.1 ConSol CM for Business Process Management

ConSol CM is a low code platform, especially suited for use as customer service software.

Using ConSol CM you can control and steer business processes with a strong focus on human communication and interaction as required in all fields of customer service management. Well-known examples of huge ConSol CM systems comprise customer service desks, RMA processes, after sales services, call centers and support centers as well as claim and complaint management environments. You can also set up customer portals, including FAQ areas, using ConSol CM. Basically, every business process that is in operation in a company can be modeled and brought to life with ConSol CM.

Starting with version 6.11, ConSol CM also provides the functionality to cover adaptive case management. In this way, you can decide, if you would like to design and live a strictly controlled business process or if a rather high level of flexibility is required. You might also combine both concepts, depending on the team or department who work with the process.

Figure 1: *Overview of potential fields of use of the low code platform ConSol CM*

Using ConSol CM, you can handle all components which are relevant in business processes to represent and control your company's processes in an optimal way. ConSol CM is used in various different industries and branches ranging from insurances and banks over fashion designing companies

to producers of ticket vending machines or car washes. The flexible process designing mechanism and workflow engine provide a perfect basis for the modeling and controlling of business processes, especially customer service processes, of different kinds.

A.2 List of Manuals

ConSol CM provides documentation for several groups of users. The following documents are available:

- **Administrator Manual**
  A detailed manual for CM administrators about the ConSol CM configuration using the Admin Tool.
- **DWH Manual**
  A detailed explanation of the ConSol CM data warehouse (DWH) concept, the database schema and a list of all table structures.
- **Operations Manual**
  A description of the ConSol CM infrastructure, the server integration into IT environments and the operation of the CM system, for IT administrators and operators.
- **Process Designer Manual**
  A guideline for workflow developers about the graphical user interface of the Process Designer and how to program workflow scripts.
- **Setup Manual**
  A technical description for ConSol CM setup in different IT environments. For expert CM administrators.
- **System Requirements**
  List of all requirements that have to be met to install ConSol CM, for IT administrators and CM administrators. Published for each ConSol CM version.
- **Technical Release Notes**
  Technical information about the new ConSol CM features. For CM administrators and key users. Published for each ConSol CM version.
- **User Manual**
  An introduction to the ConSol CM Web Client for end users.

A.3 TecDoc Server

For detailed information about all aspects of ConSol CM, please see also our **tecdoc server**, available at https://tecdoc.consol.de. You find there:

- All **ConSol CM manuals**
  The ConSol CM manuals for several target groups in English and in German
- The **release notes** for each ConSol CM version
  Detailed Release Notes documents which explain every new feature of every ConSol CM version
- The **system requirements**
  An overview of the required hardware, middleware, and software prerequisites which are required to install ConSol CM
- The **"New Features for Customers" presentations**
  Not too technical presentations which provide an overview of the new features of new ConSol CM versions

A.4 This Book's Structure

First, some basic principles of ConSol CM are explained to provide the theoretical background you need to become a CM administrator.

- In the **Introduction**, you will learn how ConSol CM is used in Business Process Management and you will get to know some basic principles of the application. Furthermore, you will learn how to start the main administration application for CM, the Admin Tool.

The following sections explain the features and functionality of the Admin Tool.

- **Access and Roles**
  In this section the basic principle and configuration concerning access permissions are explained. For example, you will learn how to define roles, assign roles to engineers (the users of the CM Web Client), and configure views (the to do lists in the system).
- **Ticket Data Model and GUI Design**
  Here, the set-up of the ticket data model and the placement of the data fields in the Web Client (GUI) are covered. For example, you will learn how to define the data fields which are required by a certain process and how to build different types of lists.
- **Customer Data Model**
  This section describes the setup of the data models for different customer groups and the respective designer GUIs. The representation of customer groups in CM is based on *FlexCDM*, the Flexible Customer Data Model. For example, you will learn how to define one set of data fields for the customer group *Reseller* and another data model for the customer group *Direct Customers*. Furthermore, customer relations and customer actions are explained, two components which help you use CM as a CRM system.
- **Global Configuration**
  In this section some general configurations are explained. For example, you will learn about Queue Management, a queue being one of the core components of ConSol CM. Furthermore, working with business calendars and projects is explained.
- **Expert**
  This is the part of the book which is targeted at CM administrators who are responsible for advanced CM system configuration. You might want to work together with your CM consultant to change system settings. This section covers topics like
  - preparing the CM system for reporting
  - ConSol CM services
  - configuration of the search module
  - the Task Execution Framework (TEF)
  - email configuration
  - import and export of configurations
  - template management
  - working with system properties

  - working with Admin Tool scripts
  - license management
  - deployment of scenarios
- **Add-Ons**
  In this section, the three CM add-ons are explained. A CM add-on is a CM module for which a separate license has to be purchased. The following add-ons are available:
  - **CM/Resource Pool**
    In this section the set-up of the data model for the Resource Pool is explained. CM/Resource Pool is a distinct CM module which has to be licensed separately. If you have purchased this module, you can learn in this section how to represent different objects like IT assets, products, SLAs or other objects as CM resources. Besides the set-up of the data model, resource relations and resource actions are described.
  - **CM/Track**
    In this section, the customer portal, CM/Track, is explained. The versions V1 and V2 are available for CM/Track, they are treated in separate chapters.
  - **CM/Phone**
    In this section, the ConSol CM CTI solution is explained.
- **Appendix**
  Here, you find lists of all important terms that are used in the book (glossary), of all annotations (important for the GUI design), and system properties (important for the CM system management). Please see also the trademarks page.

A.5 Layout Explanations

The following icons and colors are used to emphasize and highlight information:

This is an additional information.

This is an important note. Be careful here!

This is a warning!

This is a recommendation from our in-the-field consultants.

A.6 Legal Notice

Since we would like to provide a manual for you which helps you manage your CM system, but which also provides additional information about connected topics, we have inserted external links into the manual. In this way, you can get some background information about a topic if you like. This can help you better understand the required CM configuration. Despite careful review, we assume no liability for the content of those external links. The operators of sites linked to are exclusively responsible for their content.

A.7 Gender Disclaimer

As far as possible, ConSol CM manuals are written gender-neutral and often address the user with "you". When the phrasing "The user .... he ..." is used, this is always to be considered to refer to both, the feminine as well as the masculine form.

A.8 Copyright

A.9 Basic Principles of ConSol CM

A.9.1 System Components from the Users', Admins' and Customers' Points of View

ConSol CM comprises different client applications. Depending on your roles and tasks in your company you will use one or more of those applications.

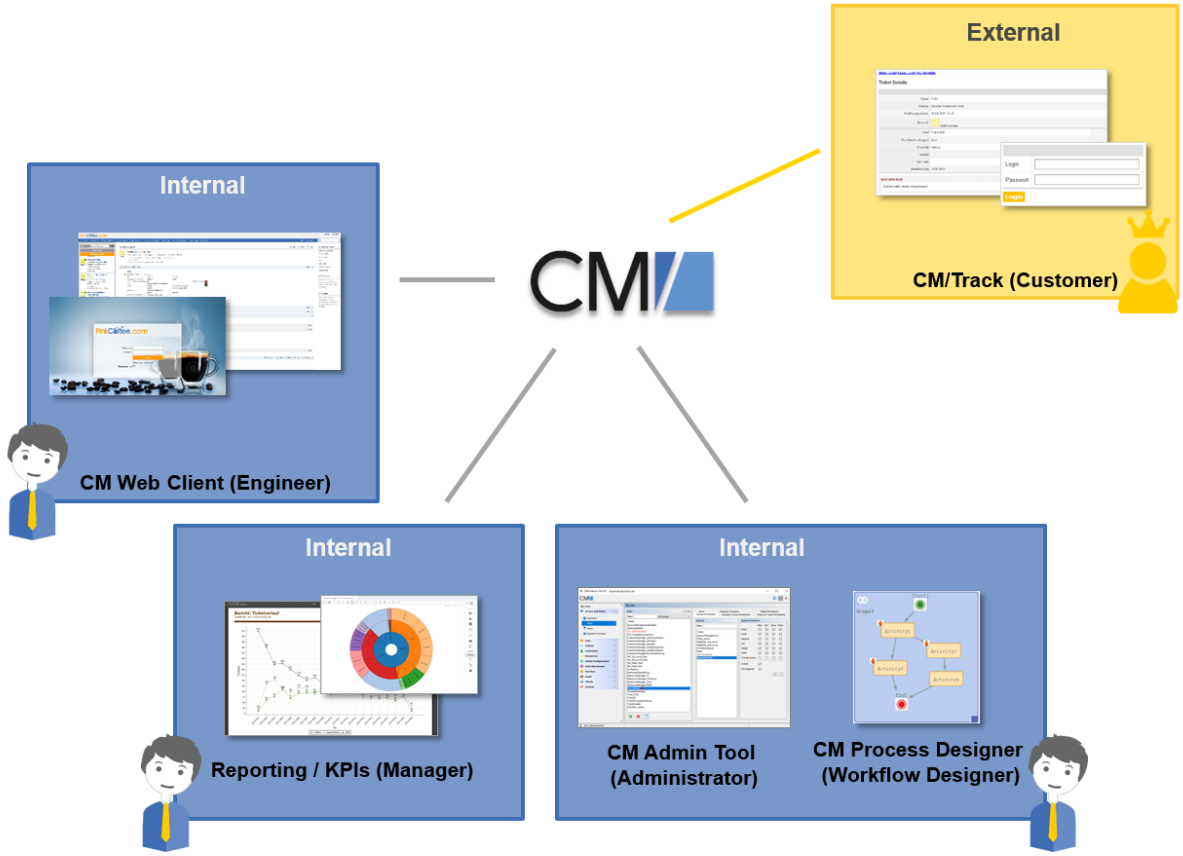


Figure 2: *ConSol CM system components*

- **Web Client**
  The primary access point to the system for engineers, an engineer being the standard user of the system. Engineers work with tickets, customer data and resources.
- **Portal**
  CM/Track, the primary access to the system for (internal or external) customers. CM/Track is a distinct CM module which requires a separate license. With this module, you can offer portal access to the tickets for your customers. Moreover, your FAQs can be made available via the web.

- **Admin Tool**
  For all system configuration tasks. As an administrator, you will primarily work with this tool. This tool is used to define the system setup. All settings (apart from workflows) are configured using the Admin Tool, and access to it is restricted to admin users.
- **Process Designer**
  For the workflow design and implementation. As a workflow developer you will primarily work with the Process Designer. In this tool, all workflows are designed graphically as well as in Groovy code.

The default scope of delivery also includes a data warehouse (DWH) that allows reporting about the data of your tickets.

Furthermore, ConSol CM is not an isolated application but can be easily integrated into your company's IT infrastructure, e.g. using Web Services and/or an Enterprise Service Bus (ESB) or the ConSol CM Webhook interface.

For a detailed explanation of the system components, described from a more technical point of view, please refer to the system administrator's section Architecture of a CM System.

A.10 Basic Technical Principles and Objects of ConSol CM

This chapter discusses the following:

A.10.1 Introduction

In order to work efficiently and correctly with ConSol CM, you have to have a profound knowledge of all components which make up a CM system. The following section will give you a first introduction into the basic CM components.

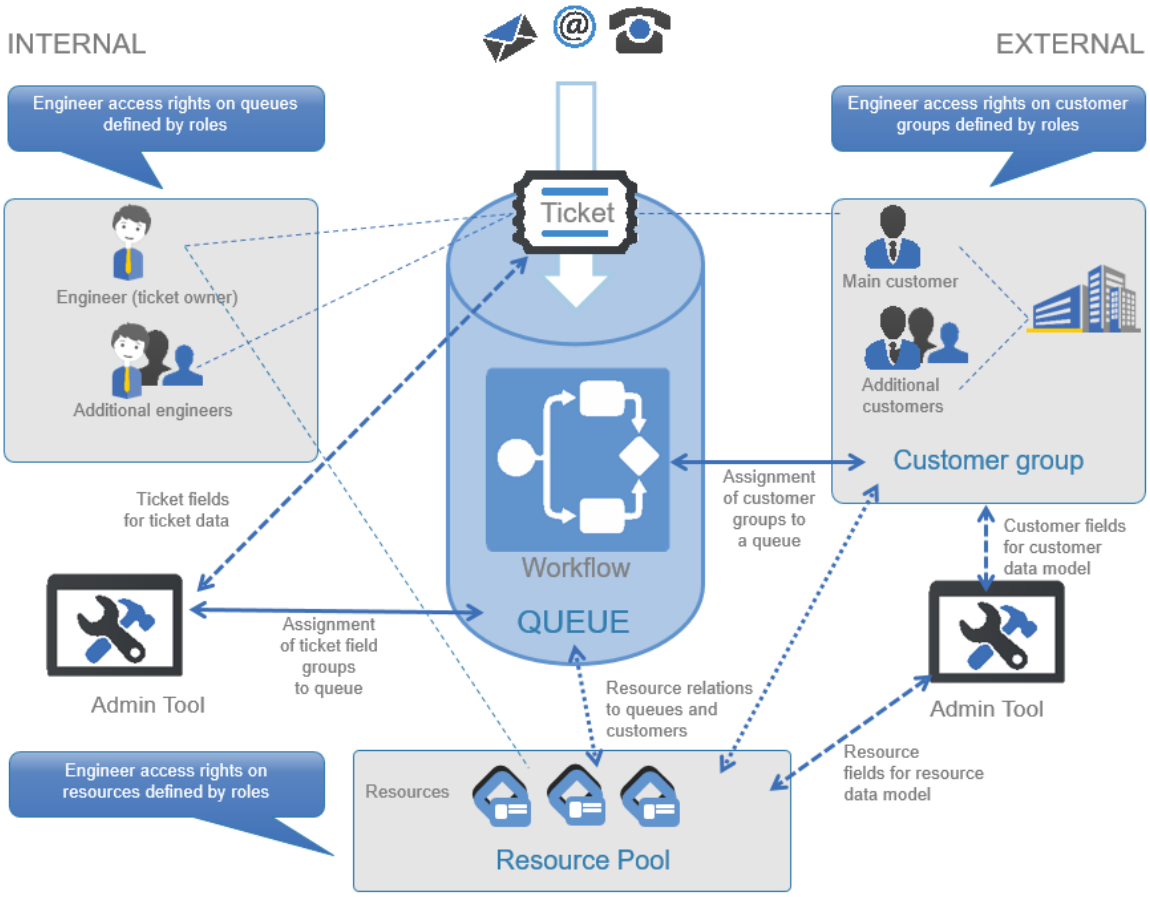


Figure 3: *ConSol CM - Basic principles (with CM/Resource Pool)*

A.10.2 ConSol CM Dogma

In ConSol CM, there is a main dogma:

ConSol CM DOGMA

External side: A ticket always has a **main customer**. This can be a contact or a company.

Internal side: A ticket can have no or one main **engineer** who has to work on the ticket.

A.10.3 Engineers

The engineers represent the **internal** side of the CM system. All users of the Web Client are called engineers, regardless of their function within the company. The engineers work on the tickets to carry out the tasks defined in the business process. Every ticket can only be assigned to **one engineer**, who is currently responsible for the ticket. But a ticket may have any number of **additional engineers,** who all have an **engineer function** representing a specific task within the process.

All engineers have a user account consisting of a user name and password, which they use to log in to the Web Client. The engineers' access permissions are managed using roles. The roles, which contain access permissions for queues, customers, and resources, are defined in the Admin Tool and assigned to the engineers.

Please see the section about *Engineer Administration* for details.

It depends on the configuration of your ConSol CM system if an engineer is called *engineer* within your system. Engineers might be called *agents*, *employees*, or similar in your Web Client. In this manual, engineers will always be called engineers for your convenience.

A.10.4 Customers

With FlexCDM, the Flexible Customer Data Model, ConSol CM provides a data model which can define contact and company data in various constellations. In this way, you can define very simple, one-level data models which only contain contact data (e.g., name, phone number, email address, address) and complex, two-level models which contain contact data (e.g., name, phone number, email address) and company data (e.g., address, zip code, company size). You can define different models within one system, you can configure relations between customers, and add activities to contacts and companies. Please refer to the *Customer Data Model* section, starting with chapter *The CM Customer Data Model - FlexCDM*, for details about all components of FlexCDM.

Each CM system uses customized customer groups and data models. Therefore, the available customer groups, hierarchical levels for customer objects, data fields, relations, and activities depend on the individual configuration of your CM system.

A.10.5 Tickets

The ticket is the request of the customer which the engineers work on. This can be an incident, a service case, or any other request. For each request, a ticket is created. The engineers work on the ticket, which means that they carry out the necessary steps as defined in the business process. The progress, including internal and external communication, is documented in the ticket. The business process can involve several engineers and different teams. When the request is solved, the ticket is closed. Closed tickets are not lost, but they represent a powerful archive and knowledge base.

In ConSol CM there are the following rules for tickets:

- A ticket must have one main customer. Every ticket can have only one main customer. It does not need to have additional customers, but it can have any number of additional customers.

The customer represents the **external** side of a ticket.

- A ticket does not have to be assigned to an engineer, but if it is, it can only be assigned to one engineer at a time. A ticket does not have to have additional engineers, but it can have any number of additional engineers. Assigning a ticket to an engineer can be done manually or automatically. The engineers represent the **internal** side of a ticket.
- A ticket always has a name, often called *ticket number*, a subject, and a ticket icon. The ticket icon shows the current scope of the ticket and can have a color indicating the value of a given data field. Every ticket also has an ID that is used internally and cannot be seen by the user.
- The ticket header always shows the current queue and scope, assigned engineer, and creation date of a ticket.
- The ticket icon in the Web Client can have (and in most cases does have) a color that represents a certain value of a list. Often the priority is used, e.g., high priority tickets are displayed in red, medium tickets in orange, and low priority tickets in yellow.

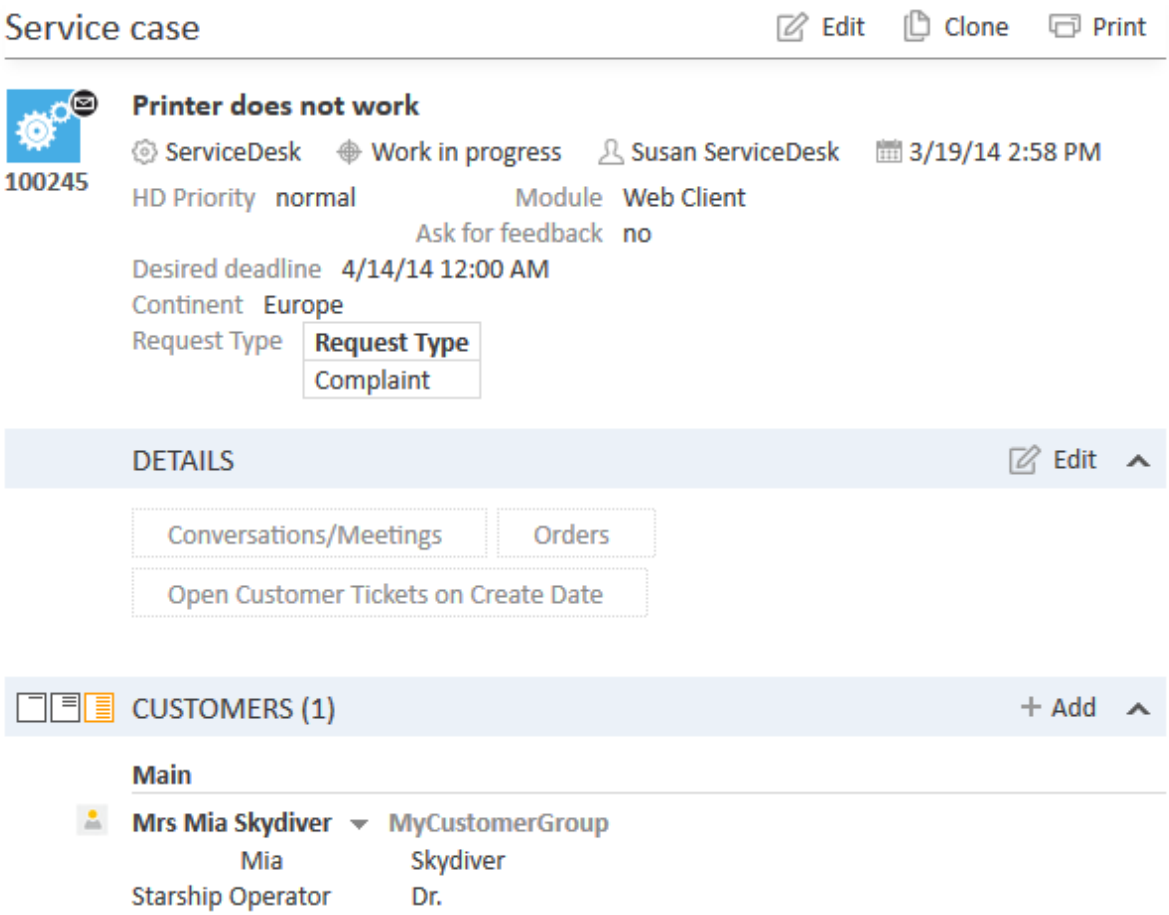


Figure 4: *Ticket in the ConSol CM Web Client*

It depends on the configuration of your ConSol CM system if a ticket is called *ticket* within your system. Tickets might be called *ticket*, *case*, *call*, *task*, or similar in your Web Client. Each queue can have its own term to refer to tickets. In this manual, tickets will always be called tickets for your convenience.

Each CM system uses customized ticket data. Therefore, the available fields, relations, and activities depend on the individual configuration of your CM system.

A.10.6 Resources

Resources can be used to manage objects which are related to the business process. Possible use cases are IT assets, SLAs, products or newsletters. All resources are saved in **CM/Resource Pool**, a separate CM module. The administrator defines the resource model, i.e., the resource types, resource data fields, the hierarchy of the resources, and the possible relations to tickets, customers, and other resources

An engineer with the required permissions can create resources and link them to existing tickets, customers, and other resources. For example, you can link a computer to an incident ticket concerning this computer or to the customer who uses it.

Please see the section about CM/Resource Pool in the Add-on section for details about this topic.

A.10.7 Queues

The queue is the **core component** of ConSol CM administration. It contains thematically related tickets which should be handled in the same way and follow the same business process. Every queue has exactly one **workflow**, which implements the desired process. The ticket data fields needed for the process are assigned to the queue. In addition, the access permissions, which are granted to the engineers via roles, are based on queues.

For example, there is one queue for the user help desk with the *User Help Desk* workflow and data fields like *Customer Service Level*, *Device that does not work*, or *Priority*. Every incident ticket passes through this *User Help Desk* process. Another queue is the *Marketing and Sales* queue where fields like *probability of contract conclusion*, *next appointment* , or *budget [$]* are defined.

Therefore, the queue determines:

- which data fields (ticket data fields) are available, e.g., *order volume* in a sales queue or *cause of the incident* in a support queue
- which customers can have tickets in the queue (customer groups)
- how its tickets are processed (workflow)
- who can work on the tickets (permissions)

Queues often reflect the organizational structure of the company. For example, there can be one queue for each department, as each department has its own processes. A ticket can be passed from one queue to another. In this case it adapts to the new queue, i.e., it receives the data fields of the new queue and only engineers with permissions for the new queue can work on it.

> It depends on the configuration of your ConSol CM system if a queue is called *queue* within your system. Queues might be called *processes*, *teams*, or similar in your Web Client. In this manual, queues will always be called *queue* for your convenience.

A.10.8 Workflows

A workflow is designed and created by a CM workflow developer using the ConSol CM Process Designer. It implements the business process which is executed in the Web Client. The workflow consists of several steps, which are called *activities*. There are manual activities, which are performed by the engineers, and automatic activities, which are performed by the system. The activities are arranged in **scopes** to illustrate the status of a ticket. The intelligence of the process, like conditions, decisions, escalations, reminders, automatically sent emails, or other actions, is also defined in the workflow. You can implement process chains or a hierarchical process structure by linking several workflows.

As an engineer, you will not work with the workflow itself, but you will see the current queue (1) and scope (2) of the ticket. The scope is indicated by the symbol of the ticket icon and scope name in the basic ticket data. Furthermore, you see the workflow activities which are available for the ticket at its current position (3). In this way, you always have a good overview of the current status of the ticket.

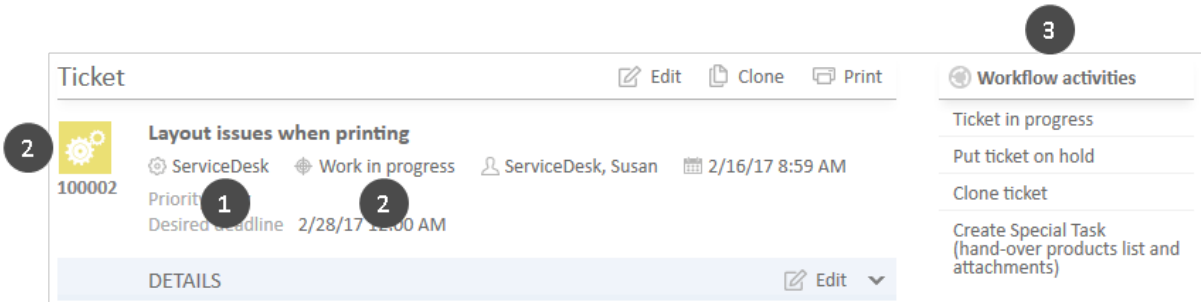


Figure 5: *Queue, scope, and workflow in a ticket*

As an administrator, you might work with the ConSol CM Process Designer to model the business processes of your company. A process can be represented by one or more workflows.

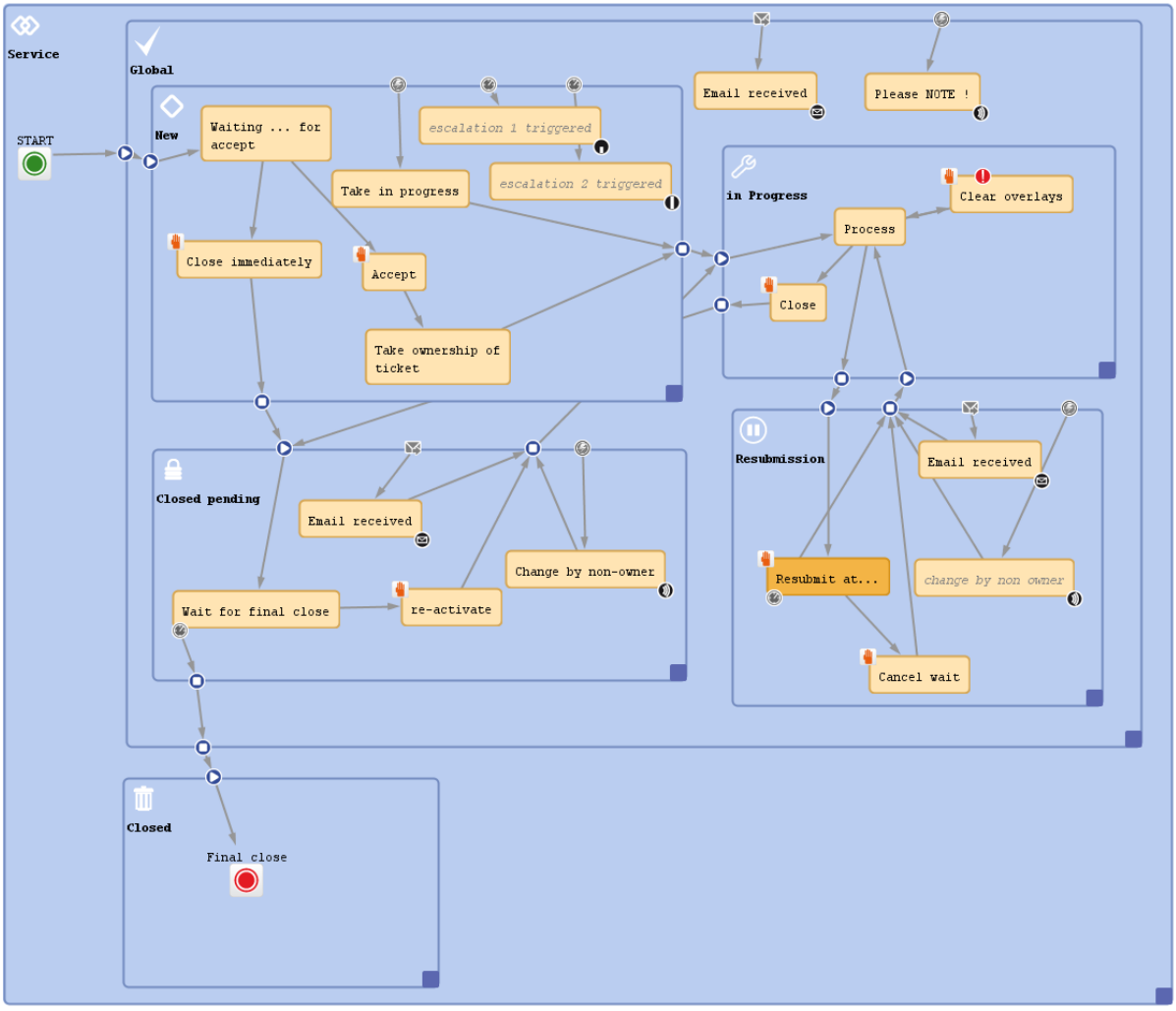


Figure 6: *Process Designer: Example of a service workflow*

Since we often deal with process chains rather than with single processes, in ConSol CM such process chains can be designed by defining a certain order for the processes. You can work with simple process chains or with a hierarchical structure.

For example, a ticket starts in an entry pool, is directed to the 1st level team, who pass it on to the 2nd level network team. Or a sales ticket starts as a customer request, becomes a lead, which eventually gets more serious and becomes an opportunity. Once the customer has signed the contract, an order ticket is created which generates so-called child tickets for the internal tasks up to billing. When all child tickets are closed, the parent ticket can be closed as well.

The intelligence of the process, like escalations, reminders, automatic generation of emails, or other actions during the process, is also defined in the workflow using Groovy scripts.

Please refer to the ConSol CM Process Designer Manual for a detailed introduction to process design and modeling using the ConSol CM Process Designer.

A.10.9 The CM Action Framework

In addition to workflow activities, which are activities executed during a certain step of a business process, activities, here called actions, can also be triggered from other objects:

Customer actions (Unit Actions) are actions which are executed based on the customer object, i.e., for a contact or for a company. In this way, you can, for example, implement a company action which updates your company-specific sales figures every night.

Resource actions are actions which are executed based on a resource, i.e., on an object in the CM/Resource Pool. In this way, you could offer an action in the Web Client where the engineer can have a list with all customers for a certain newsletter.

Search actions are actions which can be executed based on a result of a Detailed Search. In this way you can, for example, trigger an email to all customers of a list of tickets which you have retrieved using the search interface.

All actions can be either executed manually or automatically. Manual actions are offered in the Web Client like workflow activities. Automatic actions run in the background without any engineer activity being involved.

A.10.10 The Task Execution Framework

The Task Execution Framework (TEF) stores and executes long-running tasks which should not be linked to any specific activity. For example, TEF tasks can be very helpful for import scripts.

A.10.11 Accessing Objects in ConSol CM

In ConSol CM, the different objects (tickets, customers and resources) form a network. The objects are connected, e.g. a ticket is always linked to one or more customers. The connections which exist between the current object and other objects in ConSol CM are displayed on the object's page, where you can directly access the linked objects.

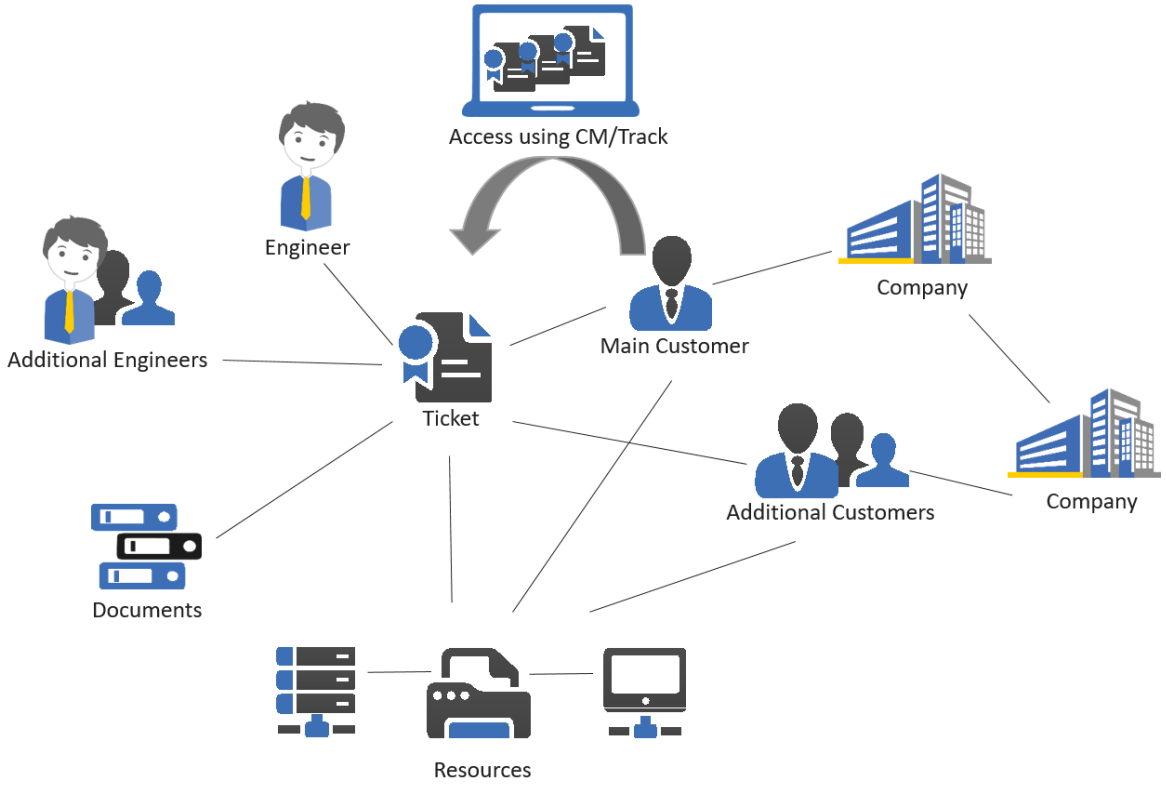


Figure 7: *Network of ConSol CM objects*

Since the ConSol CM Web Client always provides menus or links to open the objects which are related to another object, you, as a CM engineer, can move within the network easily, thus working efficiently with customer, ticket, and resource data. Once you have opened an object (e.g. a ticket), you can move from one related object (e.g. the main customer of the ticket) to other related objects (e.g. the company of the main customer or a resource which is related to the ticket or the customer). No further search is required.

An example use case could be:

1. A customer calls you and asks for a certain case, but he cannot remember the ticket number. He wants to know which SLA is used for the printer which caused the problems treated in the ticket. Is it possible to change the contract?
2. You start the quick search and look for the customer's name and the term 'printer', you find ticket #0815 and open it.
3. You check the related resources at the ticket and find printer #4711.
4. You check the SLA, a resource, related to the printer and open it. The comment in the SLA says "can be changed within two weeks". Now you have to know all other possible SLAs for this company.
5. You check the company relation of the resource (the printer #4711) and open the company page.

6. The company has three more SLA resources (i.e., resource relations on the company page) for printers. You discuss with the customer which one would be the best for this case. The customer wants to have a new SLA for the printer.
7. Since this change has to be approved by both sides, an SLA-change-ticket is required: you create a new ticket directly from the company page ...

This short example shows how easy an engineer can access all components for a certain case or service request. If customer-customer relations are managed using CM, the CRM (customer relationship management) component is even stronger.

A.10.12 ConSol CM from a System Administrator's Point of View

ConSol CM is a Java EE application which runs in a standard application server. The data is stored in a relational database. ConSol CM connects to an email server to retrieve incoming emails and sends emails using an SMTP server. Please refer to the *ConSol CM Operations Manual* for a detailed explanation of all aspects concerning running ConSol CM in an IT environment. A first introduction is provided in section *System Overview* in this manual.

A detailed list of supported operation systems, application servers, database systems, and other systems, as well as storage and CPU requirements is given in the current *System Requirements*.

A.11 Starting the Admin Tool

This chapter discusses the following:

A.11.1 Login

Most of the ConSol CM system is administrated using a Java Web Start application, called ***Admin Tool***, which is provided on the main web page of the CM application server system. To start the Admin Tool you can either use the link on the page or you can store the `jnlp` file locally and start it there. Java Web Start is part of each standard JRE.

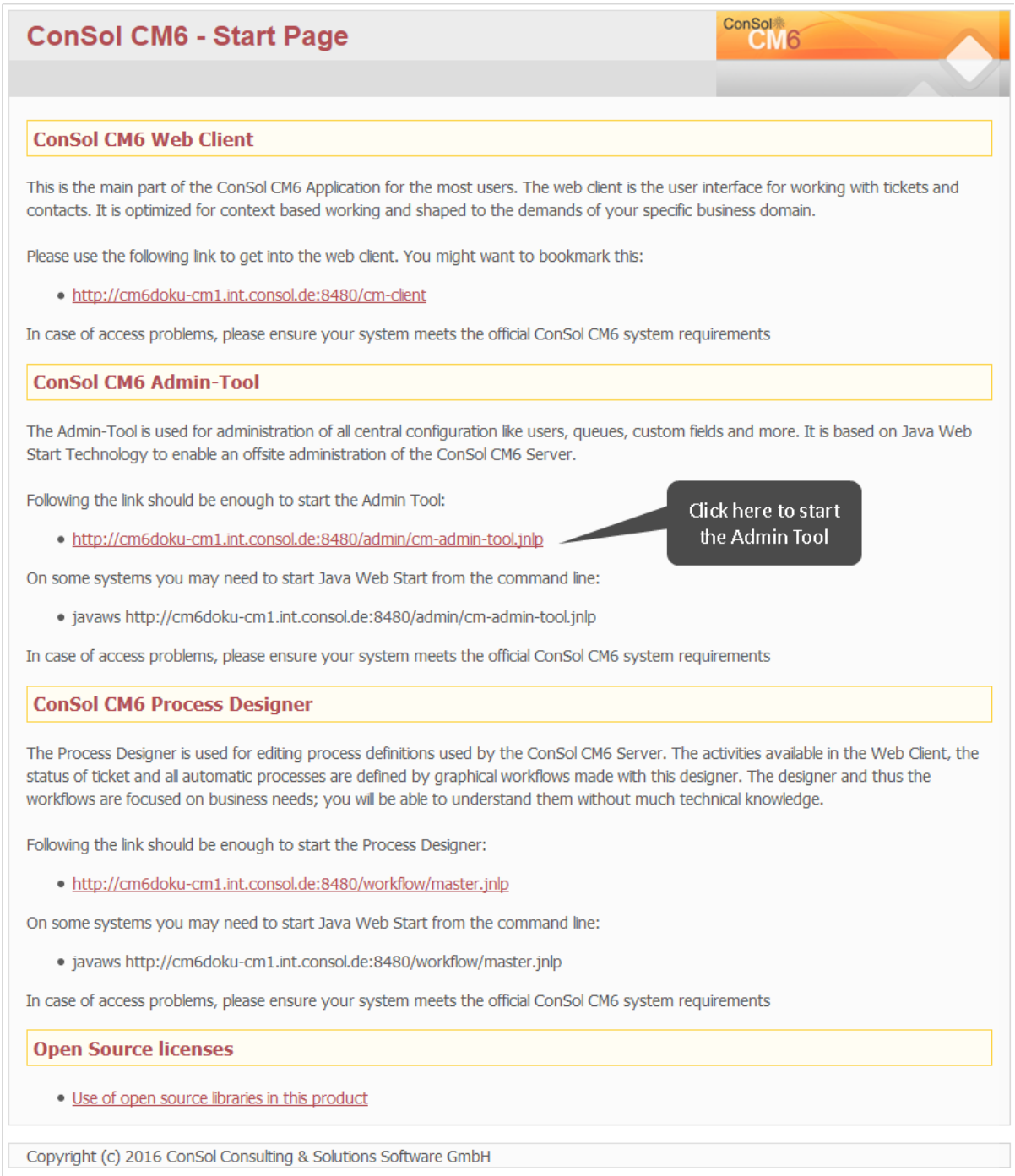


Figure 8: *ConSol CM start page*

After clicking on *Admin Tool.jnlp*, the `jnlp` file is downloaded, the Admin Tool is started, and the login window is displayed (for details see the Troubleshooting: When the Admin Tool Does Not Start section):

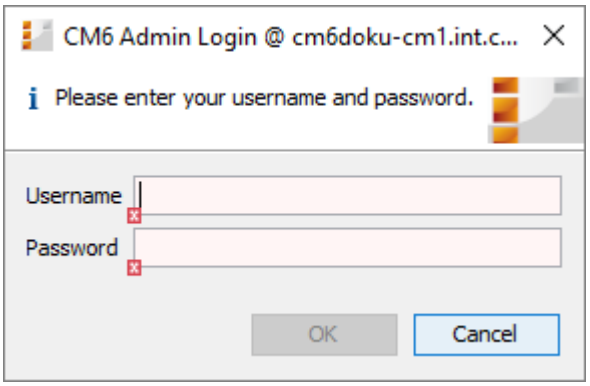


Figure 9: *ConSol CM Admin Tool - Login window*

Enter your login data to get access to the Admin Tool functions. An initial user name and password are assigned during system set-up. Further admin users can be configured later on in the Admin Tool.

After you have logged in successfully, the start page of the Admin Tool appears:

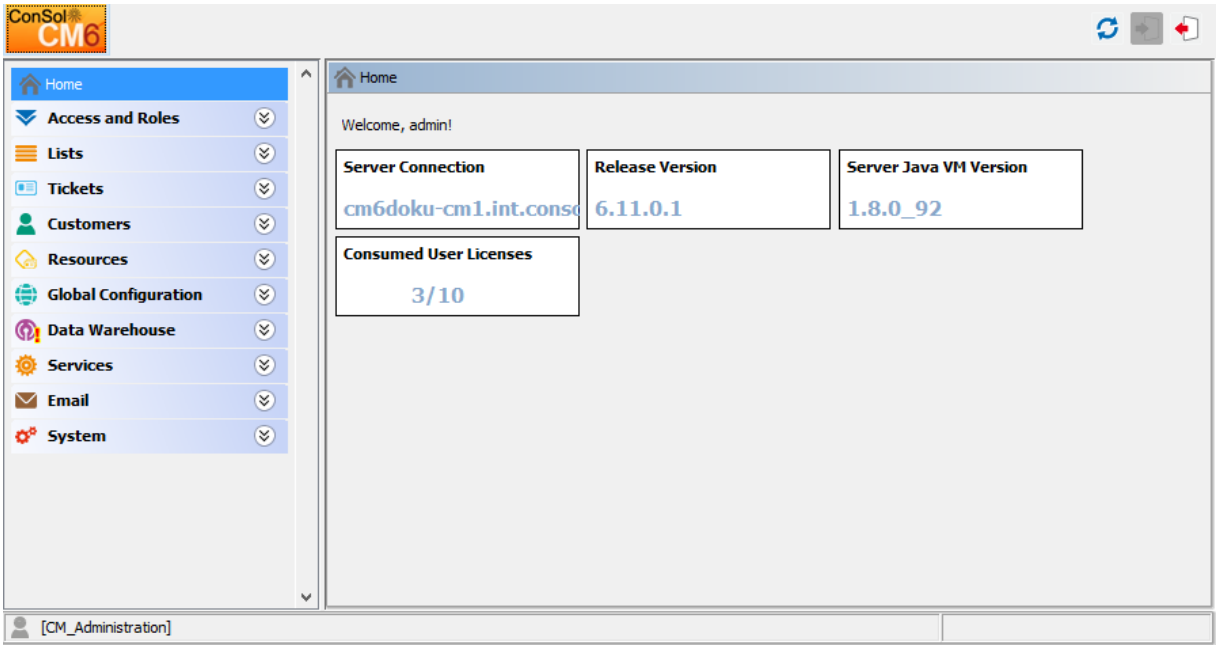


Figure 10: *ConSol CM Admin Tool - Start page*

A.11.2 Troubleshooting: When the Admin Tool Does Not Start

A.11.2.1 Correct Process

If everything is set up properly, clicking on the Admin Tool hyperlink leads to the following:

1. In a pop-up window, you are prompted as to whether you would like to open the `jnlp` file - *Java (TM) Web Start Launcher* should be offered as default application for that - or to download the `jnlp` file to your system.
   Confirm with *Open with Java (TM) Web Start Launcher*.
2. The download of the Admin Tool `jnlp` file is started. During this process, the ConSol CM logo is displayed.
3. Java Web Start starts the Admin Tool. In a pop-up window the *Verifying application* message is displayed.
4. If the Java Web Console is activated, the console is opened and you can follow the download's progress.
5. The Admin Tool GUI is displayed with the login window in the foreground.

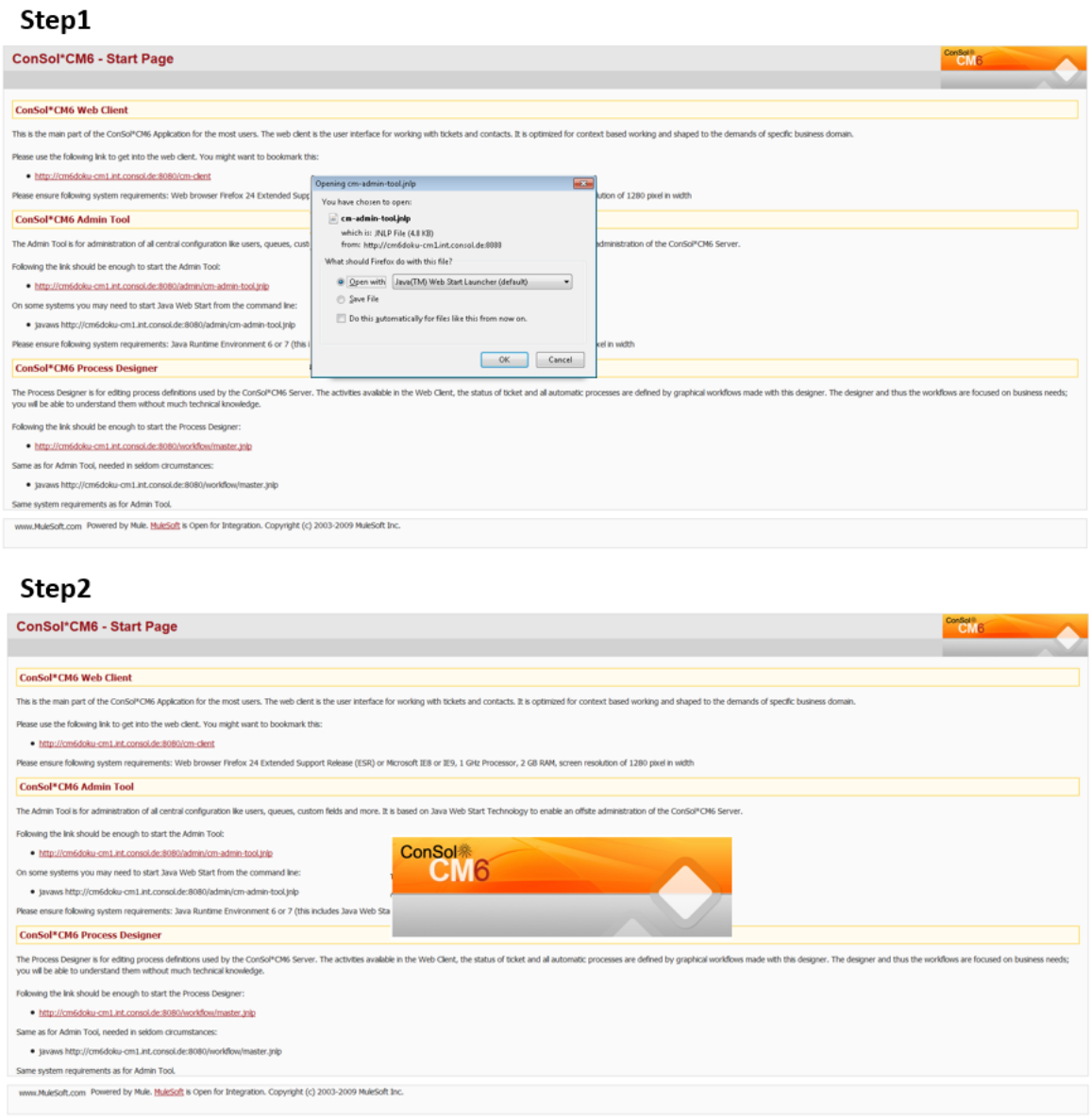


Figure 11: *ConSol CM Admin Tool - Start: Steps 1 and 2*

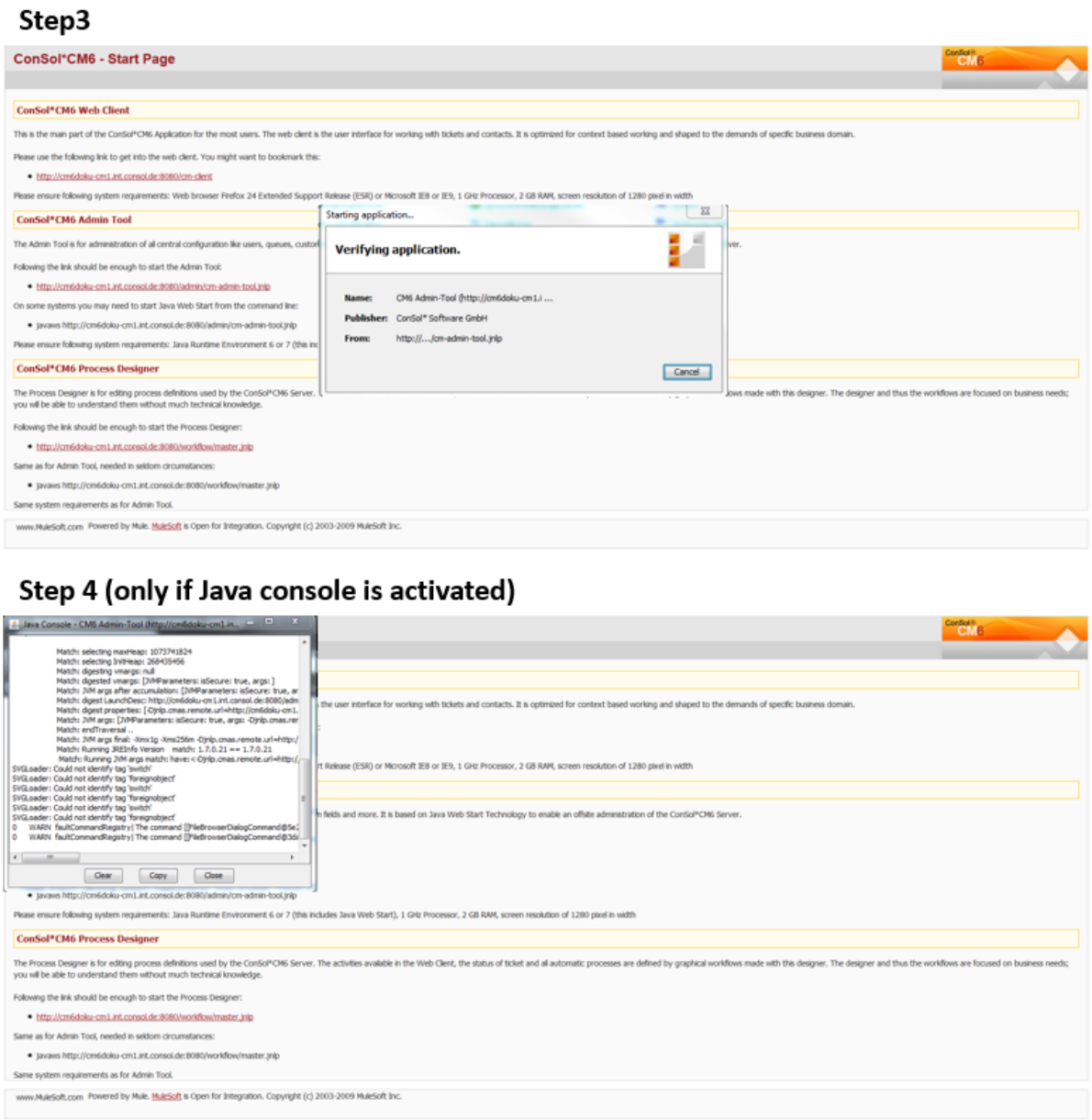


Figure 12: *ConSol CM Admin Tool - Start: Steps 3 and 4*

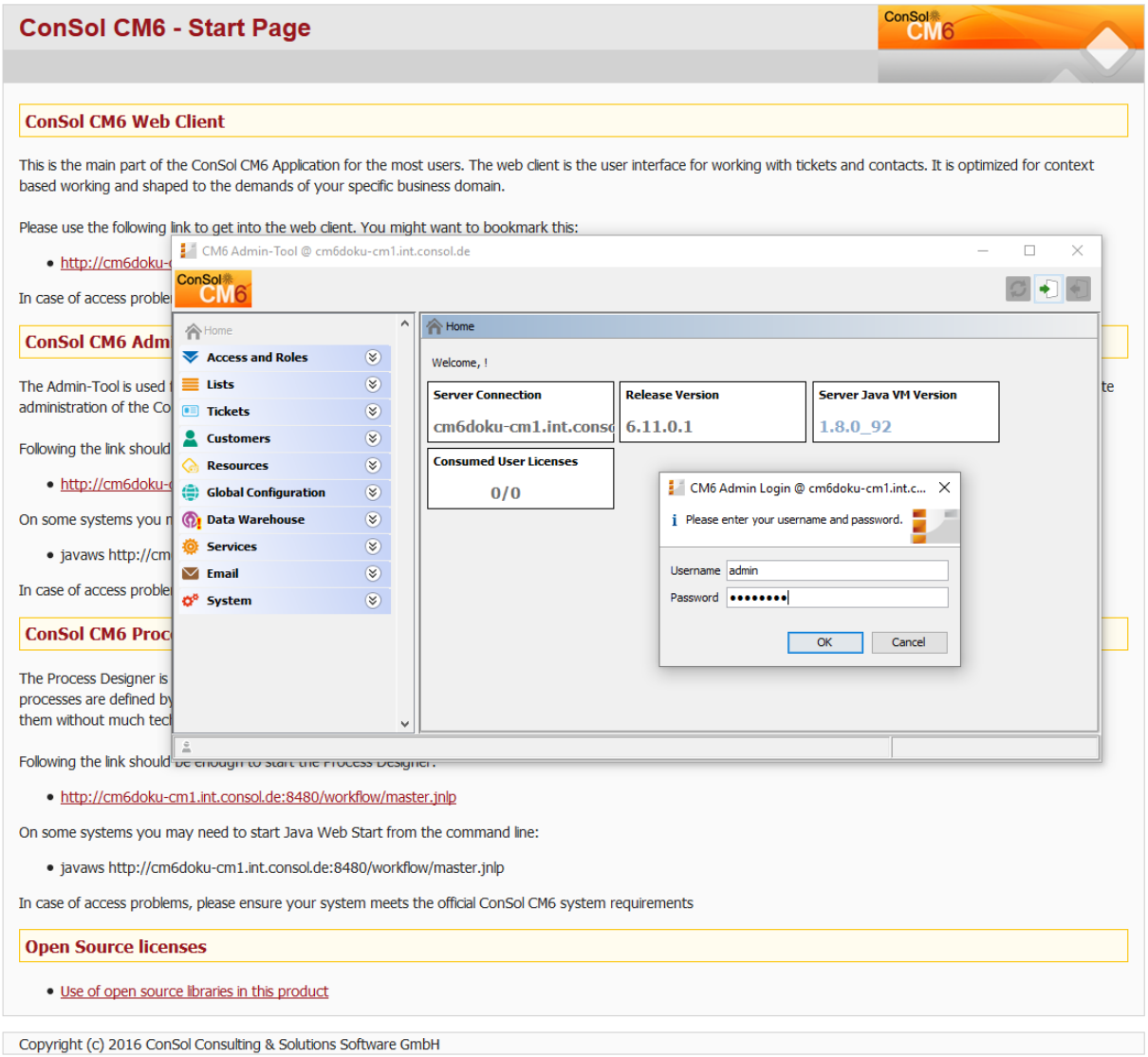


Figure 13: *ConSol CM Admin Tool - Start: Step 5*

A.11.2.2 Process with Errors

If the Admin Tool cannot be started, check the following settings:

1. **Problems with step 1:**
    a. Is a supported version of Java installed on your machine?
    b. Is the correct Java version activated?
    Under Microsoft Windows use *System settings -> Java -> Java -> Display ...*

2. **Problems with step 2:**
    a. Can your client machine connect to the ConSol CM server over the network? Can the `jnlp` file be downloaded by the web browser?
    b. Check the Java Network connection settings.
    Under Microsoft Windows use *System settings -> Java -> General -> Network settings.*
3. **Problems with step 3:**
    a. Does Java Web Start load and verify all Admin Tool application files? If not, check the network connection.
    b. For all other errors, a pop-up window with a detailed error message will be displayed.
4. **Notes concerning step 4:**
    a. To find the cause of a problem, activate the Java Console.
    Under Microsoft Windows use *System settings -> Java -> Extended -> Display console, Debugging: Tracing enabled, Debugging enabled*.
5. **Problems with step 5:**
    a. When the login window is displayed, enter your login data. If a connection error occurs then, check the proxy settings.

Please read the section The Admin Tool Graphical User Interface (GUI) to learn how to work with the Admin Tool using its GUI elements.

A.12 The Admin Tool Graphical User Interface (GUI)

This chapter discusses the following:

A.12.1 Introduction

The following section provides an overview of the Admin Tool graphical user interface (GUI). The basic principle and all icons are explained.

In the run of this manual, we refer to the names of the GUI elements as they are explained here. In order to avoid redundancies, the icons are not always shown in each section.

A.12.2 Basic Principle

On the left-hand side, you see the navigation tree (1) with the navigation groups (2: extended navigation group, 4: closed navigation groups). Each navigation group contains several navigation items (3). Click the name of a group to expand the group in the tree. Click a navigation item to open the respective tab in the working area on the right-hand side.

In the bottom left corner (5), the name of the admin user who is currently logged in is displayed with all roles which are assigned to this user.

In the top right corner (6), you find the buttons to reload the Admin Tool data, to log in and to log out. The main working area (7) contains the data of the active/opened navigation item.

In the following figure, the *Home* tab is opened. This is the start page when you open the Admin Tool.

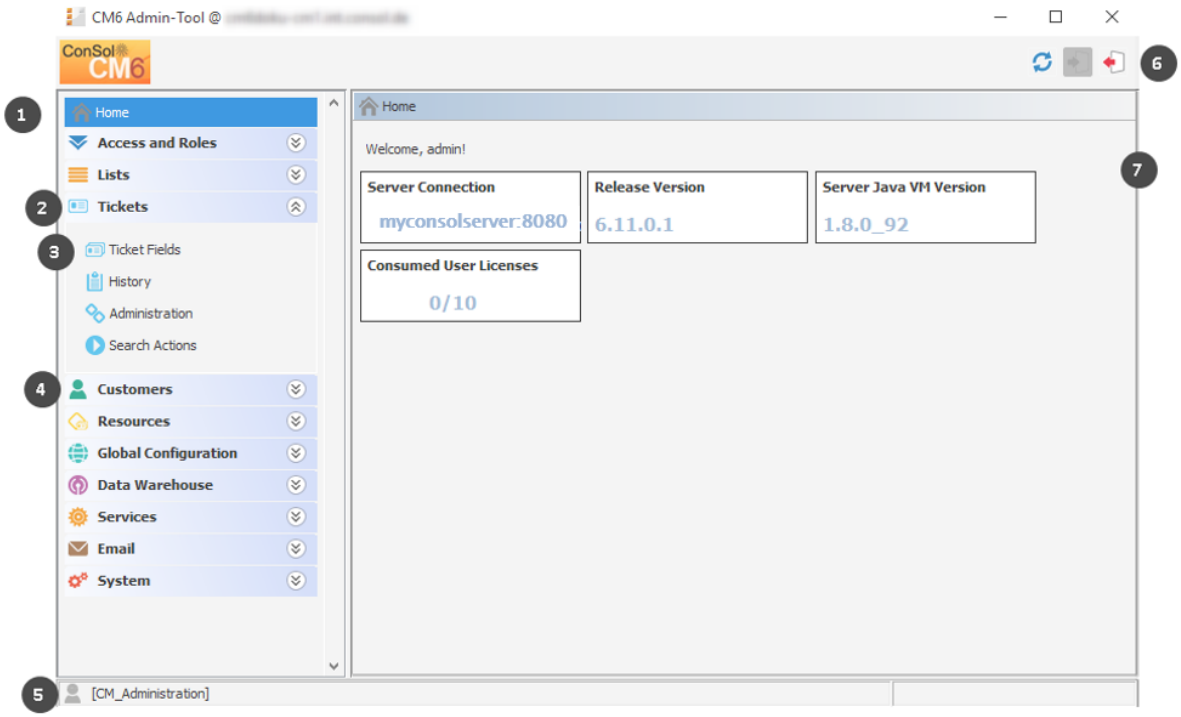


Figure 14: *ConSol CM Admin Tool - GUI overview*

The **Home tab** contains the following data:

- **Server Connection**
  The server name and port where the ConSol CM instance is running
- **Release Version**
  The exact version of the ConSol CM version
- **Server Java VM Version**
  The version of the Java VM (virtual machine) which is used for the current ConSol CM instance
- **Consumed User Licenses**
  The number of licenses which are used, compared to the number of available licenses.
  You can select one of two display modes here by manipulating the value of the CM system property cmas-app-admin-tool, admin.tool.consumed.licenses.pool.name. The value refers to a section in the license file:
  - **CONCURRENT_USERS**
    The number of engineers who are currently logged in using the Web Client is displayed.
  - **TRACK**
    Only possible if CM/Track is in operation. The number of customers who are currently logged in using CM/Track (the ConSol CM portal) is displayed.

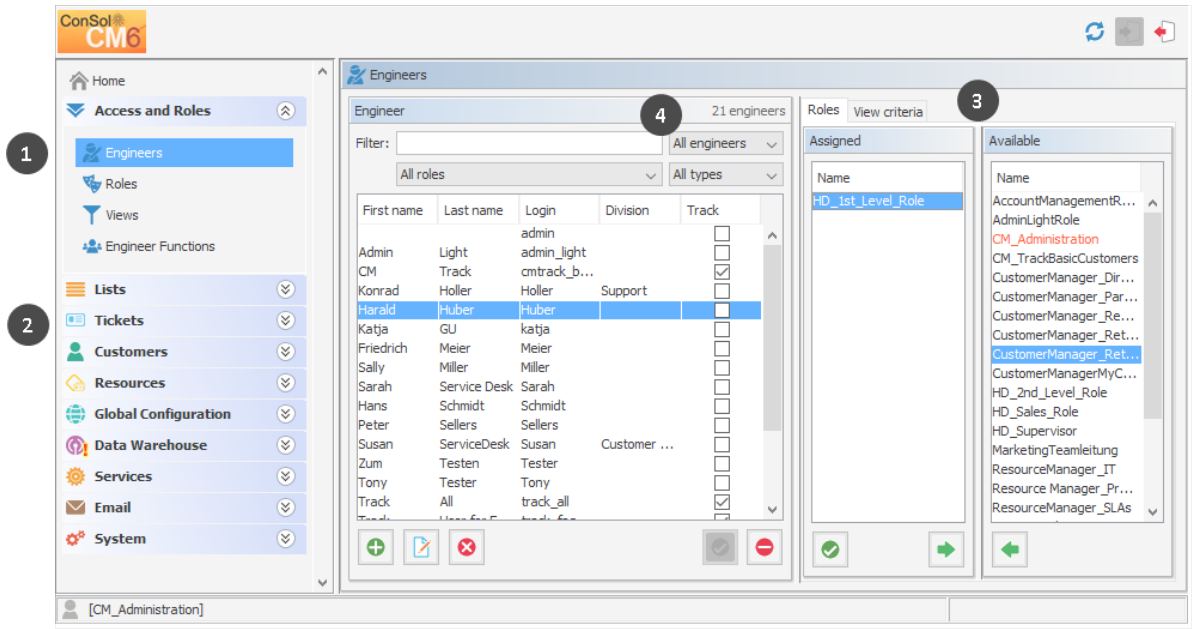


Figure 15: *ConSol CM Admin Tool - GUI overview: Navigation item open*

The user interface of all other navigation items (besides the *Home* tab) basically show the same structure. The following example shows the tab of the navigation item *Engineers*.

The left part of the screen shows the navigation tree (1, 2). The right part of the screen shows the settings for the active navigation item (Engineers, 1, in the example).

In the active tab, *Engineers* in the example, a list on the left shows the elements which can be modified. Elements can be added, edited, deleted, disabled, or enabled.

The attributes of an element are displayed on the right (3). You can move them from a list of available attributes to a list of assigned attributes either via double click or via clicking the *Assign* icon (example: *available roles* and *assigned roles*). Attributes can also be assigned via checkboxes or list boxes (not displayed here).

There are a couple of options to help you find the entries you want to edit more quickly:

- **Filters (4)**
  Filters help you find entries in lists (e.g. in the engineer list) rather quickly. There are two types of filters:
  - **Text filters**
    Type in the characters of the required word (e.g. the engineer name) and the list is updated automatically, displaying only matching entries.
  - **Drop-down menu filters**
    Select a category (e.g. *all engineers*) and only the matching list entries (e.g. engineer names) are displayed.
- **Sorting**
  You can sort the entries in ascending or descending order by clicking in one of the title fields of the list. The small up and down arrow icons denote the current sort order.

Usually all changes performed in the Admin Tool are submitted immediately without the need to synchronize any data. However, if changes have been performed in another module and the Admin Tool has to use the new data, synchronization is required. This can be achieved by clicking the *Refresh* button in the icon bar.

One example for this is the deployment of a new workflow using the Process Designer. Before the new workflow can be assigned to a new queue, you have to synchronize the data in order to let the Admin Tool know that there **is** a new workflow. The Admin Tool loads all data from the database anew, including the new workflow. This new workflow can then be used for further operations, like assigning it to a new queue.

A.12.3 Inline Validation of Input Values

For all data models (customer data model, resource data model, ticket data model), the input values are validated during input in order to avoid inconsistencies concerning the names of database fields. **Each value of a field has to be unique on the respective field level.** See the following examples:

- A new resource type cannot have the name of an existing resource type, but it can have the name of an existing resource field group.
- A resource field group cannot have the same name as another resource field group, no matter in which resource type (see example a below).
- A new ticket field cannot have the same name as another field in the same ticket field group, but it can have the same name as another field in another ticket field group (see examples b and c below).
- A new customer data model cannot have the same name as an existing customer data model.

A field which is not valid is marked in red. It is not possible to save this field value.

We recommend to use a defined terminology and syntax for data fields. Field names should be self-explanatory and unique as far as possible. There might be exceptions from this rule due to the implementation of certain special algorithms, but in general this is what you should keep in mind as a ConSol CM administrator!

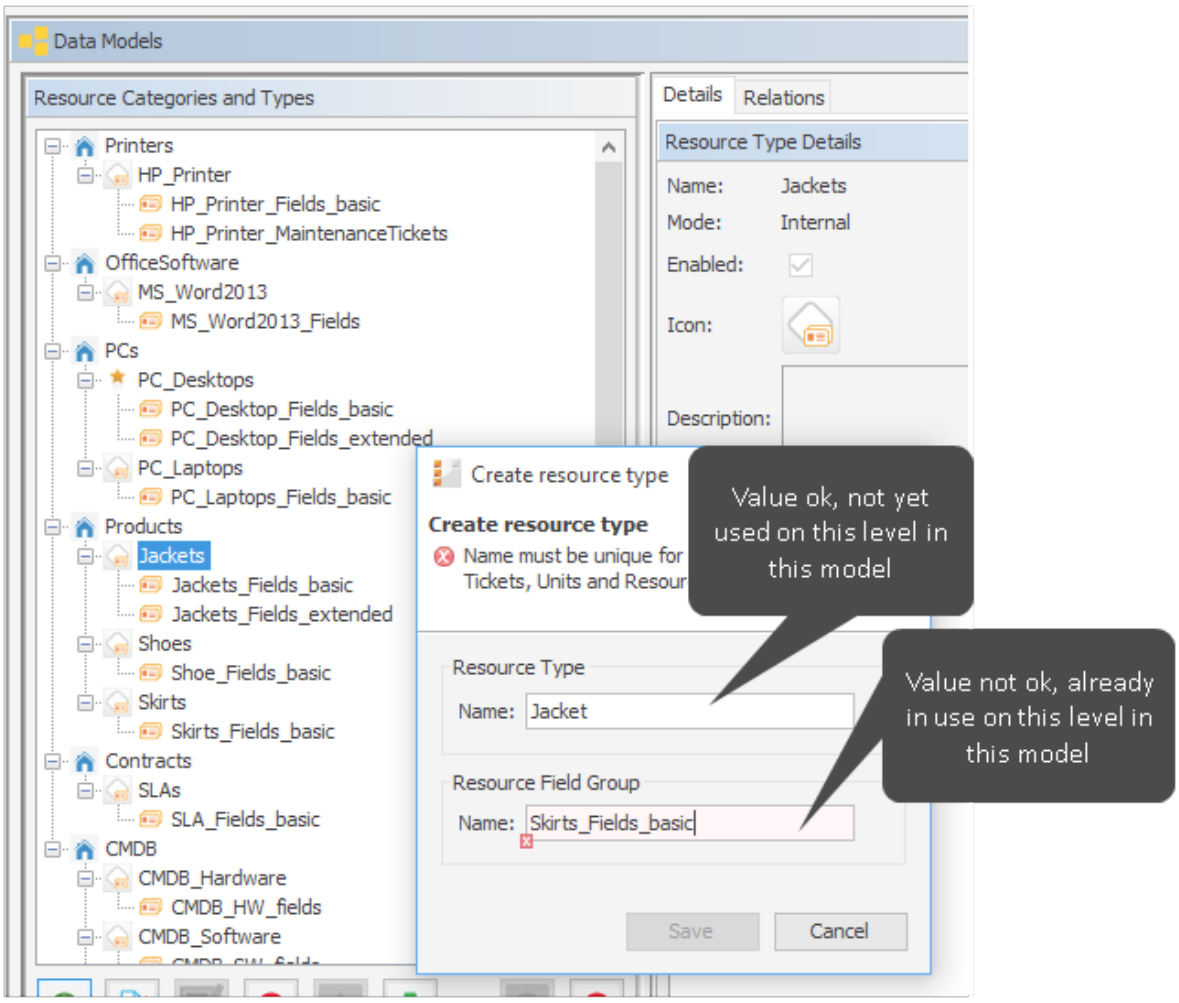


Figure 16: *Inline validation of input values during set up of the resource data model. Example a.*

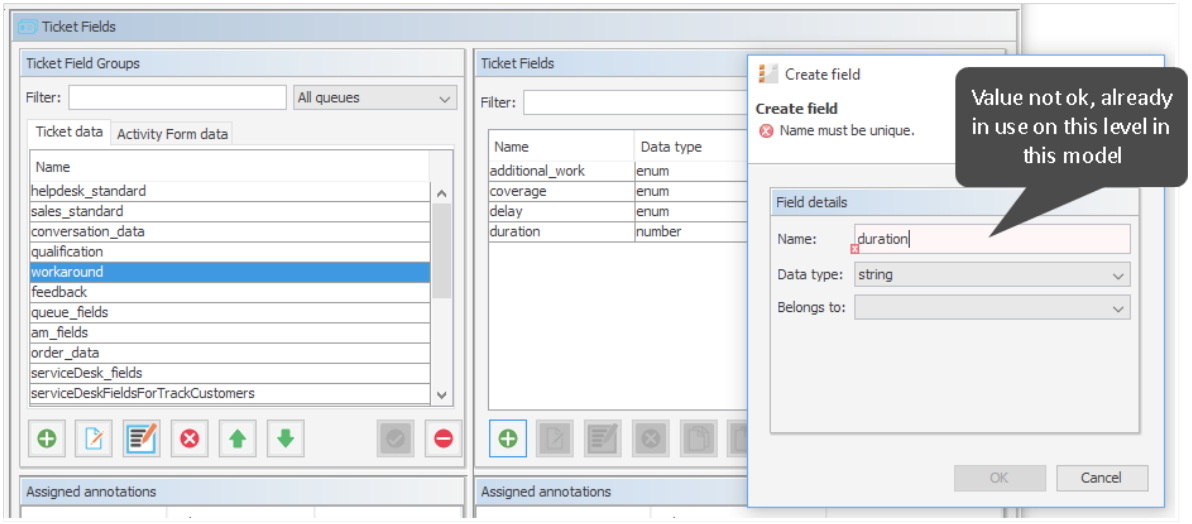


Figure 17: *Inline validation of input values during set up of the ticket data model. Example b.*

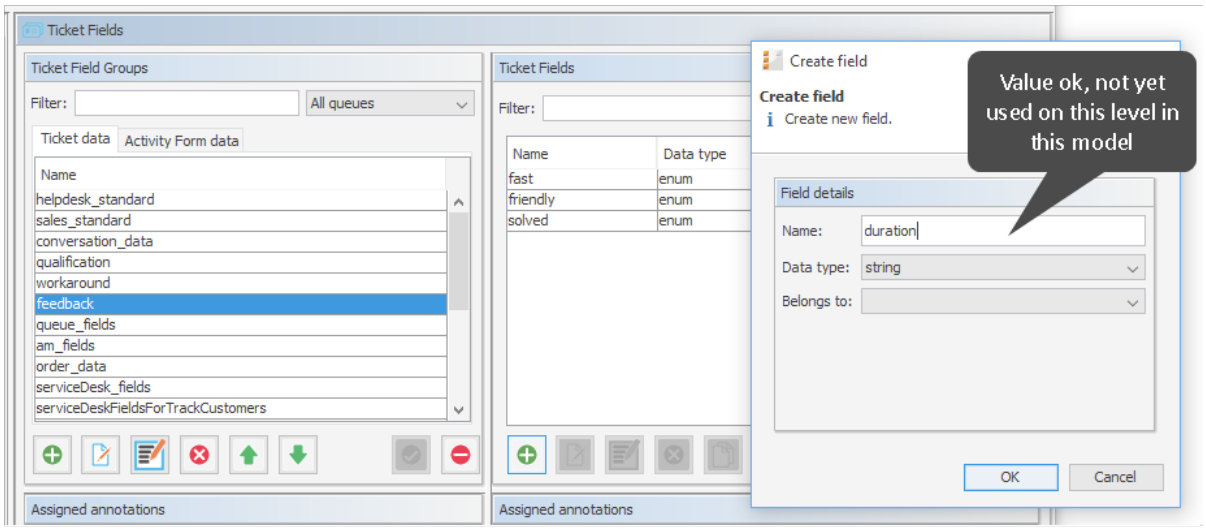


Figure 18: *Inline validation of input values during set up of the ticket data model. Example c.*

A.12.4 Icons and Other GUI Elements

You work with the following icons when you administrate a ConSol CM system. Here, the general explanations are provided, please see the respective sections for a detailed explanation of conditions and implications of applying the functionalities in the respective context.

Icon	Name	Meaning/Function	Examples
	Add / New	Add (= create) a new element of the respective type.	Add (create) a new engineer, a new view, a new script
	Edit	Edit the selected element. Usually a pop-up window is opened.	Edit an engineer, a class of text, a ticket field group
	Delete	Delete the selected element from the database. Cannot be restored.	Delete an engineer, a customer field group, a resource
	Copy	Copy the selected element	Copy a role
	Paste	Available for ticket, customer and resource fields	Paste the copied field into the selected (target) field group
	Activate	(Re-) activate an element which had been deactivated before.	(Re-) activate an engineer whose account had been deactivated.
	Deactivate	Deactivate the selected element. Might be safer than deleting it. Elements which are in use cannot be deleted, so it can be an alternative to deactivate them.	Deactivate an engineer (account), e.g. when an employee takes a sabbatical. Deactivate a customer who has canceled the contract.
	Move upwards	Move an element one step upwards in a list. This might have implications for the Web Client.	Move a view two steps upwards in the list. The view will then be displayed in the view list in the Web Client at the new position.
	Move downwards	Move an element one step downwards in a list. This might have implications for the Web Client.	Move a ticket field group one step downwards in the list. It will then be displayed in the ticket data section, maybe in the group section, at the new position.
	Unassign	Unassign/remove an element from the selected item.	Unassign a role from the selected engineer.
	Assign	Assign an element to the selected item.	Assign a role to an engineer. Assign a view to a role.

Icon	Name	Meaning/Function	Examples
	Annotate (CM versions 6.10.2 and below)	Open the *Annotation* pop-up window	Used for ticket field groups, ticket fields, customer field groups, customer fields, resource field groups and resource fields
	Annotate (CM versions 6.10.3 and up)	Open the *Annotation* pop-up window	Used for ticket field groups, ticket fields, customer field groups, customer fields, resource field groups and resource fields
	Localize / Internationalize	Open the pop-up window to enter the localized / internationalized names of the technical objects. The languages which have been configured in the Admin Tool are offered.	Localize the name of a ticket field.
	Search	Open the Search GUI.	Start the ticket search.
	Select all	Marks all elements (often checkboxes), no database action is performed, only GUI helper	Mark all permissions for a role concerning queue access
	Deselect all	Deselects all elements (often checkboxes), no database action is performed, only GUI helper	Deselect all permissions for a role concerning queue access
	Start	Start the selected element (usually a service)	Start a CM service
	Stop	Stop the selected element (usually a service)	Stop a CM service
	Upload	Open the file browser to upload a file to the CM system	Upload a script
	Download / save	Save a file on the file system	Save a script as file in the file system.
	Save and close	Saves the element (usually a script) and closes the editor in edit mode. Switches to view mode.	Save a script in the script section and switch to view mode of the script.
	Close without saving	Does not save the element (usually a script) and closes the editor in edit mode. Switches to view mode.	Do not save the edited script in the script section and switch to view mode of the script.

Icon	Name	Meaning/Function	Examples
	Refresh	Updates the data in the Admin Tool.	Update the data in the Admin Tool after making changes in the Process Designer.

A.12.5 Localization of Terms Displayed in the Web Client

ConSol CM can be configured for international environments. In this way, every engineer can work with the country-specific language in the web browser. The following principles are important concerning localization/internationalization:

The CM administrator configures the languages which will be available in the entire system. This is done using the Admin Tool and is explained in section Languages.

The languages which are configured as available languages in the Admin Tool can then be applied for all terms which have been configured for the CM system, e.g., the terms for ticket fields. Using the Process Designer, the labels for workflow activities can also be localized. This is explained in the *ConSol CM Process Designer Manual*. Thus, there are two different categories of terms in the Web Client:

1. **Standard CM terms/labels** which cannot be modified using the Admin Tool or Process Designer (however, they might have been adapted to your system in case a customer specific software has been developed). Terms like the labels for the group headers in the ticket list (e.g., *Own tickets*, *Workgroup tickets*, *Unassigned tickets*) or the labels *Favorites* and *Workspace* belong to this category.
   For these terms/labels, two standard languages are available: English and German. English is also the overall standard language. You cannot modify standard CM terms! They are available in English and in German and represent a fixed set of terms and labels.
2. **System-specific terms, i.e., terms/labels which can be localized using the Admin Tool** or the Process Designer. Localization using the Admin Tool will be explained in the following section. For an explanation of localization using the Process Designer, please refer to the *ConSol CM Process Designer Manual*.
   For system-specific fields, the localized terms/labels have to be entered manually for each language.

In the Web Clients (CM Web Client and CM/Track), the labels are displayed according to the browser locale which has been set by the engineer or the customer.

A.12.5.1 The Different Modes of Localizing Terms and Labels Using the Admin Tool

General Principle of Localization in CM

Depending on the location in the Admin Tool, there are different GUI elements which are used to set localized values. However, the general principle is valid for all those locations. The general principle works as follows (explained for the example of a ticket field).

Technical name:

Each field or object has a technical name, e.g. the ticket field *priority*. This name is set when the field or object is created and should not be changed afterward because it might be used in Admin Tool or workflow scripts which would fail when the name has been modified. CM throws a warning message and blocks the action when someone tries to modify a technical field or object name. Only when the field or object is new and has not been used yet, it is possible to change the technical name.

Localized names:

For each field, field group or object a localized name can be set for each of the languages which have been configured for the CM system (using the settings in the navigation group *Global Configuration*, navigation item *Languages*).

For example, the ticket field with the technical name *priority* is named *HD Priority* in English and *HD Priorität* in German.

In the web clients (CM Web Client and CM/Track), the labels of the fields and objects are displayed according to the browser locale of the engineer or customer. Some variants are possible:

1. Variant #1:
   The language of the browser locale has been configured and all labels and terms are displayed in this language. For example, the language *French* has been added in the Admin Tool, all system-specific terms have been localized manually for French (using the Admin Tool and the Process Designer). Then all system-specific terms are displayed in French, e.g., the workflow activities, the names of the ticket fields, and the names of the views.
2. Variant #2:
   The language of the browser locale has been configured, but not all labels and terms are displayed in this language. For example, the language *French* has been added in the Admin Tool, but only some system-specific terms have been localized manually for French (using the Admin Tool and the Process Designer). Then the localized system-specific terms are displayed in French, e.g., the workflow activities, the names of the ticket fields, and the names of the views. All non-localized terms (i.e. where the field for the localized term is empty) are displayed in the language which has been configured as default in the *Languages* settings.
3. Variant #3:
   The language of the browser locale has not been configured. For example, the localized terms and labels have been set for English, German and French, but the engineer has set the browser locale to ES (Spanish). In this case, all labels and terms are displayed in the language which has been configured as default in the *Languages* settings.

Localization of Data Fields

The localized terms for data fields are written directly into the *Labels* section in the Admin Tool. This applies to

- Ticket fields (for ticket data)
- Customer fields (for customer data)
- Resource fields (for resource data)

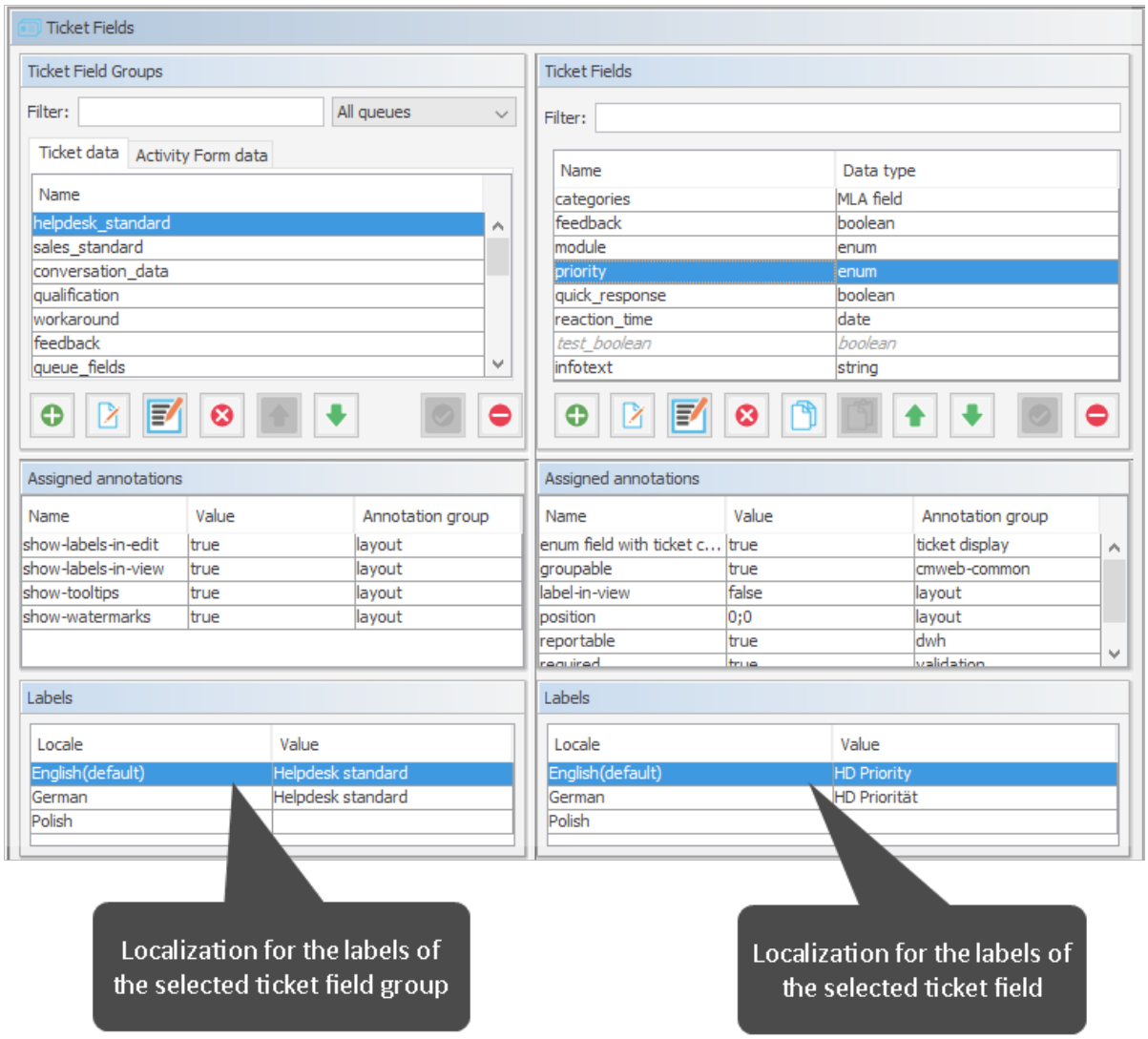


Figure 19: *ConSol CM Admin Tool - Localization of ticket fields*

In order to change the value of a field, click into the field and write the new term. Press ENTER to save the new value.

In the Web Client, the ticket field shown in the figure above is displayed as follows (in English and German):

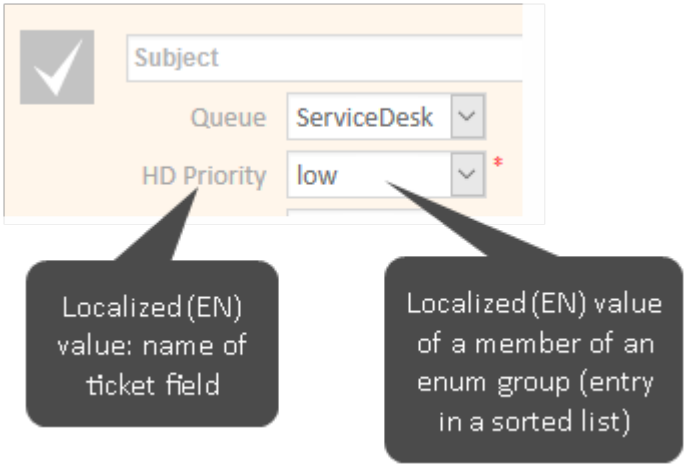


Figure 20: *ConSol CM Web Client - Ticket field displayed in English browser locale*

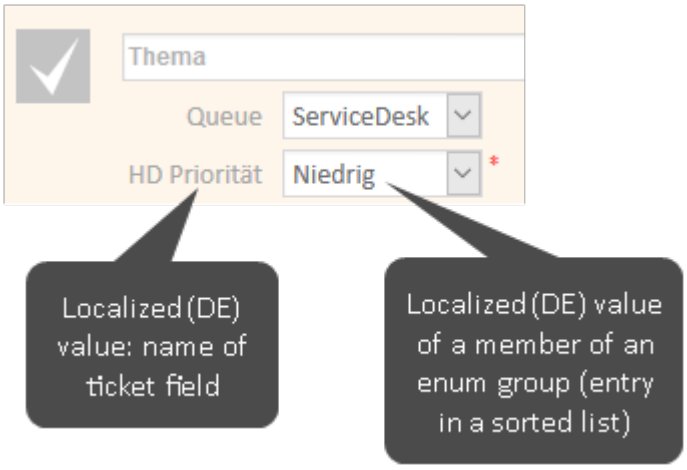


Figure 21: *ConSol CM Web Client - Ticket field displayed in German browser locale*

Localization of Objects in General, Type 1

A great number of objects can be localized using the *Localize / Internationalize* (globe icon) button which opens a pop-up window with the localization configuration.

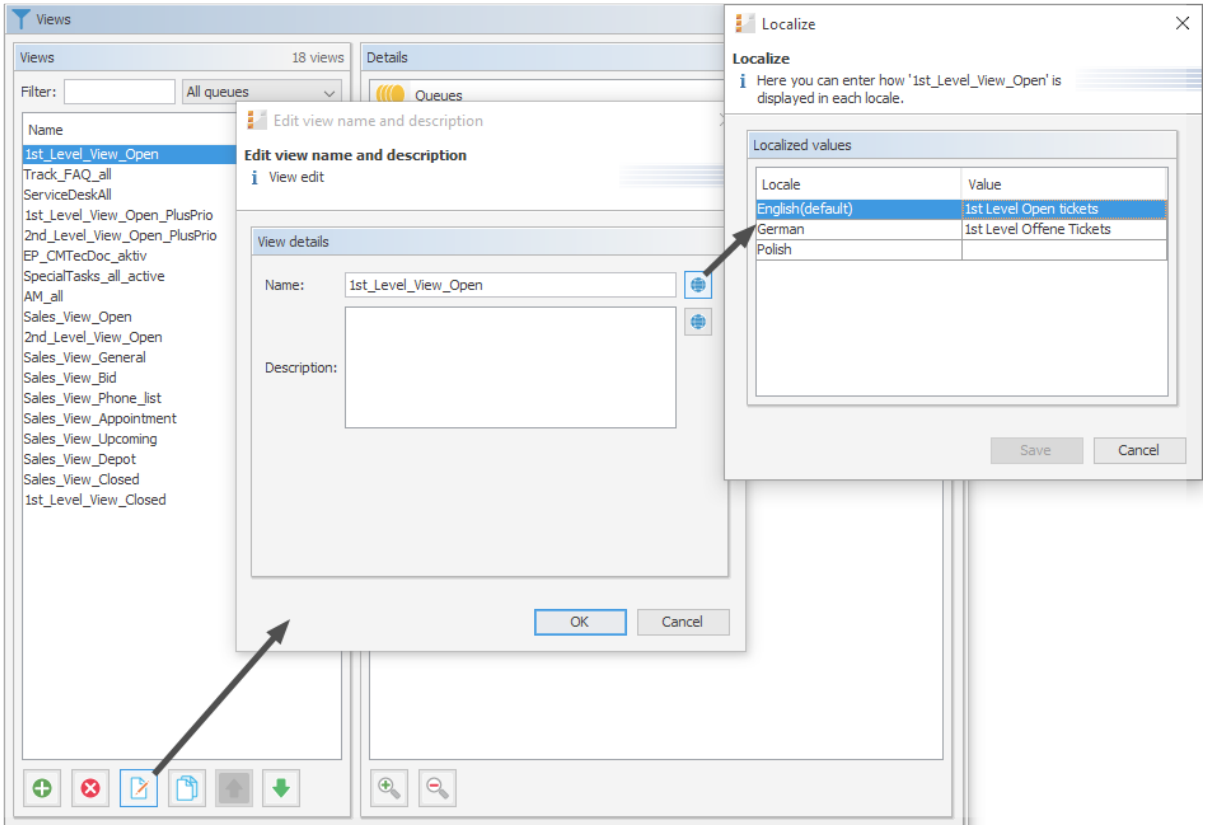


Figure 22: *ConSol CM Admin Tool - Localization of objects, example: name of a view*

In order to change the value of a field, click into the field and write the new term. Press ENTER to save the new value.

Localization of Objects in General, Type 2

Some objects are localized using the localization table within the pop-up window which is used to edit the object.

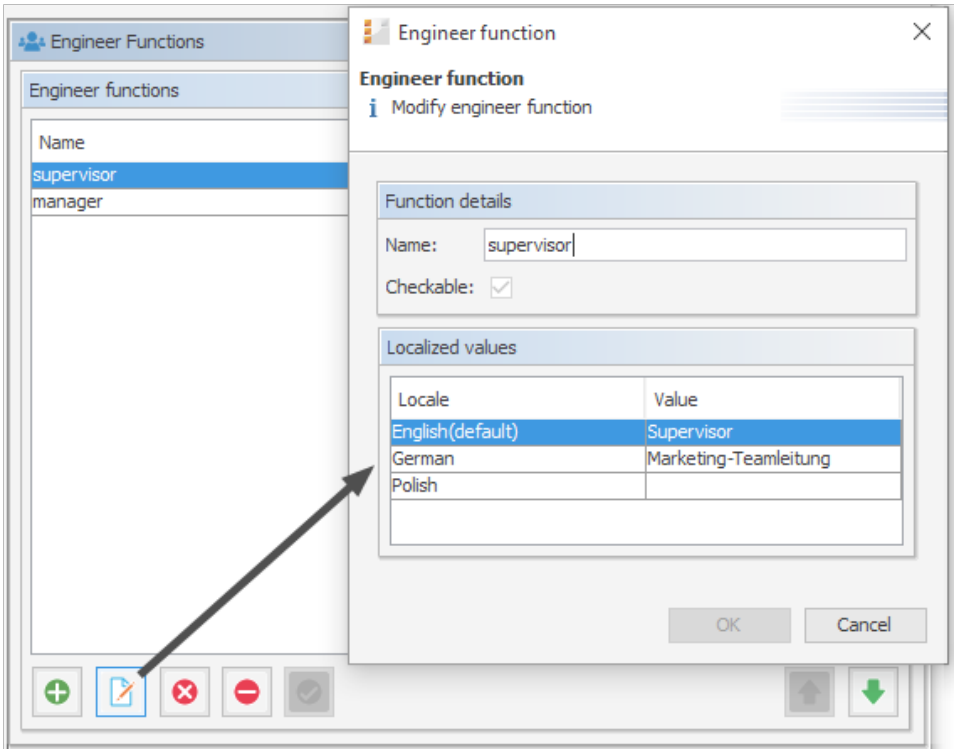


Figure 23: *ConSol CM Admin Tool - Localization of objects, example: name of an engineer function*

Using UTF-8 Symbols in Localized Fields

In order to have graphical symbols displayed in text fields, you can use UTF-8 symbols. This works for all localized fields, e.g., string fields, queue names, or enums. Just copy the UTF-8 symbol from a unicode table page and paste it into the text field in the Admin Tool.

Example:

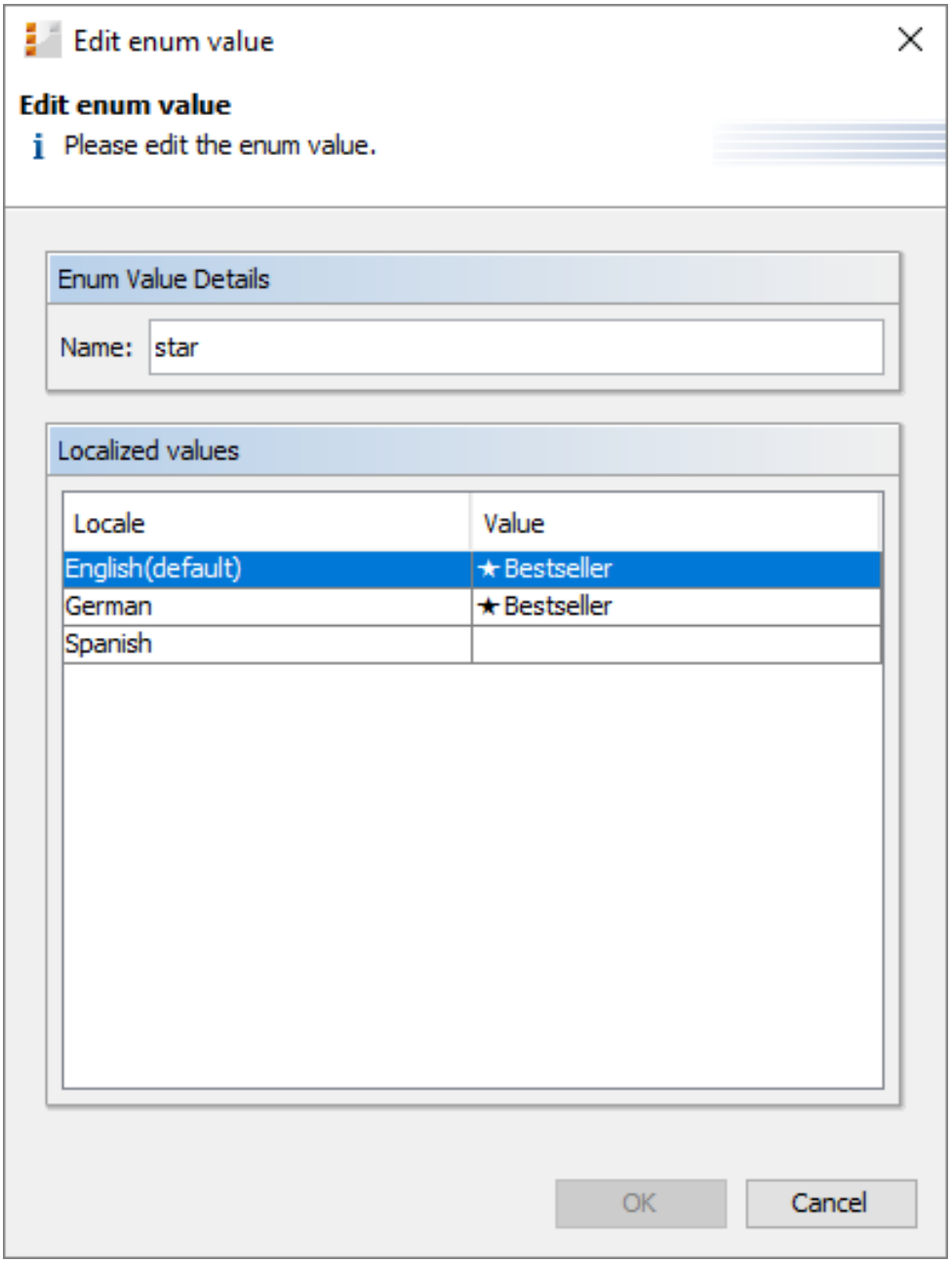


Figure 24: *UTF-8 symbol in a localized field in the Admin Tool*

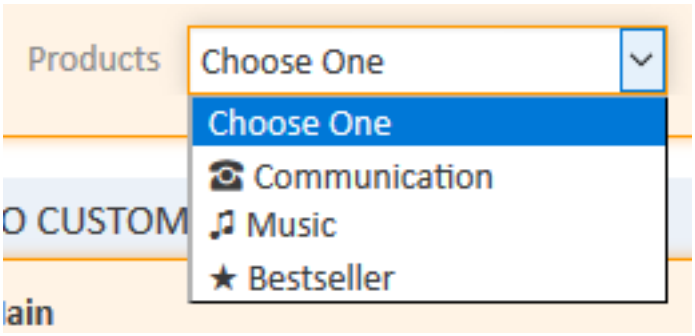


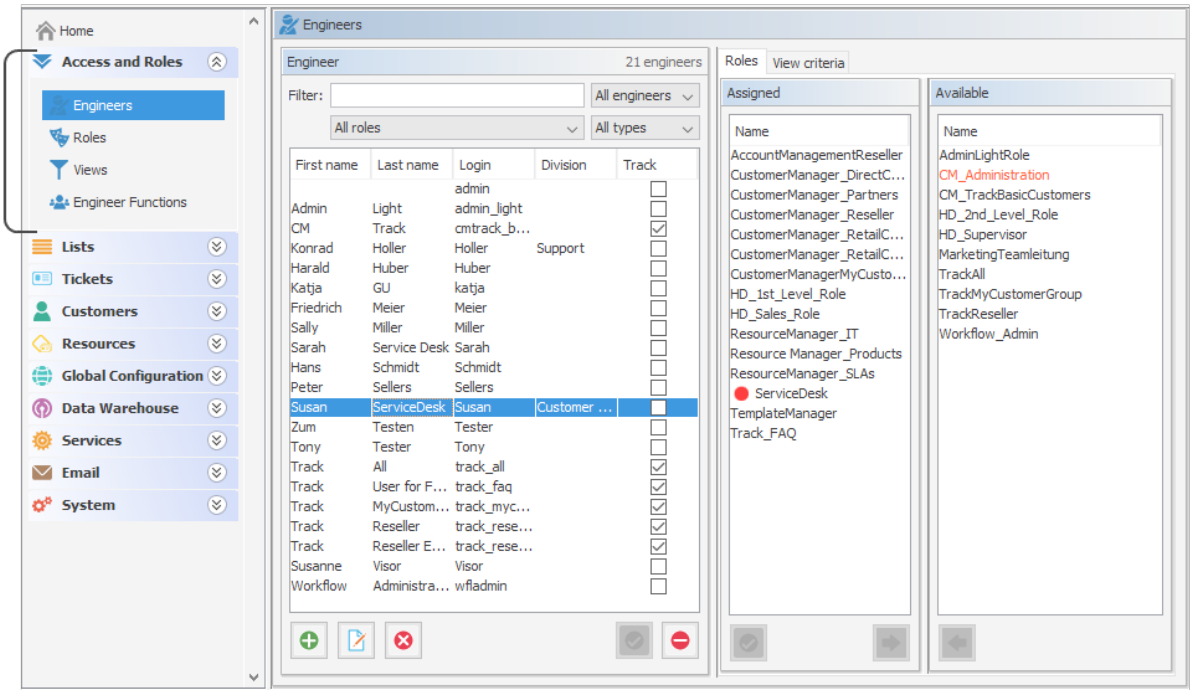
Figure 25: *Display of UTF-8 graphical symbols in the Web Client, here: enum*

> In case you work with a MySQL database, use the following settings to work with the UTF-8 graphical symbols:
>
> - CHARACTER SET = utf8mb4
> - COLLATE = utf8mb4_unicode_ci

Localization of Labels

Several terms in the Web Client can be localized using labels. This is explained in section Labels.

B - Access and Roles Section



This section provides some background knowledge about the general principle of ConSol CM access permissions and shows you how to define the engineer accounts with the required parameters.

Please read the following sections:

- Engineer Administration
- Role Administration
- View Administration
- Engineer Functions

B.1 Engineer Administration

This chapter discusses the following:

B.1.1 Introduction to Engineer Administration

An engineer account is the basic access object which allows an engineer or administrator to access the Web Client, Admin Tool, or Process Designer. During system set-up an administrator account for the first access to the Admin Tool is created. Using this account, you can set up further accounts.

Newly created engineer accounts do not have any permissions. These permissions have to be assigned through one or more roles displayed in the *Roles* tab. If you have not created any roles yet, you only see the administrator role (see Tab Roles - Assign Roles to an Engineer Account).

Views define which tickets engineers will see in the ticket list (to-do list) of the Web Client. They are created in the View Administration and assigned via roles. On the engineer administration page you can preset dynamic view criteria for specific engineers (see Tab View Criteria - Define Engineer-Specific View Criteria).

We would recommend that you create at least one role and one view first before you create any engineer accounts.

B.1.2 Engineer Administration Using the Admin Tool

To open the Engineer Administration, open the navigation group *Access and Roles* in the Admin Tool and click the navigation item *Engineers*.

B.1.2.1 List of All Engineers

When you have opened the Engineer Administration, a list of all engineers is displayed. Engineer accounts which are currently deactivated are displayed in gray. You can easily narrow down the list of engineers to engineers of only one role (for details, please refer to section Role Administration) by using the drop-down menu (filter) for roles.

B.1.2.2 Create or Edit an Engineer Account

You can create a new engineer account or edit the settings of an existing account. Both actions are performed in the same pop-up window.

You reach this screen by clicking the *Add* icon (to create a new account) or the *Edit* icon (to edit an existing account) below the list of engineers in the navigation group *Access and Roles*, navigation item *Engineers*.

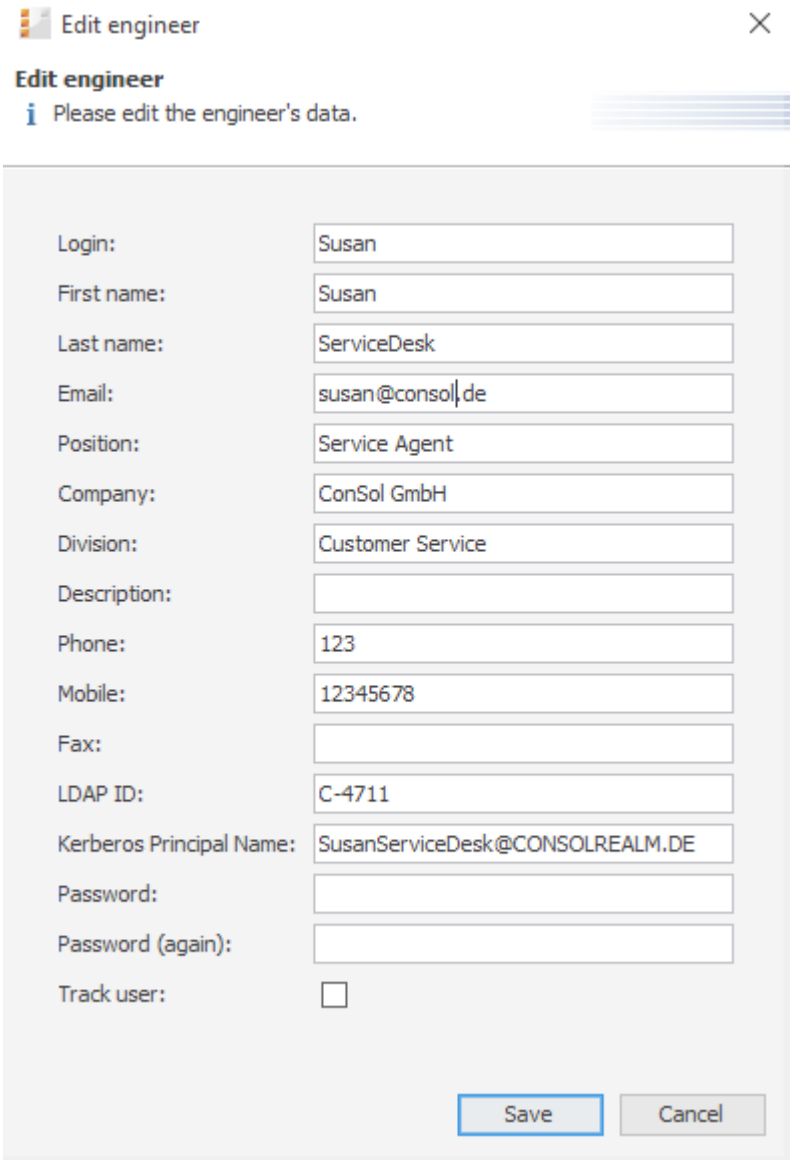


Figure 26: *ConSol CM Admin Tool - Access and Roles, Engineers: Editing an engineer account*

The window shows the parameters describing an engineer account:

- **Login:**
  Mandatory. This field contains the account name which has to be entered on the login page of the Web Client. Please use only international alphabetic and numeric characters, no blanks, punctuation marks, or special characters such as umlauts, hyphens, or the like.
- **First name:**
  Optional. The engineer's first name. This field is optional but will be displayed in the Web Client for the engineer. The entry may contain alphabetic characters, blanks, comma, periods, and hyphens. Please do not use other characters.

- **Last name:**
  Optional. The engineer's last name. This field is optional but will be displayed in the Web Client for the engineer. The entry may contain alphabetic characters, blanks, comma, periods, and hyphens. Please do not use other characters.
- **Email:**
  Mandatory. The engineer's email address. Please use only international alphabetic and numeric characters, hyphens, underscores, periods, and the @ sign. The entry of multiple email addresses in one line is not allowed.
- **Position:**
  Optional. The engineer's position or function in the company. This field is optional and has a descriptive function only. The entry may contain alphabetic characters, blanks, comma, periods, and hyphens. Please do not use other characters.
- **Company:**
  Optional. The engineer's company. This field is optional and has a descriptive function at the moment. The entry may contain alphabetic characters, blanks, comma, periods, and hyphens. Please do not use other characters.
- **Division:**
  Optional. The division in which the engineer works. This field is optional and has a descriptive function. The entry may contain alphabetic characters, blanks, comma, periods, and hyphens. Please do not use other characters.
- **Description:**
  Optional. An additional description for the engineer account. This field is optional and will **not** be displayed in the Web Client. The entry may contain alphabetic characters, blanks, comma, periods, and hyphens. Please do not use other characters.
- **Phone:**
  Optional. The engineer's phone number. This field is optional and has a descriptive function at the moment.
- **Mobile:**
  Optional. The engineer's mobile phone number. This field is optional and has a descriptive function at the moment.
- **Fax:**
  Optional. The engineer's fax number. This field is optional and has a descriptive function at the moment.

> ⚠ Several fields which contain engineer data (like *Company*, *Division*, or *Phone*) are optional fields. However, if you work with text templates which contain engineer data fields (see section The ConSol CM Text Template Manager) the emails or comments will not be formed correctly if the data is missing. For example, the field *ticket-engineer, phone* cannot be filled-in in the template if it has not been set for the engineer in Engineer Administration! So please make sure all data which will be required later on is filled-in correctly in the first place!

- **LDAP ID**
  The LDAP user ID if LDAP is used for authentication. No password has to be set here.

  > If you do not enter an LDAP ID here, the login will be used as authentication login parameter for the LDAP server (if LDAP authentication is activated)!
  >
  > For detailed information about the LDAP authentication, please see section LDAP Authentication for Engineers in the Web Client.

- **Kerberos Principal Name**
  The Kerberos principal name if Kerberos V5 protocol is used for authentication. Engineers can log in to the Web Client by using their Windows credentials. For detailed information about Single Sign-On with Kerberos, please see section Single Sign-On with ConSol CM Using Kerberos (in a Windows Domain).
- **Password:**
  Mandatory. The engineer's password is mandatory. Please use only international alphabetic and numeric characters, and punctuation marks, do **not** use any special characters such as, e.g., umlauts. The password entered will be shown as a string of asterisks. Please see section Engineer Administration for information about the optional password policy.
- **Password (again):**
  Mandatory. Please repeat the password here. This security query helps to avoid erroneous entries which would not be noticed otherwise because the password is shown as a string of asterisks. Please see section Engineer Administration for information about the optional password policy.

  > This field will only appear if the engineer authenticates against the Web Client via the CM database, i.e., if LDAP or Kerberos authentication is used, the field is not visible.

- **Track user**
  This checkbox has to be ticked if you want to create a technical engineer (or CM/Track user profile) used to define access permissions for CM/Track users. The available CM/Track users (user profiles) are shown in the Web Client when creating or modifying a customer. So, by ticking this checkbox, you do not define a real engineer (a person) with access permissions to the system but rather a user profile for CM/Track which is then assigned to one or more customers who should access the portal ***CM/Track*** using those access permissions. For a detailed description of the CM/Track access definition see also section CM/Track V1: System Access for CM/Track Users (Customers) for CM/Track V1 and CM/Track V2: System Access for CM/Track Users (Customers) for CM/Track V2.

Click *Save* afterwards to store your entries and to close the window.

B.1.2.3 Delete an Engineer Account

To delete an engineer account, select the account in the list and click the *Delete* button. Since an engineer account can only be deleted if there are no tickets (open or closed) for it anymore, you have to assign its tickets to another engineer. The name of the deleted engineer is still displayed in all history

entries in tickets and customer pages which were performed by this engineer.

In case you do not want to transfer any tickets to another engineer, you can deactivate the engineer account. See next section.

B.1.2.4 Disable or Enable an Engineer Account

If engineers should not have access to the system for a certain period of time (e.g., because they have taken a sabbatical), an account can be disabled. There will be no change regarding the tickets of these engineers, but they cannot log in anymore and other engineers cannot assign any tickets to their accounts.

To disable an engineer account, select the account and click the *Deactivate* button. The entry in the list is shown in gray italics afterwards. It is not possible to create new tickets or to edit existing tickets for this account. To re-enable the account, just click the *Activate* button at the bottom of the page.

B.1.2.5 Tab *Roles* - Assign Roles to an Engineer Account

On this tab you can assign roles to an engineer account. Select the account on the left and then the desired role(s) in the list of available roles on the right. Click the *Assign* button to move the selected roles into the list of assigned roles. Now an engineer with this account can act in the system according to the permissions set in the role(s) (see also Role Administration).

Set Roles as Main Roles

From the list of assigned roles you can choose one role as the main role for each engineer account. Select the desired role in the list and click the *Activate* button below the list. Afterwards the main role is marked with a red dot. Now the views of the main role always appear at the top of the view list in the Web Client for this engineer account.

Edit One of the Roles of an Engineer

Each role in the list of roles of an engineer offers a context menu with two items:

- **(Un-)Assign role**
  You can use this menu item to (un-)assign a role to/from an engineer. It has exactly the same effect as using the arrow buttons below the list.
- **Jump to role**
  You can use this menu item as a shortcut to quickly open the navigation element *Roles* of this role.

You reach this screen by right-clicking the name of a role in the list of assigned or available roles in the navigation group *Access and Roles*, navigation item *Engineers*.

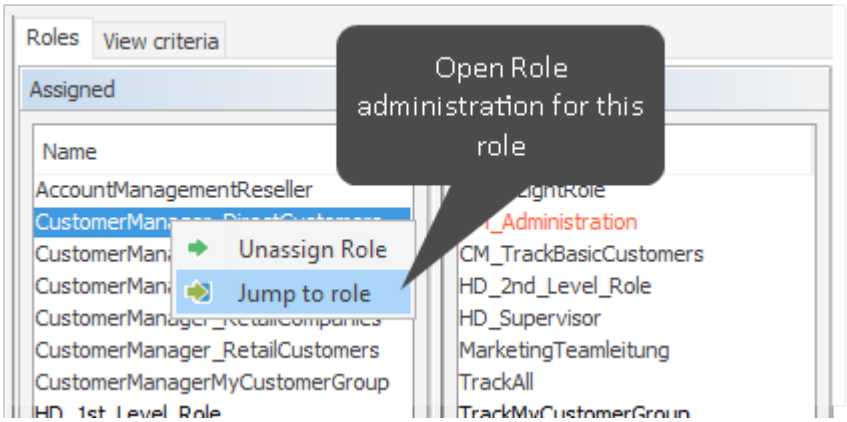


Figure 27: *ConSol CM Admin Tool - Access and Roles, Engineers: Context menu of a role*

B.1.2.6 Tab *View Criteria* - Define Engineer-Specific View Criteria

Here you can change the dynamic view criteria for an engineer. Dynamic criteria are used to give the engineer the possibility to adjust a view interactively in the Web Client (see also View Administration).

> This tab only shows view criteria if you have created a view with dynamic criteria and assigned it to the engineer's role.

You reach this screen by clicking the *View criteria* tab in the navigation item *Engineers*, navigation group *Access and Roles*.

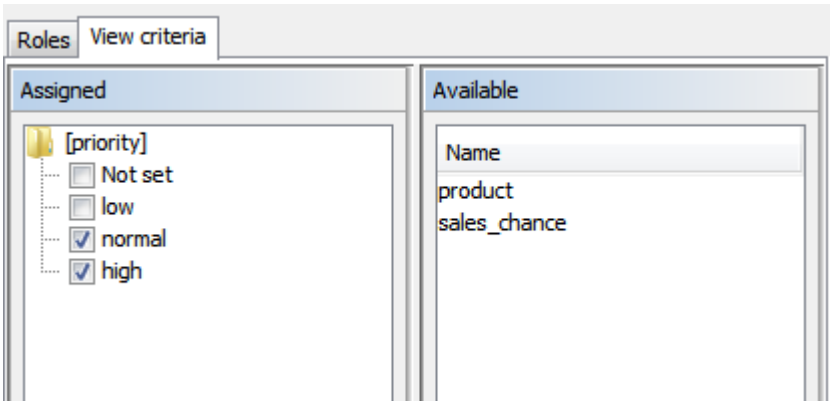


Figure 28: *ConSol CM Admin Tool - Access and Roles, Engineers: View criteria*

Select the engineer account on the left and then the desired criterion in the list of available view criteria on the right. Click the *Assign* button to move it to the list of assigned view criteria. You will see the possible values below the criterion in the list. Tick the checkboxes of the values you want to change or preset. The engineer can change these settings in the Web Client (profile page) and changes you have made in the Admin Tool are immediately visible in the engineer's profile page.

Example:
You have assigned the dynamic criterion *priority*. The list shows the values “Not set”, “low”, “normal”, and “high”. If you tick the values “normal” and “high” the engineer will only see tickets with normal and high priority after logging in to the Web Client. If you do not tick any values the engineer will see no tickets for this view. See section View Administration for details.

⚠ Please note that in a view with a dynamic criterion, only the tickets are displayed which match this criterion. So if engineers have not selected any criteria in their engineer profiles or if the administrator has removed all selections using the Admin Tool, the engineer's view will be empty! Make sure your users know about this fact and make sure you as an administrator are always aware of that fact.

B.2 Role Administration

This chapter discusses the following:

B.2.1 Introduction to Role Administration

Roles provide access rights and views, they specify what an engineer is allowed to do or to see. Without a role, an engineer can log in to the system but cannot perform any actions. Only by being assigned one or more role(s) does an engineer obtain system permissions. For each task in a company using the system there should be a role which defines its permissions. Engineers fulfilling the task should have this role.

When engineers log in to the system, they will have all permissions from all roles they have been assigned. So all permissions are added! There is no way of explicitly preventing access to objects in ConSol CM - access can only be **granted**! The sum of all granted permissions defines the final permissions for the engineer.

Roles define:

- **Access permissions for one or more queue(s)**
  E.g. read, write, and append rights are granted. The permissions are valid for all tickets in the queue(s).
- **Global permissions**
  Several system-wide permissions are managed here, e.g., the rights concerning template management, workflow design, and system administration. Using the option *Administrate access and roles*, it is possible to define an administrator "light" who can manage CM engineers with their system access permissions but who cannot modify technical system-wide settings. This is explained in section Defining an Administrator for Role and Engineer Administration Only.
- **Access permissions for customer data**
  Read, write, modify, and delete permissions for each distinct customer group.
- **Access permissions for resource data**
  Read, write, modify, create permissions, assigned on the basis of resource types.
- **Views**
  To do lists of tickets which are displayed in the ticket list in the Web Client.
- **Engineer functions**
  Additional engineer functions which can be assigned to members of this role, e.g., *approver.*

B.2.2 Role Administration Using the Admin Tool

You reach this screen by opening the navigation item *Roles* in the navigation group *Access and Roles*.

You might also see this screen when you have worked in the Engineer Administration and have selected *Jump to role* for a role in the role list of an engineer.

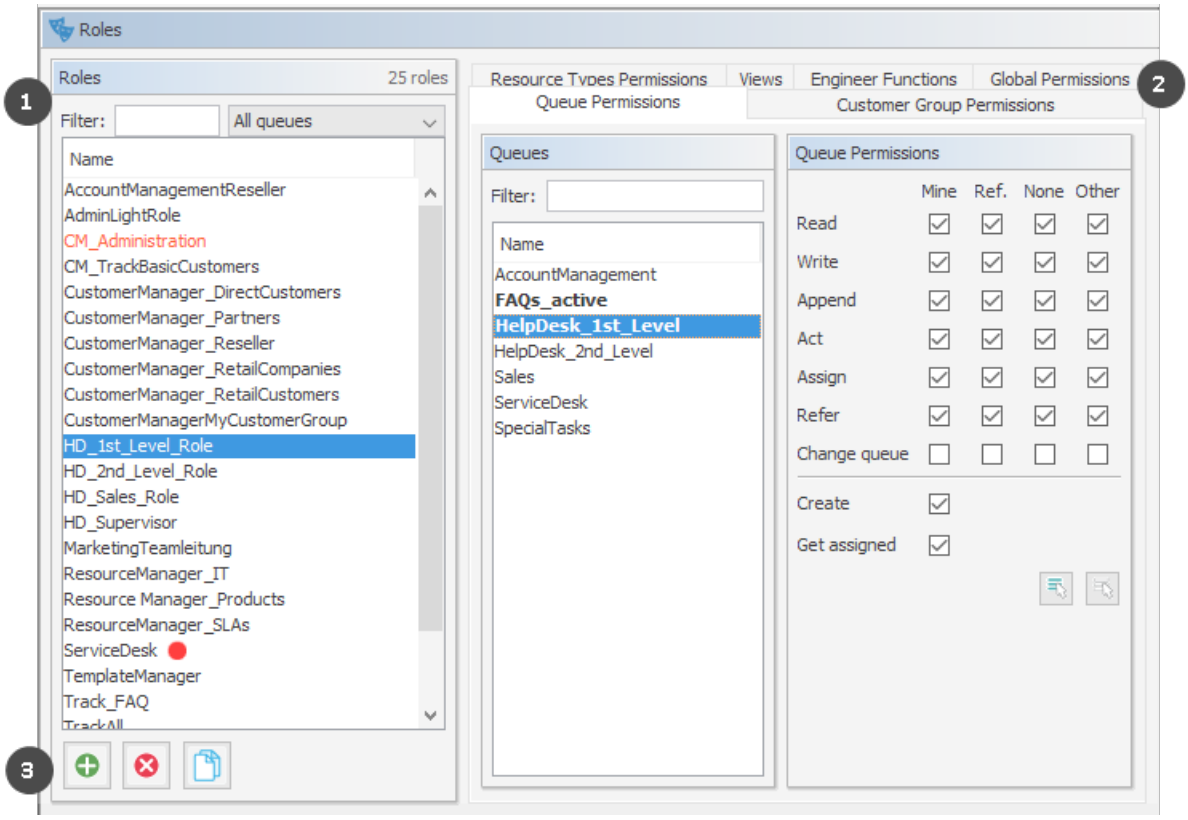


Figure 29: *ConSol CM Admin Tool - Access and Roles, Roles: Queue permissions*

You see the list of all available roles on the left-hand side (1) and the permissions which can be granted on the right-hand side (2). In the list of roles, all roles which have been set as *main role* for at least one engineer are marked with a red dot. You always work on the access permissions of the role which has been selected in the list of roles. Only one role can be selected at a time. You can use the buttons below the list of roles (3) to add, delete or copy a role.

On the right-hand side, several tabs are available:

- Tab Queue Permissions
- Tab Global Permissions
- Tab Customer Group Permissions
- Tab Resource Types Permissions
- Tab Views
- Tab Engineer Functions

All changes in the *Role Management* tabs take effect immediately or after clicking the *OK* button. You do not have to click the *Synchronize* button in the icon bar.

In the Web Client, engineers have to log in again to use their new roles. Views become effective after pressing F5 (page refresh).

Working in Role Administration, you always mark a role and then can display and modify the parameters of this role. However, it is not possible to display a list of all engineers who have been assigned this role. In order to have such a list displayed, please change to Engineer Administration (navigation item *Engineers*) and filter the engineer list for a certain role.

Please note that you can edit the name of a role by clicking on it in the list of roles and by modifying the name as required.

B.2.2.1 Create a Role

Click the *Add* button below the role list to create a new role. A pop-up window appears where you can enter the role name. Since the role name is used only for admin purposes and not displayed in the Web Client, no localization is required here. Afterwards you have to set the permissions of this role using the tabs on the right side of the page (see also the preceding picture).

Tab *Queue Permissions*

The permissions set in this tab apply to the selected role (left part of page) and the selected queue (center part of page). Without an entry here, an engineer with this role is not able to see tickets nor to perform any actions in the system.

You reach this screen by clicking the *Queue Permissions* tab in the navigation item *Roles*, navigation group *Access and Roles*.

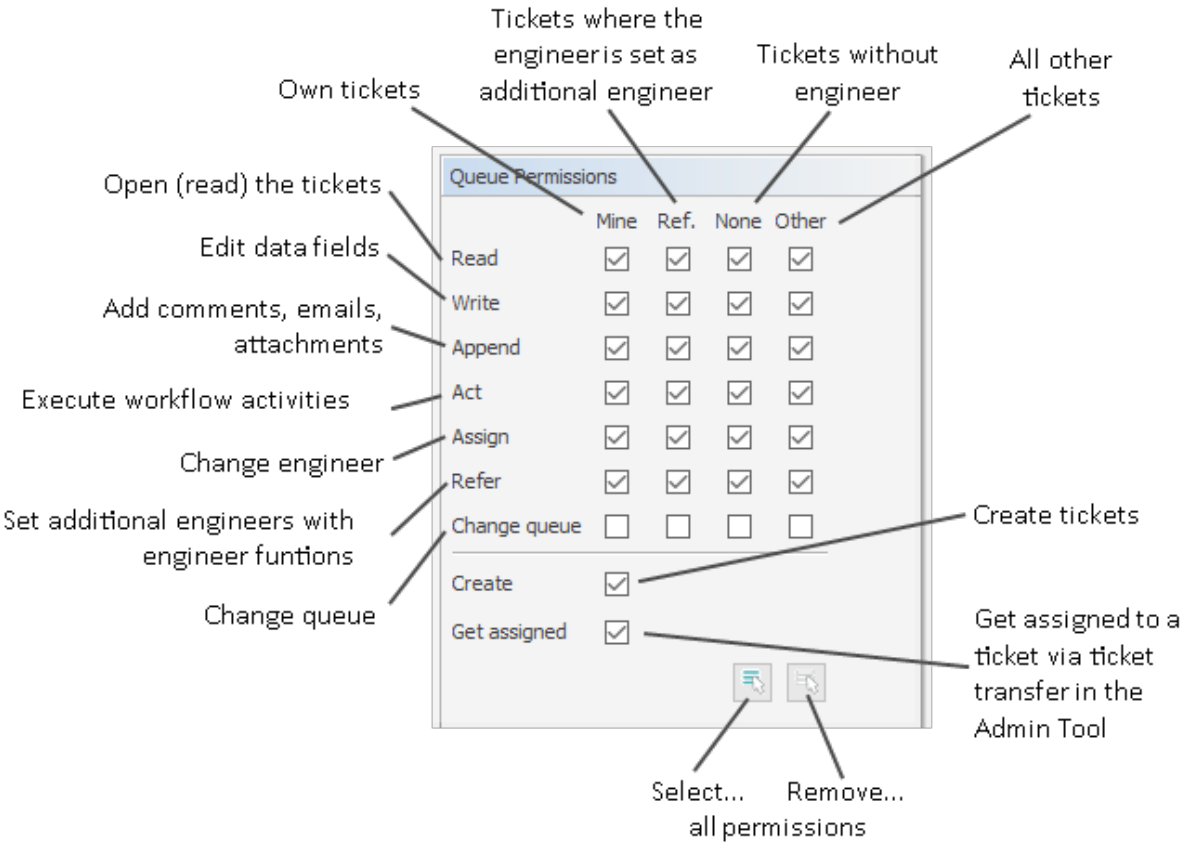


Figure 30: *ConSol CM Admin Tool - Access and Roles, Roles: Setting queue permissions*

The following permissions can be set:

- **Read**
  Read tickets.
- **Write**
  Edit data fields (default fields, ticket fields, etc.) of a ticket. The fields might be located in the ticket header section or in the group section.
- **Append**
  Add information to a ticket (comments, emails, attachments, time booking entries), i.e. add content in the ticket history.
- **Act**
  Execute workflow activities, i.e. move the ticket forward in the workflow.
- **Assign**
  Assign tickets to another engineer. The permission to assign a ticket to oneself and to accept a ticket is not relevant in this context.
  The engineer who should receive the ticket has to have at least one role with the *Get assigned* permission!

- **Refer**
  Assign an additional engineer (with engineer function, see Tab Engineer Functions) for a ticket.
- **Change queue**
  Move a ticket from this queue to another queue.
  If the current engineer has the *Change queue* permission for the respective range of tickets (mine, referenced etc., see below), the pull-down menu where the queue can be changed is displayed. All queues where the current engineer has the *Change queue* permission are listed. This means, the engineer needs the *Change queue* permission in the source as well as in the target queue. However, the engineer can only perform the entire operation if he has the permission for the right range of tickets in the target queue. For example, if an engineer wants to move a ticket which is assigned to himself to another queue, the minimum permission which is required for this operation is the *Change queue* permission for the range *mine* in the target queue. If another range was set (e.g. *referenced* or *none*), the operation would not succeed, because the ticket engineer is not changed during the operation and thus the suitable range has to be used.

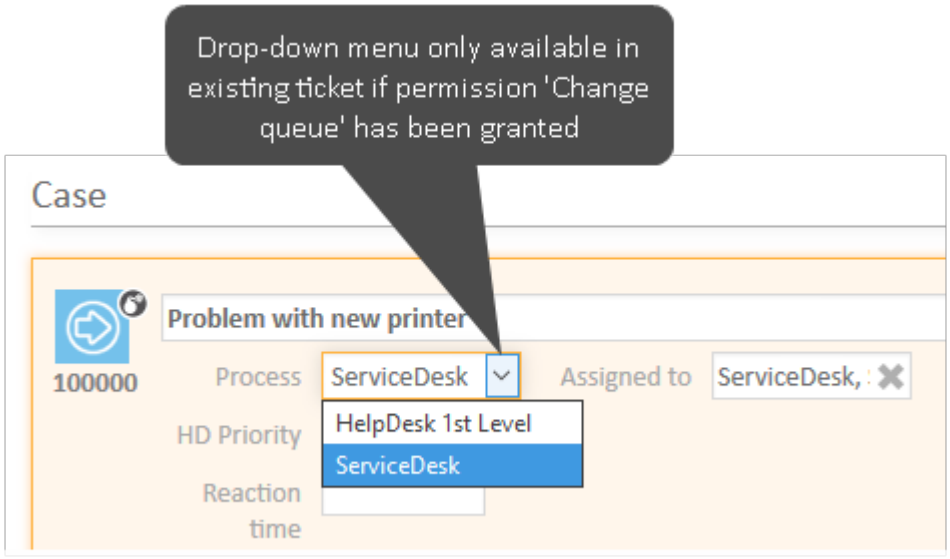


Figure 31: *ConSol CM Web Client - Drop-down menu to change queue*

It depends on the workflow of the target queue where the processing of the ticket will continue:

- If the source queue and the target queue have the same workflow, the ticket will start its processing in the target queue at the original position (i.e., its last position in the source queue).
- If the source queue and the target queue have different workflows, the ticket will start the process in the target queue at the START node.

> ⚠ Be very careful when granting the *Change queue* permission!!! Usually it is not required. On the contrary, it can destroy your process chain definition where tickets are passed from one process to another using process/workflow components, namely the Jump-in and Jump-out nodes.
>
> This permission should only be granted if it is absolutely necessary and when all side-effects have been considered thoroughly!

You can define for which range of tickets the permissions are valid:

- **Mine**
  Own tickets.
- **Ref**
  Tickets to which the engineer is assigned as an additional engineer (with engineer function, see Tab Engineer Functions).
- **None**
  Tickets without assigned engineer.
- **Other**
  Tickets assigned to other engineers.

Click the corresponding checkbox to assign one or more permissions for the desired ticket range.

Two general permissions can also be set:

- **Create**
  An engineer is allowed to create tickets in this queue.
- **Get assigned**
  Other engineers can assign tickets to an engineer who has a role with this permission (if the other engineers have the *Assign* permission!)
  An engineer can receive tickets by ticket transfer which is performed using the Admin Tool.

If you want to select all permissions simultaneously just click the *Select all* button below the list. Clicking *Deselect all* removes all selections.

Tab *Global Permissions*

Global permissions are general and queue-independent rights for a role. Setting these permissions is optional.

You reach this screen by clicking the *Global Permissions* tab in the navigation item *Roles*, navigation group *Access and Roles*.

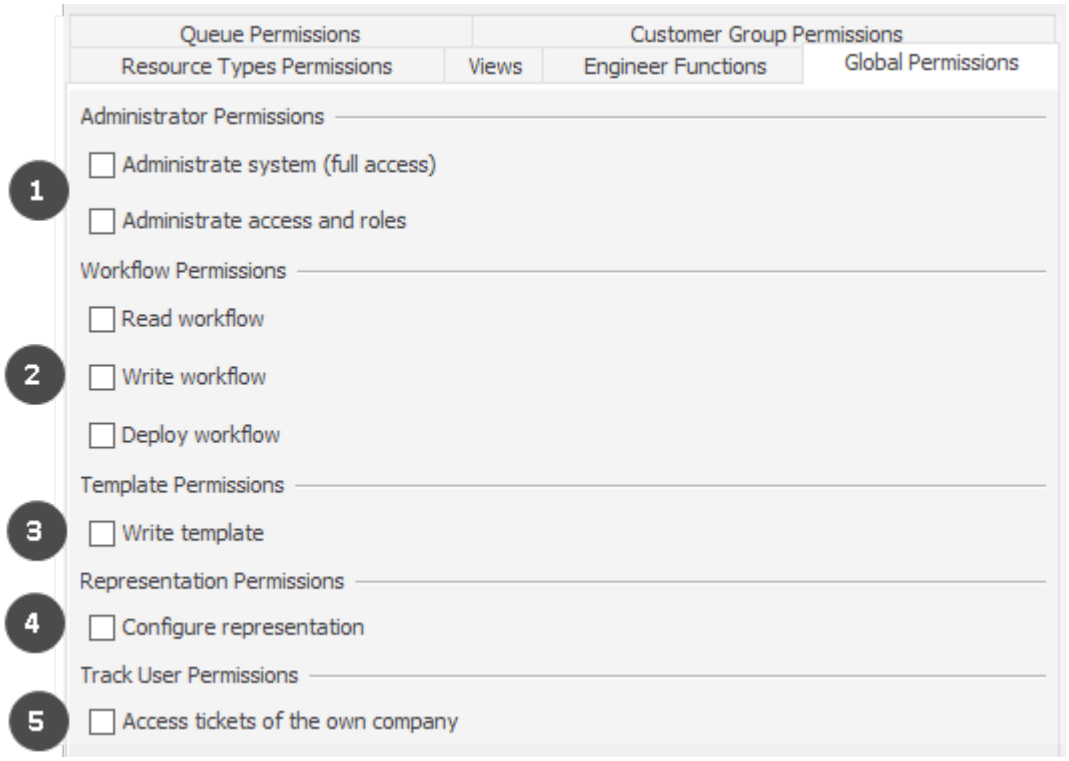


Figure 32: *ConSol CM Admin Tool - Access and Roles, Roles: Global permissions*

You can specify the following:

- **Administrator Permissions (1)**
  - **Administrate system (full access)**
    Provides administrator access to the entire CM system, this applies to the Admin Tool, the Process Designer, and admin access to the Web Client. An administrator with this role has access to all navigation groups and items in the Admin Tool.
  - **Administrate access and roles**
    Provides administrator access only to the navigation group *Access and Roles*. For details, please see section Defining an Administrator for Role and Engineer Administration Only.
- **Workflow Permissions (2)**
  Provides permissions concerning workflow design and management. These are
  - **Read**
  - **Write** (modify and store)
  - **Deploy** (install and put in operation).

- **Template Permissions (3)**
  - **Write Template** provides the permission
    - to use the Text Template Manager, which is used to create and edit email and comment templates. See section The ConSol CM Text Template Manager for details.
    - to use the Document Template Manager, which is required to define templates for CM/Doc. Only available if CM/Doc is active in the CM system.
- **Representation Permissions (4)**
  - **Configure representation**
    If this permission is set, engineers with this role can configure themselves as a representation for other engineers, e.g., who are ill and have not defined other engineers to represent them resp. if the defined engineers are not available at the moment. On the Web Client the engineers that can be represented by an engineer with this permission are shown in a list within the engineer profile.

Important information about representation configurations

Please note that there are two different scenarios for sending emails and that the CM system behavior concerning sending representation mails might differ for the two scenarios!

- **An engineer writes an email using the Ticket Email Editor**

  It depends on the value of the property cmweb-server-adapter, forward.mails.to.representatives if the representation rule is applied and the representing engineer receives a copy of the email. By default, this property is set to "false", meaning that this email is **not** sent to the representing engineer. If the property is set to "true", all emails which are sent manually using CM are sent to the original recipient **and** his current representative. The CM system checks if a representation rule is active for the respective (recipient) email address. Please keep this in mind when you configure the representation permissions in the Admin Tool and inform your CM users (engineers) about this behavior! It might lead to unwanted effects, especially when persons are registered as engineers and as contacts in the ConSol CM system (e.g., for an internal help desk).

- **An email is sent automatically from the CM system**

  It depends on the specific configuration of the CM system which engineers receive a copy of the email, the email is (!) not sent to the representing engineers automatically!
  It might be implemented that the representing engineer gets a copy, but this is not mandatory. The automatic email might be sent from a workflow script or from an Admin Tool script (which might also be called from a workflow). It depends on the implementation in this script who receives a copy of the email. For details, please refer to the *ConSol CM Process Designer Manual*.

- **Track User Permissions (5)**
  - **Access tickets of the own company**
    Users with this permission are allowed to access not only their own tickets in CM/Track, but all tickets of the company they belong to. This permission makes only sense for roles that define access rights of CM/Track users/user profiles, not for single users.

Tab *Customer Group Permissions*

In order to let engineers work with customer data from one or more customer groups, e.g. to edit reseller data sets or to create new contact data within the customer group, you have to grant access permissions concerning the customer group(s) to one or more roles.

You reach this screen by clicking the *Customer Group Permissions* tab in the navigation item *Roles*, navigation group *Access and Roles*.

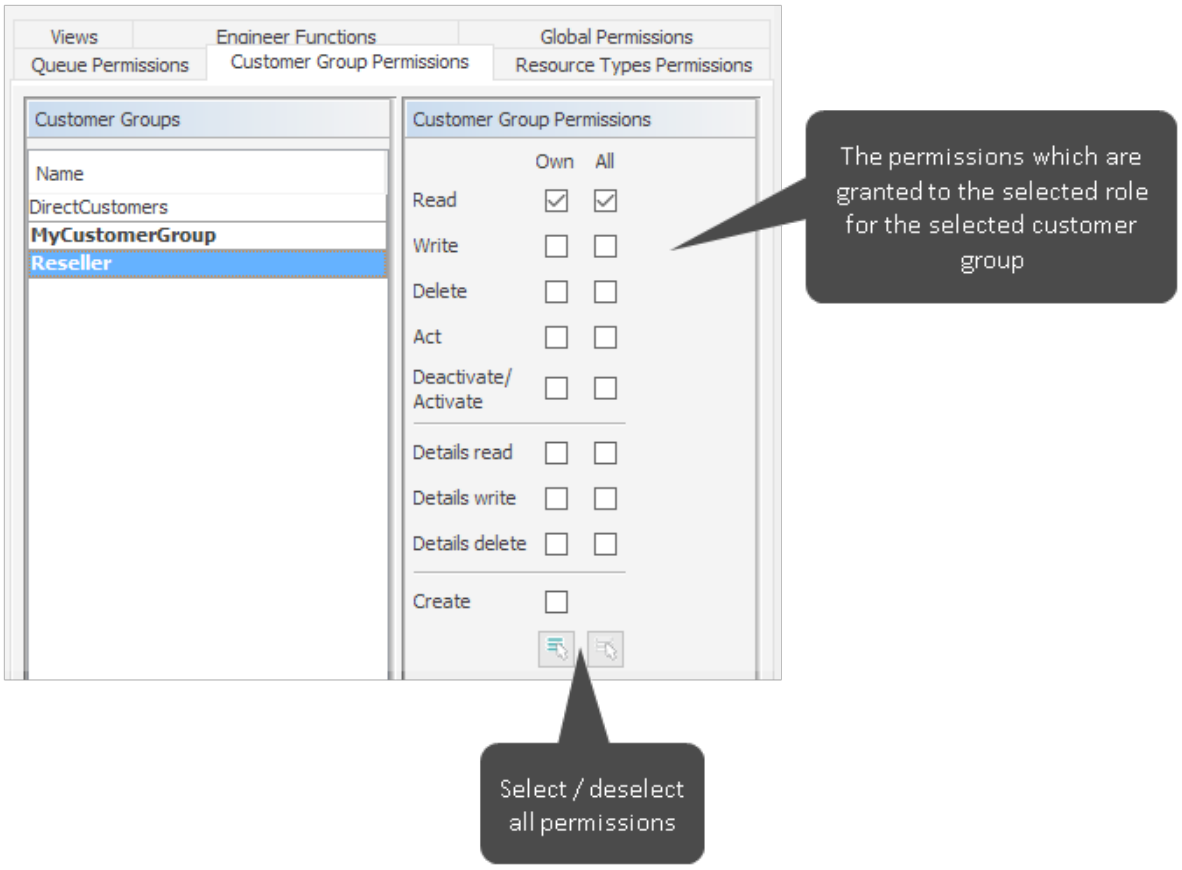


Figure 33: *ConSol CM Admin Tool - Access and Roles, Roles: Assigning permissions for customer groups to a role*

A concept which has proven very useful in various customer environments is the set-up of specific roles for customer data management. For example, there could be a role *CustomerManager_CustomerGroup1* and another role *CustomerManager_CustomerGroup2*. You can even differentiate between CustomerManager_CustomerGroup1_full and CustomerManager_CustomerGroup1 _light. In this way, you can use the assignment of the customer manager roles as a toggle and you do not mix up queue access permissions and customer management permissions. This can be very helpful in case you have a heterogeneous team in which not everyone is allowed to edit the complete customer data.

However, do not forget to grant read permissions to customer data of the required customer groups to all engineers of the respective queues! Otherwise, they cannot open their tickets at all!

The access rights which can be granted have been extended in CM version 6.9.1, compared to previous ConSol CM versions. New rights have been added which apply to a new section of the customer page. The contact page, as well as the company page in the Web Client, have a new *Comments and Attachments* section (2).

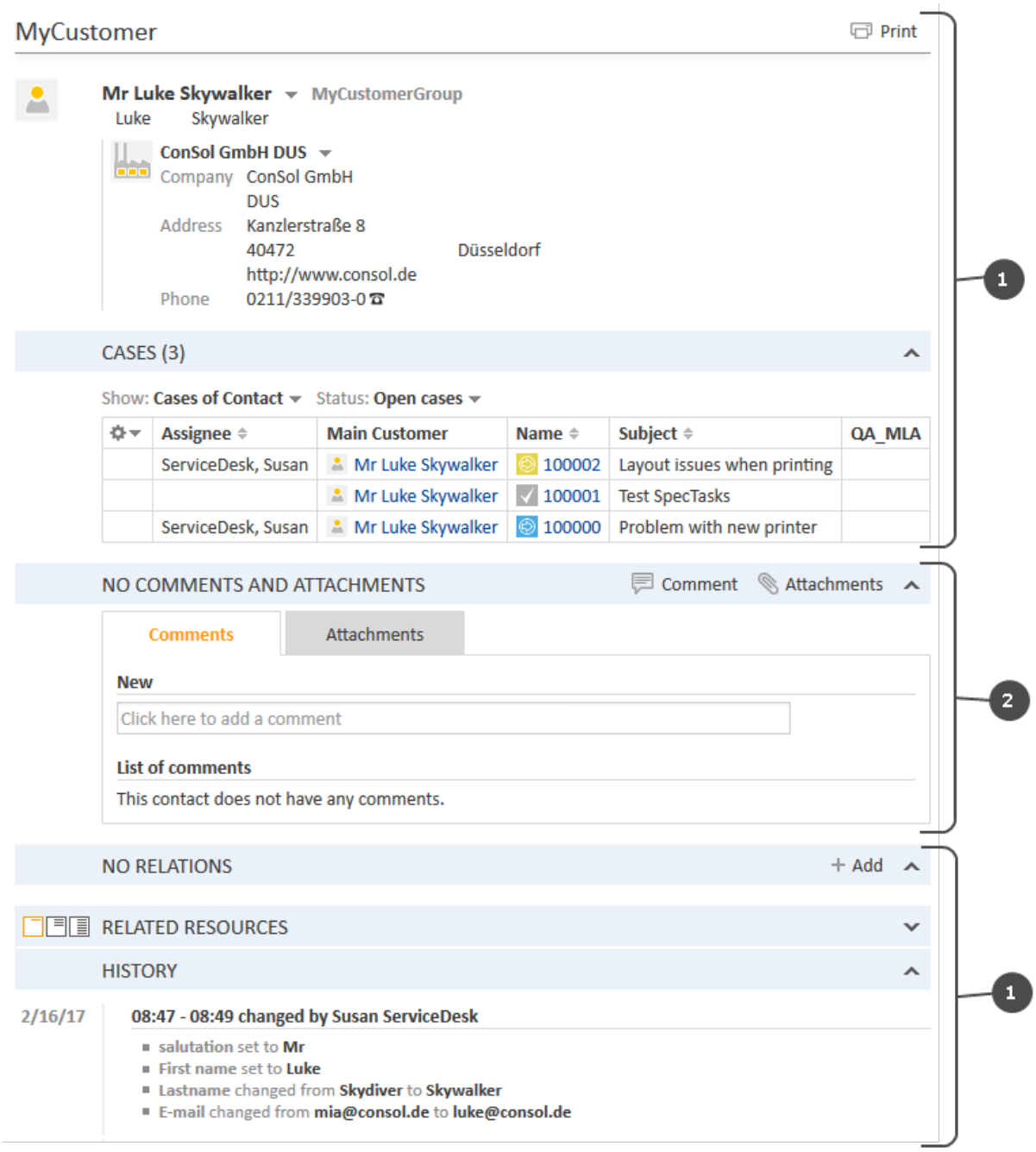


Figure 34: *ConSol CM Web Client - Sections of a contact page*

The following access permissions can be granted:

- **Customer type**
  Refers to the tickets of the customer.

- **Own**
  All (main or additional) customers of tickets which are currently assigned to the engineer or where the engineer is set as additional engineer.
- **All**
  All customers.

- **General sections (1)**
  - **Read**
    Read the customer data.
  - **Write**
    Write/modify the customer data, and change the company of a contact on the contact page using the *Change* link.
  - **Delete**
    - Delete a customer data set. This refers to companies as well as to contacts. For contacts, two types of deletion are available (with or without related data).
    - Transfer all tickets associated with a customer of this customer group to another customer.
    - Anonymize a contact
  - **Act**
    Execute actions for this customer (see section Action Framework - Customer Actions for details about customer actions).
  - **Deactivate/activate**
    - Deactivate and (re-)activate the contact or company. It is not possible to create tickets for a deactivated customer.
    - Transfer all tickets associated with a customer of this customer group to another customer.

Information concerning transfer permissions for tickets and resources

Please note that:

- in CM versions previous to 6.9.4.1, the permission *Transfer tickets* was not restricted.
- in CM versions 6.9.4.1 up to 6.10.4.3, the permission *Transfer tickets* was linked to the permission *Delete* (customer data).
- starting with CM version 6.10.4.4, the permission *Transfer tickets* is linked to the permission *Delete* (customer data) as well as to the permission *Deactivate/activate* (customer data), i.e. an engineer can have either one of these permissions to be able to transfer data.

- **Comments and Attachments section (2)**
  - **Details read**
    Read customer data in the *Comments and Attachments* section.
  - **Details write**
    Write/modify customer data in the *Comments and Attachments* section.
  - **Details delete**
    Delete customer data in the *Comments and Attachments* section.
- **General**
  - **Create**
    Create a customer data set. In a two-level customer data model this refers to contact as well as to company data sets.

Please keep in mind that an engineer must have at least read permissions for a customer group to open and/or create tickets for customers in this group!

Tab *Resource Types Permissions*

Resource types permissions control an engineer's access to resources, i.e., objects which are stored in the Resource Pool.

You reach this screen by clicking the *Resource Type Permissions* tab in the navigation item *Roles*, navigation group *Access and Roles*.

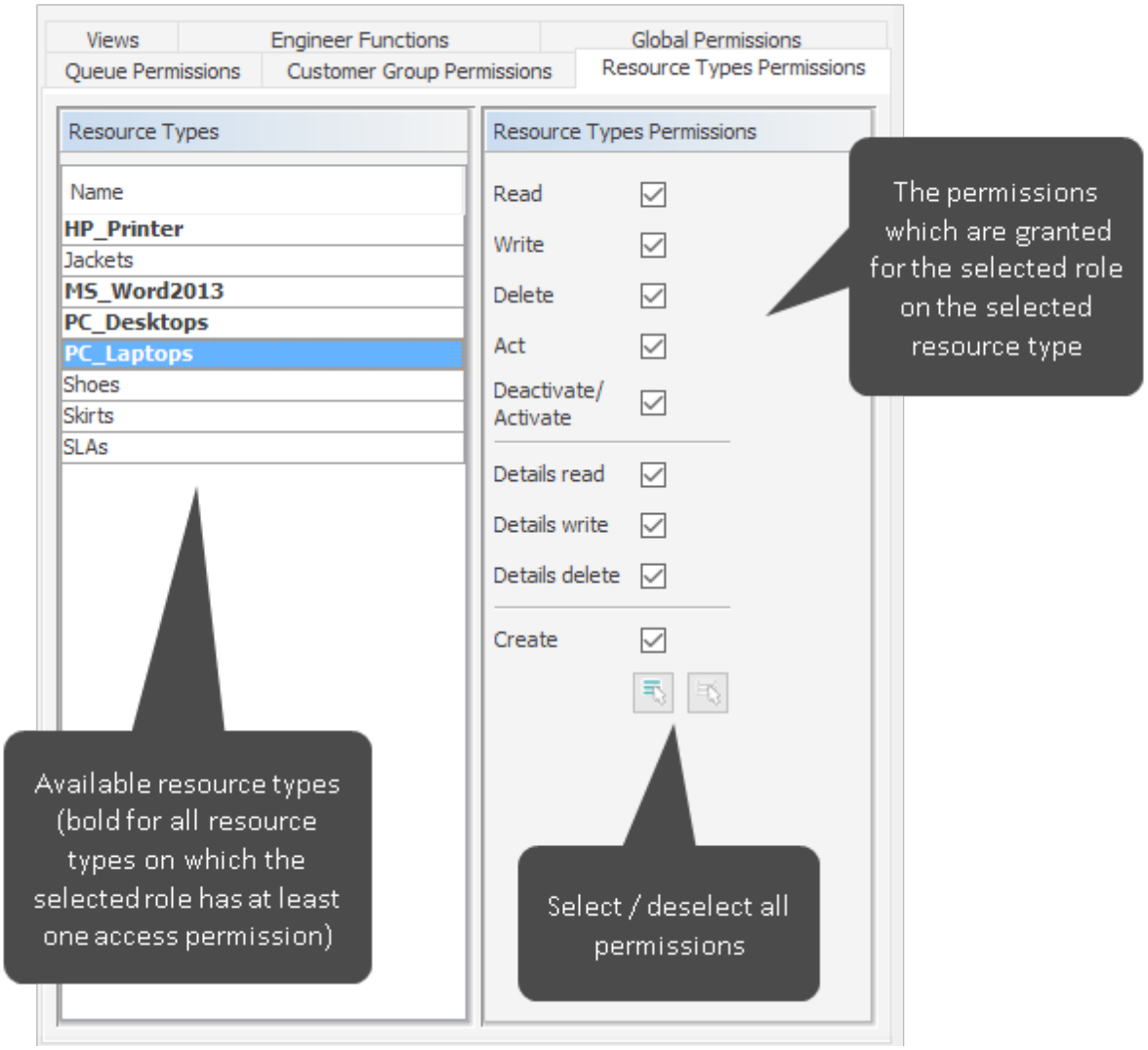


Figure 35: *ConSol CM Admin Tool - Access and Roles, Roles: Resource types permissions*

The following permissions can be granted:

- **Read**
  Load and display resources of the selected type in the Web Client.
- **Write**
  Change data fields of this type of resources.
- **Delete**
  Delete resources of the respective type from CM.
- **Act**
  Execute resource actions defined for this type of resources.
- **Deactivate/Activate**
  (De-) Activate resources of the selected type.

- **Details read**
  Load and display comments/attachments for resources of this type.
- **Details write**
  Add and change comments/attachments for resources of this type.
- **Details delete**
  Remove comments and attachments for resources of this type.
- **Create**
  Create new resource entries for the type of resources.

Tab *Views*

Views define which tickets engineers will see in the ticket list of the Web Client. This tab shows the assigned views on the left and the available views on the right (see also View Administration). The displayed views can be filtered by name and queue. Assigning views is optional.

> We recommend to assign at least one view to a role. Otherwise an engineer with this role will see no tickets in the Web Client's ticket list.

You reach this screen by clicking the *Views* tab in the navigation item *Roles*, navigation group *Access and Roles*.

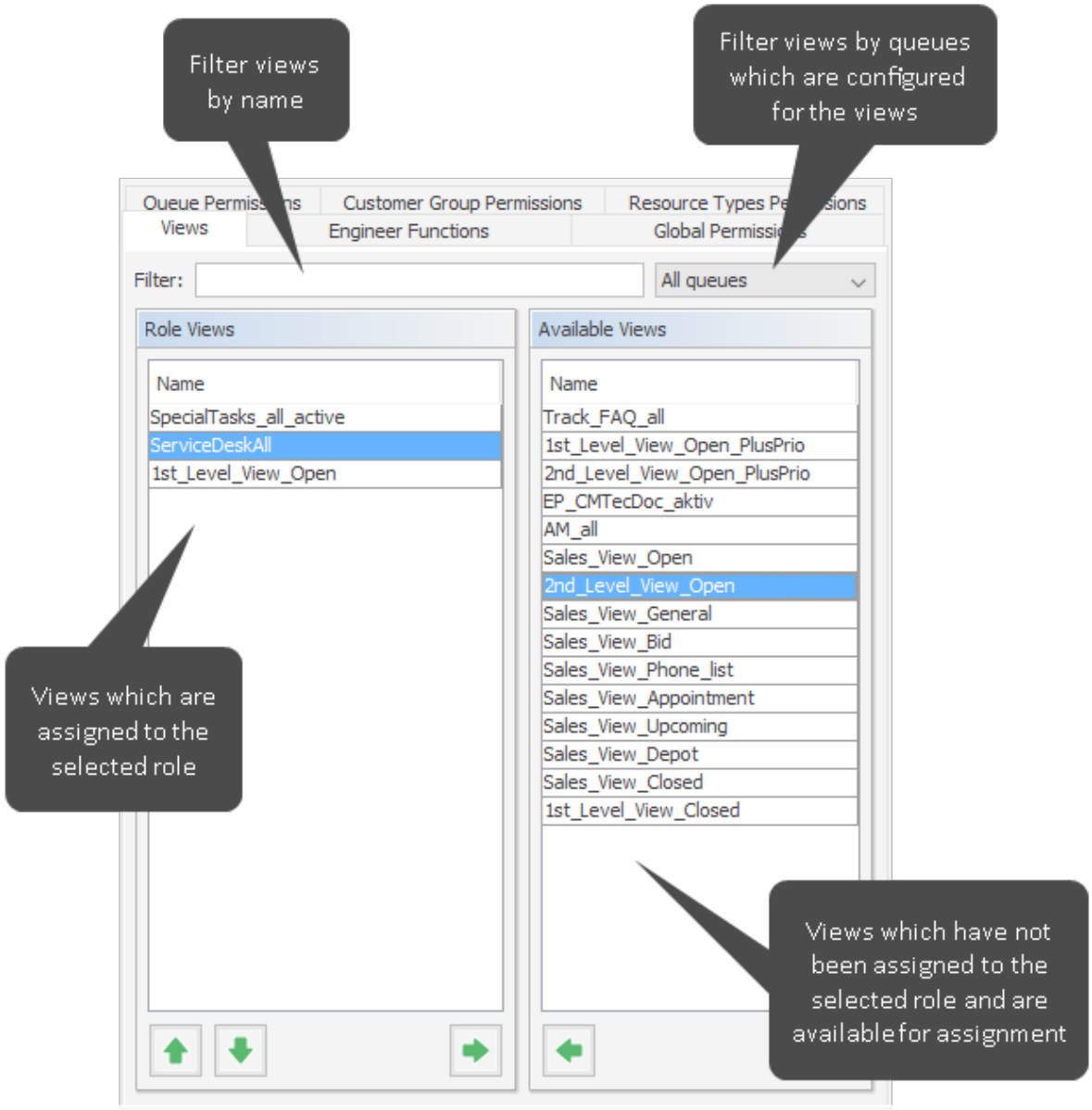


Figure 36: *ConSol CM Admin Tool - Access and Roles, Roles: Views*

Select a role on the left side of the page first and then the desired view(s) in the list of *available views*. Click the *Assign* button to move the selected view(s) to the list of *role views*. If you want to remove views from this list, select the respective views and click the *Unassign* button.

For regular roles, you cannot define the order of the views here. In the drop-down menu of the Web Client, the views will always be displayed in the order they have in the list of the view administration. Please see also section View Administration. When a role has been marked as *main role* for at least one engineer (and is thus marked with a red dot), the views can be sorted using the *Move upwards* and *Move downwards* buttons. The sorting affects the order in which the views are displayed in the Web Client.

Tab *Engineer Functions*

On this tab you can assign *engineer functions* to a role. Engineer functions are used if you need an additional engineer for a ticket, e.g., a supervisor who has to decide what to do before the ticket can be moved on in the workflow. Thus you have to assign a role with the respective engineer function to this supervisor. In the Web Client engineer functions and associated engineers are shown when assigning an additional engineer.

You reach this screen by clicking the *Engineer Functions* tab in the navigation item *Roles*, navigation group *Access and Roles*.

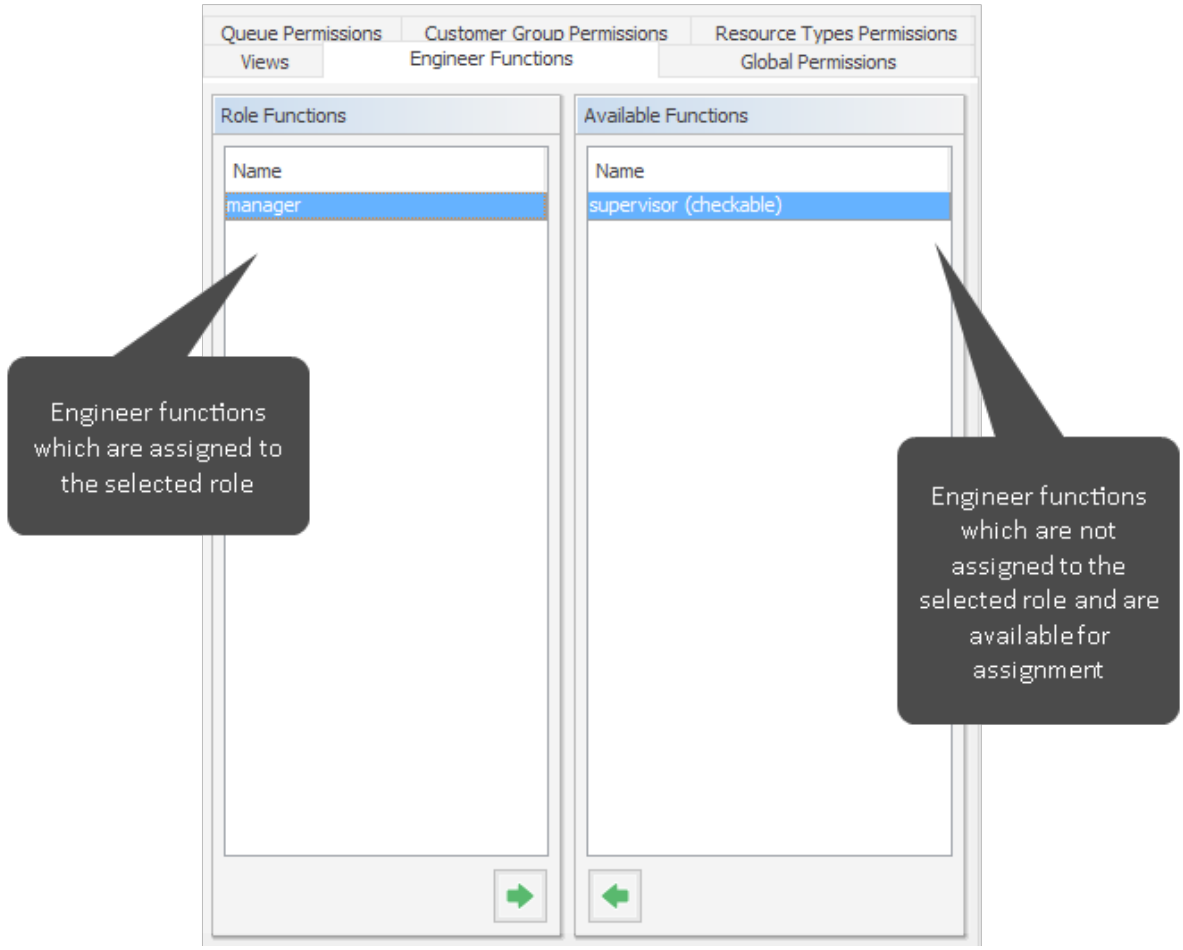


Figure 37: *ConSol CM Admin Tool - Access and Roles, Roles: Engineer functions*

Select a role on the left side of the page and then the desired engineer function(s) in the list of *available functions*. Click the *Assign* button to move the selected function(s) to the list of *role functions*. If you want to remove functions from this list, select the respective function(s) and click the *Unassign* button.

After you have defined the new role by setting permissions, views, and engineer functions in the tabs you can assign the role to the desired engineer accounts. Engineers obtain the rights of a role immediately after assignment (without an additional update of the system).

B.2.2.2 Delete a Role

Select the role you want to delete and click the *Delete* button below the role list. If you choose *Yes* in the following confirmation dialog, the role is removed from the list and the system.

> ⚠ If you delete a role, please consider that engineers with only this role will immediately lose all permissions in the system.
>
> In case tickets, e.g., from a certain queue, are not covered by any role permission, engineers and/or administrators could get the impression that tickets are missing.

B.2.2.3 Copy a Role

If you want to create a new role and use an existing role as a template you can copy it. Select the existing role and click the *Copy* button below the role list. A pop-up window appears in which you can enter the name for the copy. Afterwards you can modify the copy according to your wishes.

B.2.2.4 Edit a Role

Select the role you want to edit in the list and modify the permissions in the respective tabs as desired. The changes are immediately effective for engineers with this role. The engineer just has to login again.

B.2.3 Defining an Administrator for Role and Engineer Administration Only

Sometimes it can be necessary to define an administrator who does not have full system access but who is only allowed to manage engineers and roles. This can be used, for example, for a team manager who is allowed to create new CM engineers for new employees in his team or for a key user in a team who should be able to grant or retrieve permissions of CM engineers in a certain department.

In order to define this Administrator "light", create a new role with the global permission *Administrate access and roles*. Create a new engineer (*admin_light* in our example) and assign this role.

You reach this screen by clicking the *Global Permissions* tab in the navigation item *Roles*, navigation group *Access and Roles*.

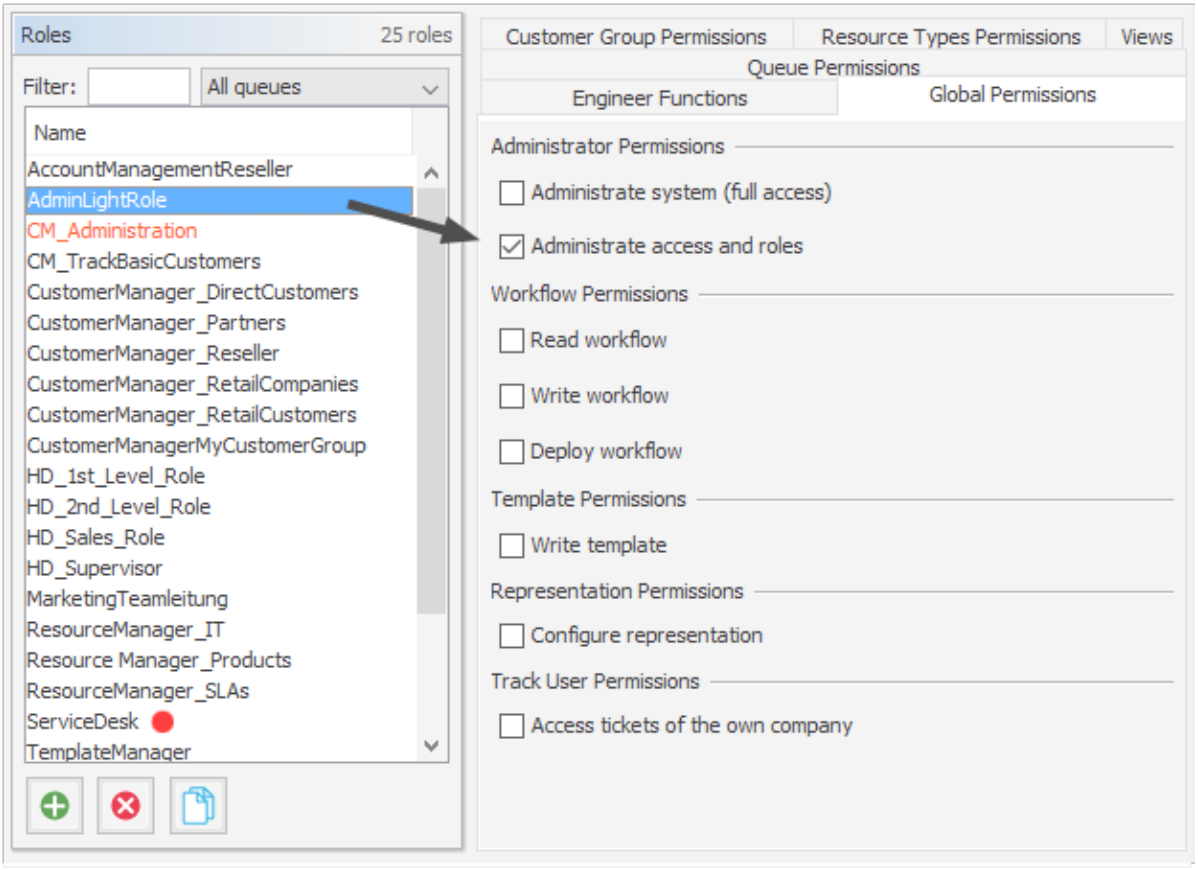


Figure 38: *ConSol CM Admin Tool - Access and Roles, Roles: Definition of an administrator role for Access and Roles*

For the "Administrator light", the navigation item *Access and Roles* is available. However, the tab *Global Permissions* of this item is not available (so the *admin_light* cannot extend his own permissions!).

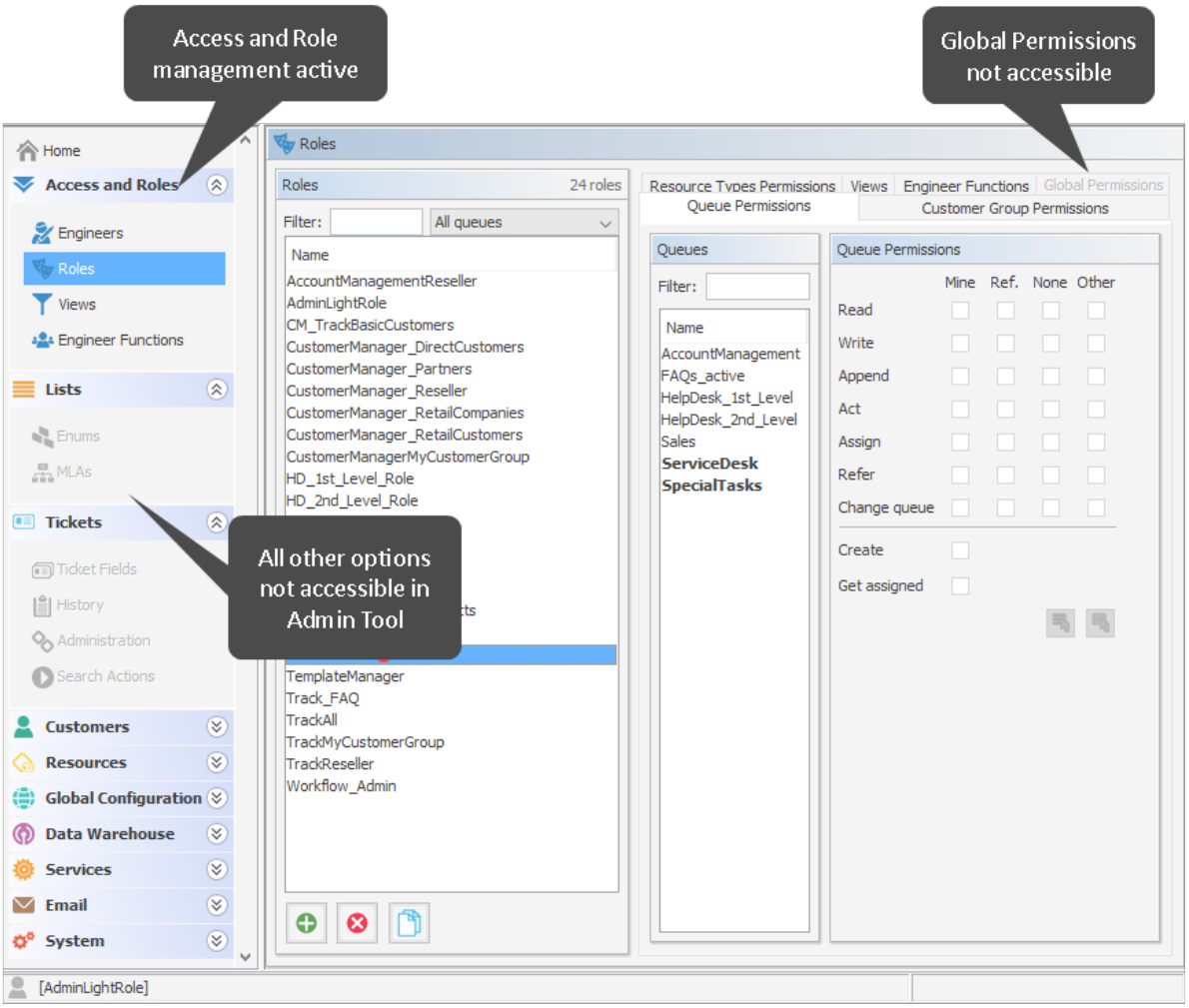


Figure 39: *ConSol CM Admin Tool - View for an administrator "light"*

B.3 View Administration

This chapter discusses the following:

B.3.1 Introduction to View Administration

Views are used to filter tickets according to certain criteria (e.g. all active tickets in the Queue *Helpdesk)* and display the resulting tickets in the ticket list of the Web Client. Since views are associated with roles engineers obtain their view(s) via the roles which are assigned to them. Engineers can switch between their views in the Web Client.

Engineers need the appropriate permissions to see all tickets filtered by a view. Permissions are not automatically granted when a view is assigned, but they have to be assigned within the definition of roles (as queue and customer group permissions). One and the same view can result in varying subsets of tickets and information therein for engineers with different roles.

The creation of views is optional. However we recommend it in order to assure central features of the Web Client. Without a view engineers will not see any tickets in the ticket list. They can only access tickets by using the search function.

B.3.2 View Administration Using the Admin Tool

To create, edit or delete views, open the navigation item *Views* in the navigation group *Access and Roles*, in the Admin Tool.

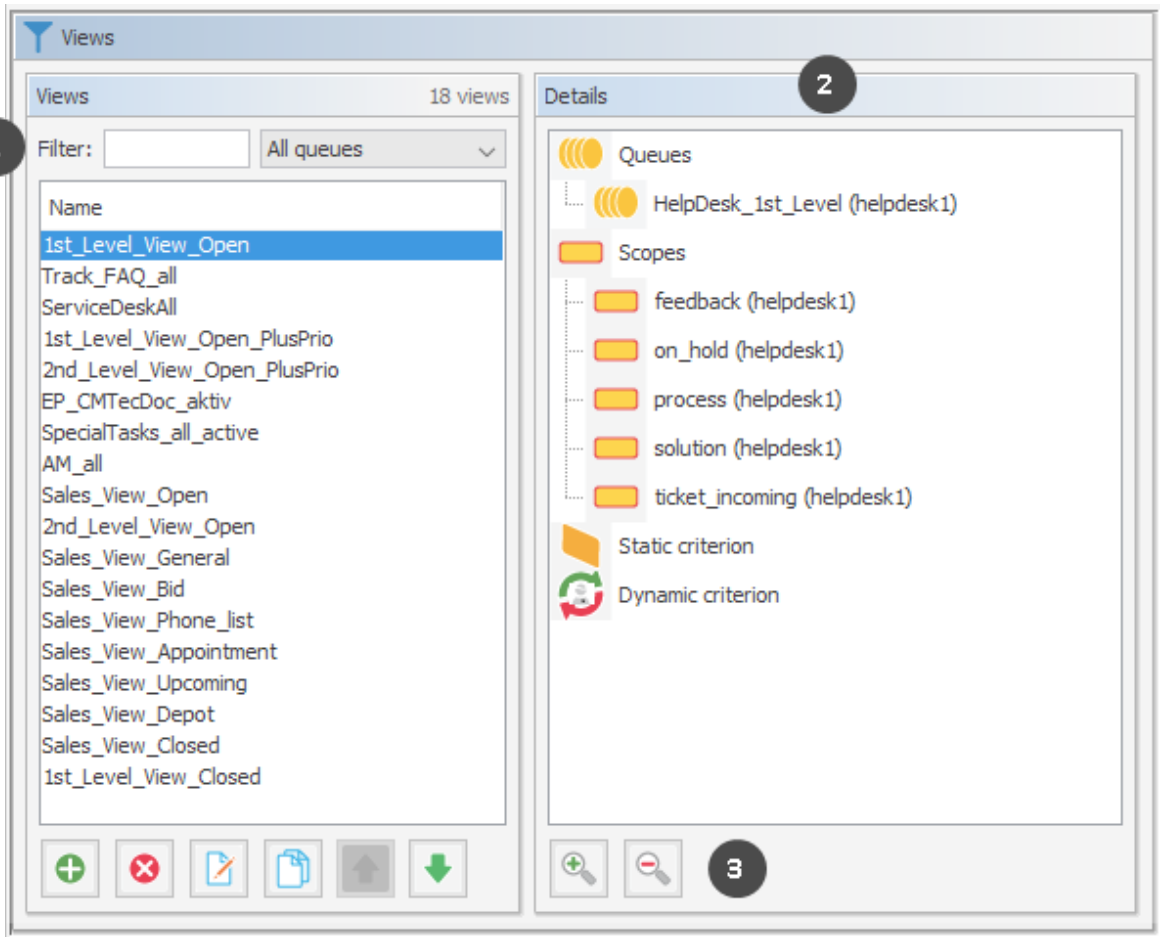


Figure 40: *ConSol CM Admin Tool - Access and Roles, Views: View administration*

The screen *Views* shows a list of all available views on the left. You can filter the list by entering the name of a view or selecting a queue (1). The details of the selected view, i.e., the queues and scopes which are configured n the view, are displayed on the right (2). You can expand or collapse the sub-trees using the buttons below the components (3).

B.3.2.1 Create a View

After clicking the *Add* button below the view list, the pop-up window *View Wizard* appears where you have to define the name for the new view first. You can also enter a description for it.

By clicking the *Localize* button you can localize view name and description. See section Localization of Objects in General, Type 1 for details.

Via *Next >* you can continue with the definition of view criteria:

- queue filter
- scope filter
- static criterion
- dynamic criterion

Queue Filter

At first you choose the queues for the new view. Select the desired queues in the list *Unassigned* and move them to the list *Assigned* by clicking the *Assign* button. To remove an assigned queue, select it and click the *Unassign* button. Continue with the *Next >* button, in order to define scope filters, too.

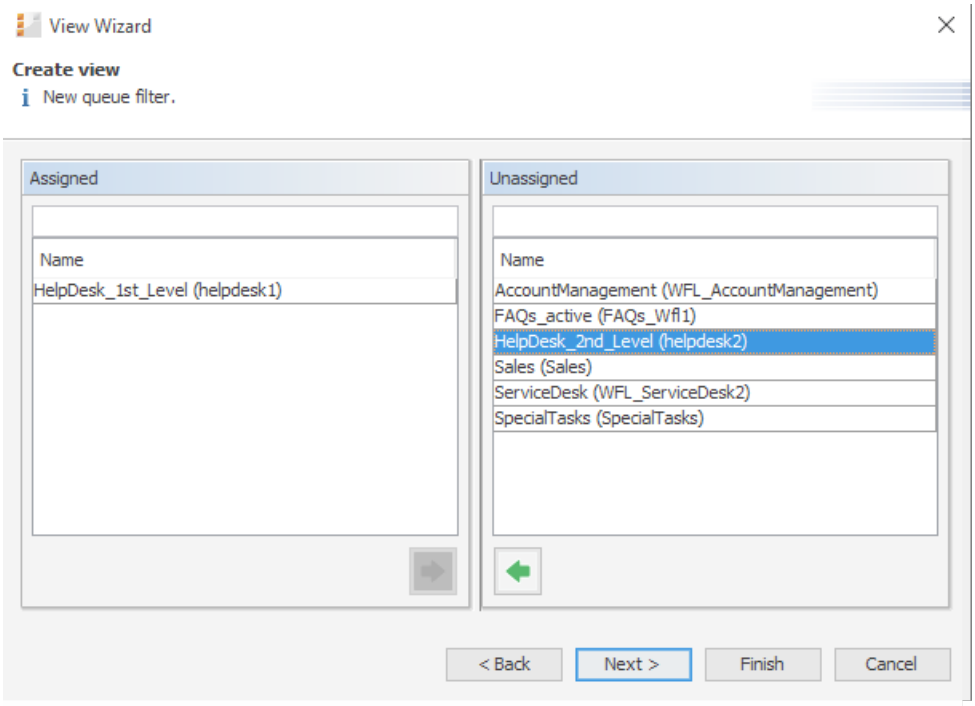


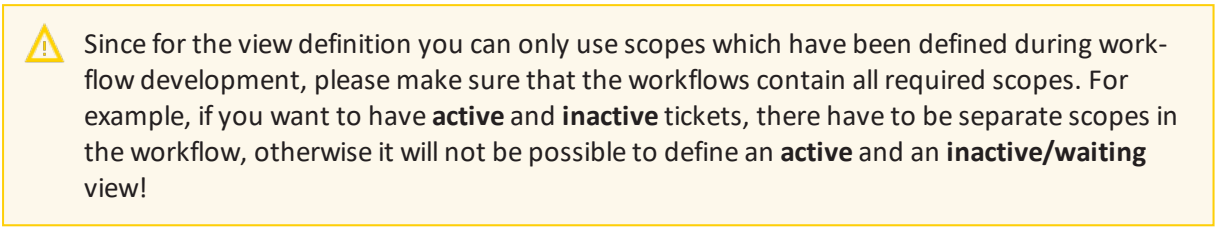
Figure 41: *ConSol CM Admin Tool - Access and Roles, Views: Selecting queues*

Scope Filter

Next you can limit the view to certain workflow scopes of the selected queue(s). Scopes group workflow activities that have a special topic in common, e.g., tickets with an appointment.

Select the desired scopes in the list *Unassigned* and move them to the list *Assigned* by clicking the *Assign* button. To remove assigned scopes, select them and click the *Unassign* button. Continue with the *Next >* button if you want to define further criteria, otherwise click *Finish* to create the view.

> If you do not assign scopes in the View Wizard, the view exists by name but will not show tickets in the Web Client.



Since for the view definition you can only use scopes which have been defined during workflow development, please make sure that the workflows contain all required scopes. For example, if you want to have **active** and **inactive** tickets, there have to be separate scopes in the workflow, otherwise it will not be possible to define an **active** and an **inactive/waiting** view!

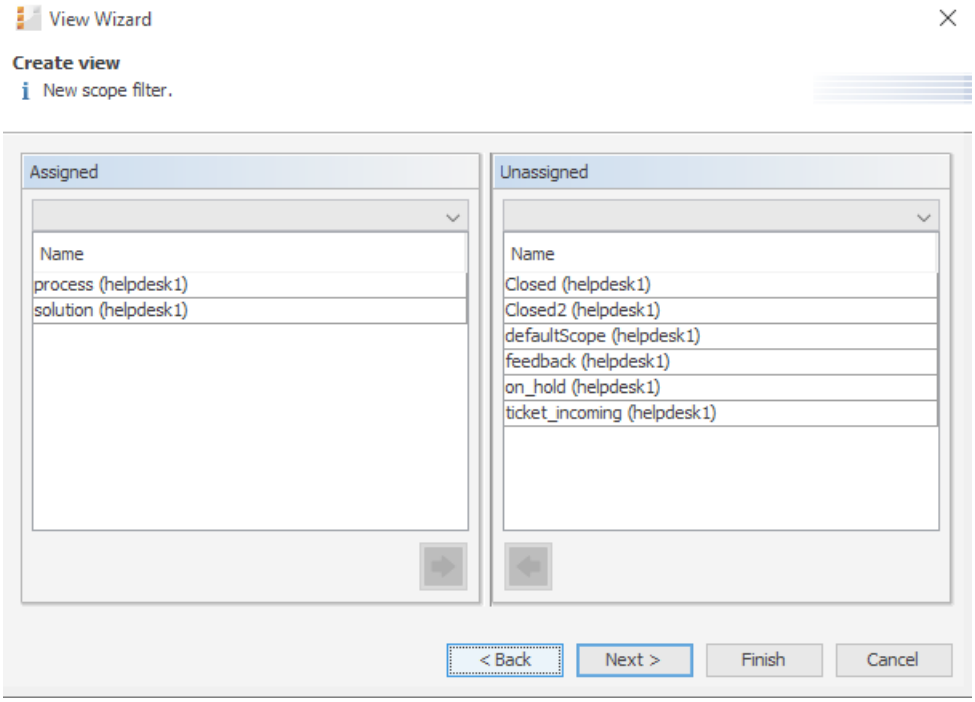


Figure 42: *ConSol CM Admin Tool - Access and Roles, Views: Selecting scopes*

Static Criterion

You can restrict the view further by a static criterion to show only tickets with a certain value in a defined data field, e.g., tickets concerning a special product or only tickets with high priority. The criterion is static because the engineer cannot change it in the Web Client. Please see the *ConSol CM User Manual* for a detailed description of working with views. Only data fields of type *enum* can be used as static criterion.

Choose the data field in the *Field* list (e.g. *product*) and select the desired value in the *Value* list below (e.g. *crm*). Continue with the *Next >* button if you want to define a dynamic criterion as well, otherwise click *Finish* to create the view.

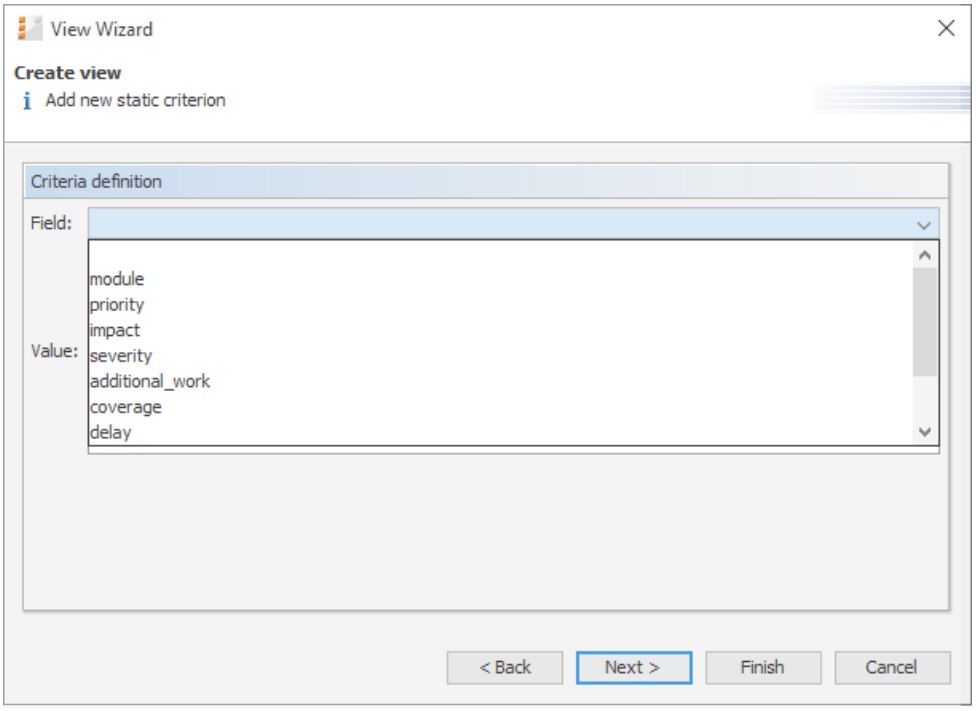


Figure 43: *ConSol CM Admin Tool - Access and Roles, Views: Selecting the static criterion*

Dynamic Criterion

Like a static criterion, a dynamic criterion is used to show only tickets with certain values in a defined data field, but in contrast to a static criterion, with a dynamic criterion engineers can choose the value (s) for the criterion themselves. This can be done in the Web Client by editing the *Engineer Profile*. Additionally, the administrator can adjust the value individually for each engineer on the *View criteria* tab of the engineer administration (see section Engineer Administration). Please see the *ConSol CM User Manual* for a detailed description of working with views. Only data fields of type ENUM can be used as dynamic criterion.

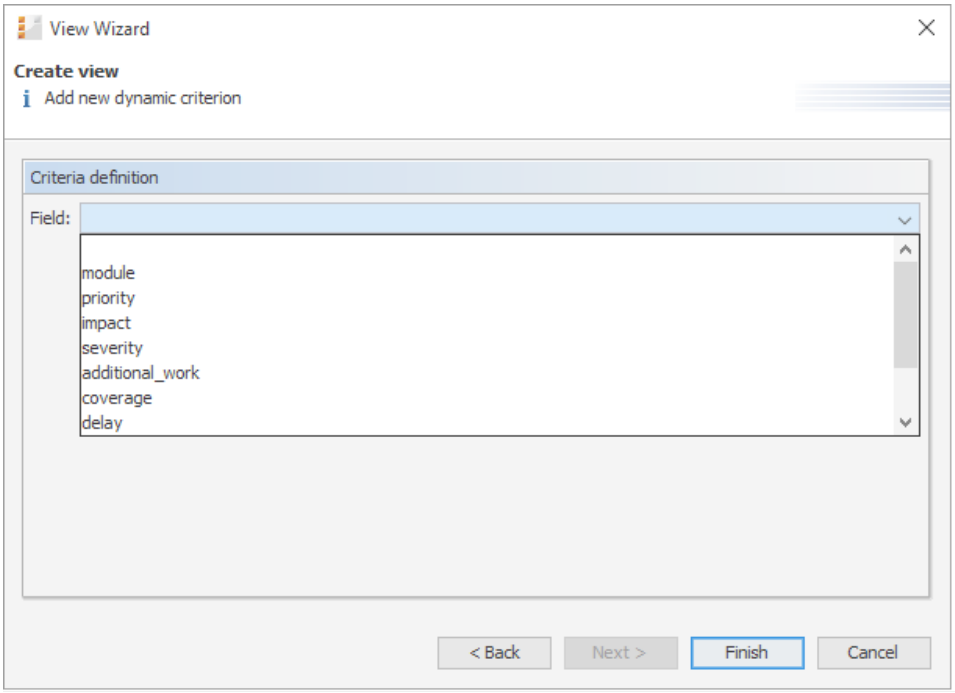


Figure 44: *ConSol CM Admin Tool - Access and Roles, Views: Selecting the dynamic criterion*

Click *Finish* to create the view. You can leave the window any time without storing by choosing *Cancel*. Via the *Back* button you can return to the previous step of the view definition.

Now you can see the new view in the view list on the left. The assigned criteria are shown in the *Details* area on the right side of the page.

⚠ Please note that in a view with a dynamic criterion, only the tickets are displayed which match this criterion. So if an engineer has not selected any criteria in his/her engineer profile, or if the administrator has removed all selections using the Admin Tool (*View criteria* in Engineer Administration), the engineer's view will be empty! Make sure your users know about this fact and make sure you, as an administrator, are always aware of that fact.

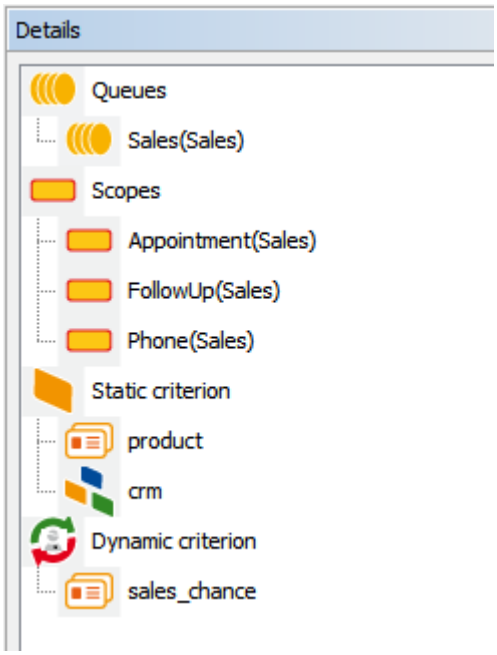


Figure 45: *ConSol CM Admin Tool - Access and Roles, Views: Viewing the details*

You can expand or collapse all details by clicking the *Plus* or *Minus* icon below the list.

⚠ We strongly recommend not to define views which contain closed tickets!

The number of closed tickets will grow considerably during work with the application. Therefore, the view of closed tickets would always reach the maximum number of tickets allowed for a view (which can be defined using a system property). This can have negative influence on the Web Client performance and in most cases the desired tickets will not even be among the first 50 or 100 tickets.

Conclusion: A view of closed tickets does not help and might decrease the speed of the system for the engineers. Only in test environments, a view for closed tickets might be an option.

B.3.2.2 Edit a View

Select the view you want to edit in the view list. The view details are shown on the right side of the page. To edit the selected view just click a filter criterion with the right mouse button. The following drop-down menu appears.

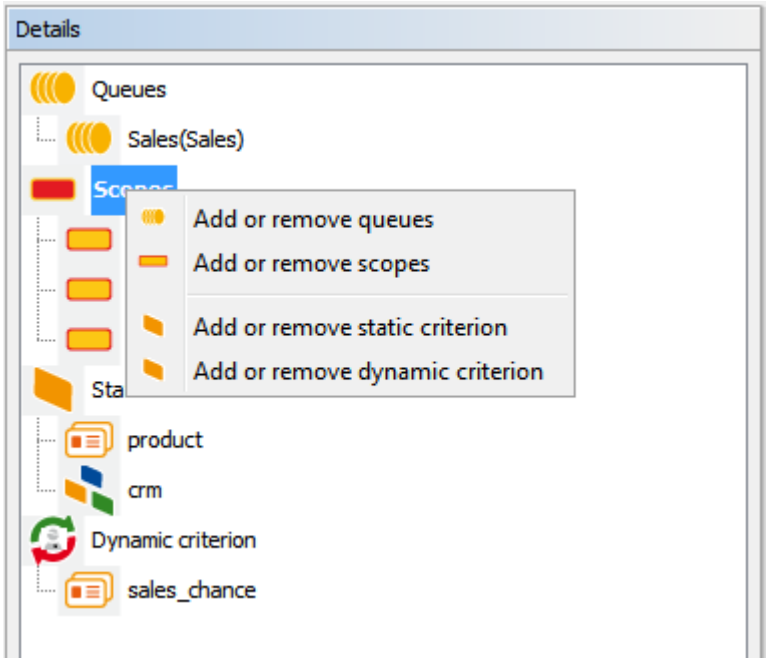


Figure 46: *ConSol CM Admin Tool - Access and Roles, Views: Editing a view*

The menu contains these options:

- Add or remove queues
- Add or remove scopes
- Add or remove static criterion
- Add or remove dynamic criterion

Just click the desired menu item. The respective window of the *View Wizard* appears where you can add or delete filter criteria as described in Create a View. Double-clicking on a filter criterion will also open the View Wizard.

> You cannot edit view criteria by clicking the *Edit* button. Here you can only modify the view's name and description.

B.3.2.3 Delete a View

Click the *Delete* button below the view list to delete the selected view. A pop-up window appears which asks whether you really want to delete the view. If you choose *Yes*, the view will not be available for any engineers. Engineer permissions are not affected by this operation.

B.3.2.4 Copy a View

The *Copy* button allows you to save time when creating a view. The selected view will be copied completely and you can edit the copy afterwards. The new view has the same name as the copied view. You can change it by double-clicking on the name or by clicking the *Edit* button.

B.4 Engineer Functions

This chapter discusses the following:

B.4.1 Introduction

Engineer functions are used if you need an additional engineer for a ticket, e.g., a supervisor who has to decide what to do before a ticket can be moved on in the workflow.

In the Admin Tool, engineer functions are managed using the navigation item *Engineer Functions* in the navigation group *Access and Roles*.

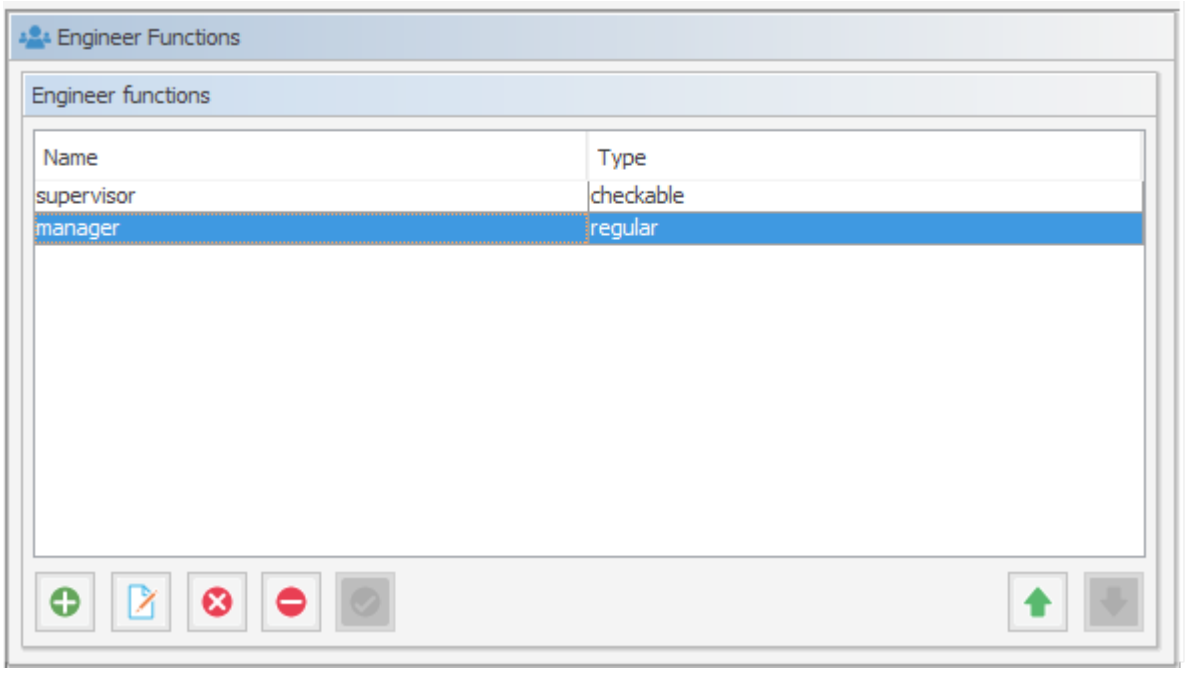


Figure 47: *ConSol CM Admin Tool - Access and Roles, Engineer Functions*

The corresponding activities for such a process have to be created in the workflow. Engineer functions are assigned to engineer roles which in turn need to be assigned to the respective engineers. In the Web Client you can choose a function and an appropriate engineer when adding an engineer to the ticket.

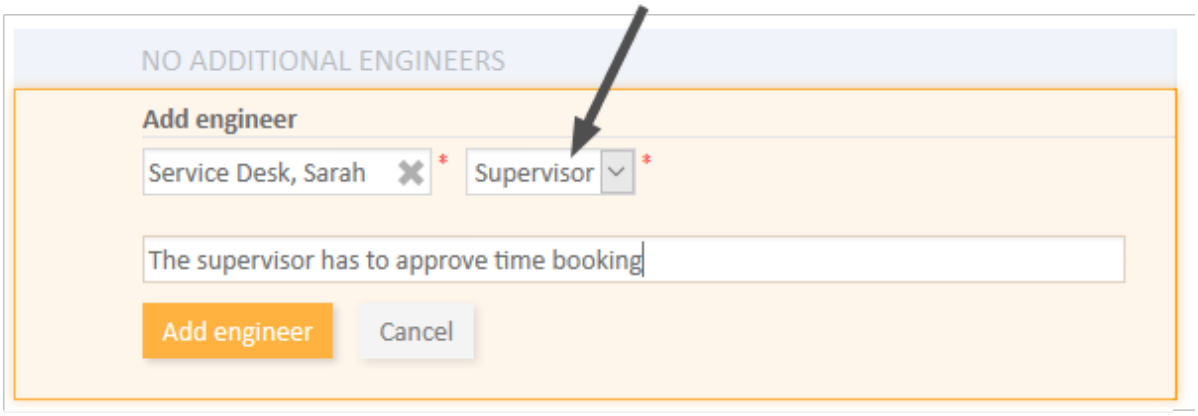


Figure 48: *ConSol CM Web Client - Assigning an additional engineer with an engineer function*

B.4.2 Create or Edit an Engineer Function

An engineer function is defined by a name. By clicking the *Add* button a pop-up window appears where you can enter the name. You will get the same window when you click the *Edit* button in order to edit an engineer function. The checkbox *Checkable* has to be ticked if additional engineers shall have the permission to execute a certain activity, e.g., give their approval before the ticket can be moved on. The approval state is then displayed in the ticket.

> ⚠ After creation of an engineer function the checkbox *Checkable* cannot be de-selected anymore.

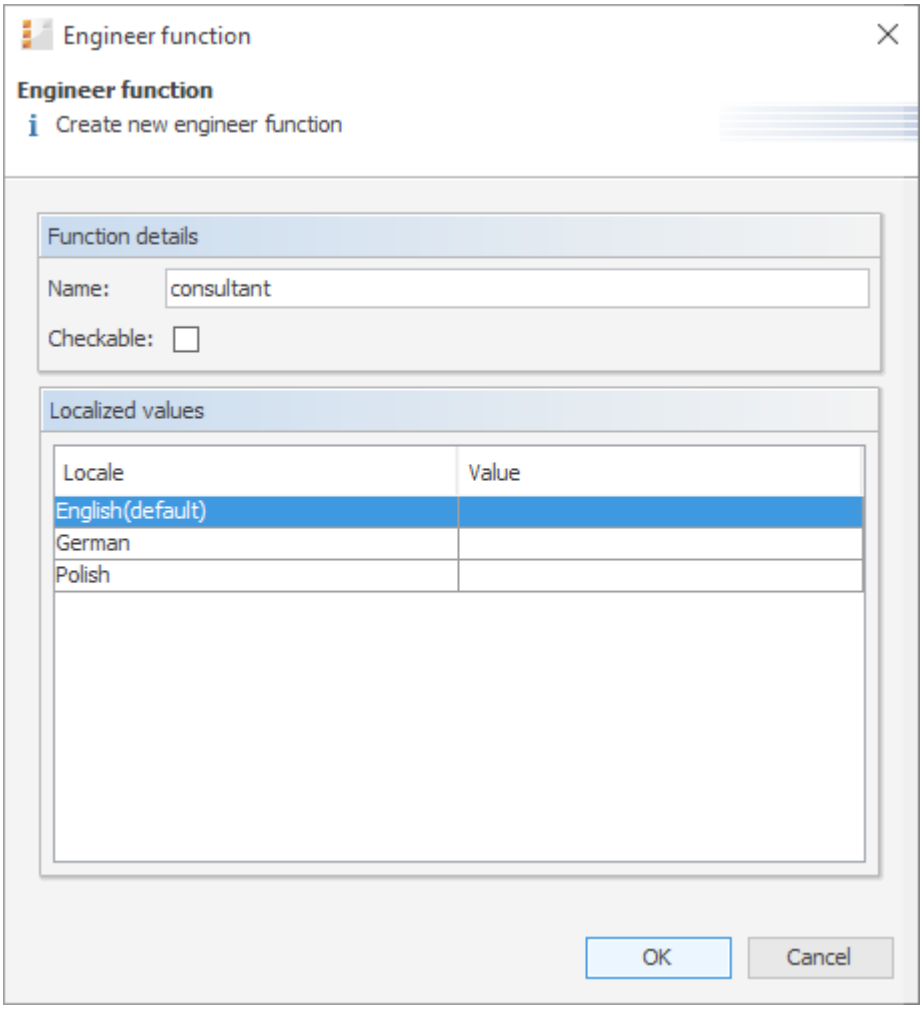


Figure 49: *ConSol CM Admin Tool - Access and Roles, Engineer Functions: Create or edit an engineer function*

You can also localize the name of an engineer function in this window. See section Localization of Objects in General, Type 2 for details.

After clicking *OK* the engineer function is created and the name will be displayed in the respective language of the engineer's locale.

B.4.3 Delete an Engineer Function

An engineer function can only be deleted if it is not assigned to any roles. Otherwise you get a warning and you can only disable this engineer function (see below).

In order to delete an engineer function, select it in the list and click the *Delete* button. After choosing *Yes* in the confirmation dialog, the engineer function will be removed from the list and the system.

B.4.4 Disable or Enable an Engineer Function

If an engineer function is still assigned to a role but is not needed anymore you can disable it. To do this select the engineer function and click the *Deactivate* button. The entry in the list is shown in italics afterwards. The engineer function cannot be assigned anymore. Just click the *Activate* at the bottom of the page if you want to enable the function again.

B.4.5 Engineer Permissions Concerning Engineer Functions

For engineers, resp. for roles, specific permissions can be granted which concern only the tickets for which the engineer who has this role is assigned as engineer with a certain function. Those permissions are granted as queue permissions (see also section Role Administration). This principle provides a good basis for a very sophisticated management of processes like approvals or other processes where people with different roles and responsibilities have to work on a case at the same time.

Using the Role Administration (navigation group *Access and Roles*, navigation item *Roles*), you can grant/withdraw the following permissions (for a detailed explanation of the permissions see section Role Administration):

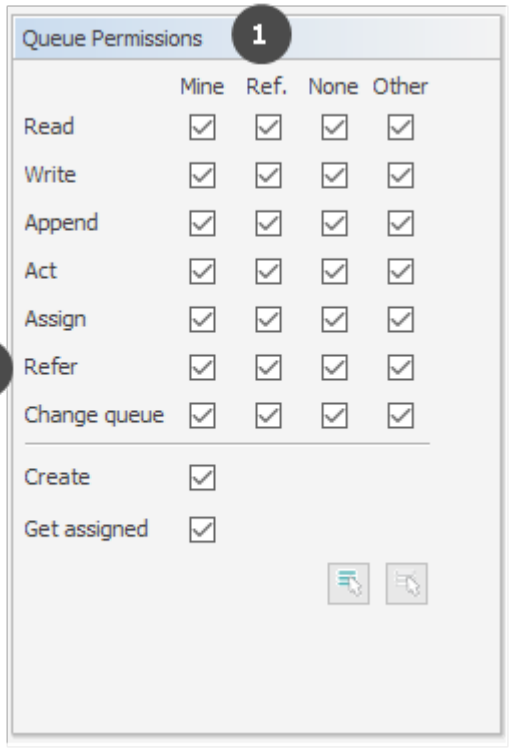
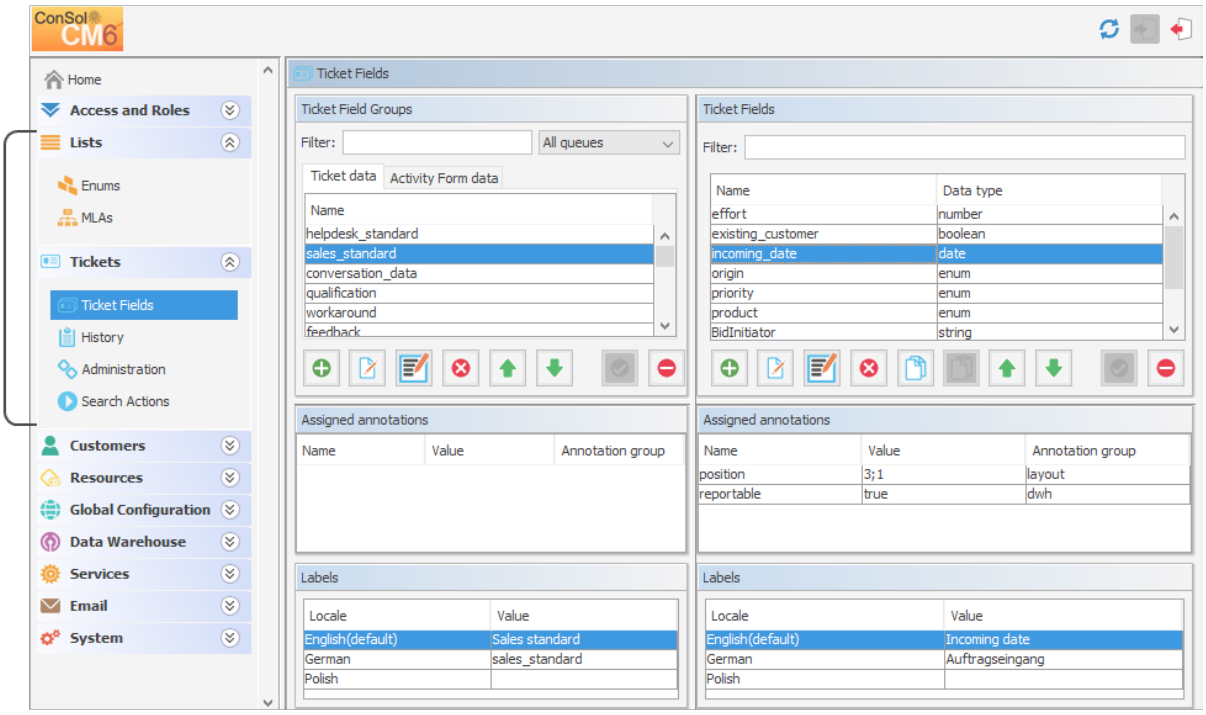


Figure 50: *ConSol CM Admin Tool - Access and Roles, Roles: Queue permissions concerning engineer functions*

- ***Ref.* column (1)**
  The permissions are valid for all tickets in this queue where a member of this role is set as additional engineer with a specific engineer function.
- ***Refer* row (2)**
  For the selected ticket types, the permission to set an additional engineer in those tickets is granted.

C - Ticket Data Model and GUI Design Section



In this section, you will learn how to define the data model for ticket data, thus this is about ticket field definition and the positioning of the ticket fields on the Web Client GUI. However, some of the data structures (enums, MLAs) might also be used for customer fields and resource fields as described in sections Customer Field Management and GUI Design for Customer Data and CM/Resource Pool - Setting Up the Basic Resource Model. The Web Client Dashboard, a new feature in versions as of 6.9.4, is included, as well as the Page Customization, a powerful mechanism to configure the layout and functionalities of the Web Client.

Furthermore you will learn how to configure labels which are displayed in the Web Client.

In this section, the following topics are covered:

- Ticket Field Administration (Setting Up the Ticket Data Model)
- Managing Sorted Lists: Enum Administration
- MLA Administration
- Ticket History
- Configuration of the Ticket List
- Configuration of the Web Client Dashboard
- Page Customization

C.1 Ticket Field Administration (Setting Up the Ticket Data Model)

This chapter discusses the following:

C.1.1 Introduction

Ticket fields are defined for tickets. By default, all tickets contain information about the assigned engineer, the creation data, and the current queue and scope. These default fields are displayed in the upper part of the ticket next to the ticket icon (1). In addition to these fields, the CM administrator can define ticket fields adapted to the use case implemented in ConSol CM. Examples for ticket fields are *priority*, *software module*, *reaction time*, or *sales potential*. Ticket fields always belong to ticket field groups, which are then assigned to queues to make the ticket fields available. The ticket fields can be displayed in the ticket header (2) or in the Details section (3).

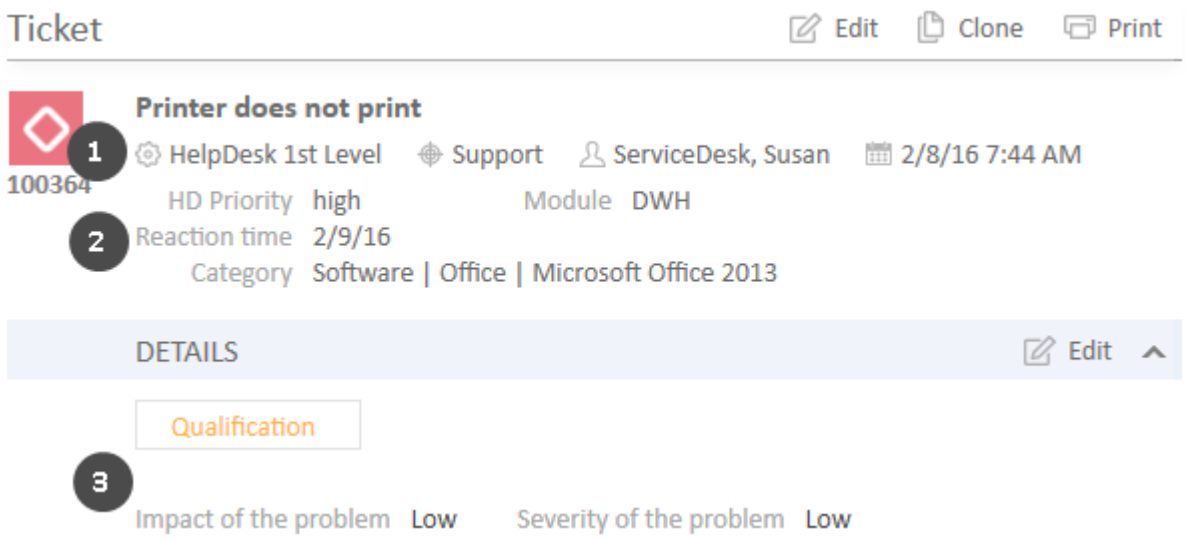


Figure 51: *ConSol CM Web Client - Ticket fields*

A **ticket field group**:

- can be assigned to a queue. For example, the ticket field group *helpdesk_fields* can be assigned to the queue *HelpDesk*.

- can be faded in and out in the Web Client GUI during the process. For example, you can fade in the ticket field group *solution* later in the process, so it is displayed at the point when the engineer found a solution to the issue reported by the customer. You cannot fade in or out single ticket fields.
- can be displayed below the default fields or as a tab in the *Details* section of a ticket. In the latter case, the title and mouse-over of the tab is the (localized) name of the ticket field group. If the ticket field group is displayed in the ticket header, the group name is not shown.
- is configured using group annotations. Annotations are set to define special parameters and characteristics of a ticket field group, e.g. the initial display mode on the user interface. Please see List of Group Annotations for further information about the available group annotations.
- is placed on the Web Client GUI based on its position in the list of ticket field groups, which defines the order of the ticket field groups in the ticket header and/or the order of the tabs.

A **ticket field**:

- is always defined within a ticket field group.
- is assigned to a queue as a part of its ticket field group.
- can be made invisible using annotations, but cannot be faded in or out as single field during the process.
- is configured using field annotations. Annotations are set to define special parameters and characteristics of a ticket field, e.g. the position on the user interface. Please see List of Field Annotations for further information about the available field annotations.
- is placed on the Web Client GUI based on the value of its `position` annotation or based on its position in the list (the first ticket field in the list is displayed first on the GUI) if `position` is not set.

To define new **ticket fields** for a queue, you have to perform the following steps:

1. Define a ticket field group and set the respective annotations.
2. Define all sorted lists which you will need for the ticket fields (see Managing Sorted Lists: Enum Administration). For example, if the ticket field *priority* should contain a list of priorities, you have to define this list first.
3. Define all ticket fields within the new ticket field group.
4. Assign the new ticket field group to the queue where its ticket fields are required.
5. Test the results in the Web Client GUI. You do not need to log in again, it will be sufficient to refresh a page which shows a ticket in the respective queue.

All these steps are explained in detail in the following sections.

In case some data fields should be offered in a form during one or more processes, Activity Control Forms (ACFs) can be defined. This is explained in the section Tab Activity Form Data.

C.1.2 Ticket Field Administration Using the Admin Tool

In the Admin Tool, ticket fields are managed using the navigation item *Ticket fields* in the navigation group *Tickets*.

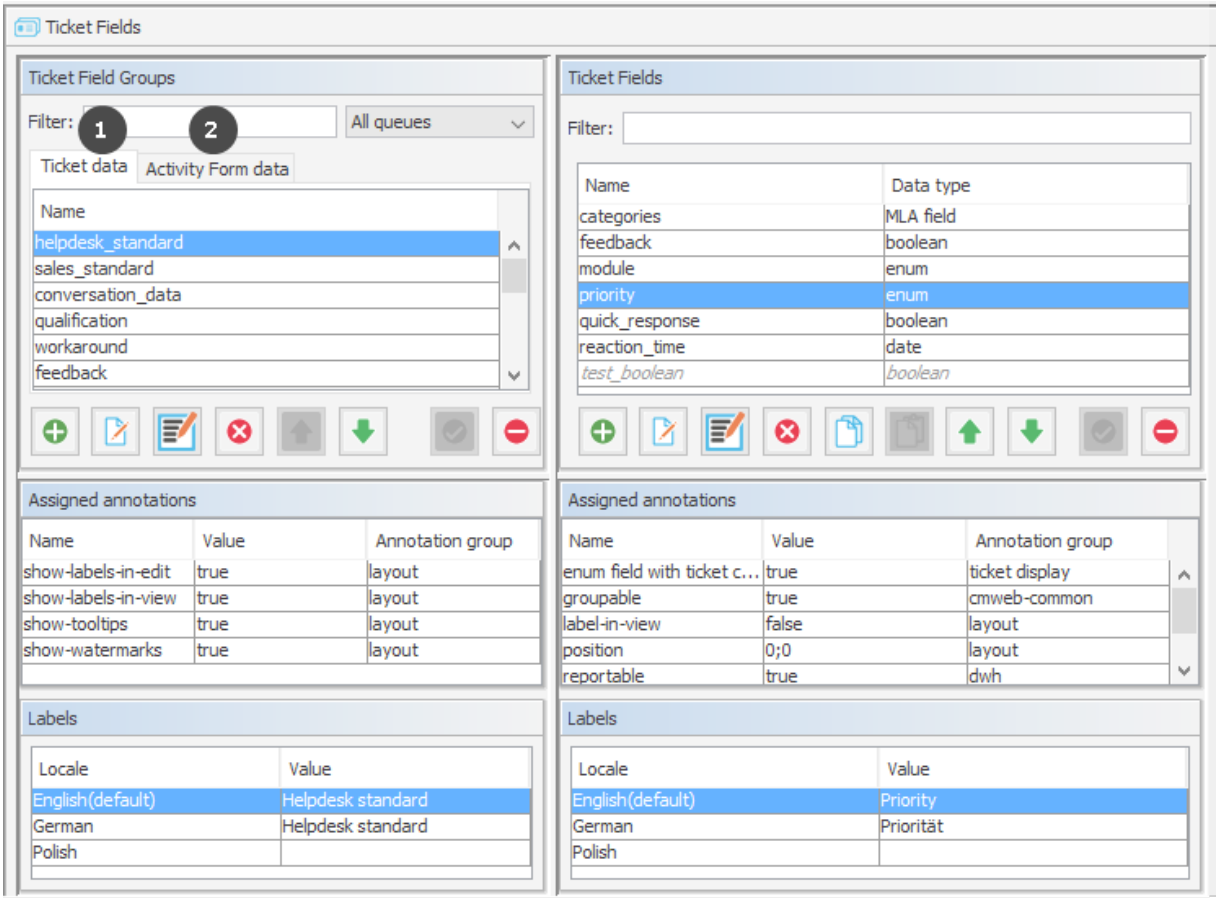


Figure 52: *ConSol CM Admin Tool - Tickets, Ticket Fields*

The navigation item *Ticket fields* has two tabs:

- Tab Ticket Data (1): This tab is used to define ticket field groups and ticket fields.
- Tab Activity Form Data (2): This tab is used to define Activity Control Forms.

Both tabs are explained in detail in the following sections. You can switch between the tabs by clicking the tab header.

C.1.3 Tab *Ticket Data*

On this tab, you can define groups and fields for ticket data.

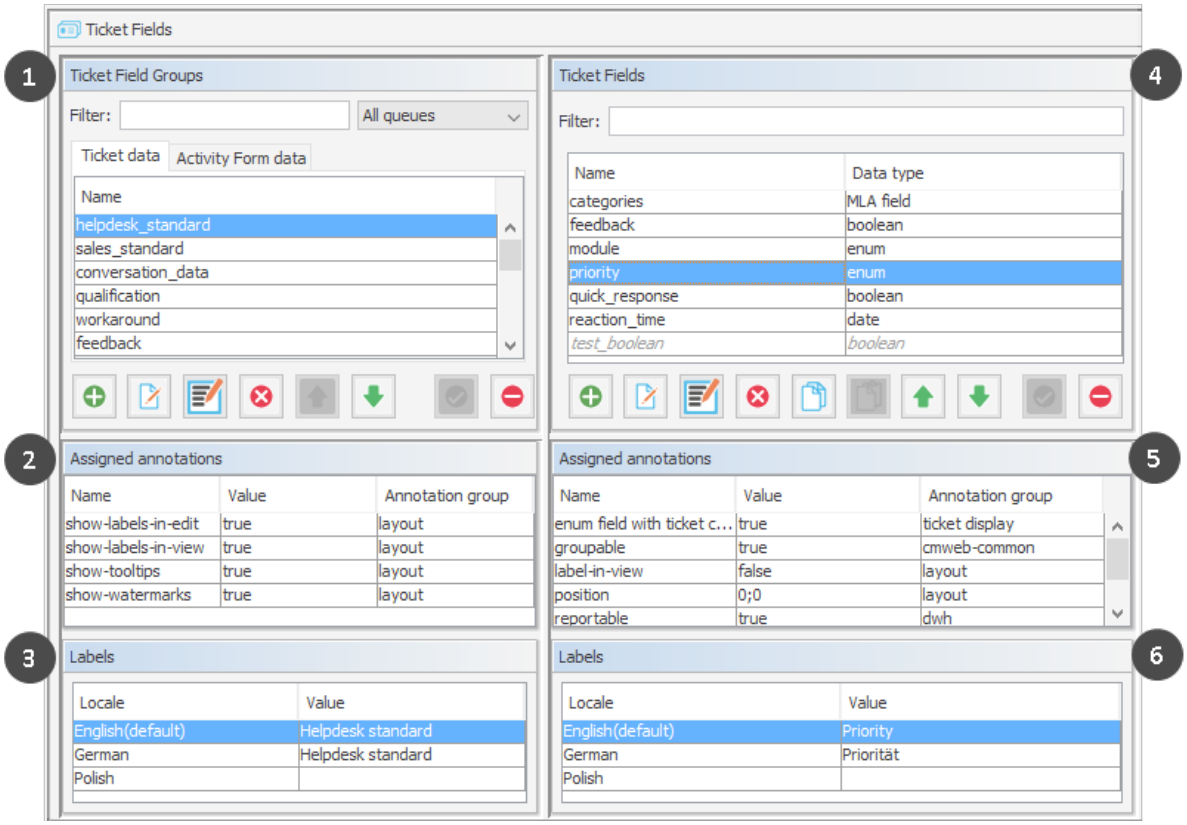


Figure 53: *ConSol CM Admin Tool - Tickets, Ticket Fields*

The *Ticket data* tab consists of six sections:

- Ticket field groups (1)
- Annotations of the selected ticket field group (2)
- Labels, i.e., localized values, of the selected ticket field group (3)
- The ticket fields of the selected ticket field group (4)
- Annotations of the selected ticket field (5)
- Labels, i.e., localized values, of the selected ticket field (6)

C.1.3.1 Create a Ticket Field Group

To create a new ticket field group just click the *Add* button below the list on the left side of the page. The following pop-up window appears.

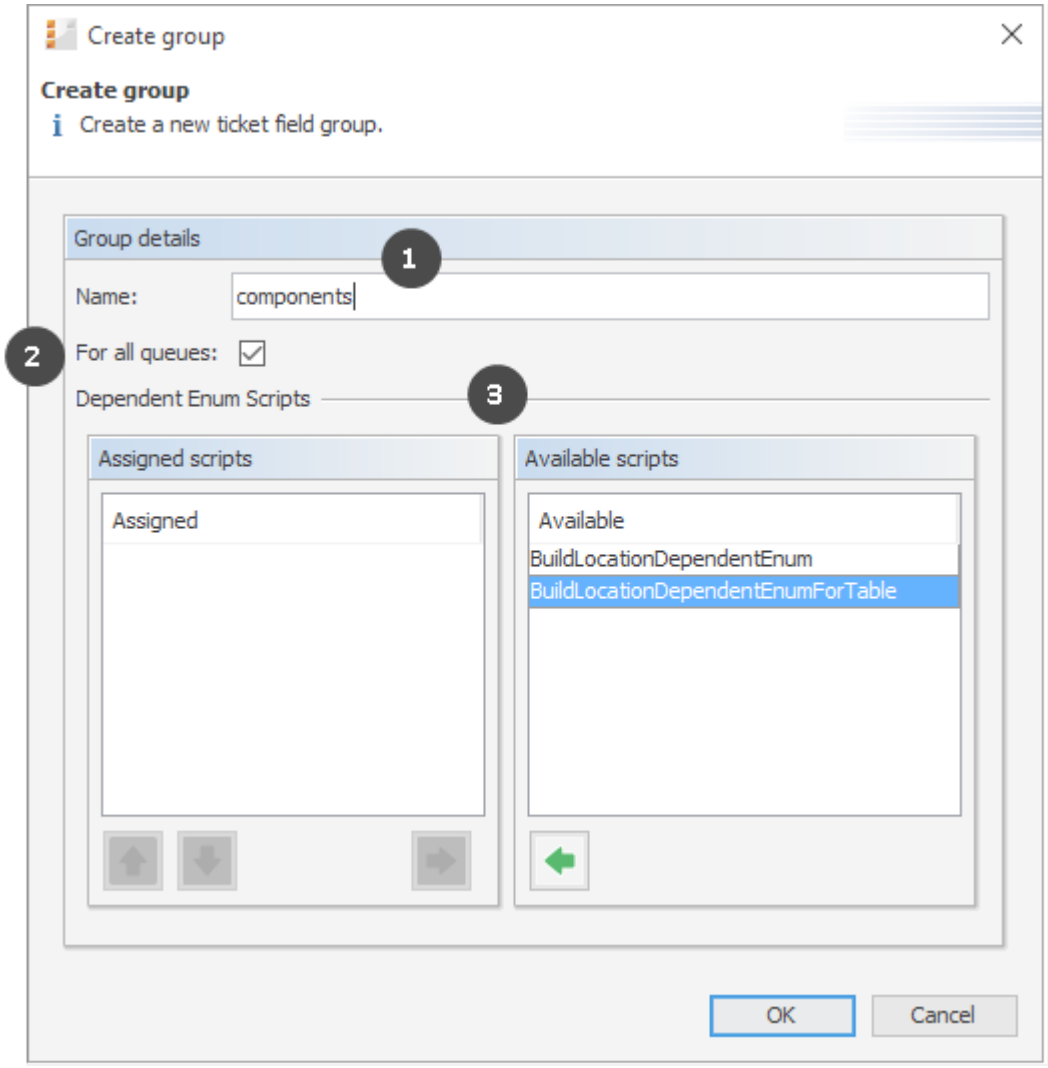


Figure 54: *ConSol CM Admin Tool - Tickets, Ticket Fields: Creating a ticket field group*

- **Name (1)**
  Enter a name for the ticket field group. The name must be unique. The localized names for a ticket field group are set in the main *Ticket Fields* panel. For details, please refer to section Localization of Data Fields.
- **For all queues (2)**
  If this check box is activated, this group's ticket fields are automatically assigned to all queues. A ticket field group which is marked as *for all queues* will also be assigned automatically to each newly created queue. Usually ticket field groups for ticket data are only valid in specific queues (see Queue Administration).
- **Available scripts, i.e., Dependent Enum Scripts (3)**
  Dependent enum scripts define the structure of DEPENDENT ENUMS (hierarchical multi-level lists) used in ticket fields of this ticket field group. With DEPENDENT ENUMS you can limit the choices in multi-level lists. You select an element in a list and, based on this selection, only

matching results will be shown in the next lower hierarchy level of the list. The ENUMS (single lists) for the ticket fields have to be created within the Managing Sorted Lists: Enum Administration while the scripts that couple the lists to create the DEPENDENT ENUM are created using an Admin Tool script, see section Admin Tool Scripts.
To assign dependent enum scripts to a ticket field group select the desired script(s) in the list *Available scripts* and move them to the list *Assigned scripts* by clicking the *Assign* button.

C.1.3.2 Edit a Ticket Field Group

If you want to edit a ticket field group, select it in the list and click the *Edit* button. The same window as described above for creating a ticket field group will appear. You can modify all fields and save your changes by clicking *OK*.

C.1.3.3 Annotate a Ticket Field Group

Ticket field groups are annotated to define their characteristics, e.g., where a group is displayed in the Web Client, if a group is indexed, or if it should be visible. You can define, e.g., whether a group is visible in the Web Client (annotation `group-visibility`) or whether it is shown in the *Details* section of the Web Client (annotation `show-in-group-section`). To assign annotations select a group and click the *Annotate* button. The following pop-up window appears:

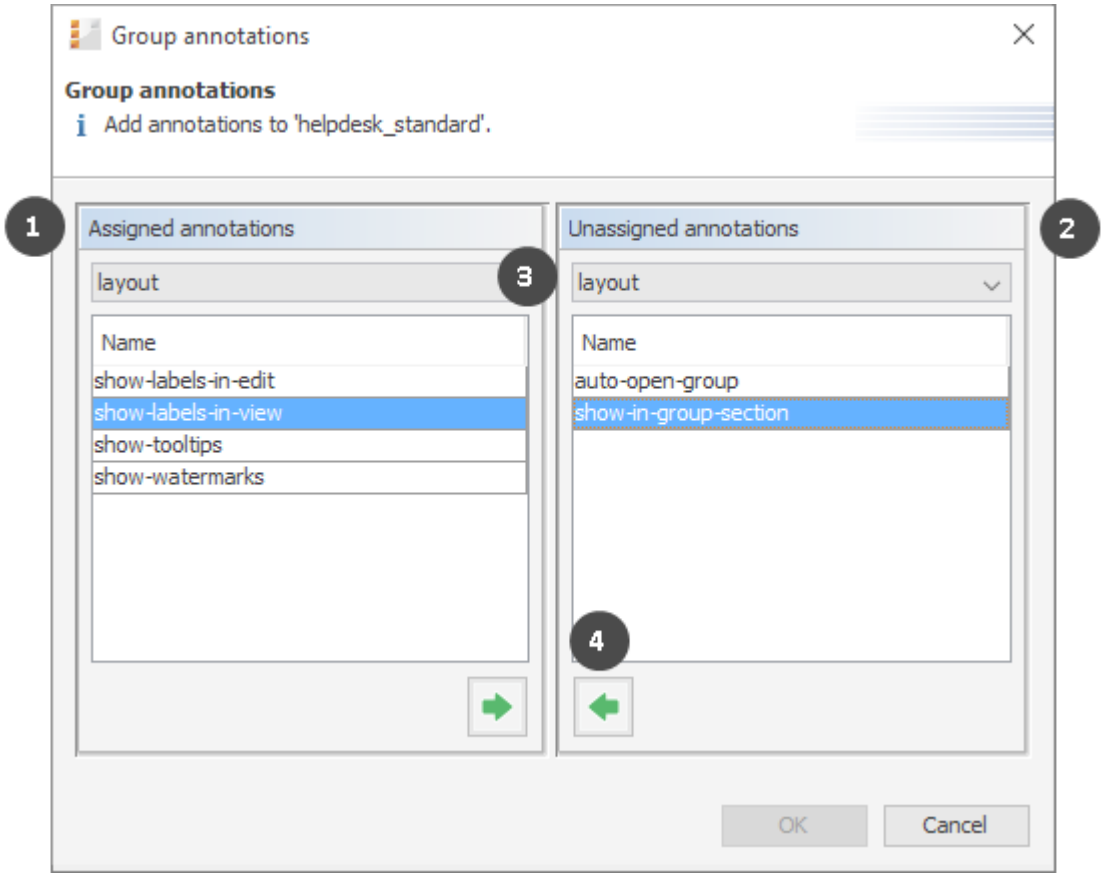


Figure 55: *ConSol CM Admin Tool - Tickets, Ticket Fields: Assigning annotations to a ticket field group*

The right part of the window contains the available annotations (2). Using the selection field above the list you can filter the display by annotation type, e.g., *common* or *layout* (3). Select the desired annotations and move them to the *Assigned annotations* list on the left (1) by clicking the *Assign* button (4). This list can also be filtered by annotation type. Click *OK* to assign the annotations to the ticket field group and to close the window. See Annotations, section *Group Annotations* for detailed information.

The annotations are now shown with a default value (if available, e.g., “true” or “false”) in the bottom left-hand corner of the administration page. The value can be modified by double-clicking into the corresponding *Value* field and typing the desired value. Press the Enter key afterwards.

Ticket field groups will appear in the Web Client as they are ordered in the list. Select a group and use the *Move upwards* and *Move downwards* buttons and if you want to change the position of this group in the list.

C.1.3.4 Delete a Ticket Field Group

A ticket field group can only be deleted if it is not assigned to a queue or a ticket. Otherwise you get a warning stating you can only disable this group (see below). In order to delete a ticket field group select it in the list and click the *Delete* button. If you confirm the following dialog with *Yes*, the group with its corresponding fields is removed from the list and the system.

C.1.3.5 Enable or Disable a Ticket Field Group

If you cannot delete a ticket field group, or if you do not want to delete it because you might need it again, you can disable it. To do so select the group and click the *Deactivate* button. The entry in the list is shown in italics afterwards. Disabled ticket field groups are not displayed in the Web Client. Just click the *Activate* button below the group list if you want to enable the group again.

C.1.3.6 Create a Ticket Field

Ticket fields contain the data for tickets, e.g., priority, service level, deadline, or hardware module. The fields of a ticket field group are created in the right part of the page. To create a new ticket field, select the desired group first on the left and then click the *Add* button below the *Ticket Fields* area on the right. The following pop-up window appears:

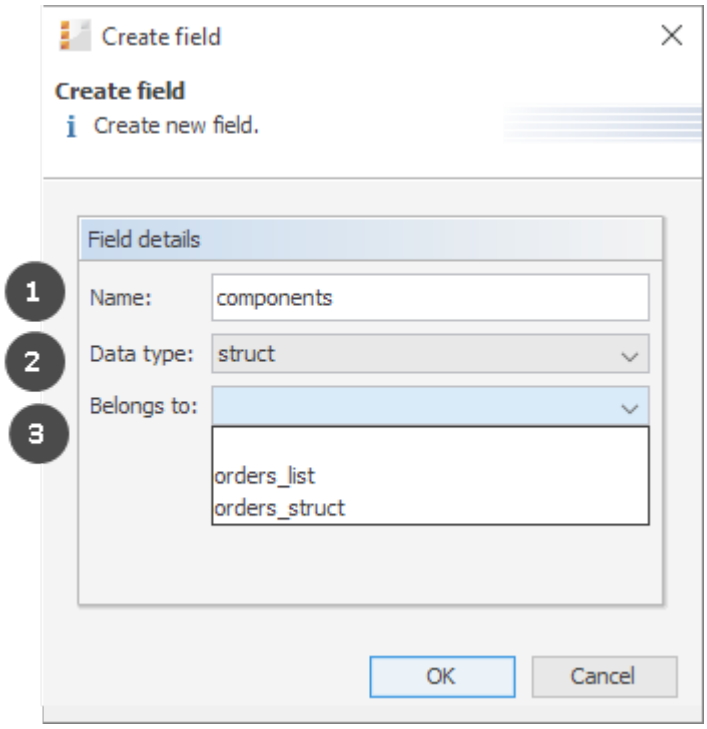


Figure 56: *ConSol CM Admin Tool - Tickets, Ticket Fields: Creating a ticket field*

Fill out the following information:

- **Name (1):**
  Enter a name for the ticket field. The name must be unique within the ticket field group. The localized names for a ticket field are set in the main *Ticket Fields* panel. For details, please refer to section Localization of Data Fields.
- **Data type (2):**
  Choose one of the available data types for the new ticket field. Please see Types of Data Fields.
- **Belongs to (3):**
  This field shows the available ticket fields of the data types *list* and *struct* used to create lists or tables. Choose in the drop-down box to which list or structure the ticket field belongs (if applicable).

Types of Data Fields

The following data types are available for ticket fields, customer fields and resource fields.

- **autocomplete**
  A data field which contains a scripted autocomplete list. This is a dynamic list which is based on

a script of type *Text Autocomplete*. A detailed explanation of scripted autocomplete lists is given in section Scripted Autocomplete Lists.

- **boolean**
  Values: true/false. Depending on the annotation `boolean-type`, the value is displayed as checkbox, radio buttons, or drop-down list.

  > If you work with scripts, either in CM workflows or in the Admin Tool, please note that the behavior of boolean fields which are represented as checkboxes, i.e. with annotation `boolean-type = checkbox` (default) is different depending on the CM version!
  >
  > - **In CM versions prior to 6.9.4.0:**
  >   If a boolean field has not been touched, its value is "false". If it is checked, its value is "true", and if it is unchecked again, its value is "false "again.
  > - **In version 6.9.4.0 and up:**
  >   If a boolean field has not been touched, its value is "NULL". If it is checked, its value is "true", and if it is unchecked again, its value is "false".
  >
  > Fields which have already been filled with values in the database will not be changed during an update from a version prior to 6.9.4.0 to a version 6.9.4.0 and up.
  >
  > Boolean fields represented as radio buttons (annotation `boolean-type = radio`) or drop-down menu (annotation `boolean-type = select`) are always shown and behave as described for versions 6.9.4.0 and up, i.e., with "NULL "value if untouched.

- **date**
  Format and accuracy can be set by annotations.
- **enum**
  For sorted lists. The engineer can choose one of the enum values in the Web Client. Enums and values have to be created previously within the Managing Sorted Lists: Enum Administration. Select the desired *Enum type* and *Enum group* in the fields below.
- **list**
  A data field of this data type is the first step to creating a list (one column) or a table (multiple columns) of input fields in the Web Client.
  - For a **table** the next step will be to create another field of type *struct* (see below) to contain the input of the individual list fields (which will become the columns of the table). So, if you want to create a table you have to define a field of the type *struct* first (see below) before you can add the fields for the table columns.
  - For a **simple list**, the next step will be to create fields which belong to the list. No *struct* is required.

  For all fields belonging to a list or table you have to set the dependencies in the field *Belongs to* (see below). For example, a table field (which is a regular data field) always belongs to a *struct*, a struct always belongs to a *list.*

- **struct**
  A data field of this type defines a data structure (line of a table) which groups one or multiple fields. It is the second step to building a table after you have created a field of the type *list*. Add the fields for the columns of the table in the next step. The dependencies have to be set for each field in the *Belongs to* field (see below), i.e., a *struct* always belongs to a *list*.

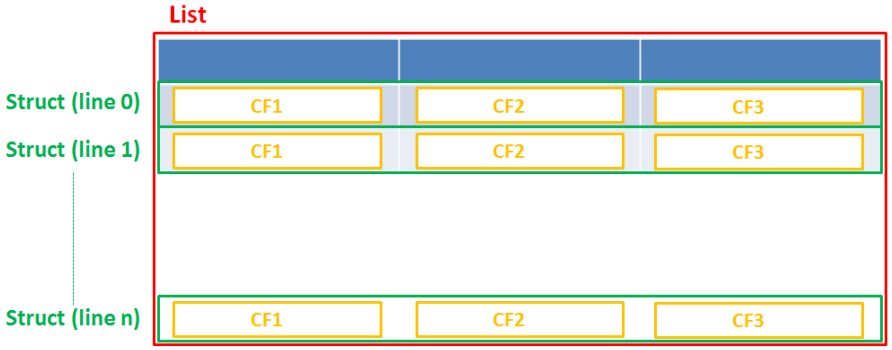


Figure 57: *Scheme: List of structs*

Technically spoken, the list is an array which contains a map (= key:value pairs) in each field.

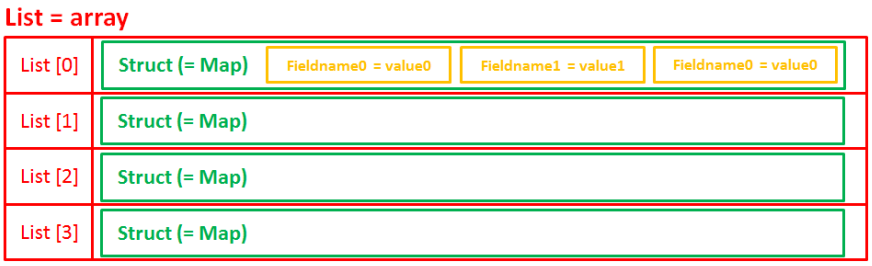


Figure 58: *List of structs, technical principle*

- **number**
  For integer values.
- **fixed point number**
  For numbers with a fractional part, e.g., currencies. You have to enter the total number of digits (*Precision*) and the number of digits that fall to the right of the decimal point (*Scale*) in the respective fields below.
- **string**
  For up to 4000 alphanumeric characters.

> ⚠ Restriction when using an Oracle database: at most 4000 bytes can be saved in UTF encoding. Starting with Oracle12c.

- **long string**
  For large objects.

  > ⚠ For *long strings* the limit depends on the database system used for ConSol CM: MS SQL Server: 2 GByte; MySQL: 4 GByte; Oracle: 16 - 64 GByte (depending on page size of tablespace).

- **short string**
  For up to 255 alphanumeric characters.

  > ⓘ For *string* fields, you can use specific annotations to fine-tune the field definition. For example, a *string* field can be defined to contain a URL which will automatically be displayed as hyperlink or can be the hook for an autocomplete list. Please read the following section.

- **contact data reference**
  Special data type used internally for referencing the contacts associated with a ticket. In FlexCDM (i.e., starting with CM version 6.9.0), this data type is no longer displayed but only used internally in the CM system.
- **MLA field**
  This data type is used for fields that contain hierarchical lists with a tree structure called *MLA* (Multi Level Attributes). The name of the field is the name of the new MLA that has to be defined within the MLA Administration. The group of the field has to be referenced when the MLA is created.

> ⚠ The data type you choose on creating a data field cannot be changed afterwards!

Details about String Fields: Use Annotations to Fine-Tune Strings

String fields are widely used for customer, ticket, and resource data and strings can be used to contain various content, for example, a text box with a comment, a simple input field with only 20 characters, a URL or a password. The fine-tuning of string fields is implemented using specific annotations which are all listed on the Annotations page. However, since work with these annotations is an every-day task of CM administrators, the most important and most commonly used annotations will be explained here as well.

How can I ...

... insert a **text box** instead of a single line?

Value for annotation `text-type`: “textarea”

The size of the text box can be adjusted, displayed as standard text box depending on web browser. Use the `field-size` annotation in case a specific size of the text box is required.

... hide the input of the fields for **passwords**?

Value for annotation `text-type`: “password”

Only dots will be displayed. This annotation does **not** define the field to contain a password! It only defines the display mode! Use the `password` annotation to define a string field to contain the CM/Track password.

... display a **hyperlink**, display the name instead of the link?

Value for annotation `text-type`: “url”

Input will be displayed as a hyperlink in view mode. String has to match a specific URL pattern:

- "^((?:mailto\:|(?:(?:ht|f)tps?)\://)1\S+)(?: (?:\|)?(.*))?$"

First part of the string is the link (url), second part is the name which should be displayed.

Example: "http://consol.de ConSol"

... display a **file link**?

Value for annotation `text-type`: “file-url”

Input will be displayed as a link to a file on the file system. The web browser has to allow/support those links!

Example: Enabling file:// URLs in a Firefox browser

Add the following lines to either the configuration file `prefs.js` or to `user.js` in the user profile. On a Windows system usually in a folder like `C:\Users\<USERNAME>\AppData\Roaming\Mozilla\Firefox\Profiles\uvubg4fj.default`

- user_pref("capability.policy.localfilelinks.checkloaduri.enabled", "allAccess");
- user_pref("capability.policy.localfilelinks.sites", "http://cm-server.domain.com:8080");
- user_pref("capability.policy.policynames", "localfilelinks");

Alternatively a Firefox browser add-on like ***Local Filesystem Links*** can be installed for better access to the referenced files and folders.

The link will also be displayed as tooltip.

The URL is correctly formed if the following conditions are met:

- It starts with `file:` followed by regular slashes:
  - three slashes "///" for files on the same computer as the browser (alternatively "//localhost/") or
  - two slashes followed by the server name followed by another slash for files on file servers accessible from the computer running the browser.
- These are followed by the full path to the file ending with the file name.
- The path on Microsoft Windows systems is also written with forward slashes instead of backslashes.

- The drive letter of a local path on Microsoft Windows systems is noted as usual, for example `C:`.
- Paths with spaces and special characters like "{, }, ^, #, ?" need to be percent encoded ("%20" for a space for example) for Microsoft Windows systems.

Example URLs:

- file://file-server/path/to/my/file.ext
- file:///linux/local/file.pdf
- file:///C:/Users/myuser/localfile.doc

See also the explanation about `file-url` in the section text-type

... define a **label**?

Value for annotation `text-type`: “label”

This will be a read-only field which is displayed in gray, use the ***label-group*** annotation to link label and input fields which belong together. Please take a look at the annotations for labels (`show-label-in-edit`, `show-label-in-view`) before implementing special label fields!

... define a field for the **valid email addresses**?

Value for annotation `email`: “true”

The field may only contain valid email addresses. Input will be validated according to standard email format <name>@<domain>.

... define a scripted autocomplete list?

Value for the annotation `text-type` = “autocomplete”

Optional: value for the annotation `autocomplete-script` = <name of the respective script>

A scripted autocomplete list is used to provide a drop-down menu which is filled dynamically using the input the engineer has provided so far. For example, when the user types "Mil", the possible values "Miller", "Milberg", and "Milhouse" are displayed as list and the engineer can select the one required for the field. You know this behavior from other autocomplete fields, e.g., the search for engineers for a ticket or the search for customers while creating a ticket. However, in these cases, CM generates the list automatically. The behavior cannot be influenced or customized. Scripted autocomplete lists, on the contrary, can be implemented by the CM administrator. The values are based on a result set which is dynamically created. The result set can contain strings, engineers, customers (Units), and resources.

A detailed description of scripted autocomplete lists is provided in section Scripted Autocomplete Lists.

... define a field containing personal data?

Value for annotation `personal-data`: “true”

This annotation can be assigned to ticket and contact fields. Contact fields with this annotation will be deleted when a contact is anonymized. Ticket fields with this annotation will be deleted when the main customer of the ticket is anonymized. See Example 8: Removing Customer Data for information about how to anonymize a contact.

When defining a field to contain personal data, please take into account that the deletion of the field during the anonymization process is treated as a regular update. Therefore, business event triggers reacting on changes to ticket fields fire, and the contact update action script is executed.

This might lead to unwanted side-effects.

C.1.3.7 Copy a Ticket Field

You do not necessarily have to create every data field using the pop-up menu. You can also copy an existing data field. This might be a ticket field, customer field, or resource field. Select the respective data field in the data field group and click the *Copy* button. Then navigate to the data field group where the copied field should be placed. This might be a ticket field group, a customer field group, or a resource field group. You can copy a field from one object type (e.g. ticket) to another (e.g. customer). Click into the list of data fields of the target field group and click the *Paste* button.

Every type of field can be copied, from simple string fields over enums to entire lists of structs. To copy a list of structs or a list of fields, mark and copy only the list, the child objects (e.g., structs with fields) will be copied automatically.

These things will be copied:

- almost all annotations with their values
- all localized values

These things will not be copied:

- the following annotations
  - *ldapid*: This annotation can be only once in a data model. Duplication would cause an invalid model.
  - *username*: This annotation can be only once in a data model. Duplication would cause an invalid model.
  - *password*: This annotation can be only once in a data model. Duplication would cause an invalid model.
  - *position*: The value for this annotation must be unique, so the annotation is duplicated, but the value is removed from the copy.
  - *ticket-list-position*: This annotation should be used very specifically and, thus, it should be present only for very few fields. Duplicating it would potentially multiply it although it should not be used for the majority of copies.

Naming of the copied field(s):

- Copy to other field group: field name is used
- Copy to same field group: <field name>_copy is used

Besides the Copy/Paste buttons you can also use the keyboard shortcuts:

- Copy: CTRL-C
- Paste: CTRL-V

C.1.3.8 Edit a Ticket Field

If you want to edit a ticket field, select it in the list and click the *Edit* button. The same window as described above for creating a ticket field will appear. Except for *data type*, *enum type*, and *enum group* you can modify all fields and save your changes by clicking *OK*.

C.1.3.9 Annotate a Ticket Field

Just like ticket field groups, ticket fields are annotated to define the properties of the field, e.g., is it read-only, should it be indexed, and where it should be displayed on the Web Client GUI (please see section Annotations for a list of all available annotations). Select a field and click the *Annotate* button below the list. The following pop-up window appears:

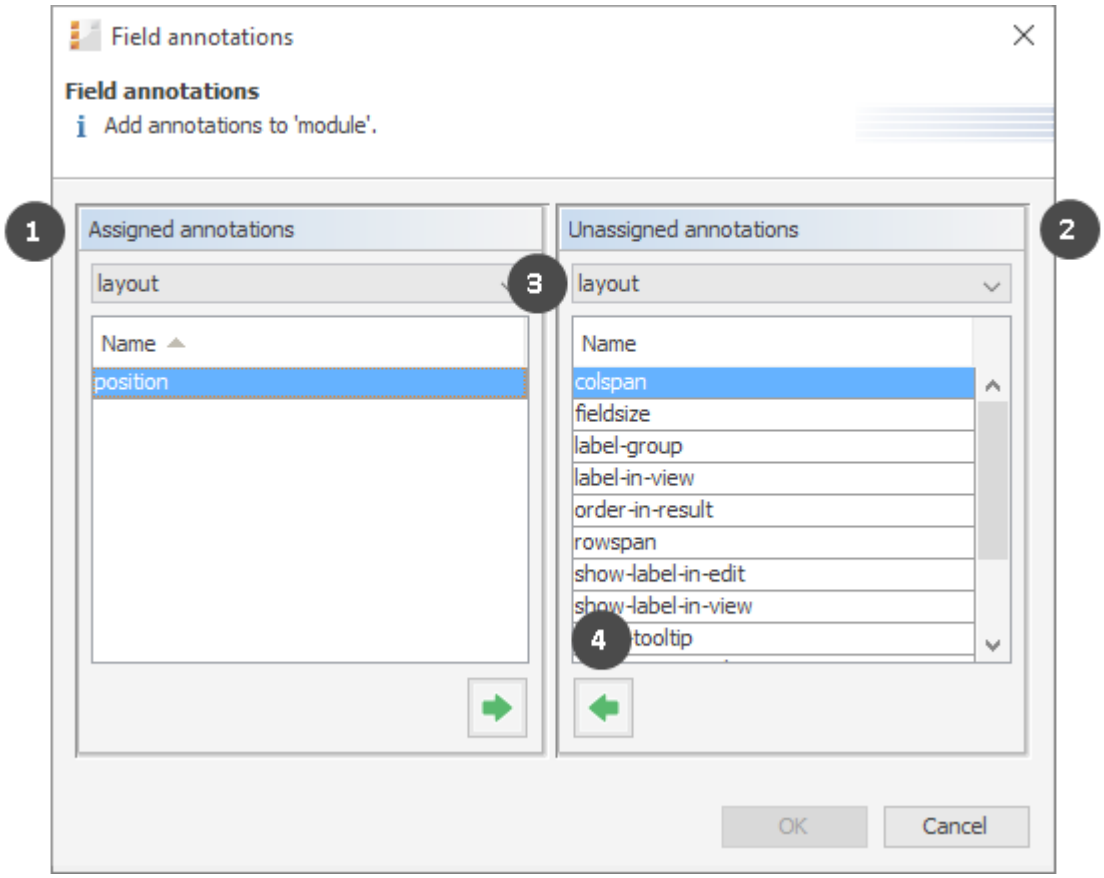


Figure 59: *ConSol CM Admin Tool - Tickets, Ticket Fields: Assigning annotations for a ticket field*

The right part of the window contains the available annotations (2). Using the selection field above the list you can filter the display according to annotation type (3). Select the desired annotations and move them to the *Assigned annotations* list on the left (1) by clicking the *Assign* button (4). This list can also be filtered according to annotation type. Click *OK* to assign the annotations to the ticket field and to close the window.

The annotations for the selected field are now shown with a default value (if available, e.g., “true” or “false”) in the bottom right-hand corner of the administration page. The value can be modified by double-clicking the corresponding *Value* field and typing the desired value. Press the Enter key afterwards.

Ticket fields will appear in the Web Client as they are ordered in the list unless you have assigned a layout via the `position` annotation. You can change the position of a field in the list by using the *Move upwards* and *Move downwards* buttons below.

Note on the layout of the ticket data:

You can define several columns of data fields on each line, e.g., the position annotation can end with 0, 1 or 2.

0;0 0;1 0;2

1;0 1;1 1;2

C.1.3.10 Delete a Ticket Field

A ticket field can only be deleted if it is not assigned to a queue or a ticket, otherwise you get a warning stating you can only disable this field (see below). In order to delete a ticket field, select it in the list and click the *Delete* button. If you confirm the following dialog with *Yes*, the ticket field will be removed from the list and the system.

C.1.3.11 Enable or Disable a Ticket Field

If you cannot delete a ticket field, or if you do not want to delete it because you might need it again, you can disable it. To do so select the field and click the *Deactivate* button. The entry in the list is shown in italics afterwards. A disabled ticket field is not displayed in the Web Client. Just click the *Activate* button below the ticket field list, if you want to enable the field again.

C.1.3.12 Visibility of Ticket Data in CM/Track

If CM/Track, a ConSol CM Add-on which provides a customer portal, is active in your CM system, you will have to configure ticket fields which are specific for CM/Track. Furthermore, the visibility of the ticket fields in the customer portal has to be configured. This is explained in detail in section CM/Track V2: Data Availability For Customers.

C.1.3.13 Using Scripted Field Visualization for Ticket Fields

Using Scripted field visualization, you can enhance the display of data in ticket fields. Please see section Scripts of Type Field Visualization for details.

C.1.4 Tab *Activity Form Data*

C.1.4.1 Introduction

An Activity Control Form is a web form which is displayed during the process, i.e., when the engineer works on the ticket. If the engineer clicks a workflow activity, the ACF is displayed first. The engineer has to fill out certain data fields and save the changes in order to proceed and execute the workflow activity.

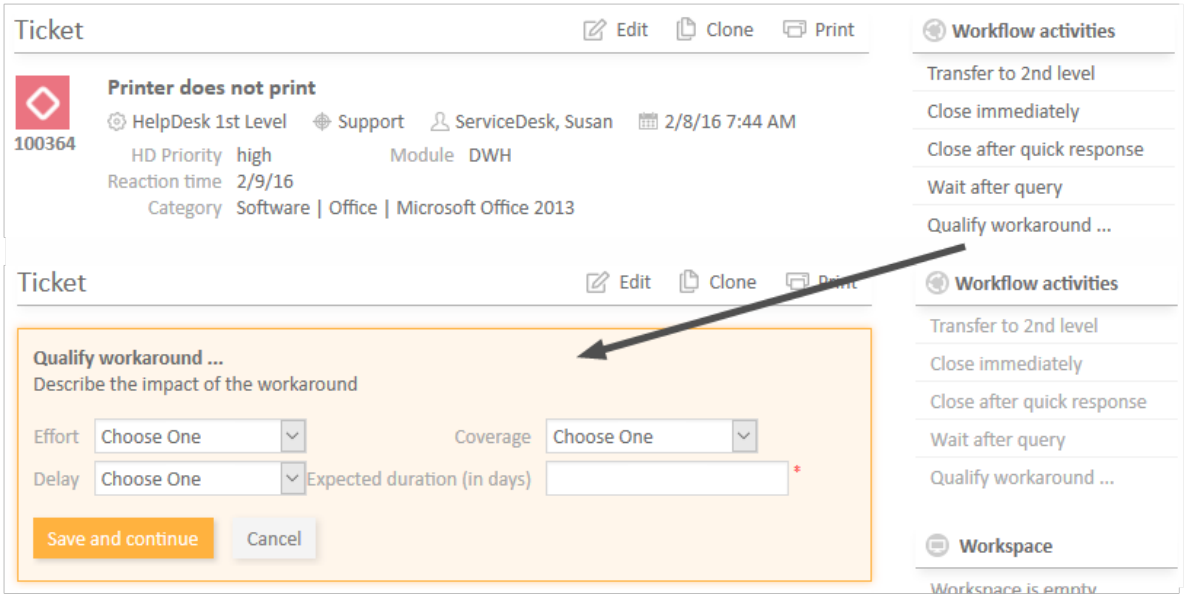


Figure 60: *ConSol CM Web Client - Example for ACF*

The ticket fields which are used in an ACF have to be defined first, as regular ticket fields in the ConSol CM Admin Tool. ACFs can include ticket fields from one or more ticket field groups. The ACFs is integrated into the process (workflow) using the ConSol CM Process Designer. There, you can also define which ticket fields should be required and which fields are optional. The display of an ACF can depend on a condition, e.g., the ACF *Reasons for dismissal of request* is only displayed if the customer has *Gold* or *Platinum* status. ACF dependencies are configured using scripts. This is explained in detail in the *ConSol CM Process Designer Manual*.

C.1.4.2 Definition of Activity Control Forms (ACFs)

In this tab, you can create activity control forms (ACF) which can be assigned to activities in the Process Designer. They are used to gather input in the Web Client if a manual workflow activity needs more information for the next step, e.g., if a ticket has to be qualified before it can be moved on or if you want feedback for a ticket. ACFs are basically a set of ticket fields already created in the tab *Ticket data*. An ACF can contain ticket fields of more than one ticket field group. However, all ticket field groups have to be assigned to the queue to which the workflow using the ACF is assigned. Please read the chapter on ACFs in the *ConSol CM Process Designer Manual* for detailed information about the process flow with ACFs and the features provided using programming ACF scripts.

The following figure shows the available ACFs. You reach this screen by opening the navigation item *Ticket Fields* in the navigation group *Tickets* and selecting the *Activity Form data* tab.

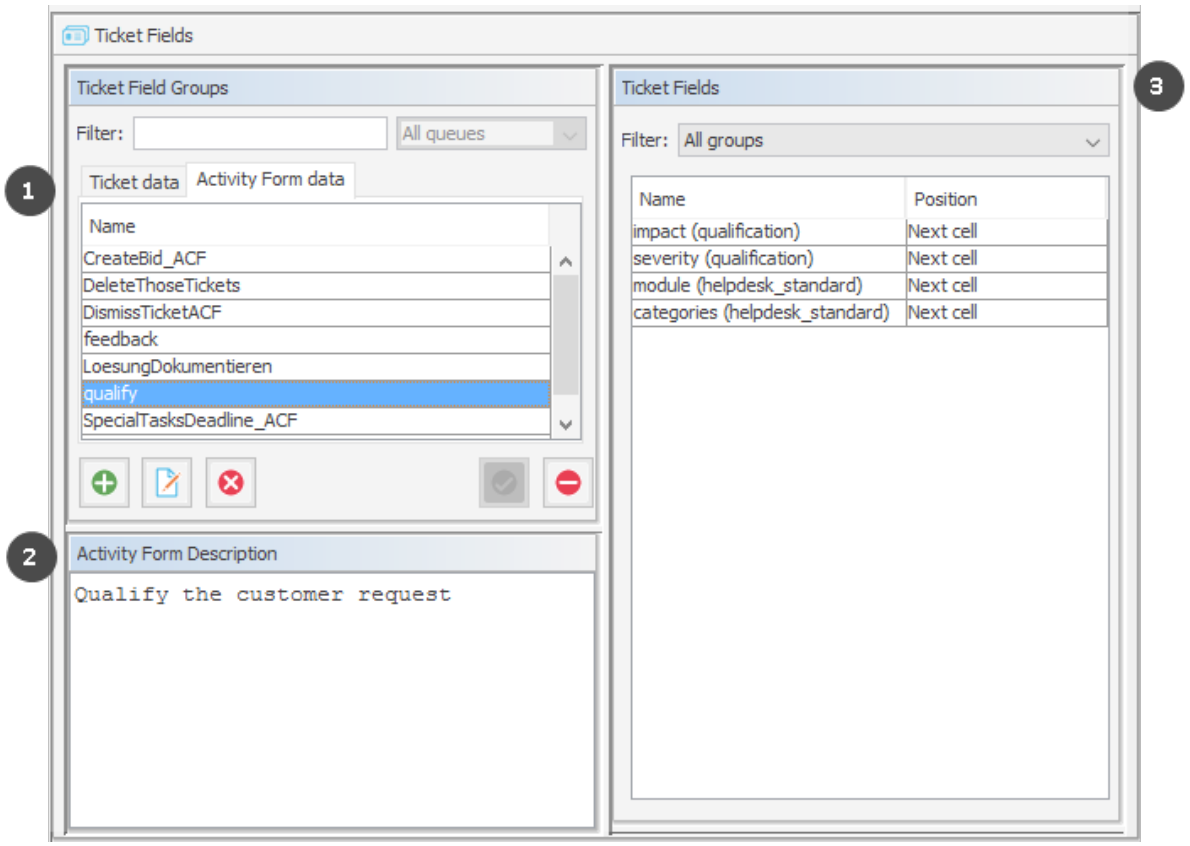


Figure 61: *ConSol CM Admin Tool - Tickets, Ticket Fields: Activity Form data*

The *Activity Form data* tab consists of three sections:

- **Activity Form data (1):**
  Contains the available Activity Forms
- **Activity Form Description (2):**
  Contains the description of the selected Activity Form
- **Ticket Fields (3):**
  Contains the ticket fields of the selected Activity Form

C.1.4.3 Create an Activity Control Form

To create an ACF just click the *Add* button below the list on the left side of the page. The following pop-up window appears (this is the same window for creating and for editing an ACF).

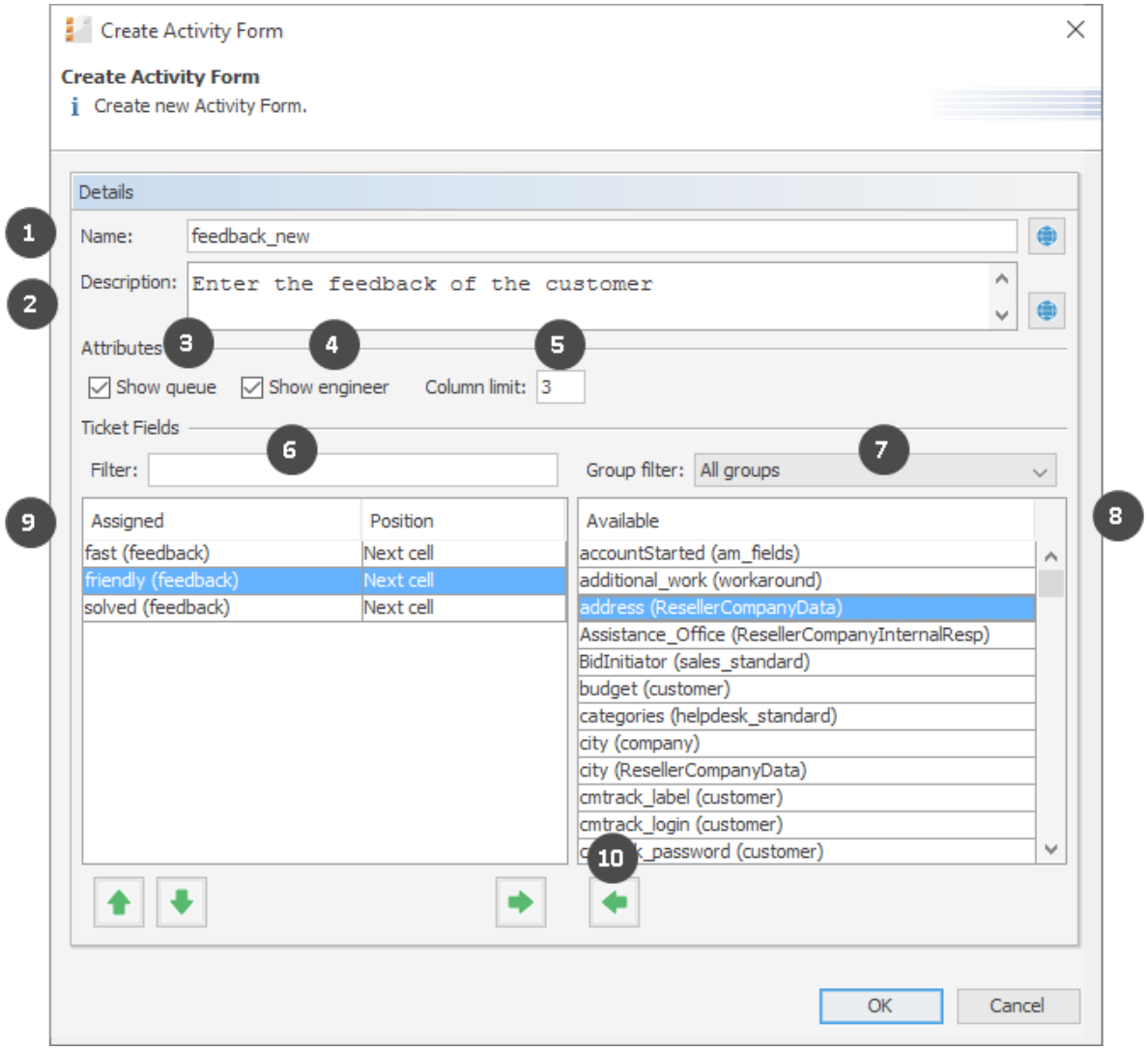


Figure 62: *ConSol CM Admin Tool - Tickets, Ticket Fields: Creating an Activity Control Form*

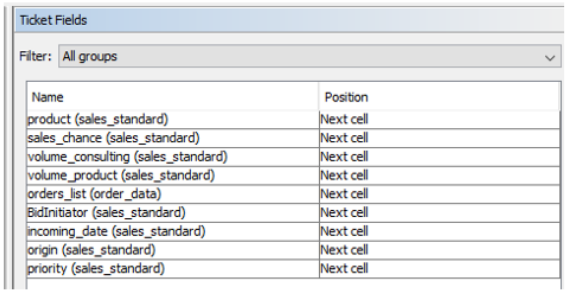
Please enter or select the following data:

- **Name (1)**
  Enter the name of the ACF in this field. You can localize the name by clicking the *Localize* button. For details, see section Localization of Objects in General, Type 1.
- **Description (2)**
  Enter a description for the ACF in this field. The description is shown as the title of the ACF in the Web Client. You can localize the description by clicking the *Localize* button. For details, see section Localization of Objects in General, Type 1.
- **Show queue (3)**
  Check this box if the queue of the ticket should be displayed with the ACF in the Web Client.
- **Show engineer (4)**
  Check this box if the engineer of the ticket should be displayed with the ACF in the Web Client.

- **Column limit (5)**
  Number field. Defines the number of columns for the display of ticket fields in the ACF. *0* means no column limit, i.e., all ticket fields in a single row. You can also work with the parameter *Display in new row* for each ticket field, see below.
- **Filter (6)**
  You can enter a string of characters into this field to filter the assigned ticket fields by name.
- **Group filter (7)**
  Select a group of ticket fields from this list if you want to display only ticket fields belonging to this group in the list of available ticket fields below.
- **Ticket field lists (8/9)**
  The list on the right (8) shows the available ticket fields with their respective ticket field group. You can sort the entries in ascending or descending order by clicking into the title field of the list. The small up and down arrow icons show the sort order. Select the ticket fields for the ACF in this list and move them to the list *Assigned* on the left (9) by clicking the *Assign* button (10). For each assigned ticket field you can define the position. The following values are possible:
  - **Next row**
    A new row will be displayed, starting with this field.
  - **Next cell** (default)
    The field will be displayed next to the previous field, no new row.
  - **New table**
    A new table will be started in a new row. This option should be used for data structures (e.g., a LIST OF STRUCTS which use a lot of space thereby destroying the readability of the ACF by using a lot of space for one column. All other columns would be moved more to the right without this *new table* option. To get an impression of this feature, please see the following four figures.

The assigned ticket fields will appear in the Web Client as they are ordered in the list. You can rearrange the list by selecting an item and clicking the *Move upwards* or *Move downwards* button. To remove assigned ticket fields, select them and click the *Unassign* button.

Click *OK* afterwards to store your entries and to close the window.



column limit = 3

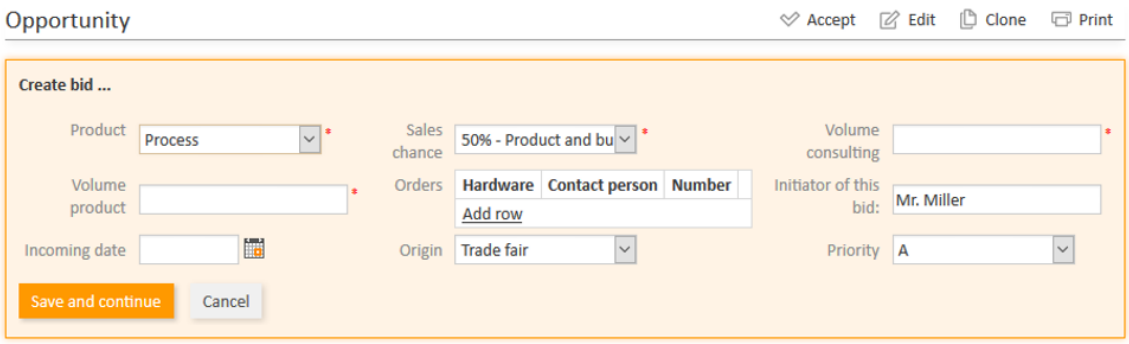


Figure 63: *Example for of ACF layout (1), not yet optimized*

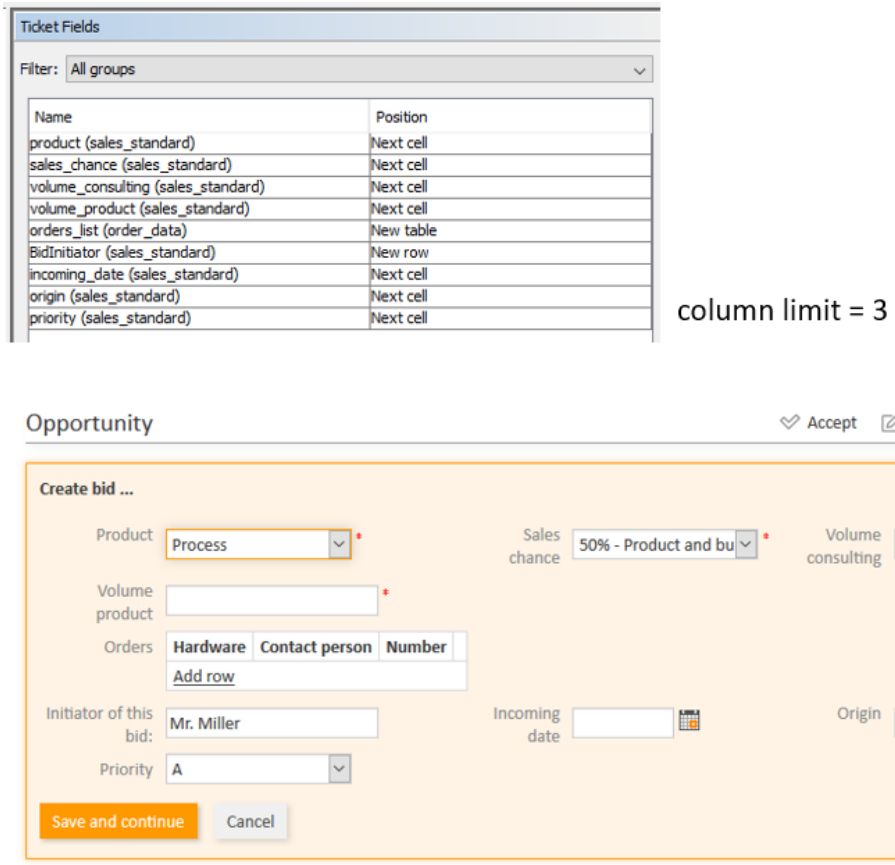


Figure 64: *Example for optimization of ACF layout (2), optimized, table in one row*

C.1.4.4 Edit an Activity Control Form

If you want to edit an ACF, select it in the list and click the *Edit* button or just double-click the name of the ACF. The same pop-up window as for creating an ACF will appear, where you can modify the details. Store your changes by clicking *OK*.

C.1.4.5 Delete an Activity Control Form

An ACF can only be deleted if it is not assigned to a workflow activity, otherwise you get a warning stating that you can only disable this ACF (see below). In order to delete an ACF, select it in the list and click the *Delete* button. If you confirm the following dialog with *Yes*, the ACF will be removed from the list and the system.

C.1.4.6 Enable or Disable an Activity Control Form

If you cannot delete an ACF, or if you do not want to delete it because you might need it again, you can disable it. To do so select the ACF and click the *Deactivate* button. The entry in the list is shown in italics afterwards. A disabled ACF is not available in the Process Designer. ACFs which are in use cannot be disabled. Just click the *Activate* button below the ACF list if you want to enable the form again.

C.1.4.7 Localize an Activity Control Form

By clicking the *Localize* button in the create or edit window you can localize name and description of an ACF. For details, please refer to the section Localization of Objects in General, Type 1.

C.1.5 Frequently Used Annotations

Here are some frequently used annotations on field level. You can find a complete list of group and field annotations in Annotations.

groupable

- **Type**: cmweb-common
- **Description**: Enables grouping of the ticket list by this field. Please see section Grouping for a detailed explanation.
- **Values**:
  - *true*: Used only with ENUM data fields. Remove the annotation if you want to disable grouping.

sortable

- **Type**: cmweb-common
- **Description**: Used to enable sorting of the ticket list by this field. Please see also the detailed explanation in section Frequently Used Annotations
- **Values**:
  - *true*: Used for data fields of type DATE or of type ENUM. Remove the annotation if you want to disable sorting.
    For ENUM fields: Works only if order index is set for all values of the ENUM field.

readonly

- **Type**: common
- **Description**: Used to indicate that the ticket field cannot be modified.
- **Values**:
  - *true / false*: Field is read-only if value is set to “true”. Lack of value, or any value except “false”, is treated as “true”.

visibility

- **Type**: common
- **Description**: Defines when the field is visible.
- **Values**:
  - *edit*: Field will be displayed in edit mode.
  - *view*: Field will be displayed in view mode.
  - *none*: Field is not visible.
  - If any other, or no value, is set then the field will always be visible.

text-type

- **Type**: component-type
- **Description**: Defines the possible types of a STRING field.

- **Values**:
  - *text* (default): Single-line input field
  - *textarea*: Multi-line input field
  - *password*: Input field for passwords.
    Password will be displayed as ******* in view mode.
  - *label*: Input will be displayed as a label, i.e., the field is displayed only, no input is possible.
  - *autocomplete*: The field will be used as autocomplete list. Please see the detailed explanation in section Scripted Autocomplete Lists.
  - *url*: The input will be displayed as a hyperlink in view mode. If no protocol is provided, *http://* is added automatically in front of the entered string. If a protocol, e.g., "http", "https", "mailto", "file", or "ftp" is used, the URL is rendered as is. The display text for the URL can be entered after a whitespace.
    Example: "http://consol.de ConSol"
  - *file-url*: Input will be displayed as a link to a file on the file system. The web browser has to allow/support those links! See section Details about String Fields: Use Annotations to Fine-Tune Strings on how to achieve this. The link will also be displayed as tooltip.

    The URL is correctly formed if the following conditions are met:

    It starts with `file:` followed by regular slashes:

    - three slashes "///" for files on the same computer as the browser (alternatively "//localhost/") or
    - two slashes followed by the server name followed by another slash for files on file servers accessible from the computer running the browser.

    These are followed by the full path to the file ending with the file name. The path on Microsoft Windows systems is also written with forward slashes instead of backslashes.

    The drive letter of a local path on Microsoft Windows systems is noted as usual, for example C:. Paths with spaces and special characters like "{, }, ^, #, ?" need to be percent encoded ("%20" for a space for example) for Microsoft Windows systems.

    Example URLs:

    - file://file-server/path/to/my/file.ext
    - file:///linux/local/file.pdf
    - file:///C:/Users/myuser/localfile.doc

    See also the explanation in the section Details about String Fields: Use Annotations to Fine-Tune Strings.

reportable

- **Type**: dwh
- **Description**: Indicates that the field is reportable and that it should be transferred to the DWH.

- **Values**:
  - *true / false*: Field is reportable if value is set to “true”.

field indexed

- **Type**: indexing
- **Description**: Indicates that a database index will be created for this field. If it should be possible to sort result tables (in the Web Client) according to a column (by clicking on the column header), the respective field has to be indexed!
- **Values**:
  - *transitive* (default): All data is displayed (ticket data, customer data and resource data).
  - *unit*: Used for customer data. Only the unit and the parent unit (i.e., company) is given as a search result, no tickets are provided.
  - *local*: Used for customer data. Only the unit is given as a search result, no company and no tickets are displayed.
  - *<annotation not set>*: Field is not indexed.

position

- **Type**: layout
- **Description**: Defines the position of a field within a grid layout or defines the position of a field within a list (STRUCT).
- **Values**:
  - **<number>;<number>**: Values define row and column (row;column), numbering starts at 0;0. If no values are set, the data field will take the next free grid cell.
  - **0;<number>**: Only the column value is used, the row value is ignored.

colspan

- **Type**: layout
- **Description**: Defines how many columns are reserved for the field in the layout.
- **Values**:
  - **<number>**: Number of columns.

⚠ This annotation only works if the annotation `position` is also set for the field.

rowspan

- **Type**: layout
- **Description**: Indicates how many rows within the layout are occupied by this field.
- **Values**:
  - **<number>**: Number of rows.

⚠ This annotation only works if the annotation `position` is also set for the field.

field-group

- **Type**: layout
- **Description**: Allows grouping of fields in view mode. Annotation is ignored in edit mode.
- **Values**:
  - **<string>**: To group fields the same string value has to be set in the annotation of each field. Two or more data fields are bound when they share the same value for this annotation. The group of coupled data fields is shown only if all of them have values set.
- Removed in ConSol CM version 6.11.0.1.

fieldsize

- **Type**: layout
- **Description**: Displayed field size within the ticket layout.
- **Values**:
  - **<rows>;<cols>**: Displayed field size for textareas

    For STRING fields with `text-type` = `textarea`: rows;cols (corresponds to <textarea rows="" cols="">).
  - **<number>**: Displayed field size for strings and numbers

    For STRING fields with `text-type` other than `textarea` and NUMBER fields: n indicates the number of characters in the fields; for string fields this is the number of monospaced capital M characters.

    For ENUM data fields: Defines how many values are directly visible in the list box. Used only for layout purposes. This annotation is not relevant for ENUM autocomplete fields.

ⓘ This is only a layout configuration, for validation use `maxlength` of type `validation`.

enum field with ticket color

- **Type**: ticket display
- **Description**: Defines the background color of the ticket icon for ticket list and ticket.
- **Values**:
  - *true / false*: The field has to exist within Enum Administration where lists, values, and colors are defined.

accuracy

- **Type**: validation
- **Description**: For ticket, customer and resource fields of type DATE. To define the level of detail displayed
- **Values**:
  - *date (default)*: Show date without time.
  - *date-time*: Show date with time.
  - *only-time*: Show only time, no date.

format

- **Type**: validation
- **Description**: Used for validating the format of date fields.
- **Values**:
  - *<date format>*: The pattern for the date is based on `SimpleDateFormat`, e.g., dd.MM.yyyy.
    Remember to set the proper colspan when including hours/minutes in the format. See http://docs.oracle.com/javase/6/docs/api/java/text/SimpleDateFormat.html for the format reference.

This annotation is applied to date fields both in the Web Client and in CM/Track V2. If it is not set, the standard date format of the browser locale is used in the Web Client, and the German standard date format (dd.MM.yyyy) is used in CM/Track V2.

maxLength

- **Type**: validation
- **Description**: Defines the maximum length of input for STRING data fields.
- **Values**:
  - **<number>**: May be used with STRING data fields.

minLength

- **Type**: validation
- **Description**: Defines the minimum length of input for STRING data fields.
- **Values**:
  - **<number>**: May be used with STRING data fields.

required

- **Type**: validation
- **Description**: Indicates that this is a required field.

- **Values**:
  - *true / false*: Field is required if value is set to “true”. The user cannot save the ticket without having entered a value in a required field. In the Web Client, required fields are marked by a red asterisk.

C.2 Managing Sorted Lists: Enum Administration

This chapter discusses the following:

C.2.1 Introduction

Enums, also called sorted lists, are lists with predefined list values. You define an enum in the navigation item *Enums* from the navigation group *Tickets*. Enums are defined once and can be used in several places:

- as a selection list (in drop-down menus) for ticket fields, customer fields, or resource fields of type ENUM
- as hierarchical lists for ticket fields, customer fields, or resource fields of type *MLA field* (Multi Level Attributes, see section MLA Administration)
- as dependent enums, i.e., as enums that form a hierarchy, a data construct implemented by Scripts of Type Dependent Enum

You only define the lists, i.e., the structures with various list values, in the enum administration. To display the enum in the Web Client (as ticket field values, customer field values, or resource field values), you have to complete one of the following steps:

- Create a ticket field of type *enum* and assign the respective enum there.
- Create a customer field of type *enum* and assign the respective enum there.
- Create a resource field of type *enum* and assign the respective enum there.
- Create an MLA which is linked automatically as ticket field to the ticket field group, as customer field to the customer field group, or as resource field to the resource field group that has been indicated during MLA set-up.
- Create a dependent enum script and assign it to a ticket field group, customer field group, or resource field group.

Examples:

A list of country names (Germany, Italy, France, etc.) is used in the customer field *Country* belonging to an address data set. The same list can also be used in the ticket field *Machine location* and in other data fields. Priority lists (high, normal, low, etc.) are other typical examples.

Depending on the value of the `enum-type` annotation, an enum field is displayed in the Web Client as follows:

- **drop-down menu**
  enum-type not set or enum-type = select
- **radio buttons**
  enum-type = radio

- **self-completing (autocomplete) list**
  enum-type = autocomplete

If the annotation `enum-type` is not set, a drop-down menu is displayed by default (see example in the picture below).

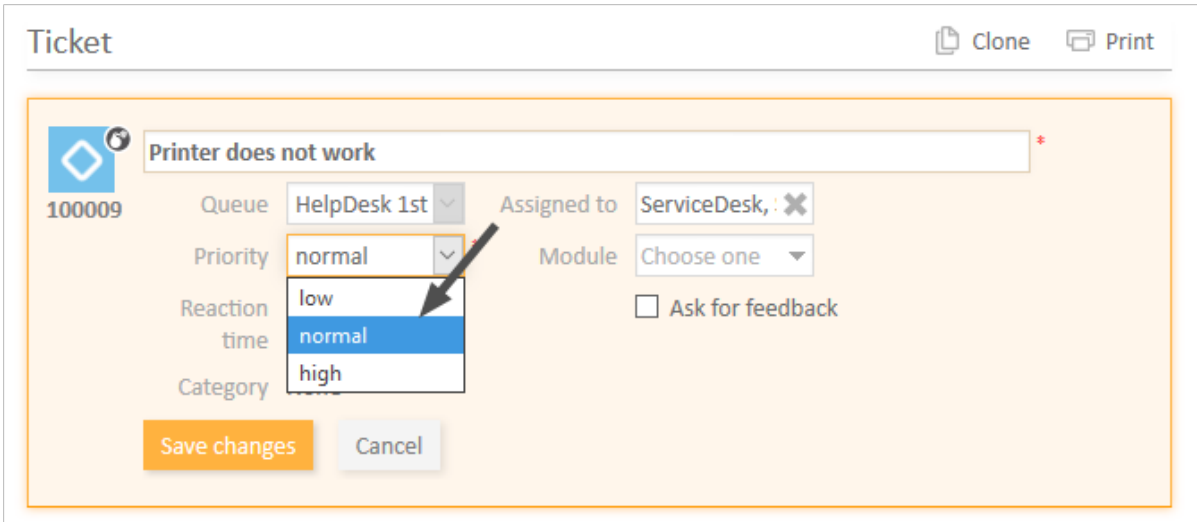


Figure 65: *ConSol CM Web Client - Enum for Priority (localized enum values displayed as list items)*

C.2.2 Enum Administration Using the Admin Tool

The following figure shows the available sorted lists. You reach this screen by opening the navigation item *Enums* in the navigation group *Lists*.

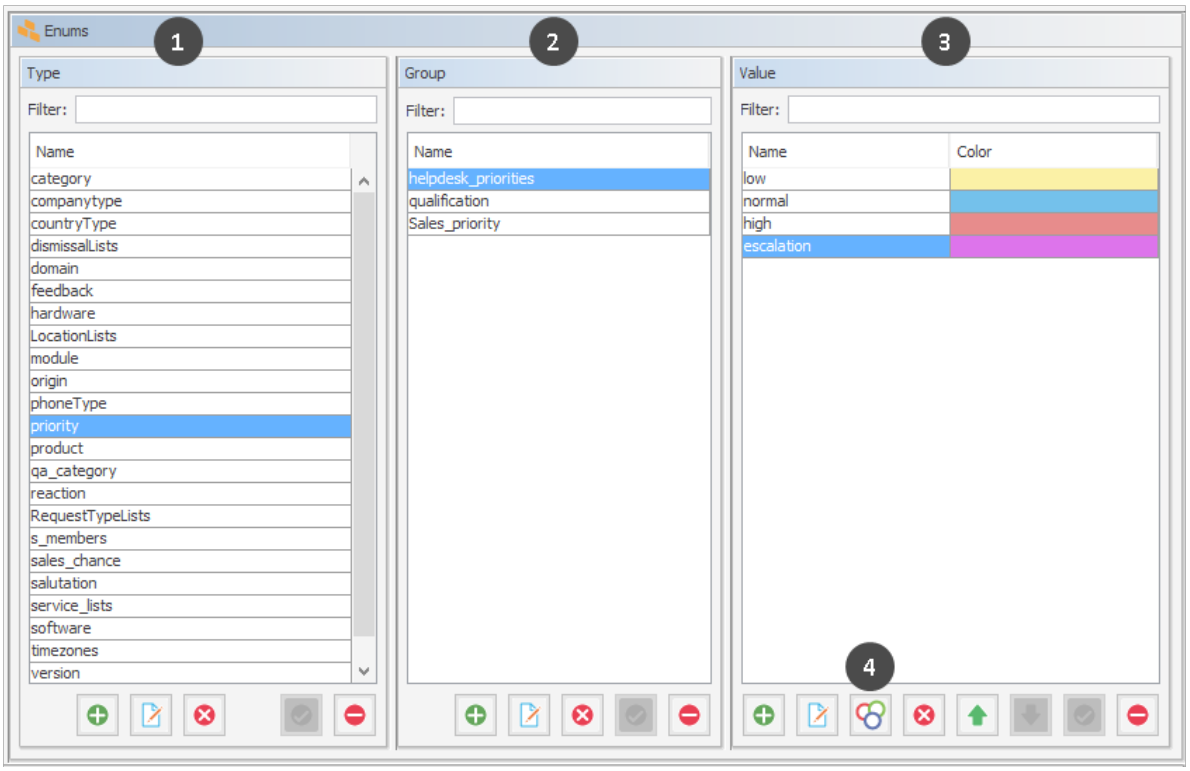


Figure 66: *ConSol CM Admin Tool - Lists, Enums*

Enums are organized in three levels:

- **Type (1)**
  The *type* helps to organize your lists within the Admin Tool. Its name is never displayed in the Web Client and does not have any other implications.
- **Group (2)**
  The *group* represents a group of enum values, i.e., the list.
- **Value (3)**
  The *value* represents one value within a list. You can pick a color for each value (4), which will be used as a background color for the ticket icon in the defined queue. You can use the up and down arrows to determine the value's position within the list in the Web Client.

C.2.2.1 Enum Types

Create an Enum Type

To create a new enum type just click the *Add* button below the list in the *Type* area on the left of the window. The following pop-up window appears.

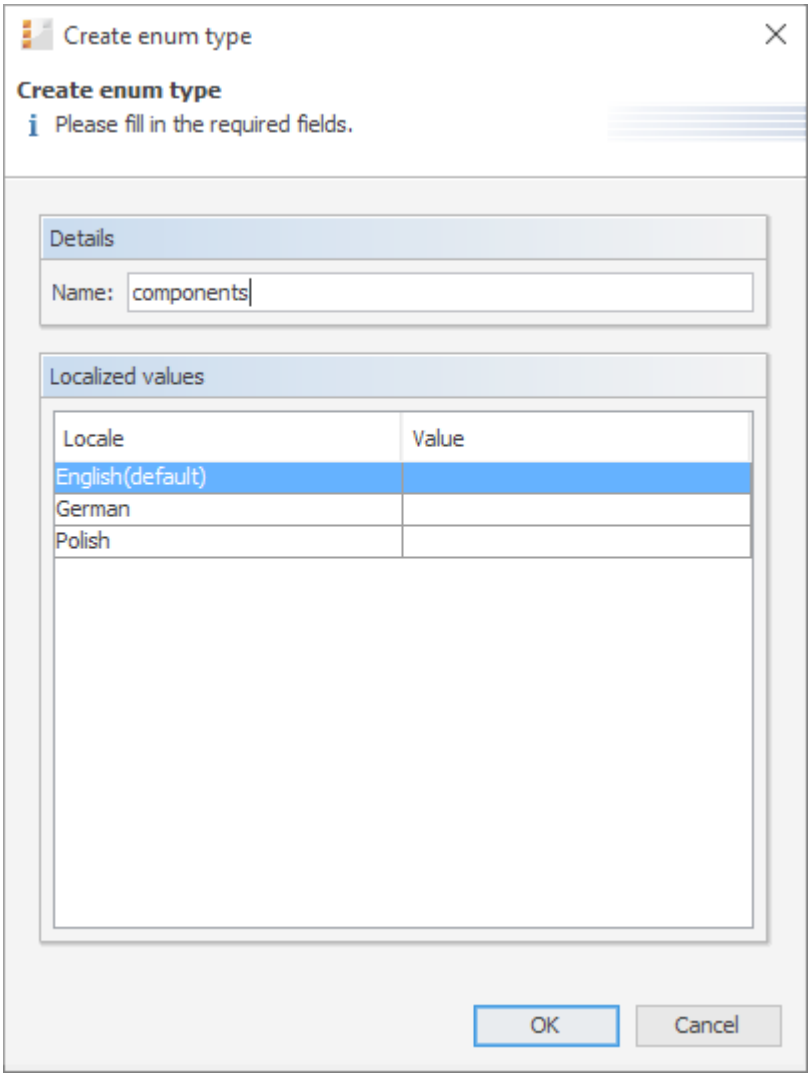


Figure 67: *ConSol CM Admin Tool - Lists, Enums: Creating an enum type*

- **Name:**
  Enter a name for the new enum type. The name must be unique.
- **Localized values:**
  Enter the corresponding type name in the *Value* field for each additional language. For details, please refer to the section Localization of Objects in General, Type 2.

Click *OK* to create the enum type and to close the window.

Edit an Enum Type

If you want to edit an enum type, select it in the list and click the *Edit* button. The same window as described above for creating an enum type will appear. You can modify all fields and save your changes by clicking *OK*.

Delete an Enum Type

An enum type can only be deleted if there are no enum groups for it anymore. Either you have to delete all groups belonging to this type first or you have to assign them to another type. In order to delete an enum type, select it in the list and click the *Delete* button. If you confirm the following dialog with *Yes*, the type will be removed from the list and the system.

Enable or Disable an Enum Type

If you do not want to delete an enum type because you might need it again, you can disable it. To do so select the type and click the *Deactivate* button. The entry in the list is shown in italics afterwards. Just click the *Activate* button below the type list, if you want to enable the type again.

C.2.2.2 Enum Groups

Create an Enum Group

An enum group represents a list, i.e., the enum group is a collection of list (enum) values. All groups of an enum type (i.e., all lists which belong to this type) are created and managed in the middle part of the *Enum Administration* window. To create a new enum group select the desired type on the left, then click the *Add* button below the *Group* area. The following pop-up window appears.

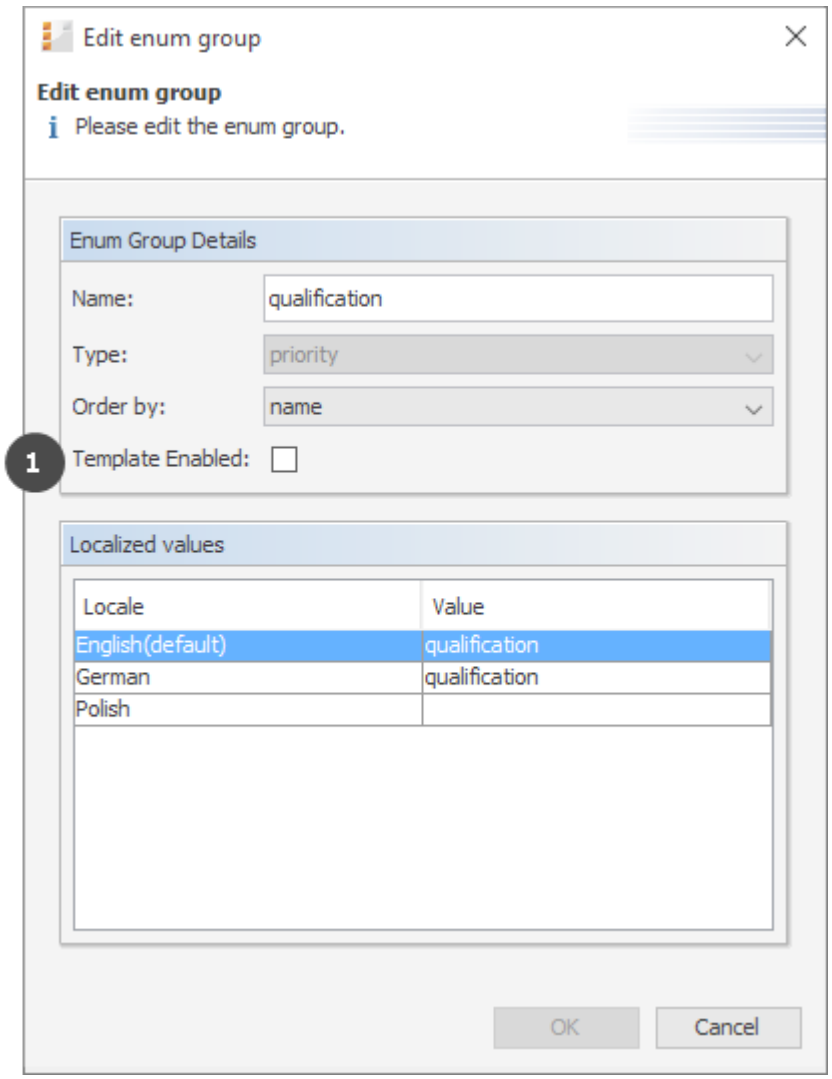


Figure 68: *ConSol CM Admin Tool - Lists, Enums: Creating an enum group*

- **Name:**
  Enter a name for the enum group. The name must be unique.
- **Type:**
  This field shows the selected enum type for the group. You can also choose any other type from the selection list, e.g., if you want to assign the group to a different type.
- **Order by:**
  Here you can define the way the values of a group shall be ordered:
  - **user-defined**
    You can specify the sort order by means of arrow icons below the value list.
  - **name**
    The values will be ordered alphabetically.

- **Template Enabled (1):**
  When this value is set to *true* (checkbox ticked), the enum is offered as *enum parameter* in the Text Template Manager. This is the only module where this value is relevant. If you do not know what to do with the checkbox, leave it empty. Please see section The ConSol CM Text Template Manager, String and Enum Parameters.
- **Localized values:**
  Enter the corresponding group name in the *Value* field for each additional language. For details, please refer to the section Localization of Objects in General, Type 2.

Click *OK* to create the enum group and to close the window.

Edit an Enum Group

If you want to edit an enum group, select it in the list and click the *Edit* button. The same window as described above for creating an enum group will appear. You can modify all fields and save your changes by clicking *OK*.

Delete an Enum Group

An enum group can only be deleted if it is not used in a ticket or an MLA. In order to delete a group, select it in the list and click the *Delete* button. If you confirm the following dialog with *Yes*, the group will be removed from the list and the system.

Enable or Disable an Enum Group

An enum group cannot be disabled if it is still used in an MLA.

If you cannot delete an enum group or if you do not want to delete it, because you might need it again, you can disable it. To do so select the group and click the *Deactivate* button. The entry in the list is shown in italics afterwards. Just click the *Activate* button below the group list, if you want to enable the group again.

C.2.2.3 Enum Values

Create an Enum Value

The individual values of an enum group (i.e., the list values) are created in the right part of the window. Select the desired group and click the *Add* button below the *Value* area. The following pop-up window appears.

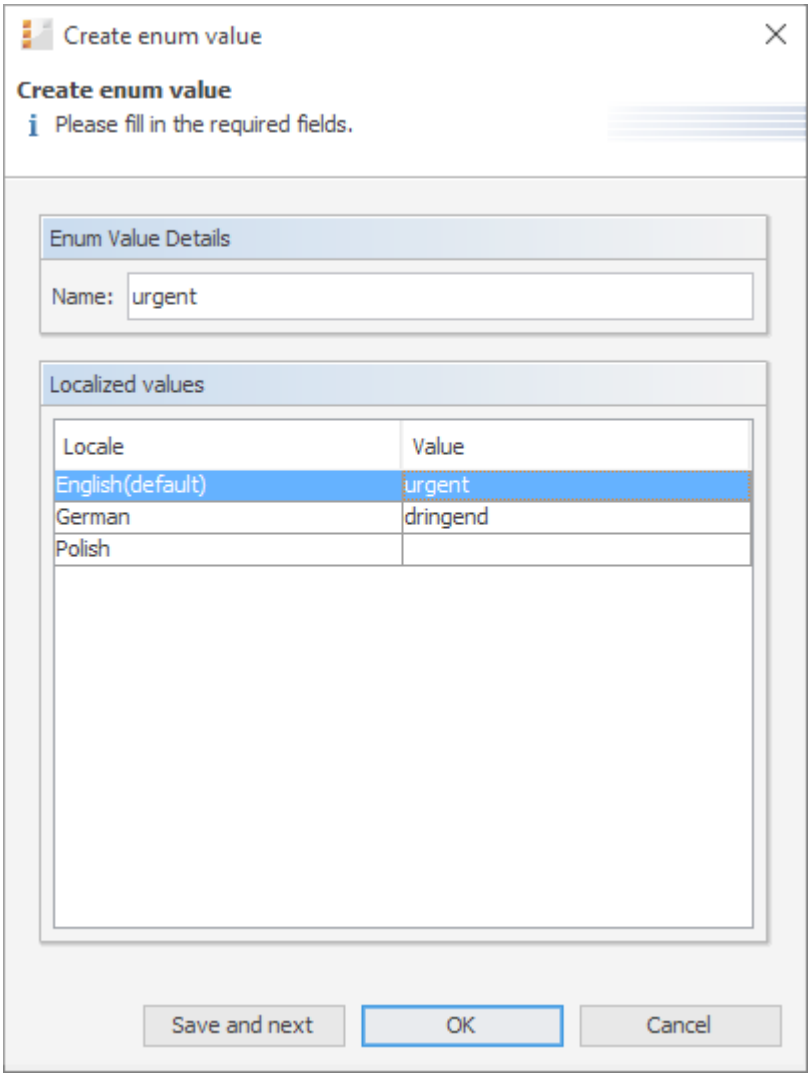


Figure 69: *ConSol CM Admin Tool - Lists, Enums: Creating an enum value*

- **Name:**
  Enter a value which will be displayed in the sorted list on the Web Client.
- **Localized values:**
  Enter the corresponding value name in the *Value* field for each available language. For details, please refer to the section Localization of Objects in General, Type 2.

Click *Save and next* if you want to continue to create values for this enum group. To finish the creation of the list click *OK*.

Edit an Enum Value

If you want to edit an enum value, select it in the list and click the *Edit* button. A pop-up window will appear where you can modify the name and the localized values. Click *OK* to save your changes.

Set a Background Color

You can assign a color to a selected enum value by clicking the *Color* button. This can be used, for example, for priorities. The priority of a ticket in the Web Client can be recognized immediately by the background color of the ticket icon. This will be effective when the respective annotation is set, see the following info box.

> Please note that only one enum can determine the color of the ticket icon for a queue. You have to assign the annotation `enum field with ticket color` to the respective ticket field of type ENUM in the Ticket Field Administration (Setting Up the Ticket Data Model). For example, you can use the ticket field *priority* to determine the ticket icon color in the *helpdesk* queue and the ticket field (enum) *likelihood of closing a deal* in the *sales* queue.

The pop-up window contains a range of colors from which you can choose the desired background color. Click the desired color to set it for the marked list value. You can check the selected color in the *Preview* area. Click *OK* to save your choice. Click *Reset* if you want to restore the last saved color.

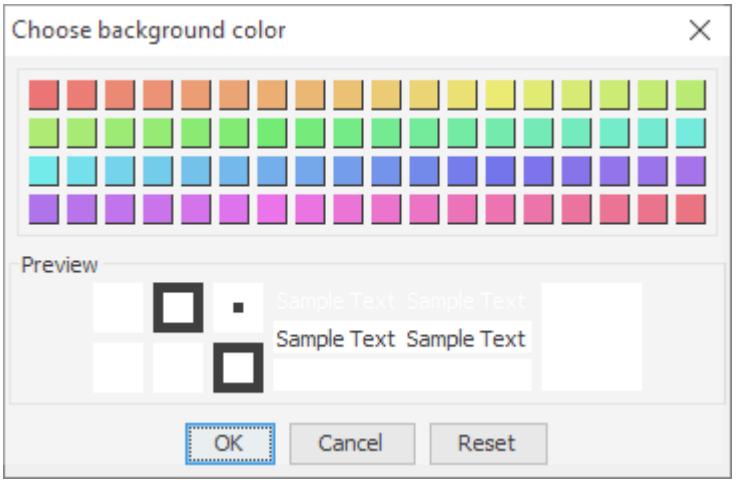


Figure 70: *ConSol CM Admin Tool - Lists, Enums: Setting a background color for an enum value*

Delete an Enum Value

An enum value can only be deleted if it is not used in an MLA. To delete a value, select it in the list and click the *Delete* button. If you confirm the following dialog with *Yes*, the value will be removed from the list and the system.

> Before you delete an enum value, make sure there are no references to it in workflow scripts! This is not checked in the Admin Tool!

Change the Order of the Value List

If you have chosen *user-defined* in the *Order by* field of an enum group, you can arrange the enum values by using the arrow icons below the list. Click the *Move upwards* button to move the selected value one line up resp. click the *Move downwards* button to move it one line down. If you change the

value for *Order by* from *user-defined* to *name*, the enum values will automatically be ordered by name in alphabetical order.

Enable or Disable an Enum Value

An enum value cannot be disabled if it is still used in an MLA.

If you do not want to delete an enum value because you might need it again, you can disable it. To do so, select the value and click the *Deactivate* button. The entry in the list is shown in italics afterwards and the value is not available in the Web Client any longer. Just click the *Activate* button below the value list if you want to enable the value again.

C.2.2.4 Placing an Enum in the Data Model

Enums can be used in ticket fields (i.e. for ticket data), customer fields (i.e. for customer data) and resource fields (i.e. for resource data). The example below shows an enum for ticket data.

Enums for Ticket Data

In order to place an enum in the ticket data model, i.e., to make it available in queues and visible in the Web Client, a ticket field of type ENUM has to be defined. Select ENUM as a data type (1) and select the enum type and group which the enum belongs to (2). Please see section Ticket Field Administration (Setting Up the Ticket Data Model) for a detailed explanation of the work with ticket fields.

Enums for Customer Data

In order to place an enum in the customer data model, i.e., to make it available for company or contact data in the Web Client, a customer field of type ENUM has to be defined. Select ENUM as a data type (1) and select the enum type and group which the enum belongs to (2). Please see section Customer Field Management and GUI Design for Customer Data for a detailed explanation of the work with customer fields.

Enums for Resource Data

In order to place an enum in the resource data model, i.e., to make it available in resources in the Web Client, a resource field of type ENUM has to be defined. Select ENUM as a data type (1) and select the enum type and group which the enum belongs to (2). Please see section CM/Resource Pool - Setting Up the Basic Resource Model for a detailed explanation of the work with resource fields.

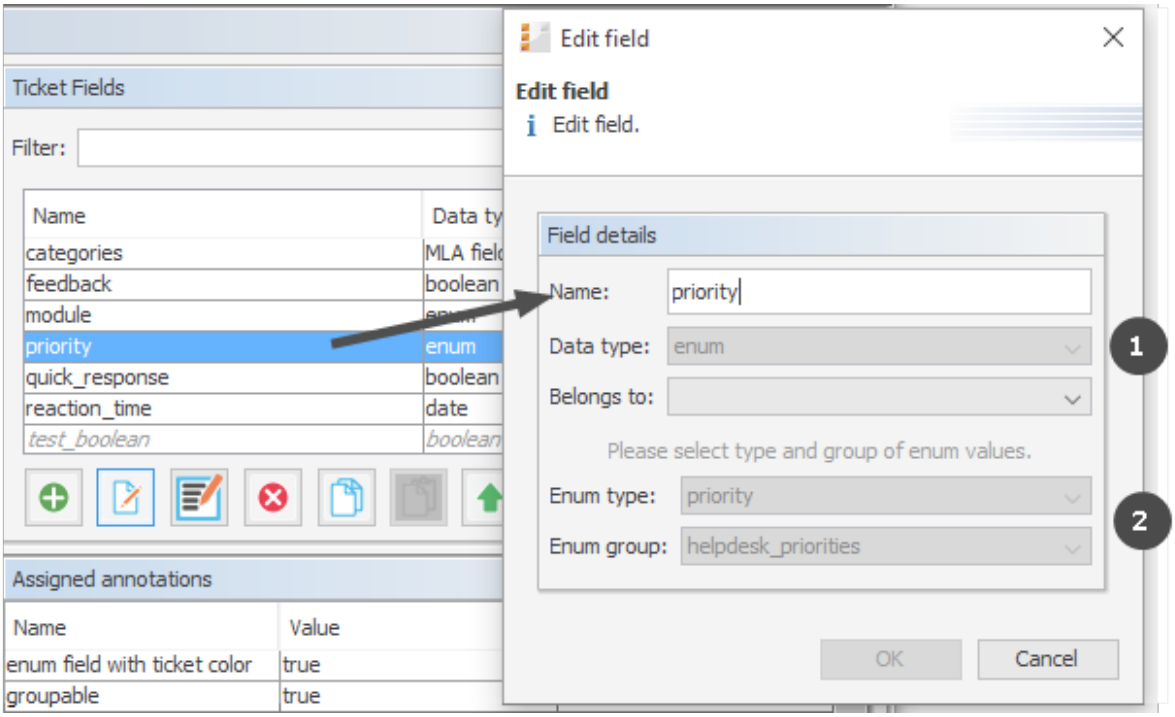


Figure 71: *ConSol CM Admin Tool - Tickets, Ticket Fields: Defining a ticket field of type enum*

C.3 MLA Administration

This chapter discusses the following:

C.3.1 Introduction

MLA is the abbreviation for Multi Level Attribute. An MLA is used to represent a hierarchical data set and consists of several lists which form a tree structure. Each item of a list can lead to a list of the next level with the item name being the name of the subordinate list. An MLA can be used for ticket data, customer data or resource data.

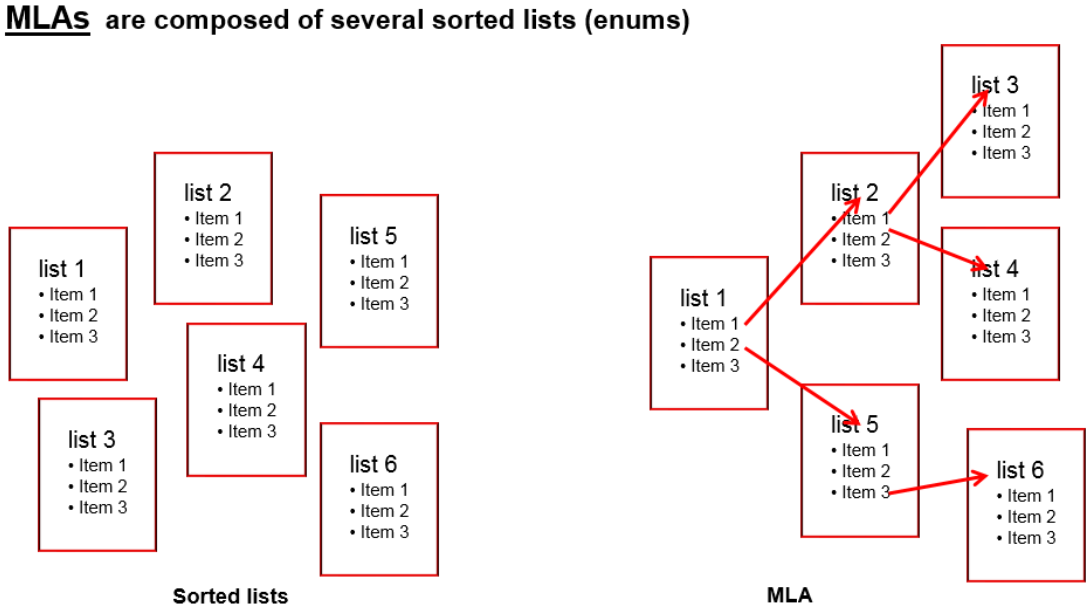


Figure 72: *ConSol CM Admin Tool - MLA construction principle*

Example:

For quality management you need to specify hardware or software products in a ticket. For this purpose you can create an MLA with the name *QA_MLA*. The next step will be to create the first level with the items *Hardware* and *Software*. For each item of a level you can create further levels, e.g. *Graphics Card*, *Monitor*, and *Motherboard* for item *Hardware* and so on. The picture below shows such an MLA in the Web Client.

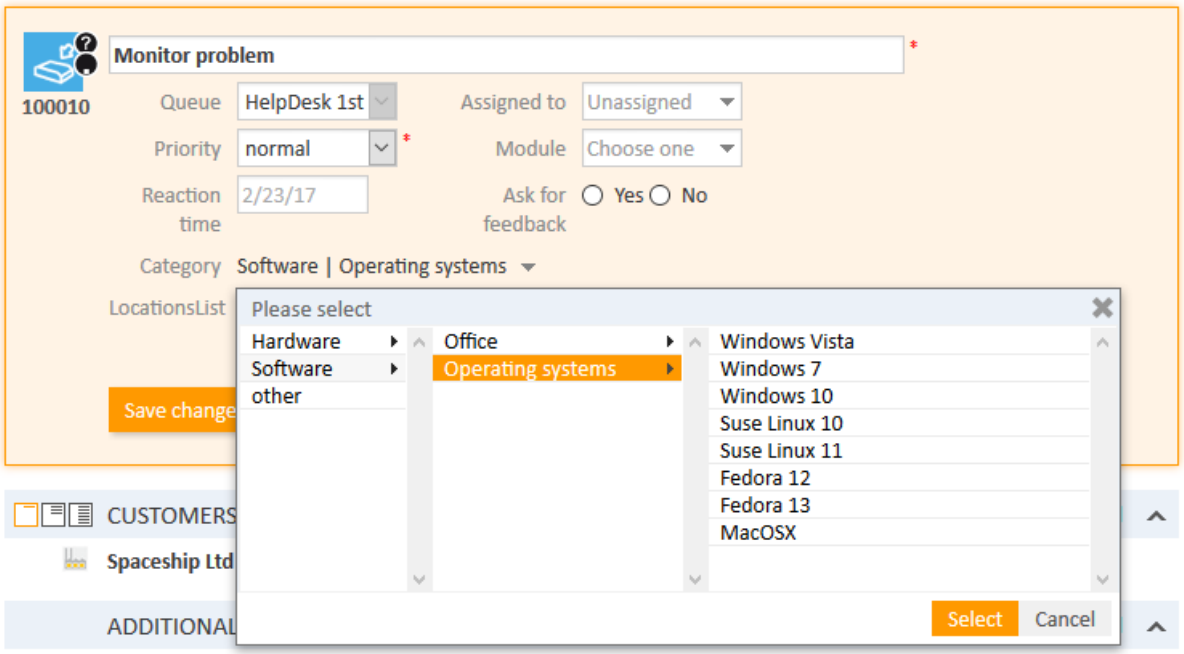


Figure 73: *ConSol CM Web Client - MLA for hardware and software selection*

The sorted lists (enums) for each level of an MLA ...

- can be created within the Managing Sorted Lists: Enum Administration and are only referenced when a new MLA level is defined.
- can be completely created within the MLA administration during the set-up of a new MLA.

C.3.2 MLA Administration Using the Admin Tool

The following figure shows the available MLAs. You reach this screen by opening the navigation item *MLAs* in the navigation group *Lists*.

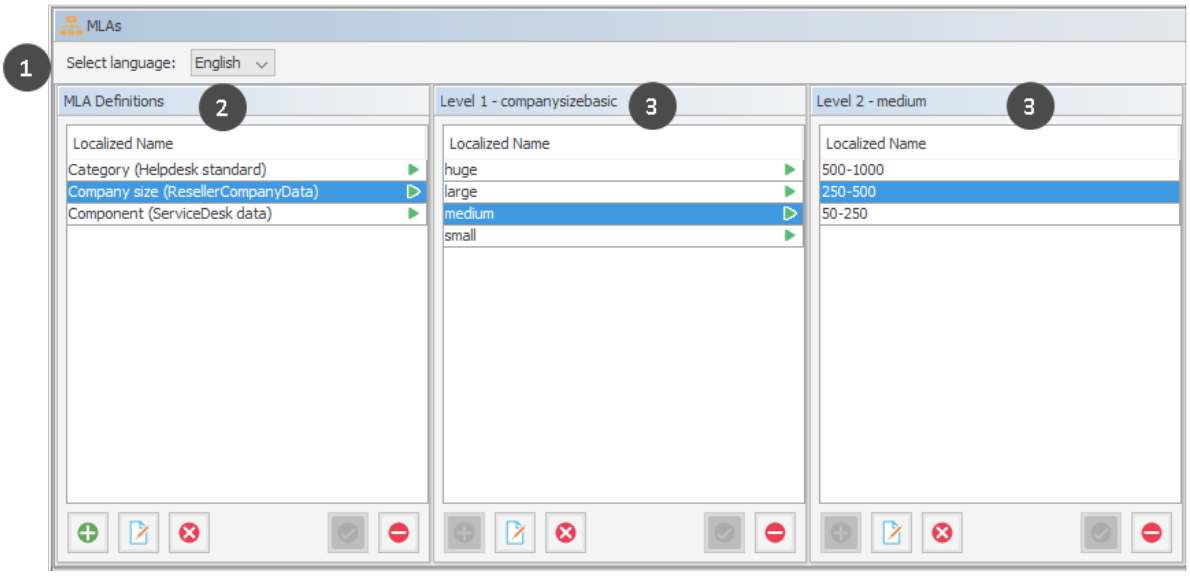


Figure 74: *ConSol CM Admin Tool - Lists, MLAs*

All entries are shown with their localized names (i.e., how they are displayed on the Web Client) in the selected language. You can change the display language of this page by choosing a different locale in the *Select language* field (1) above the list. The first column of the list contains the available MLAs (2). The following columns contain the ENUM values for the respective levels (3).

C.3.2.1 Create an MLA

To create an MLA click the *Add* button below the MLA list in the bottom left corner of the page. The following pop-up window appears.

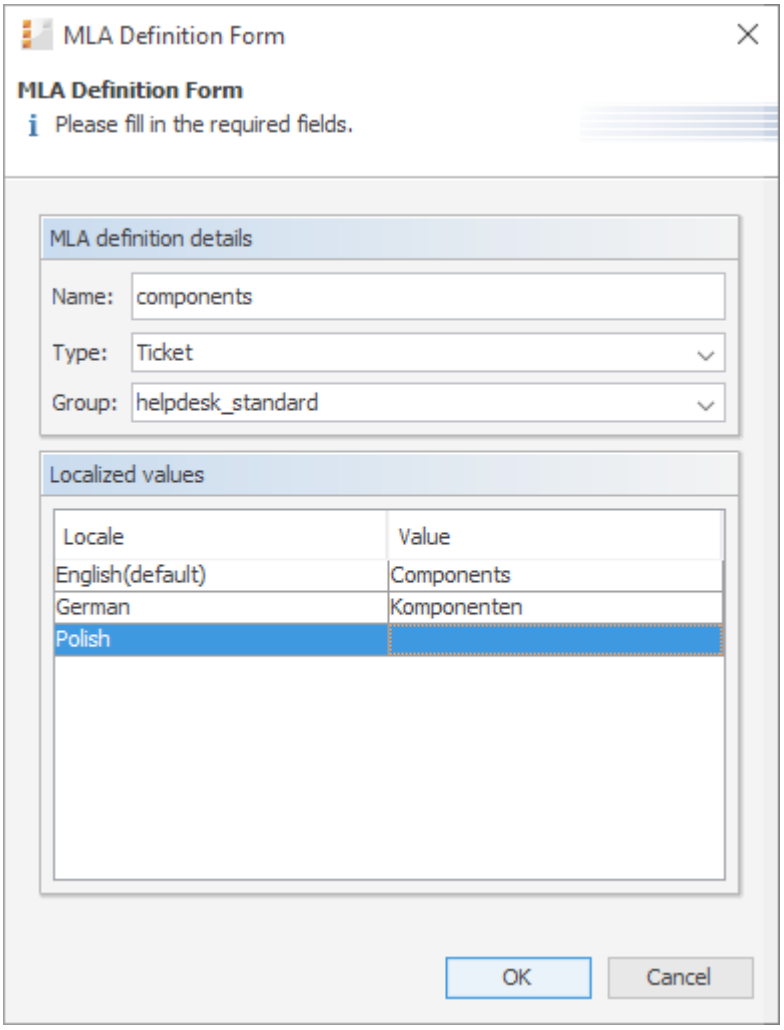


Figure 75: *ConSol CM Admin Tool - Lists, MLAs: Creating an MLA*

- **Name:**
  Enter a name for the new MLA. The name must be unique.
- **Type:**
  - **Ticket**
    MLA will be used in ticket data, i.e., in a ticket field.
  - **Customer object**
    MLA will be used in customer data, i.e., in a customer field.
  - **Resource**
    MLA will be used in resource data, i.e., in a resource field.
- **Group:**
  Choose the required ticket field group (ticket data), customer field group (customer data) or resource field group (resource data) in the list box. For the new MLA a ticket field, customer field or resource field of type *MLA field* will be created automatically in this group. This is necessary to display the MLA in the Web Client. The ticket field, customer field or resource field can

be annotated as described in sections Ticket Field Administration (Setting Up the Ticket Data Model), Customer Field Management and GUI Design for Customer Data, and CM/Resource Pool - Setting Up the Basic Resource Model.

- **Localized values:**
  Enter the corresponding MLA name in the *Value* field for each additional language. For details, please refer to section Localization of Objects in General, Type 2.

Click *OK* to save the details of the new MLA.

> You can also create the ticket field, customer field, or resource field for the MLA first. In this case you will find the localized name of the ticket field, customer field, or resource field already in the list of available MLAs.

Create an MLA Level

Having created a name and a ticket field, customer field, or resource field for the MLA you can go on with the definition of levels. Select the MLA in the list and click the *Add* button below *Level 1*. You will get the *Enum level form* where you can specify an enum for this level.

This menu is only available during the process of creating a new MLA, not when editing an existing MLA.

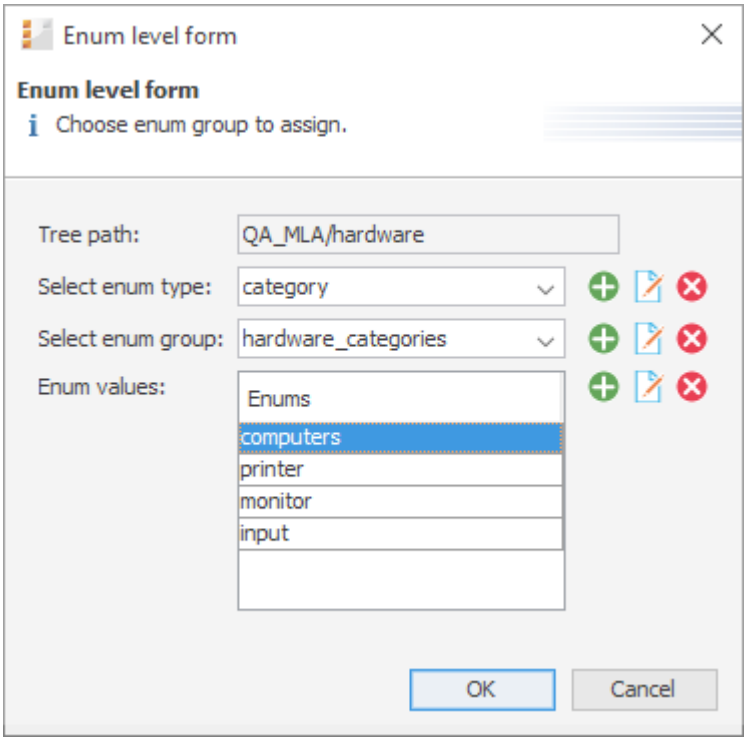


Figure 76: *ConSol CM Admin Tool - Lists, MLAs: Creating an MLA level*

- **Tree path:**
  This field shows the tree path of the new MLA level. Thus you can always see the position of the

level within the MLA. The field is read-only.

- **Select enum type:**
  Choose an enum type from the list to use the corresponding enum groups (which have been created in enum administration first) or create a new enum type directly here in the MLA administration. The new type will also be visible in enum administration afterwards.
- **Select enum group:**
  Choose the desired enum group for this level from the lists in enum administration which are located within the selected type. If you have created a new enum type in the previous step, you also have to create a new enum group. The new enum group will also be visible in enum administration afterwards.
- **Enum values:**
  These are the list values of the new level which will be displayed in the Web Client. You can either take the list as-is or you can enter/add or delete values. The changes will be immediately visible in enum administration. If you have created a new enum group, you also have to create one or more enum values for the new group. You can either create all the enums you need before you start to define an MLA, or create an enum during the definition of a level in the MLA Administration by clicking the *Add* button next to the respective fields in the window. By clicking the *Edit* button or the *Delete* button, you can also edit or delete enum types, groups, and values here, but please consider that changes will affect other MLAs using the same enum. You cannot delete an enum if it is used in another MLA.

Click *OK* to create the new MLA level and to close the window.

For each value of a level you can create further levels as previously described. Just select the value in the list and click the *Add* button below the next level area to the right.

If you have finished your MLA definition and see that you need an additional value for one of the levels, you have to create that value in the respective enum group within Managing Sorted Lists: Enum Administration.

Please note that an ENUM can only be used once within an MLA. It can be re-used in other MLAs though.

Edit a Level Value

If you want to edit a value of the level, select it in the list and click the *Edit* button. You can change the object name and the localized values but please consider that changes will affect other MLAs using the same enum.

Delete a Level

A level can only be deleted if it is not used in a ticket. In order to delete it click the *Delete* button below the respective level. If you confirm the following dialog with *Yes*, the level and all its dependent levels will be removed from the list and the system.

Enable or Disable a Level

If you cannot delete a level, or if you do not want to delete it because you might need it again, you can disable it. Just click the *Deactivate* button below the respective level. The level values (including the values of dependent levels) are shown in italics afterwards. Click the *Activate* button if you want to enable the level again.

C.3.2.2 Edit an MLA

If you want to edit an MLA, select it in the list and click the *Edit* button. The same window as described above for creating an MLA will appear. You can modify all fields except the ticket field group, customer field group, or resource field group. Click *OK* to save your changes.

C.3.2.3 Delete an MLA

You can only delete an MLA if it has not been used. If it has been used, you get a warning stating you can only disable this MLA (see below). In order to delete an MLA select it in the list and click the *Delete* button. If you confirm the following dialog with *Yes*, the MLA (and the ticket field within *Ticket Field Administration*, customer field within *Customer Data Model*, or resource field within *Resource Model*) will be removed from the list and the system.

C.3.2.4 Enable or Disable an MLA

If you cannot delete an MLA, or if you do not want to delete it because you might need it again, you can disable it. To do so select the MLA and click the *Deactivate* button. The entry in the list is shown in italics afterwards. A disabled MLA will not be displayed in the Web Client. Just click the *Activate* button below the MLA list if you want to enable the MLA again.

C.4 Ticket History

This chapter discusses the following:

C.4.1 Introduction

For each ticket, the ticket history is stored in the ConSol CM database, i.e., the entire life cycle of a ticket can be traced. This comprises actions like workflow activities or incoming emails as well as automatic and system actions like the change of the responsible engineer or to check if a time-based trigger has to fire. For those of you familiar with ConSol CM version 5: this is the equivalent to the ticket protocol.

The ticket history is displayed in the *History* section on the ticket page.

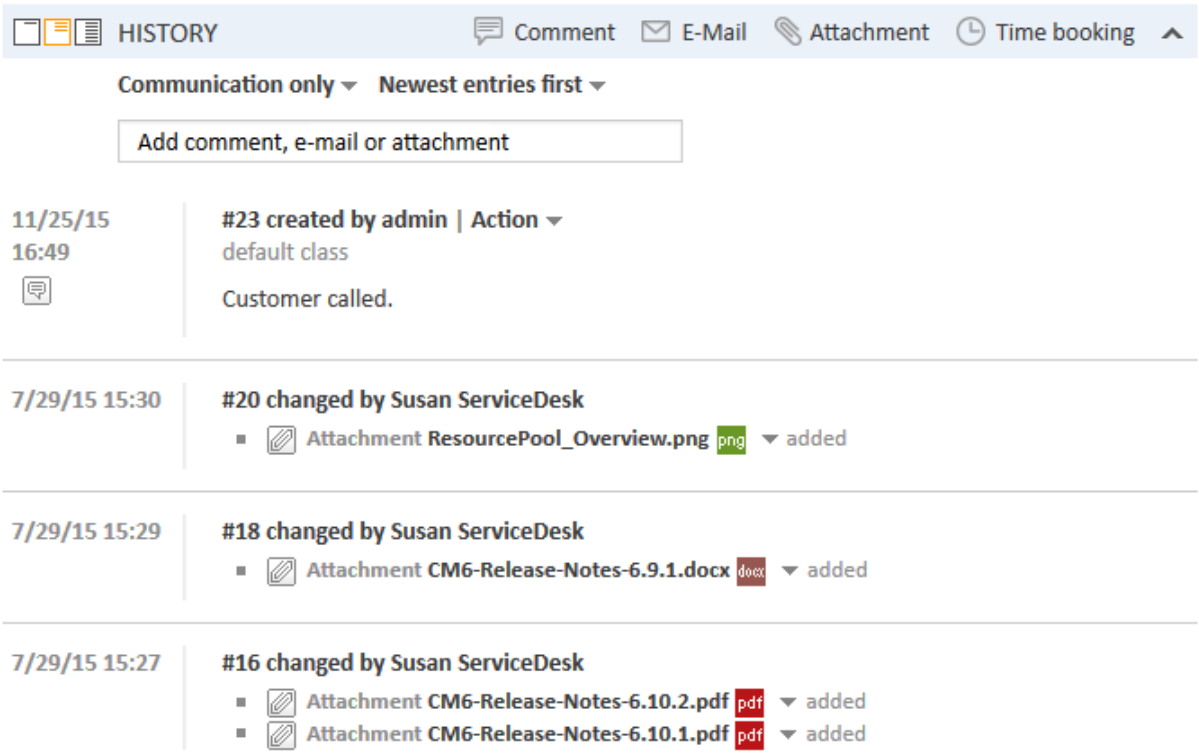


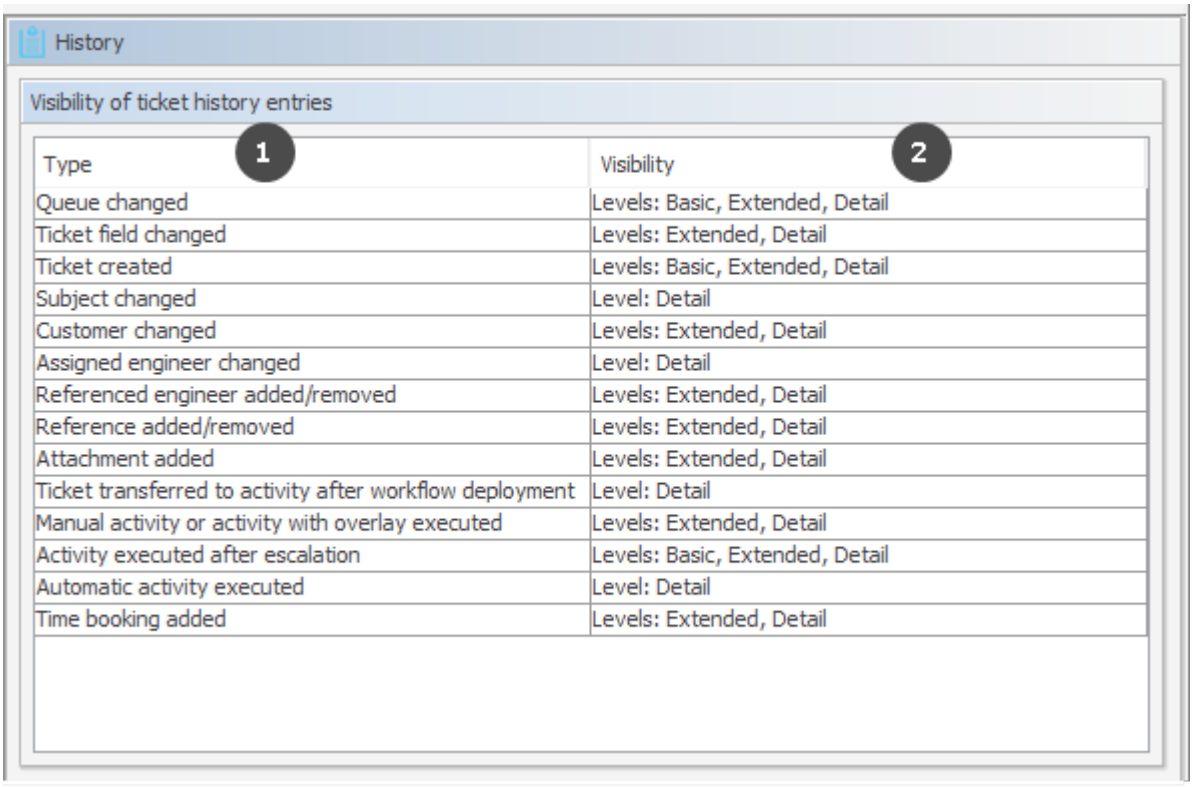
Figure 77: *ConSol CM Web Client: History section*

In case the DWH is in use (see section Data Warehouse (DWH) Management), you can configure if the history data should be transferred to the DWH, see section Ticket History Storage and Transfer to the Data Warehouse (DWH)on this page.

C.4.2 Display Modes of Ticket History in the Web Client

The display mode for the ticket history can be configured on the navigation item *History* in the navigation group *Tickets*. On this item you can configure the visibility level for each action or event that has taken place concerning a ticket. The entries of the indicated type(s) will be visible in the ticket history if the user has selected the respective visibility level. This is of importance if the display mode *Display all entries* is used.

The visibility of emails, comments and attachments (i.e., their visibility level and if the text entries are displayed full or short) can be controlled by Classes of Text. The visibility of entries with no class of text assigned to them is determined by the default class of text (in a default installation of ConSol CM, the default class for emails and comments is named *default_class* and the default class for attachments is named *default_attachment_class*).



Type	Visibility
Queue changed	Levels: Basic, Extended, Detail
Ticket field changed	Levels: Extended, Detail
Ticket created	Levels: Basic, Extended, Detail
Subject changed	Level: Detail
Customer changed	Levels: Extended, Detail
Assigned engineer changed	Level: Detail
Referenced engineer added/removed	Levels: Extended, Detail
Reference added/removed	Levels: Extended, Detail
Attachment added	Levels: Extended, Detail
Ticket transferred to activity after workflow deployment	Level: Detail
Manual activity or activity with overlay executed	Levels: Extended, Detail
Activity executed after escalation	Levels: Basic, Extended, Detail
Automatic activity executed	Level: Detail
Time booking added	Levels: Extended, Detail

Figure 78: *ConSol CM Admin Tool - Tickets, History*

The editing panel for the ticket history shows a list of all configured values, each with:

- **Type (1)**
  The type of action that has been performed.

- **Visibility (2)**
  The visibility level in the Web Client. There are three levels:
  - Basic (1st level)
  - Extended (2nd level)
  - Detail (3rd level)

The following figures show the action type *time booking added* configured for *Basic* and *Extended*.

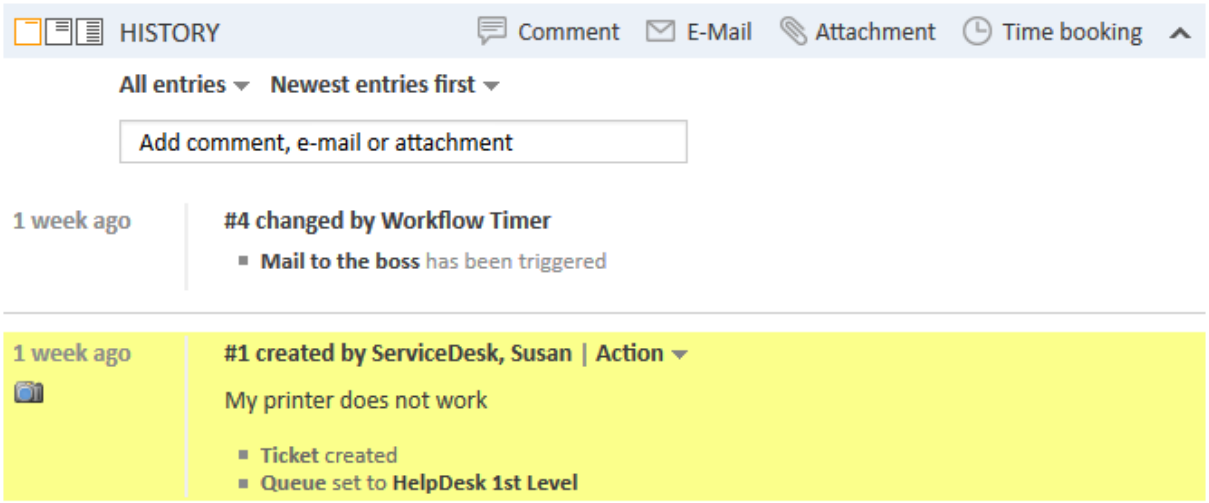


Figure 79: *ConSol CM Web Client - Time booking entry not visible on Basic level*

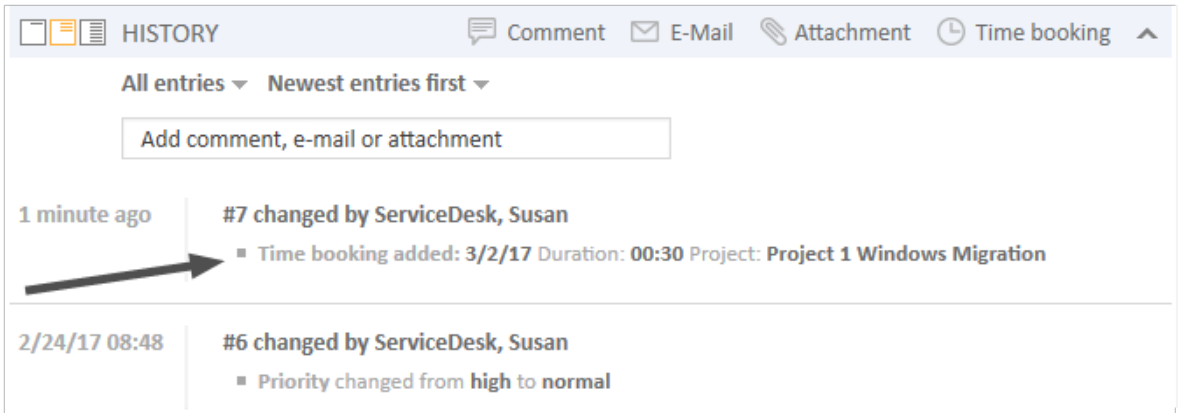


Figure 80: *ConSol CM Web Client - Time booking entry visible on Extended level*

It is not possible to add new action types to the list. To edit the visibility for an existing entry, double-click the visibility value you would like to modify and select the desired option from the drop-down menu.

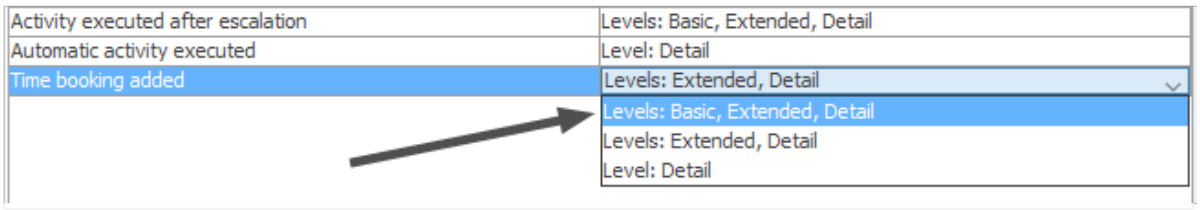


Figure 81: *ConSol CM Admin Tool - Tickets, History: Changing the visibility level for an action type*

The next picture shows the visibility for the action type *time booking added* after the setting has been modified on the navigation item *History* in the navigation group *Tickets*:

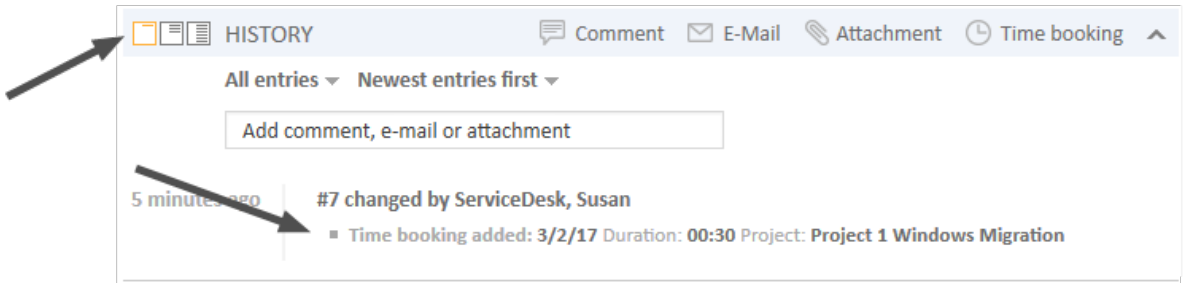


Figure 82: *ConSol CM Web Client - Time booking entry visible on Basic level*

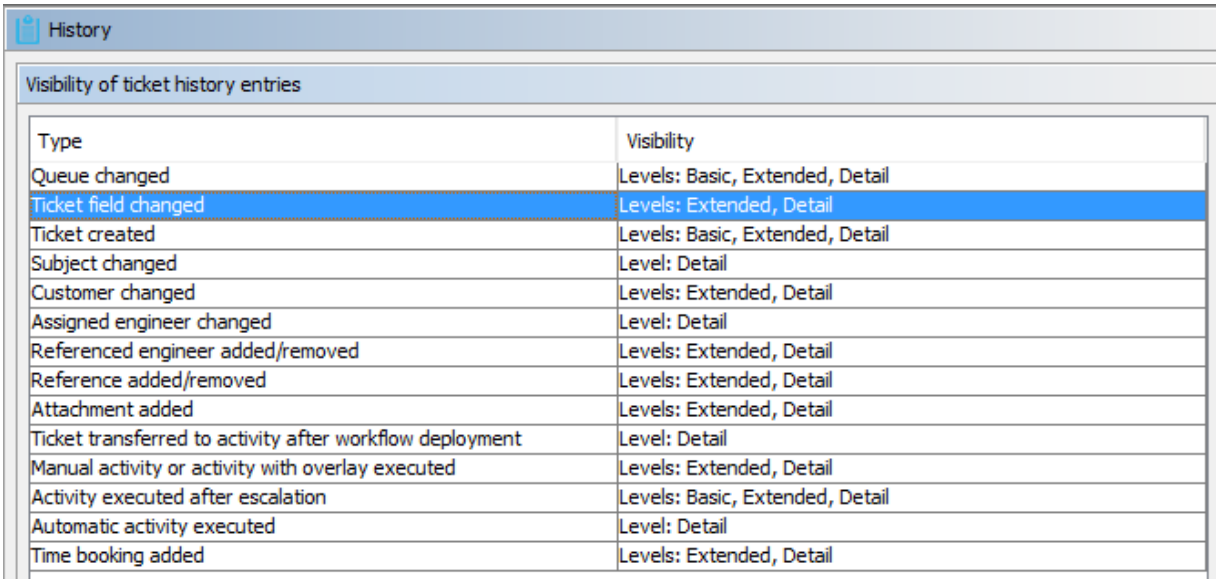
C.4.2.1 Setting the Mode for the Display of Ticket Field Changes in the Ticket History

Please note that the visibility level in the history can be set

- globally, using the visibility parameter *Ticket field changed*
- specifically for one or more ticket fields using the annotation `visibility configuration`

> Please note that the visibility of attachments is configured only here (*Attachment added*). Text classes (see section Classes of Text) are not taken into consideration for attachments.

In the following example, changing a ticket field (i.e. setting a new value) will be displayed in the *Extended* and *Detail* levels of the ticket history. This is the system-wide setting done in the navigation item *History*, navigation group *Tickets*.



Type	Visibility
Queue changed	Levels: Basic, Extended, Detail
Ticket field changed	Levels: Extended, Detail
Ticket created	Levels: Basic, Extended, Detail
Subject changed	Level: Detail
Customer changed	Levels: Extended, Detail
Assigned engineer changed	Level: Detail
Referenced engineer added/removed	Levels: Extended, Detail
Reference added/removed	Levels: Extended, Detail
Attachment added	Levels: Extended, Detail
Ticket transferred to activity after workflow deployment	Level: Detail
Manual activity or activity with overlay executed	Levels: Extended, Detail
Activity executed after escalation	Levels: Basic, Extended, Detail
Automatic activity executed	Level: Detail
Time booking added	Levels: Extended, Detail

Figure 83: *ConSol CM Admin Tool - Tickets, History: Visibility configuration for ticket field changed*

For the ticket field *volume_product*, the visibility configuration is set to *on every level*. In this way, the change of this ticket field would always be visible in the ticket history, no matter which visibility level the engineer has selected (*Basic*, *Extended*, or *Detail*). This setting is done in the navigation item *Ticket Fields*, navigation group *Tickets*.

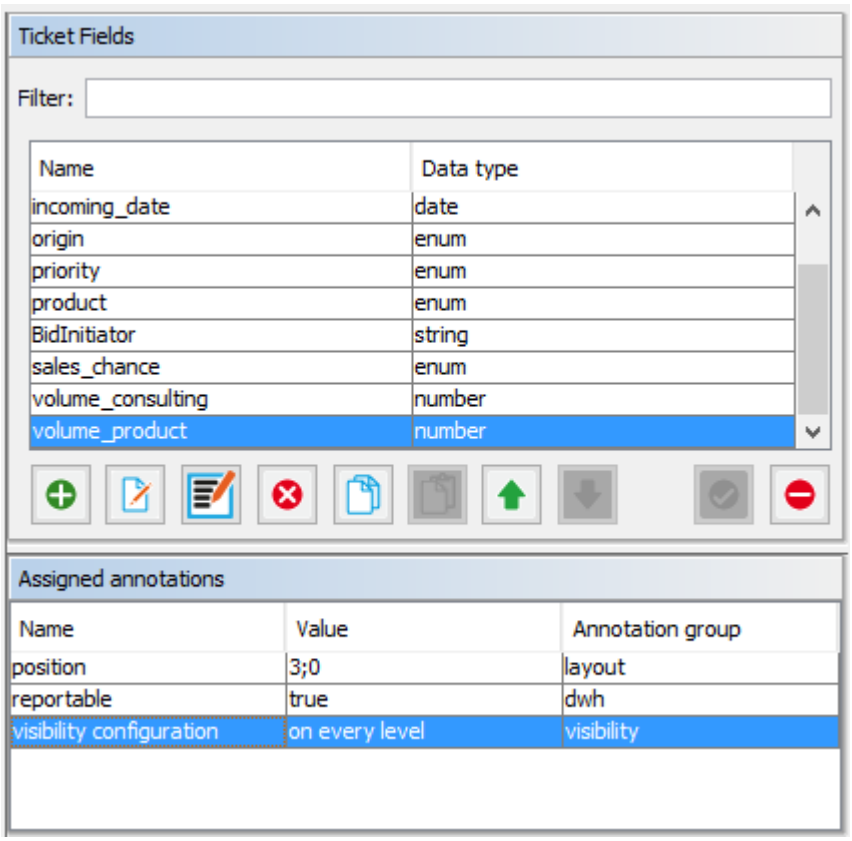


Figure 84: *ConSol CM Admin Tool - Tickets, Ticket Fields: Setting the visibility configuration for one specific ticket field*

C.4.2.2 Modifying the Display Behavior for the Basic and Extended Level to Optimize Ticket History Display

Tickets with a great number of history entries, e.g. comments, might be too large to get a quick overview of all entries. You can use the *lazy loading* feature (see Page Customization attributes *headHistoryElementsCount* and *tailHistoryElementsCount*) to reduce the required space of a ticket.

You can also modify the number of lines which are displayed for comments in the basic and extended visibility level. This can be done for ticket history entries which are marked with a class of text which applies to the first two "short" levels, i.e. for classes of text with the visibility *Basic (short)* and *Extended (short)*.

The configuration is done using the two page customization attributes of type *acimSection*, scope *ticketEditPage/acimSection*, see acimSection (Type or Subscope).

- **basicViewCharactersLimit** (formerly **standardViewCharactersLimit**)
  for Basic (short) level. Defines the number of characters displayed, default 150.
- **extendedViewCharactersLimit**
  for Extended (short) level. Defines the number of characters displayed, default 350.

In the following example, both view limits have been set, but only when the level is short (according to the class of text, here: *default_class*), the limit is applied.

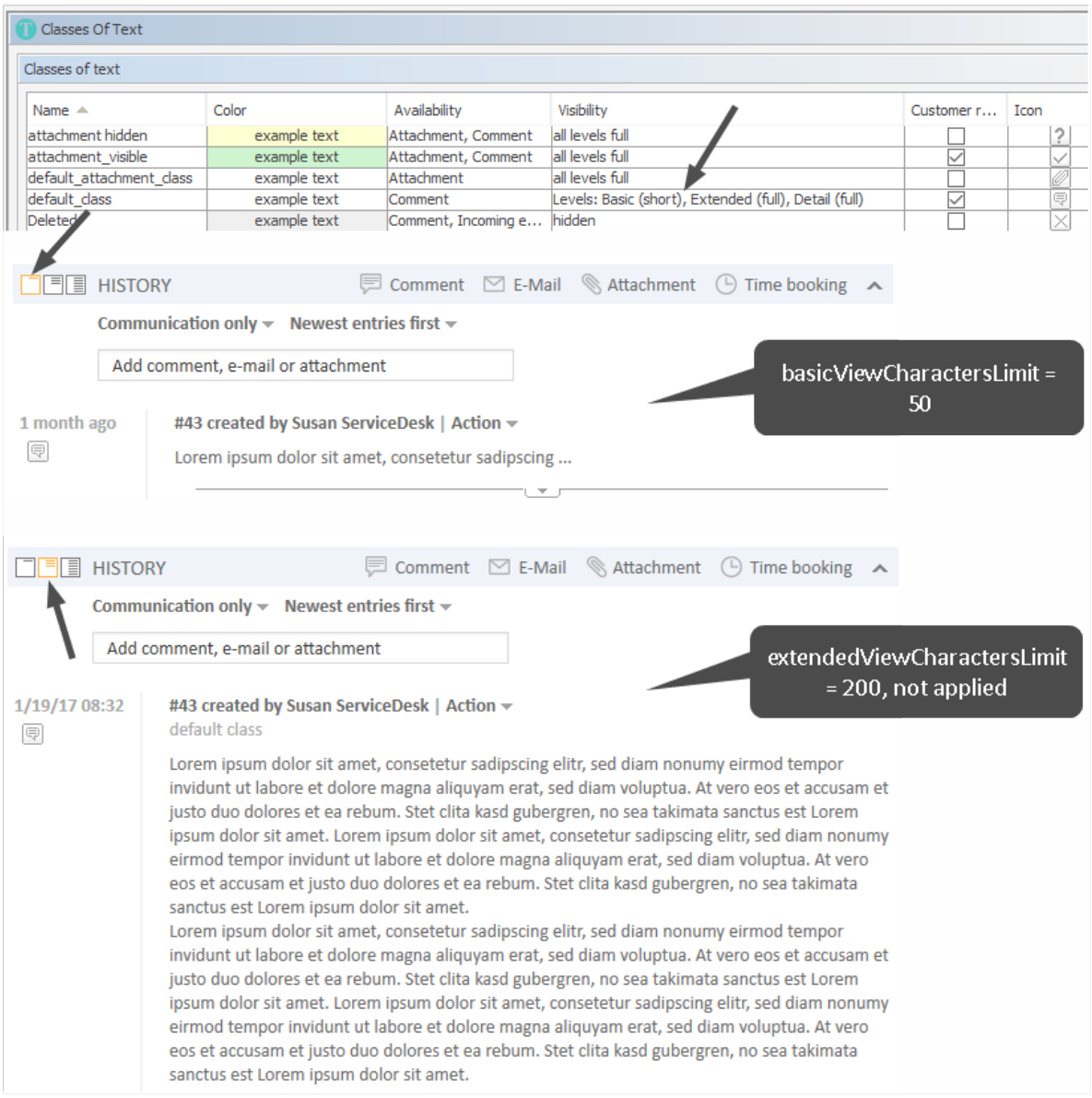


Figure 85: *Configuration of comment display using a class of text plus page customization (ViewCharactersLimit), 1*

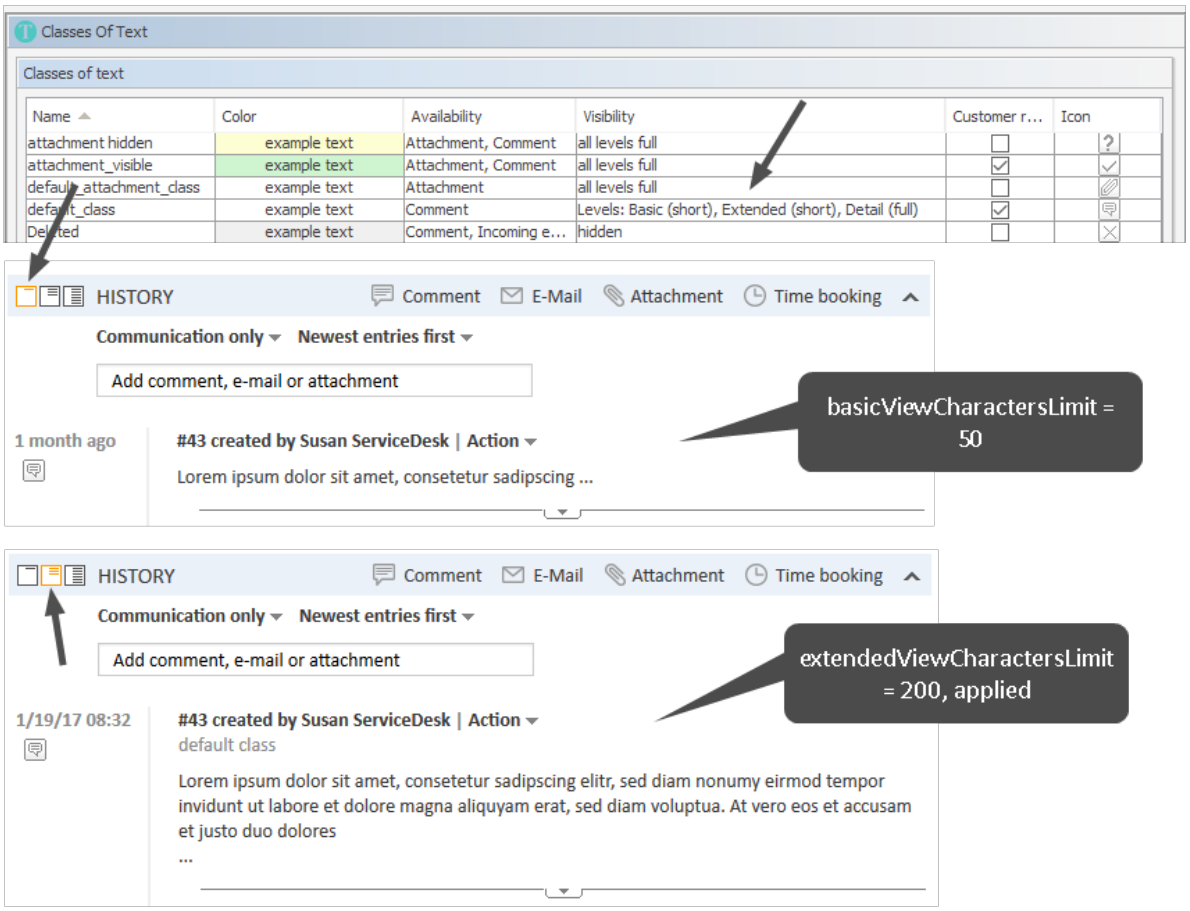


Figure 86: *Configuration of comment display using a class of text plus page customization (ViewCharactersLimit), 2*

C.4.3 General Information about the Visibility of Ticket History Entries in the Web Client

There a **two visibility settings** for ticket history entries:

- The chosen display mode
- The chosen visibility level (including the additional information, whether entries should be displayed *full* or *short* within the chosen visibility level).

There are **two kinds of ticket history entries** for which you can define the visibility settings:

- **Comments/Communication**
  Refers to comments, emails and attachments. Their *visibility level*, and whether the text entries are displayed *full* or *short*, can be controlled by classes of text. The visibility of entries with no class of text assigned to them is determined by the *default class of text* (in a default installation of ConSol CM, the default class for emails and comments is named *default_class* and the default class for attachments is named *default_attachment_class*).
- **Other entries**
  For example, entries about changes to a ticket's queue or scope, assigning and unassigning engineers, or the execution of automatic activities. The full list of all possible types of *other entries* can be found in the Admin Tool in the navigation item *Ticket History*, where you can also set the *visibility level* for this entries.

C.4.3.1 The Display Mode in the Web Client

The display mode determines which kinds of ticket history entries are generally shown in the ticket history. Like the two kinds of ticket history entries (*comments/communication* and *other entries)*, there are two display modes for the ticket history:

- Display mode *Display communication* (shows communication entries only)
- Display mode *Display all entries* (shows communication and all other entries)

Display mode *Display communication*

Shows only comments, emails, and attachments (not the full attachment, but an entry that an attachment has been added and a link to open the attachment). Whether a communication entry is shown, and how much of its contents are shown, depends an the selected visibility level (including the additional information, e.g., whether entries shall be displayed in *full* or *short* forms within the chosen visibility level).

Display mode *Display all entries*

Shows all other entries in addition the communication entries (e.g., entries about changes to a ticket's queue or scope). The full list of all possible types of *other entries* can be found in the Admin Tool in the navigation item *Ticket History*, where you can also set the *visibility level* for these entries.
These *other entries* are visible only if the display mode *Display all entries* is chosen. After this requirement is met, the selected visibility level determines whether the different kinds of entries are shown. In contrast to communication entries, there is no differentiation between *full* and *short* visibility for these *other entries*.

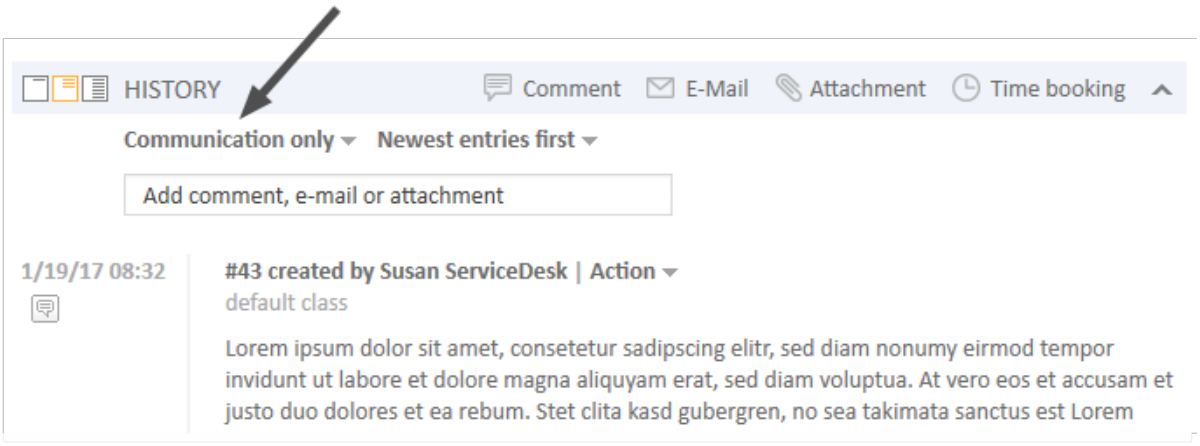


Figure 87: *ConSol CM Web Client - Setting the display mode*

C.4.3.2 The Visibility Level in the Web Client

There are three ticket history levels in the Web Client:

- Basic (1st level)
- Extended (2nd level)
- Detail (3rd level)

Visibility level for communication entries

In which visibility level (Basic, Extended or Detail) communication entries will be shown or hidden in the ticket history is determined by classes of text. You can create and manage them in the navigation item *Classes of Text*. The visibility of entries with no class of text assigned to them is determined by the *default class of text* (in a default installation of ConSol CM, the default class for emails and comments is named *default_class* and the default class for attachments is named *default_attachment_class*).

In addition to the general visibility of a communication entry, you can also determine if an entry of this class should be displayed in full length (*full*) or shortened after a certain number of characters (*short*).

Visibility level for other entries

In which visibility level (Basic, Extended or Detail) other entries will be shown or hidden in the ticket history can be configured in the navigation item Ticket History. In contrast to communication entries, there is no differentiation between *full* and *short* visibility for these other entries.

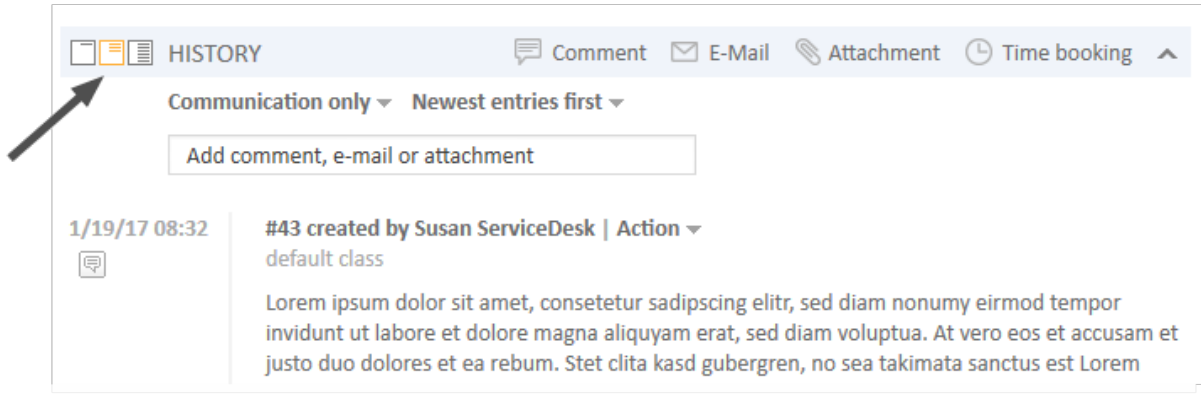


Figure 88: *ConSol CM Web Client - Setting the visibility level*

C.4.4 Ticket History Storage and Transfer to the Data Warehouse (DWH)

In a standard CM installation which uses the Data Warehouse, the ticket history is stored in the CM database and transferred to the DWH.

You can use annotations to prevent the writing of the ticket history into the CM database and to prevent the transfer of ticket history data to the DWH:

Available in CM versions	Group annotation	Field annotation	Triggered behavior
6.6.0 and up	`no-history`	`no-history-field`	For the ticket field groups or single ticket fields with this annotation, no history data will be written into the CM database. So no history data will be available at all - the `dwh-no-history` annotation does not have to be set for the respective ticket fields or ticket field groups.
6.10.2 and up	`dwh-no-history`	`dwh-no-history-field`	For the ticket field groups or single ticket fields with this annotation, no history data will be transferred to the DWH.

Thus, if you want to see the history in the Web Client, but there are good reasons not to transfer all the data to the DWH, use `dwh-no-history` and `dwh-no-history-field`. In case you do not want to write history data at all, e.g., if some internal programming data is written into ticket fields, use the `no-history` and `no-history-field` combination. Of course you can combine the two variants and use different values for each field or field group.

Please note that there might have been changes concerning the history transfer configuration during the life of a CM system. This might result in a CM database which contains (old) history data even though, currently, the annotation `no-history` is set.

Please note the CM behavior during an update to versions 6.10.2 and higher:

To keep old system configurations valid, for all ticket fields and ticket field groups which are annotated with `no-history-field` and `no-history`, the new annotations `dwh-no-history-field` and `dwh-no-history` are added automatically.

C.5 Configuration of the Ticket List

This chapter discusses the following:

C.5.1 Introduction

In the Web Client, the ticket list is displayed on the left hand side of the user interface. The list serves as to-do list for the engineer.

Figure 89: *ConSol CM Web Client - Ticket list*

The configuration of views and the assignment of views to certain roles define which tickets will be displayed in the ticket list. Please see section View Administration for a detailed introduction to the configuration of views performed by the administrator.

C.5.2 Configuration of Grouping and Sorting of the Ticket List

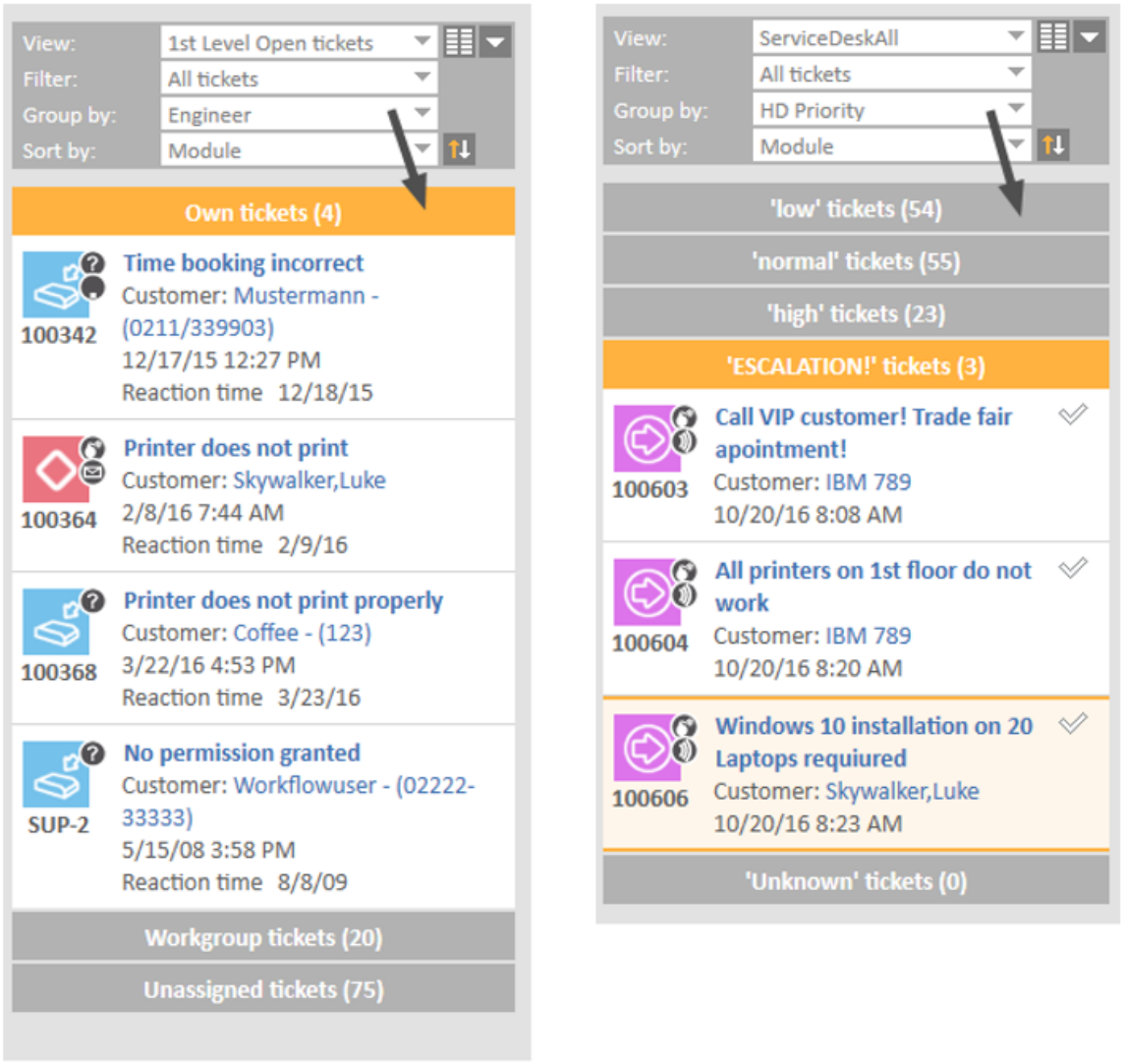
C.5.2.1 Grouping

The ticket list can be displayed in groups. In the image above, the groups are based on which engineer is assigned to the ticket resulting in three groups: *Own tickets*, *Workgroup tickets* and *Unassigned tickets*. The grouping can be changed by the engineer who is working with the Web Client by using the drop-down menu *Group by* in the ticket list configuration.

Three standard values are always offered:

- **No grouping**
  All tickets are shown in one group named *All tickets*
- **Engineer**
  Tickets are divided into three groups. The names of the groups depend on your individual CM configuration, but the groups always serve the same purpose:
  - Tickets assigned to the currently logged in engineer (e.g., Own tickets)
  - Tickets assigned to any other engineer but the currently logged in engineer (e.g., Workgroup tickets)
  - Tickets not assigned to any engineer (e.g., Unassigned)
- **Queue**
  Tickets are divided into as many groups as there are queues in your CM system, but you can only see the groups for queues for which you have at least read permission. The queues are sorted alphabetically by their localized queue names.

In addition to these values, you, as an administrator, can define further parameters by which the ticket list can be grouped. Every ticket field of type *enum* can be configured to be used for this purpose. All you have to do is setting the field annotation `groupable` for the respective ENUM value. In the following example, the annotation has been set for the ticket field *priority* which has the English localization *HD priority*.



Ticket list grouped by engineer

Ticket list grouped by HD priority

Figure 90: *ConSol CM Web Client - Two different groupings of the ticket list (engineer: standard, HD priority: customized)*

In the Admin Tool, the annotation `groupable` is set for the ticket field in the navigation group *Tickets*, navigation item *Ticket fields*.

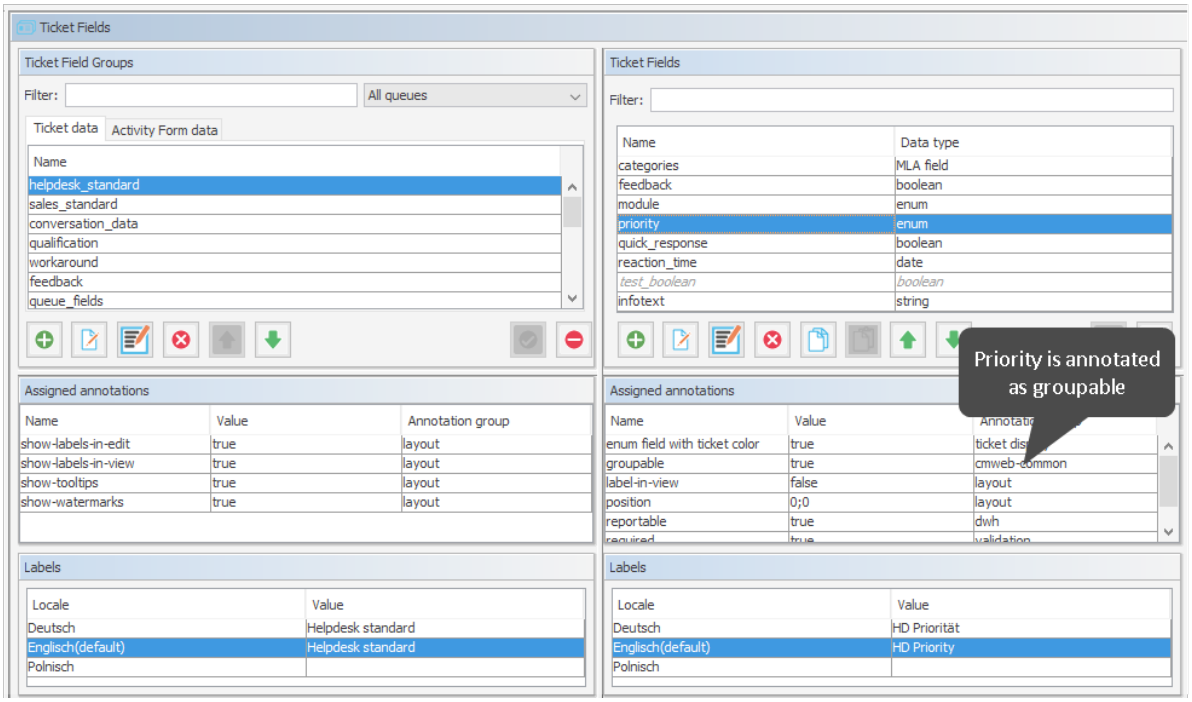


Figure 91: *ConSol CM Admin Tool - Tickets, Ticket Fields: Annotation groupable set for a ticket field*

C.5.2.2 Sorting

Within a group of the ticket list, the tickets can be sorted. The sorting can be changed by the engineer who is working with the Web Client by using the drop-down menu *Sort by* in the ticket list configuration. There are three standard values by which the tickets can be sorted:

- **Scope**
  Sorts the tickets within the groups by the logical order of the scopes they are currently in. The logical order of the scopes is the order of the process steps in your business processes, e.g., Open Ticket - Ticket in progress - Query the technical department - Give solution to customer - Close Ticket.
- **Creation date**
  Sorts the tickets within the groups by their creation date.
- **Modification date**
  Sorts the tickets within the group by the date of the last modification. A modification is one of the following types of event whereby the event can be triggered manually (via Web Client) or automatically (via workflow):
  - ticket field edition
  - details edition
  - addition of additional customer
  - changing a role of customer
  - setting a customer as main customer

- adding a relation
- deleting a relation
- adding an additional engineer
- deleting an additional engineer
- adding a resource relation
- deleting a resource relation
- adding a comment
- sending an email
- reply, forward, retry an email saved on history
- adding an attachment
- deleting an attachment
- adding time booking

In addition to these values, you, as an administrator, can define further parameters by which the tickets within a group can be sorted. Every ticket field of type *enum* can be configured to be used for this purpose. All you have to do is setting the field annotation `sortable` for the respective ENUM value. In the following example, the annotation has been set for the ticket field *priority* which has the English localization *HD priority*.

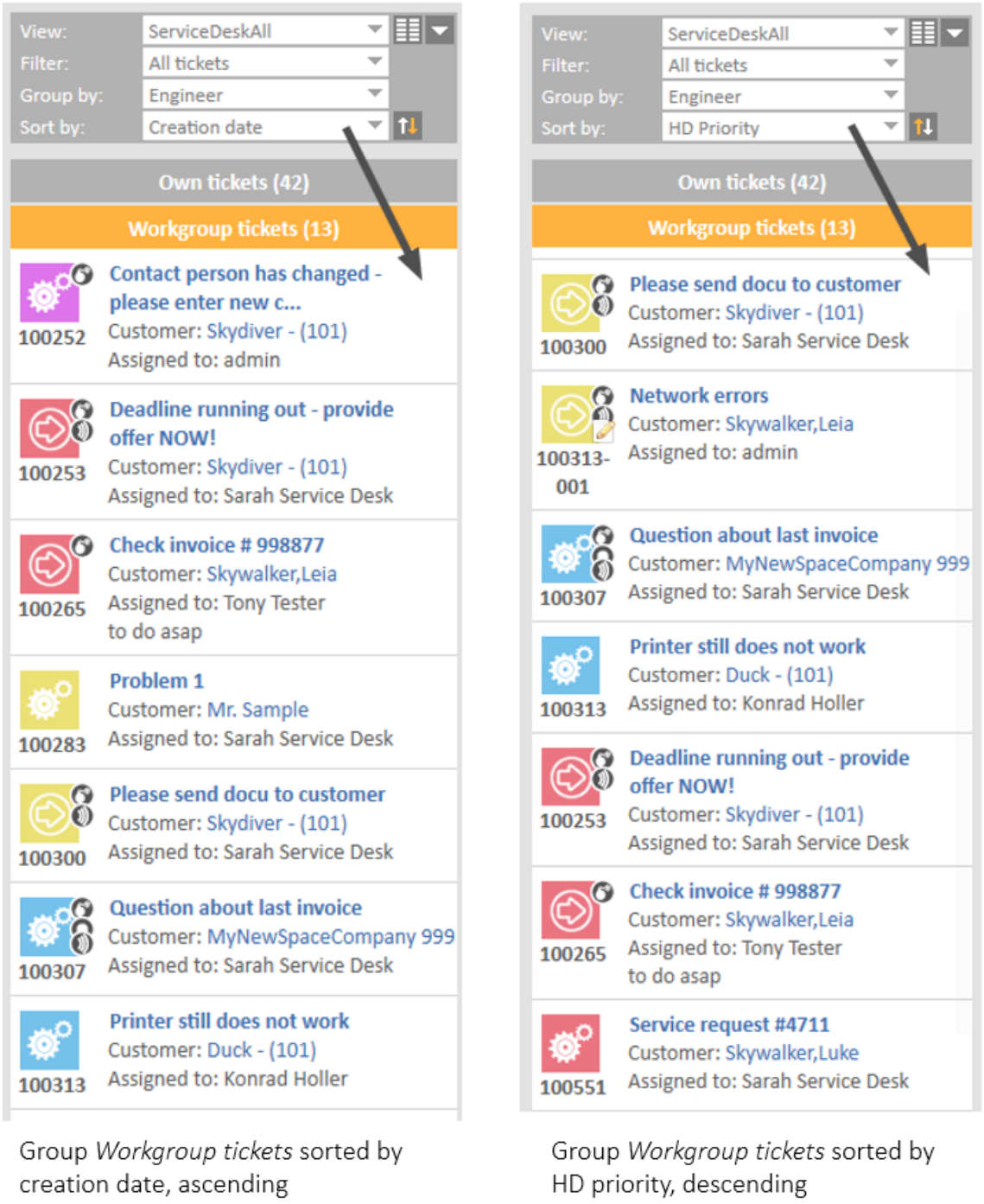


Figure 92: *ConSol CM Web Client - Two different sortings of the ticket list, grouped by engineer, group Workgroup tickets (creation date: standard, HD priority: customized)*

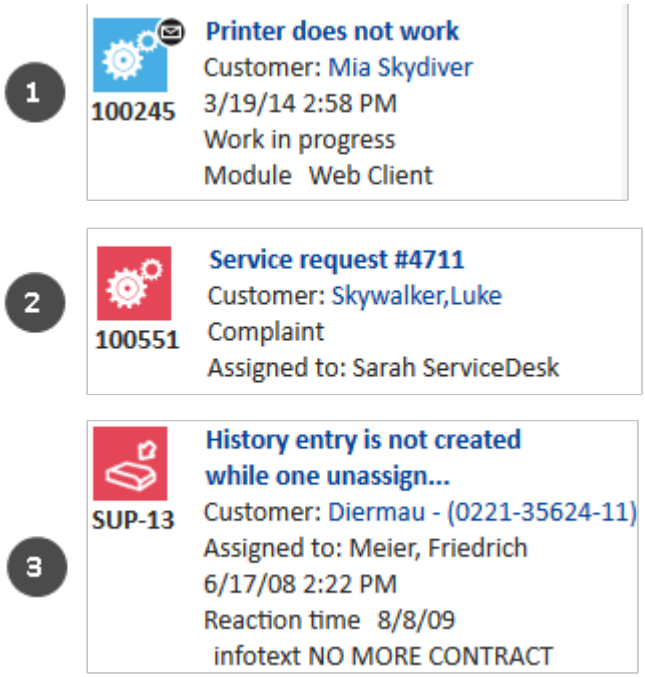
In the Admin Tool, the annotation `sortable` is set for the ticket field in the navigation group *Tickets*, navigation item *Ticket fields*.

C.5.3 Configuration of Parameters Which Are Displayed for One Ticket

In the current section, you will learn how to configure the parameters which are displayed for one ticket. A ticket in the ticket list can look different in different systems, depending on the following parameters:

- the definition of customer templates
- the annotation of ticket fields
- the page customization for the ticket list-specific attributes

Please see the following examples for different ticket appearances:



In the example displayed above, the following data is displayed:

(1) Service Desk ticket example

- Ticket subject (standard)
- Customer (formatted by customer template)
- Creation date (standard for unassigned tickets)
- Scope (formatted by page customization)
- Ticket field (here: Module, an enum; formatted by annotation)

(2) Customer Service ticket example

- Ticket subject (standard)
- Customer (formatted by customer template)
- Ticket field (here: Ticket type, an enum)
- Assigned to (standard for assigned tickets)

(3) Support ticket example

- Ticket subject (standard)
- Customer (formatted by customer template, here with phone number)
- Assigned to (standard for assigned tickets)
- Creation date
- Ticket field (here: Reaction time deadline, a DATE field)
- Ticket field (here: a String field, infotext)

C.5.4 The Definition of Customer Templates

The appearance of customer data (e.g. name only or name and phone number) is defined by the template which is assigned in the customer object. This can be the *Ticket list* template or, if the *Ticket list* template is not defined, the *Default* template. Please see section Templates for Customer Data for a detailed explanation of templates for customer data.

C.5.5 The Annotation of Ticket Fields

C.5.5.1 Ticket List-Specific Annotations

There are three annotations which are specific for the display of ticket field data in the ticket list:

- ticket-list-position
- ticket-list-colspan
- ticket-list-rowspan

ticket-list-position

This annotation can be set for ticket fields. It defines the position of the given ticket field in the ticket list. Please be aware that all fields which are annotated this way have to fit in a common matrix. The principle is the same as for the positioning of customer fields on the ticket GUI, see section Ticket Field Administration (Setting Up the Ticket Data Model).

ticket-list-colspan

This annotation can be set for ticket fields. It defines the number of columns which should be used by the respective field.

ticket-list-rowspan

This annotation can be set for ticket fields. It defines the number of rows which should be used by the respective field.

C.5.5.2 General Annotations Which Help Designing the Ticket List

The following annotations help designing the ticket list, but they are not ticket-list-specific, i.e., they can be used for any other ticket field as well:

- show-label-in-view

show-label-in-view

This annotation can be set for ticket fields, for customer field and for resource fields. Regarding the ticket list configuration, only ticket fields are relevant.

If the annotation is set to “true”, the label (name of the ticket field, technical or localized) is displayed in the view mode (here: in the ticket list). If you do not want the label to be displayed, set the value to “false”.

C.5.5.3 Example

The following example shows a ticket in the ticket list, configured using several parameters.

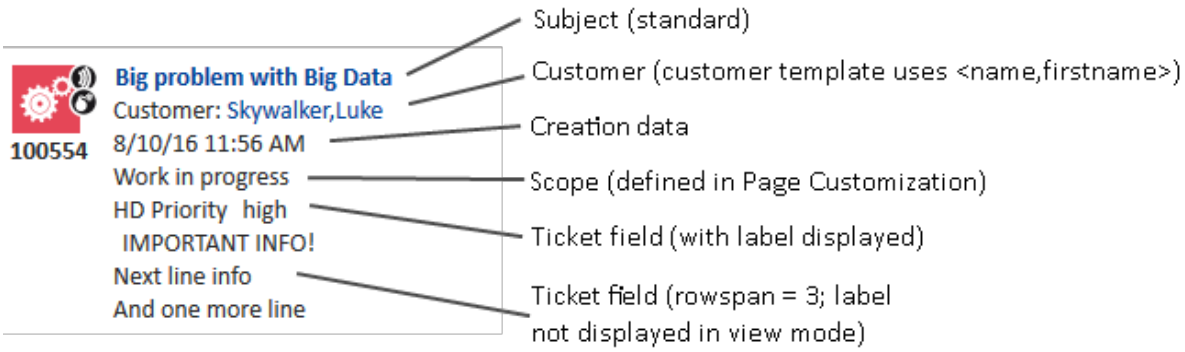


Figure 93: *ConSol CM Web Client - Ticket in ticket list, formatted using several parameters*

C.5.6 The Page Customization for the Ticket List-Specific Attributes

Four page customization attributes play role for configuring the appearance of a ticket in the ticket list. They are set for the page customization type *accordionTicketList*.

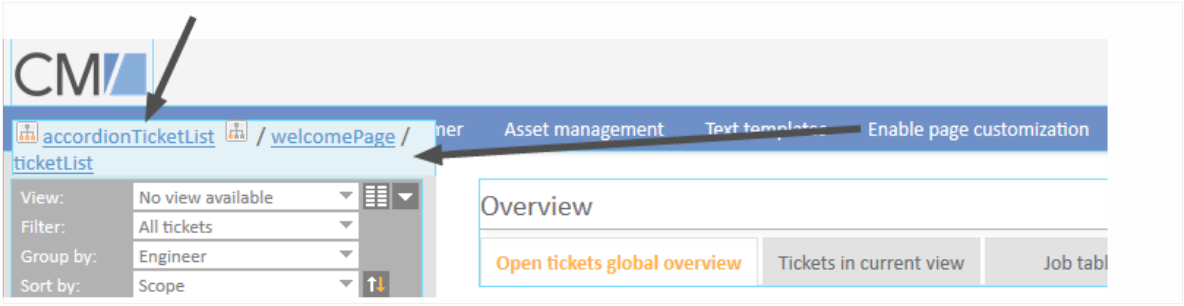


Figure 94: *ConSol CM Web Client - Page Customization for the ticket list*

The following attributes can be used:

- hideEmptyGroups
- ticketDataConfigQueueGroupingScript
- ticketDataConfigEngineerGroupingScript

- ticketDataConfigCustomGroupingScript
- ticketDataConfigNoGroupingScript

See also section accordionTicketList (Type).

In order to define the layout of each ticket in the ticket list, the following keywords can be used (for a detailed explanation of the required syntax, please see section Formatting Principle and Syntax):

- CREATION_DATE
- QUEUE
- SCOPE
- ENGINEER
- CUSTOMER

Additionally, each type of grouping (see following sections) uses an additional type-specific set of keywords.

C.5.6.1 ticketDataConfigQueueGroupingScript

Defines the ticket information display when grouping by queue is selected. The following additional keywords can be used (for a detailed explanation of the required syntax, please see section Formatting Principle and Syntax):

- Queue names

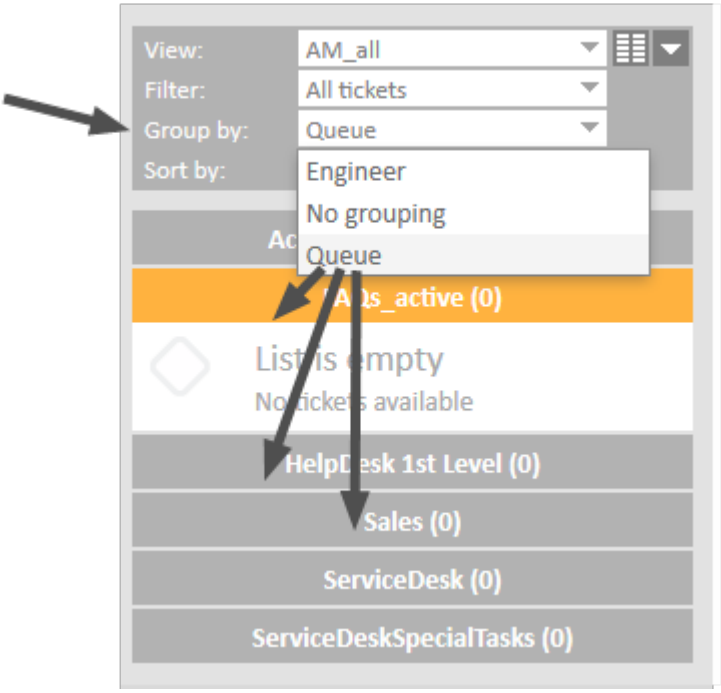


Figure 95: *ConSol CM Web Client - Ticket list grouped by queue*

C.5.6.2 ticketDataConfigEngineerGroupingScript

Defines the ticket information display when grouping by engineer is selected. The following additional keywords can be used (for a detailed explanation of the required syntax, please see section Formatting Principle and Syntax):

- MINE
- GROUP
- UNASSIGNED

C.5.6.3 ticketDataConfigCustomGroupingScript

Defines the ticket information display when grouping by a ticket field is selected. The following additional keywords can be used (for a detailed explanation of the required syntax, please see section Formatting Principle and Syntax):

- enum_grp_name.enum_value_name

C.5.6.4 ticketDataConfigNoGroupingScript

Defines the ticket information display when no grouping criterion is selected. No additional keywords can be used.

- Only the DEFAULT setting can be used.

C.5.6.5 Formatting Principle and Syntax

The formatting principle and syntax are identical for all four attributes.

The configuration String can contain 2048 characters. In case you have to use a longer construct, use an Admin Tool script (of type page customization).

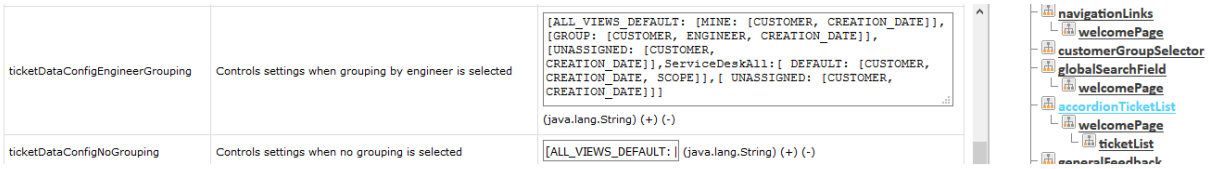


Figure 96: *ConSol CM Web Client - Page Customization attributes for accordionTickeList (two examples displayed)*

To define the ticket list display with page customization (for the attributes mentioned above), a hierarchical configuration is used.

The **first** level is represented by the view definition

- ALL_VIEWS_DEFAULT
- View names like, e.g., *2nd_Level_View_Open*

On the **second** level, you can define for which subset the parameter should be displayed.

For queue-based grouping (i.e. in `ticketDataConfigQueueGroupingScript`), the following parameters can be used:

- Queue names

For engineer-based grouping (i.e. in `ticketDataConfigEngineerGroupingScript`), the following parameters can be used:

- MINE
- GROUP
- UNASSIGNED

For custom-defined grouping (i.e. in `ticketDataConfigCustomGroupingScript`), the following parameters can be used:

- enum_grp_name.enum_value_name, e.g. helpdesk_standard.module.misc.

For each type of grouping, a DEFAULT value can be set which will be used for all constellations which are not defined specifically.

The **third** level is represented by the parameters to display. This can be

- CUSTOMER
- SCOPE
- ENGINEER
- CREATION_DATE
- QUEUE

Please see the following example (for an engineer-specific grouping) for a first impression. To improve readability, we have inserted line breaks. In CM, please do not set any line break in the statement.

```
[ALL_VIEWS_DEFAULT:
 [MINE: [CUSTOMER, CREATION_DATE]],
 [GROUP: [CUSTOMER, ENGINEER, CREATION_DATE]],
 [UNASSIGNED: [CUSTOMER, CREATION_DATE]],
 ServiceDeskAll:
 [ DEFAULT: [CUSTOMER, CREATION_DATE, SCOPE]],
 [ UNASSIGNED: [CUSTOMER, CREATION_DATE]]]
```

The meaning of the template above:

For all views, it is set that the engineer's assigned tickets contain customer and creation date, that the group tickets contain customer, the assigned engineer and the creation date, and the unassigned tickets contain customer and creation date. For tickets in the queue ServiceDesk, a specific configuration is provided: for all types of tickets (i.e. tickets assigned to the engineer and group tickets), the customer data, creation date and scope should be displayed. For unassigned tickets, only customer data and creation date are displayed.

The ticket list, in the example, is displayed as shown in the following two figures.

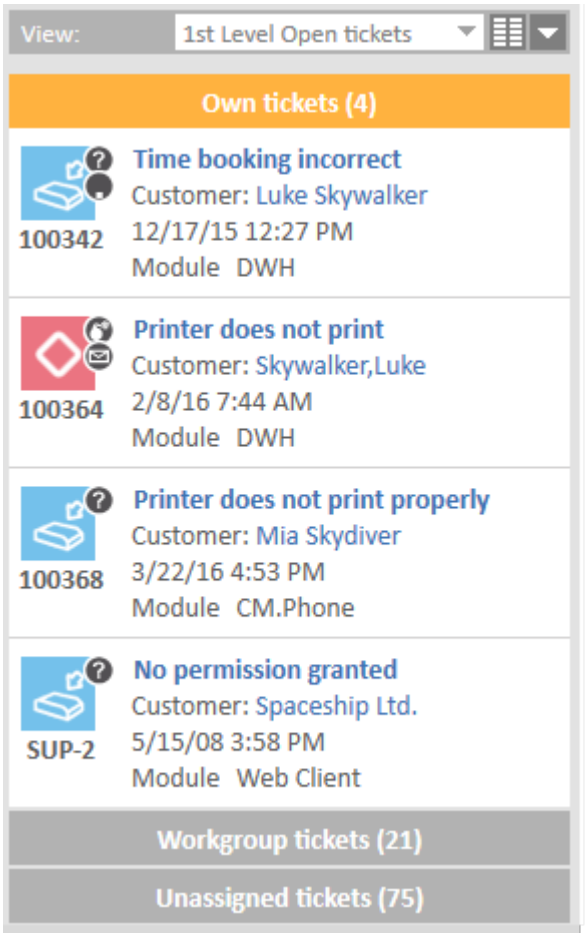


Figure 97: *ConSol CM Web Client - Ticket list for view HelpDesk 1st Level, defined by ALL_VIEWS_DEFAULT*

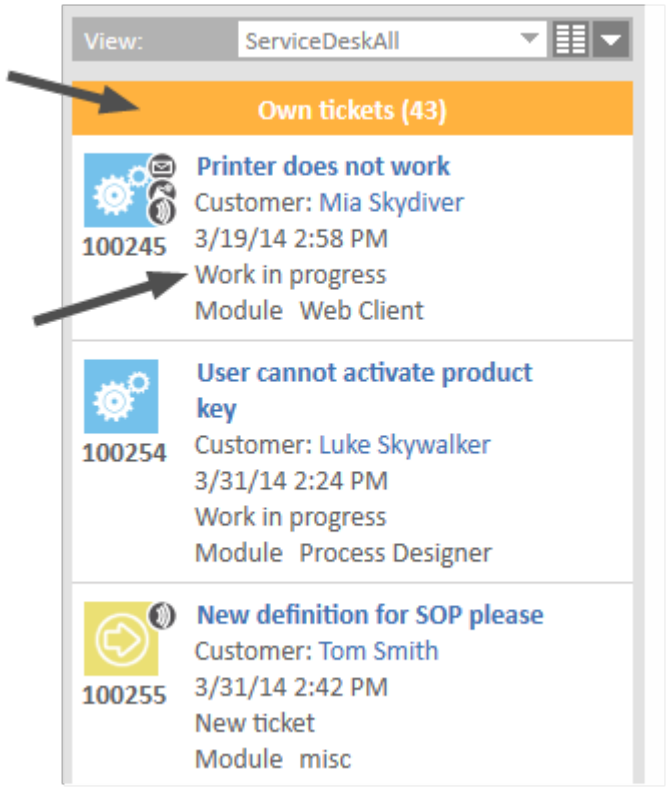


Figure 98: *ConSol CM Web Client - Ticket list for view ServiceDeskAll, defined view-specific, scope displayed for the engineer's tickets*

Since two of the definitions are identical, you could work with the DEFAULT value here. The following JSON statement represents a configuration identical to the example shown above:

```
[ALL_VIEWS_DEFAULT:
 [DEFAULT: [CUSTOMER, CREATION_DATE]],
 [GROUP: [CUSTOMER, ENGINEER, CREATION_DATE]],
 ServiceDeskAll:
 [ DEFAULT: [CUSTOMER, CREATION_DATE, SCOPE]],
 [ UNASSIGNED: [CUSTOMER, CREATION_DATE]]]
```

Restore default setting for a configuration:

- DEFAULT_CONFIGURATION_SCRIPT

C.5.6.6 Default System Settings

If no values are changed, or if the default configuration is restored (using the parameter DEFAULT_CONFIGURATION_SCRIPT), the following values are set:

- **Queues** (ticketDataConfigQueueGrouping):

```
[ ALL_VIEWS_DEFAULT: [ DEFAULT: [CUSTOMER, CREATION_DATE] ] ]
```

- **Engineers** (ticketDataConfigEngineerGrouping)

```
[ ALL_VIEWS_DEFAULT: [ MINE: [CUSTOMER, CREATION_DATE ] ],
  [ GROUP: [CUSTOMER, ENGINEER, CREATION_DATE] ],
  [ UNASSIGNED: [CUSTOMER, CREATION_DATE ] ] ]
```

- **Ticket fields** (ticketDataConfigCustomGrouping):

```
[ ALL_VIEWS_DEFAULT: [ DEFAULT: [CUSTOMER, CREATION_DATE] ] ]
```

- No grouping (ticketDataConfigNoGrouping):

```
[ ALL_VIEWS_DEFAULT: [ DEFAULT: [CUSTOMER, CREATION_DATE] ] ]
```

C.5.6.7 Configuring Page Customization for the Ticket List Using a Script

As for all page customization attributes, instead of setting the values for each single attribute using the Web Client, you can set the required attributes using an Admin Tool script. For a detailed explanation, please refer to the section about Page Customization.

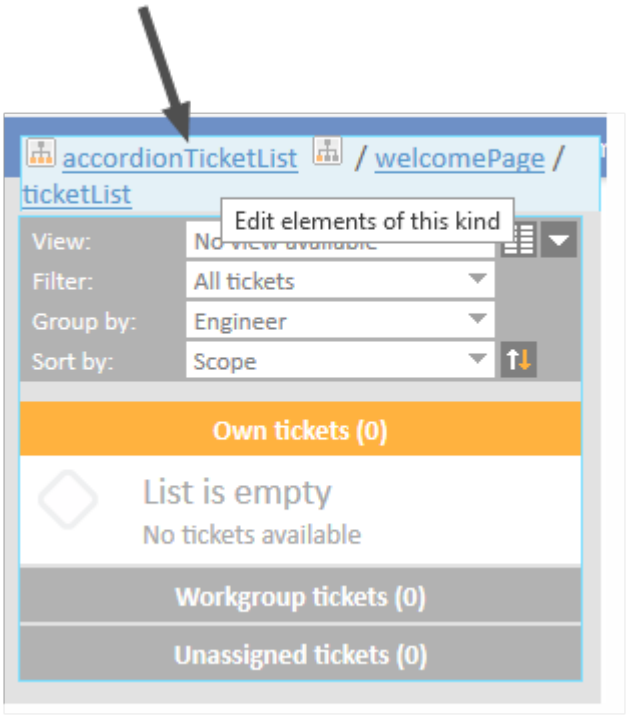


Figure 99: *ConSol CM Web Client - Page Customization for the ticket list*

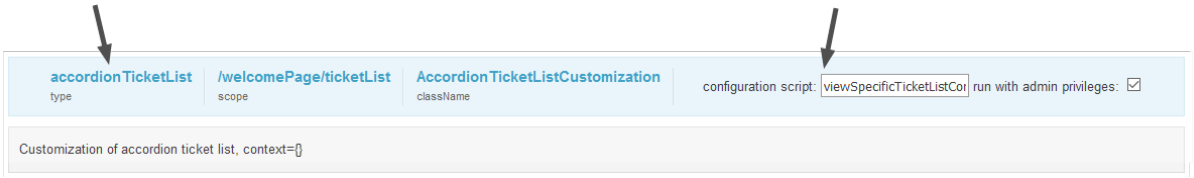


Figure 100: *ConSol CM Web Client - Configuring a script for the Page Customization for the ticket list*

The attributes for the ticket list have to be set using a script for the type *accordionTicketList*. The script can either return only one of the attributes for the type or more, or even values for all attributes. The following example shows a script which returns values for the two attributes *ticketDataConfigNoGrouping* and *ticketDataConfigQueueGrouping*.

```
return [ticketDataConfigNoGrouping: "[ALL_VIEWS_DEFAULT: [DEFAULT: [CUSTOMER,
 CREATION_DATE]], MultiCGView: [DEFAULT: [ENGINEER]], 1st_Level_View_Open:
 [DEFAULT: [SCOPE]]]",ticketDataConfigQueueGrouping: "[ALL_VIEWS_DEFAULT: [DEFAULT:
 [CUSTOMER, CREATION_DATE, ENGINEER, ENGINEER]], [HelpDesk_1st_Level: [CUSTOMER,
 SCOPE, CREATION_DATE, QUEUE]], [HelpDesk_2nd_Level: [CUSTOMER, QUEUE, SCOPE,
 CREATION_DATE]]]"]
```

Code example 1: *Example script for ticket list configuration (viewSpecificTicketListConfig.groovy)*

The following ticket list configurations will be displayed in the Web Client, see following figure.

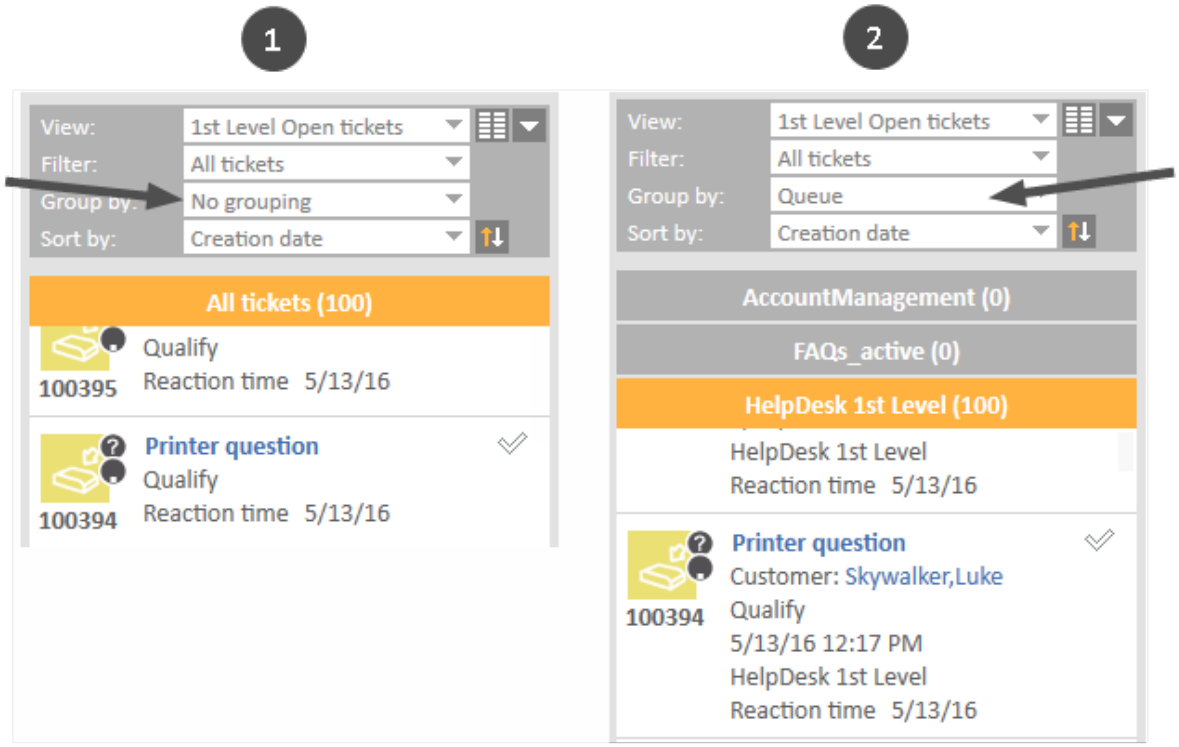


Figure 101: *Parts of two ticket lists according to the configuration in the example script*

- (1) No grouping. HelpDesk_1st_Level:
  DEFAULT: SCOPE
  (Reaction time stems from annotation!)
- (2) Queue grouping. HelpDesk_1st_Level:
  CUSTOMER, SCOPE, CREATION_DATE, QUEUE displayed
  (Reaction time stems from annotation!)

C.6 Configuration of the Web Client Dashboard

C.6.1 Introduction

Starting with version 6.9.4, the ConSol CM Web Client provides a dashboard for engineers, displayed on the Web Client *Overview* page. On the dashboard you can display statistical values that show important information about the work areas of your engineers.

The dashboard is composed of one or more so-called widgets. In its default configuration, the dashboard contains only one widget which displays a graphic with the number of tickets in all groups of the selected view. Dashboards can contain interactive elements, e.g., to show or hide elements of the graphic. However, persistent engineer-specific (personal) adaption is not possible.

> After an update from a previous version to CM version 6.9.4 and up, the Web Client Dashboard will be disabled.
>
> In a new installation CM 6.9.4 or higher, the Web Client Dashboard will be enabled by default. A default dashboard configuration is provided (see example below).

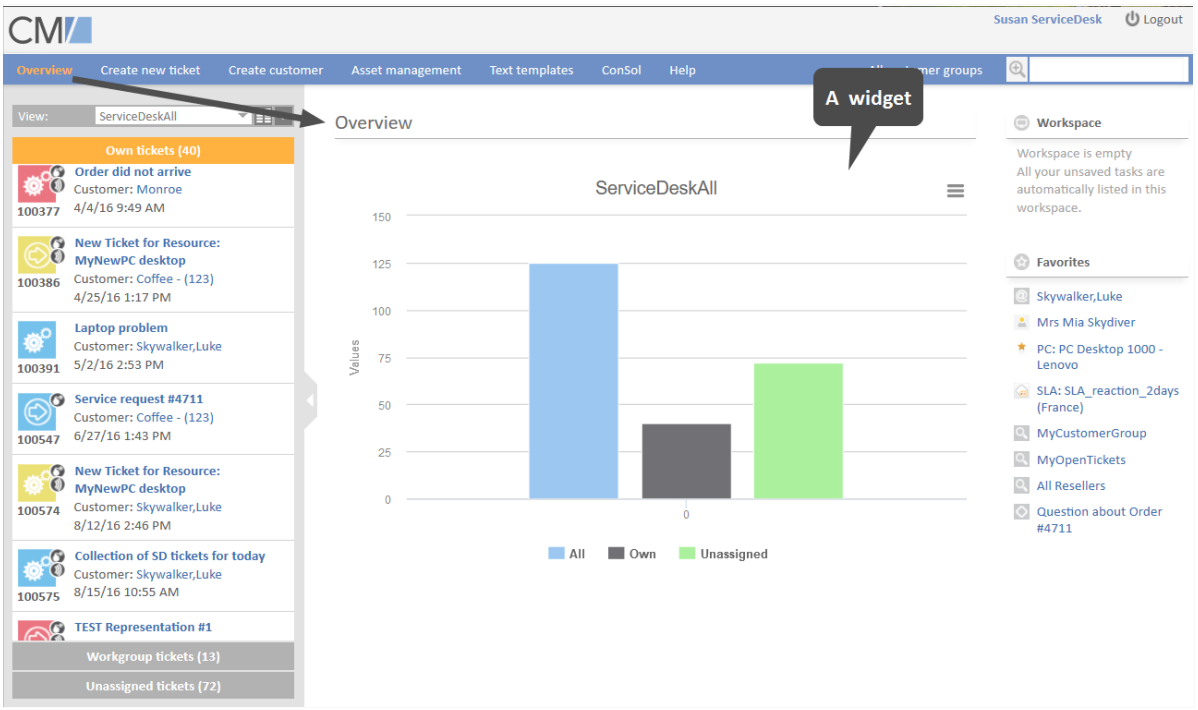


Figure 102: *ConSol CM Web Client - Web Client Dashboard example*

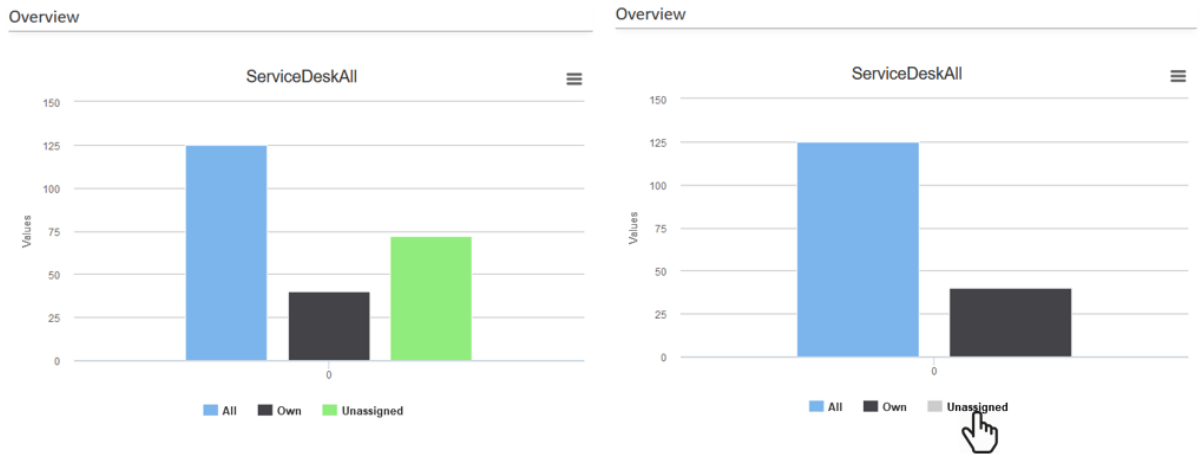


Figure 103: *ConSol CM Web Client - Interactive widget (fade-in and -out parts of the graphic)*

Widgets which are used for the dashboard are of one of the following types:

- **Chart**
  for graphical representation of data
- **Table**
  for table representation of data
- **KPI**
  for special representation of key performance indicators
- **Recently visited**
  Standard widget which displays a list of the objects the current engineer has recently viewed. Can be switched on or off. The engineer can select if only his tickets or all tickets should be displayed. In both cases, the engineer's access permissions determine which tickets are displayed.
- **Recent changes**
  Standard widget which displays a list of the objects which have recently been changed. The engineer can select if only his changes or the changes made by all engineers should be taken into consideration. The engineer's access permissions determine which tickets are displayed.

As an administrator you can design the layout of the dashboards as required, i.e., you can place KPI, chart, and table widgets on the dashboard page. They are placed on the page based on a grid layout.

C.6.2 Examples

Please look at the following examples to get a first impression of what can be displayed on the Web Client Dashboard.

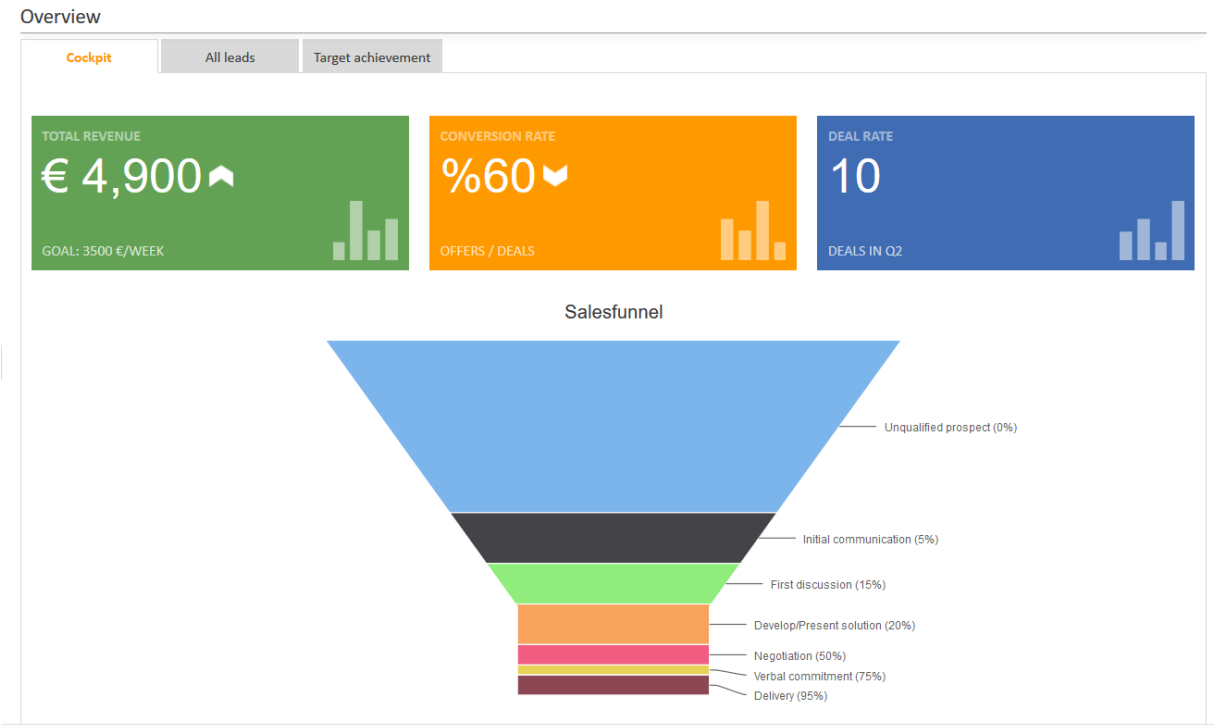


Figure 104: *ConSol CM Web Client - Example of a dashboard tab with two chart widgets (KPI, funnel) in one column*

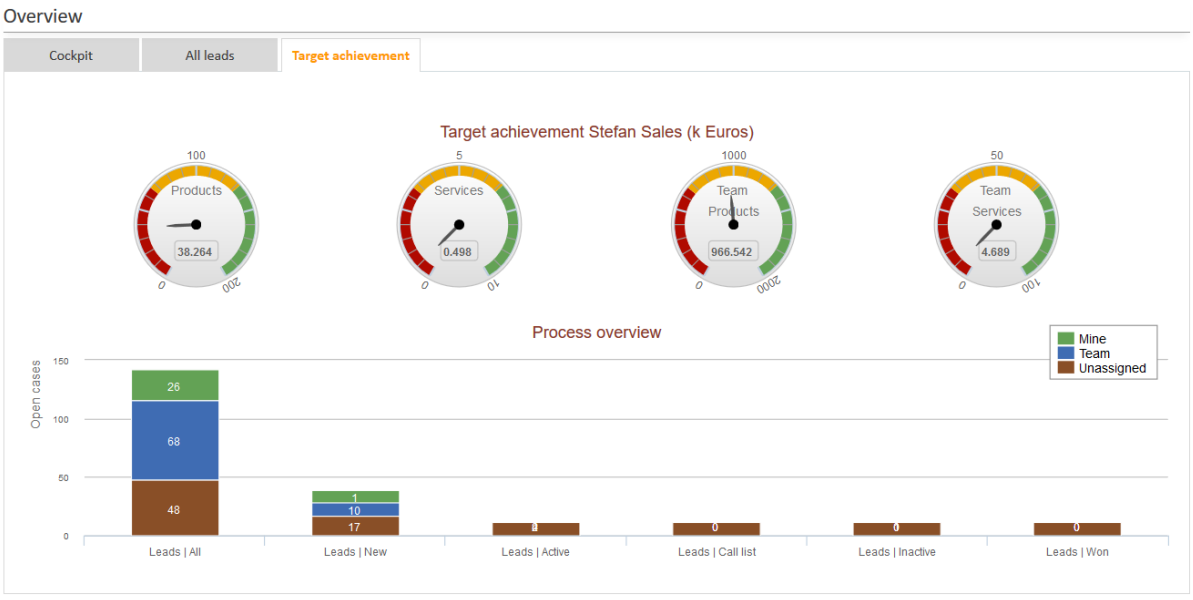


Figure 105: *ConSol CM Web Client - Example of a dashboard tab with two chart widgets in one column*

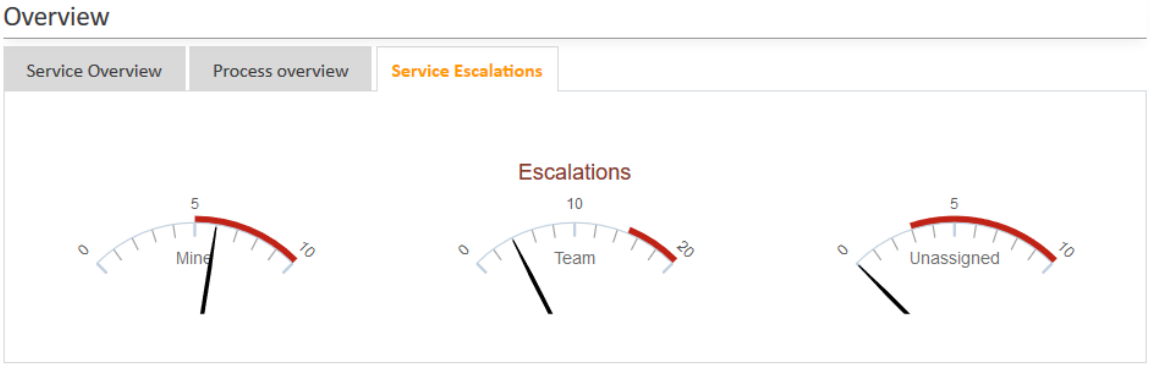


Figure 106: *ConSol CM Web Client - Example of dashboard tab with one widget*

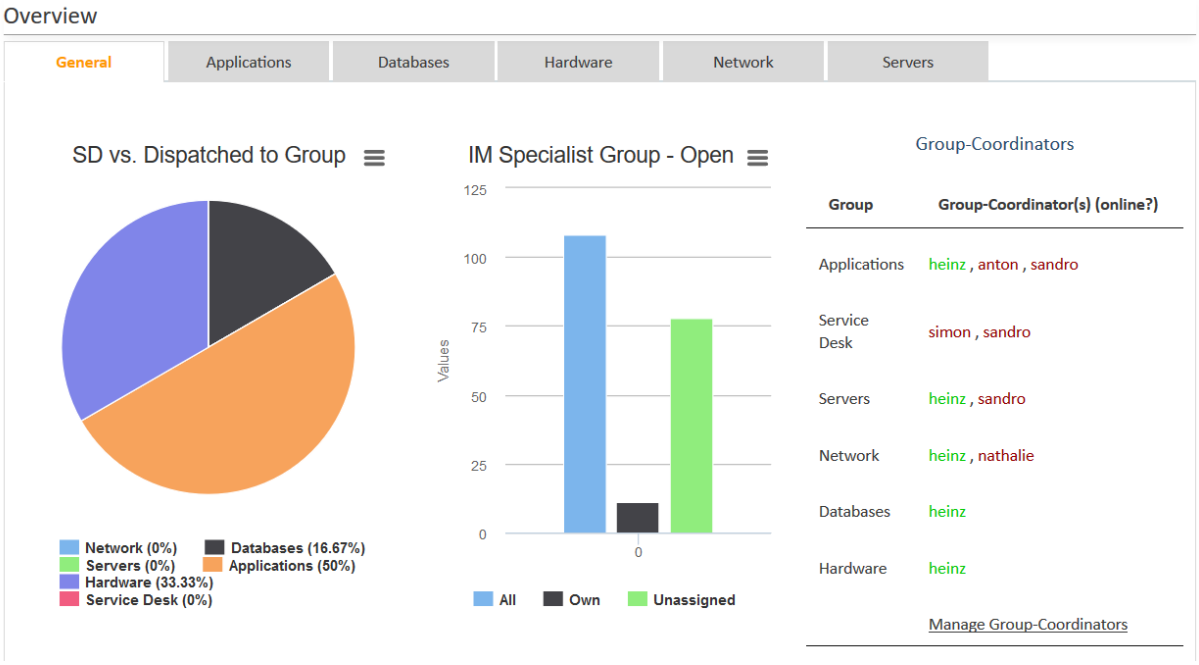


Figure 107: *Example of a dashboard tab with three different widgets in one row*

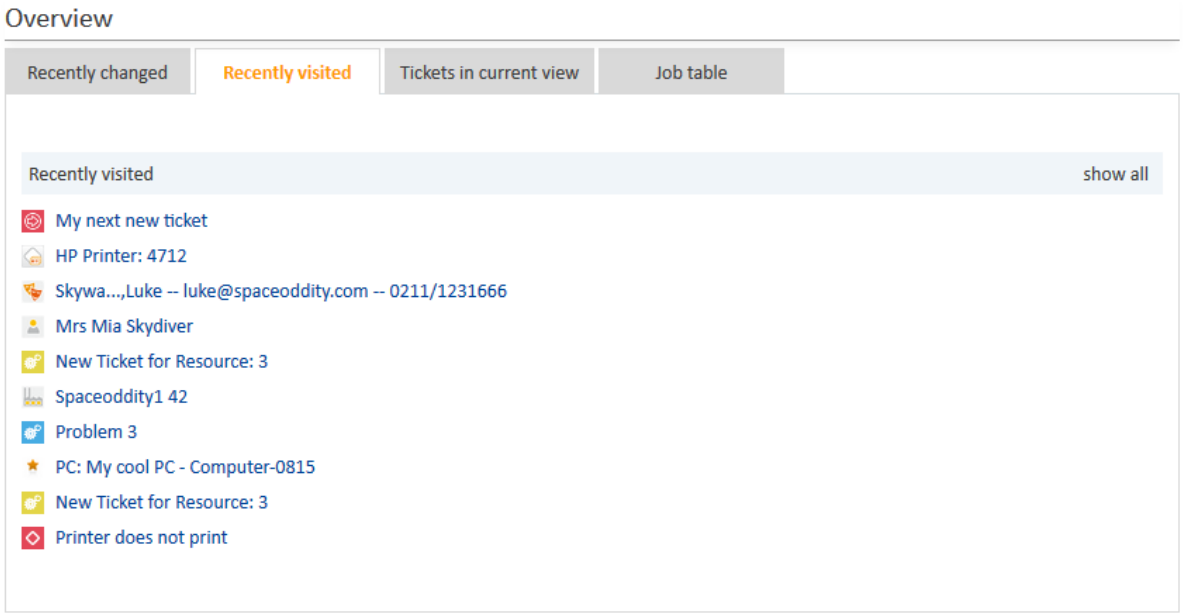


Figure 108: *Example of a recentlyVisitedWidget*

Please note that the *Recently Visited* widget might look different in your CM system, because it has been configured in a different way using Attributes and Settings for the recentlyVisitedWidget.

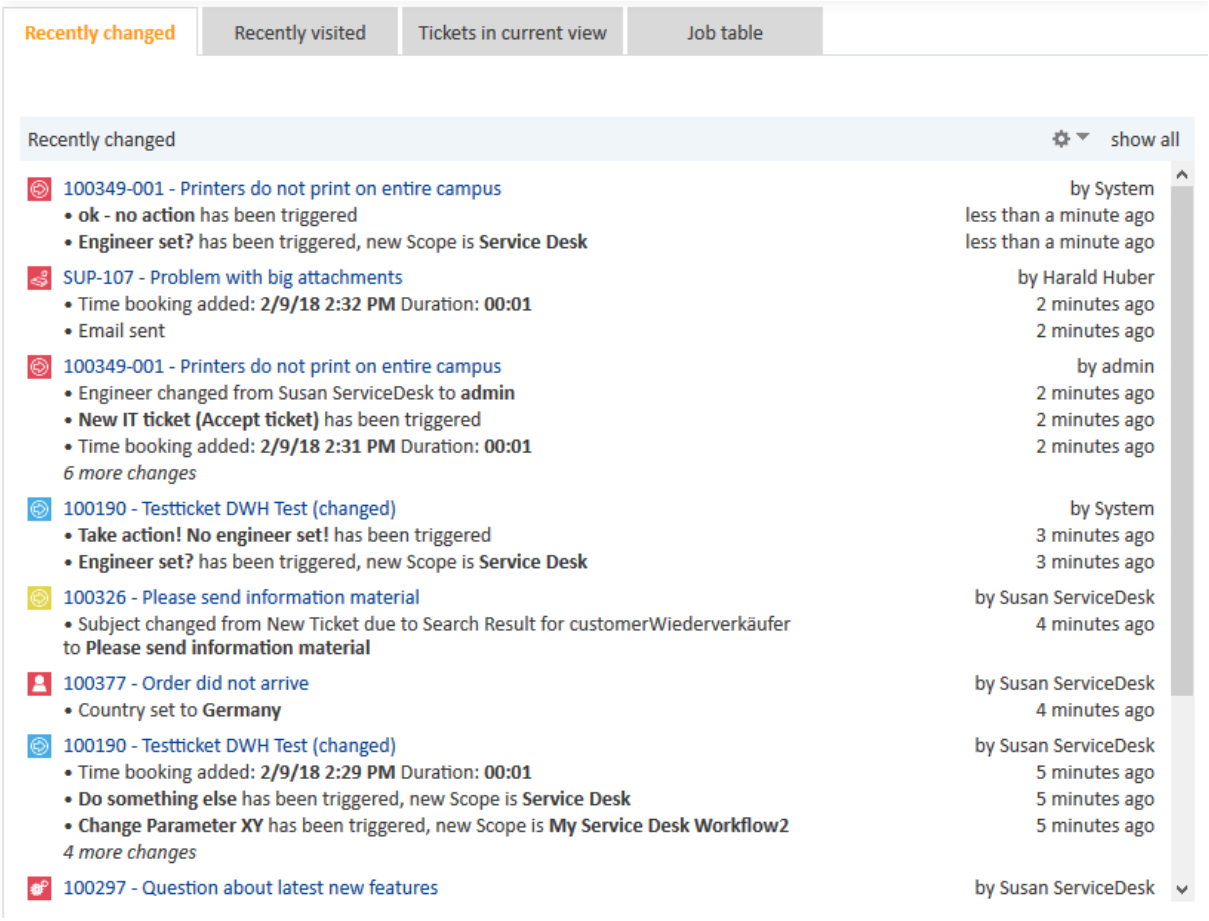


Figure 109: *Example of a recentChangesWidget*

Please note that the *Recent Changes* widget might look different in your CM system, because it has been configured in a different way using Attributes for the recentChangesWidget.

The originators of the changes in the Recent Changes widget are displayed as follows:

- engineer has first name and last name: first name and last name are displayed
- engineer has only last name: last name is displayed
- engineer has only first name: first name and login are displayed
- engineer has only login: login is displayed

Note for all widgets:

The data for the graphs and/or tables in the widgets are retrieved using Groovy statements which are placed in an Admin Tool script.

The dashboard is configured using Page Customization for the Web Client Dashboard.

C.7 Configuration of the Graphical Representation of Relations

C.7.1 Configuration of the Graphical Display in Standard Mode

In several locations of the ConSol CM Web Client, relations between objects can be established and are displayed:

- On the ticket page
  - Relationen of the ticket to one or more customers
  - Relations of the ticket to other tickets (references or parent/child or master/slave relations)
  - Relations of the ticket to one or more resources (if CM/Resource Pool is active)
- On the company page:
  - Relations of the company to tickets
  - Relation of the company to (its) contacts
  - Relations of the company to other customers (contacts, companies)
  - Relations of the company to one or more resources (if CM/Resource Pool is active)
- On the contact page:
  - Relations of the contact to tickets
  - Relations of the contact to other customers (contacts, companies)
  - Relations of the contact to one or more resources (if CM/Resource Pool is active)

If CM/Resource Pool is active:

- On the resource page:
  - Relations of the resource to one or more other resources
  - Relations of the resource to customers (contacts, companies)
  - Relations of the resource to tickets

The relations can be displayed as list and as graph. The list display is activated per default and cannot be switched off. The graph view can be additionally activated using **page customization**. If it is activated, the engineer can select between list and graph view.

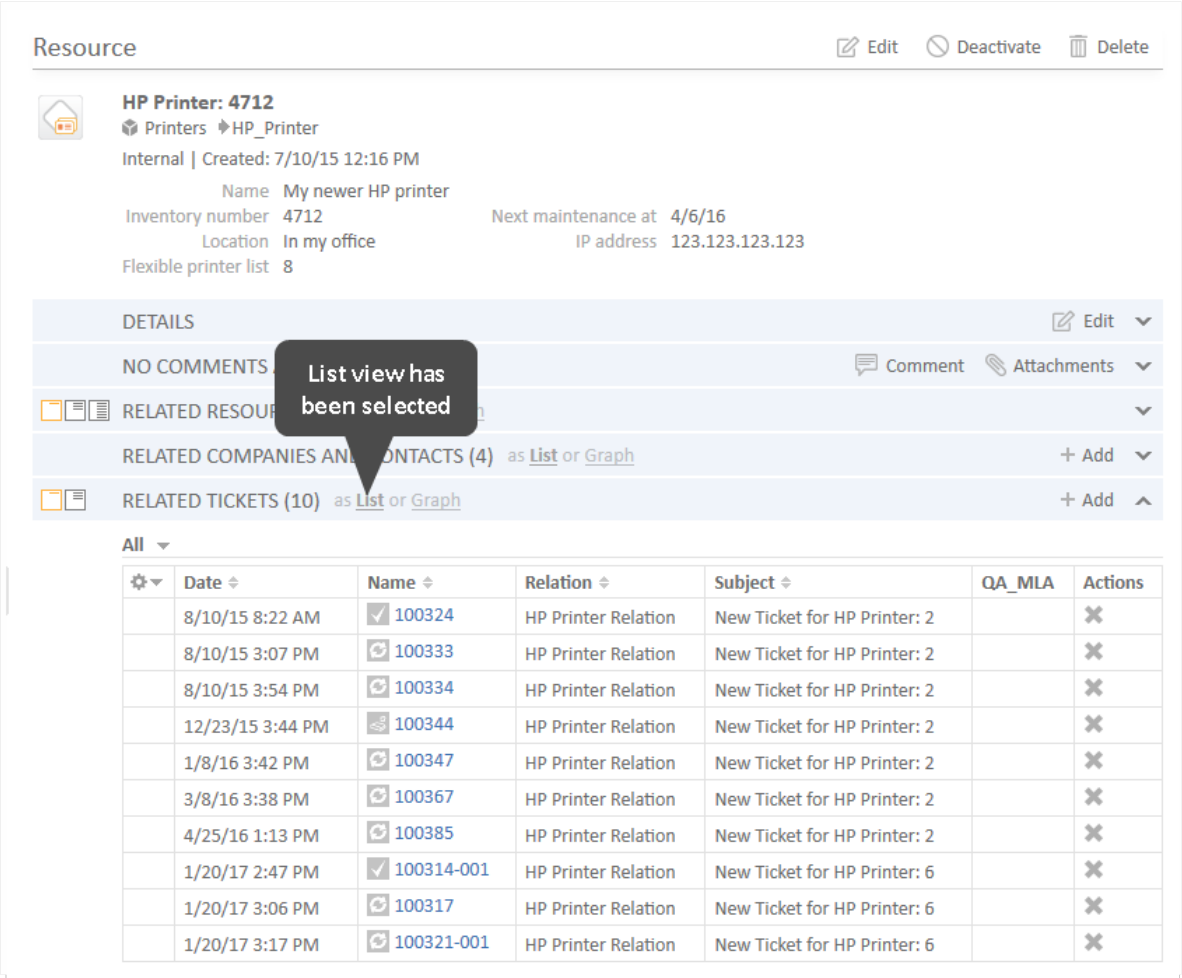


Figure 110: *ConSol CM Web Client - Resource page of a printer: Related tickets section in list mode*

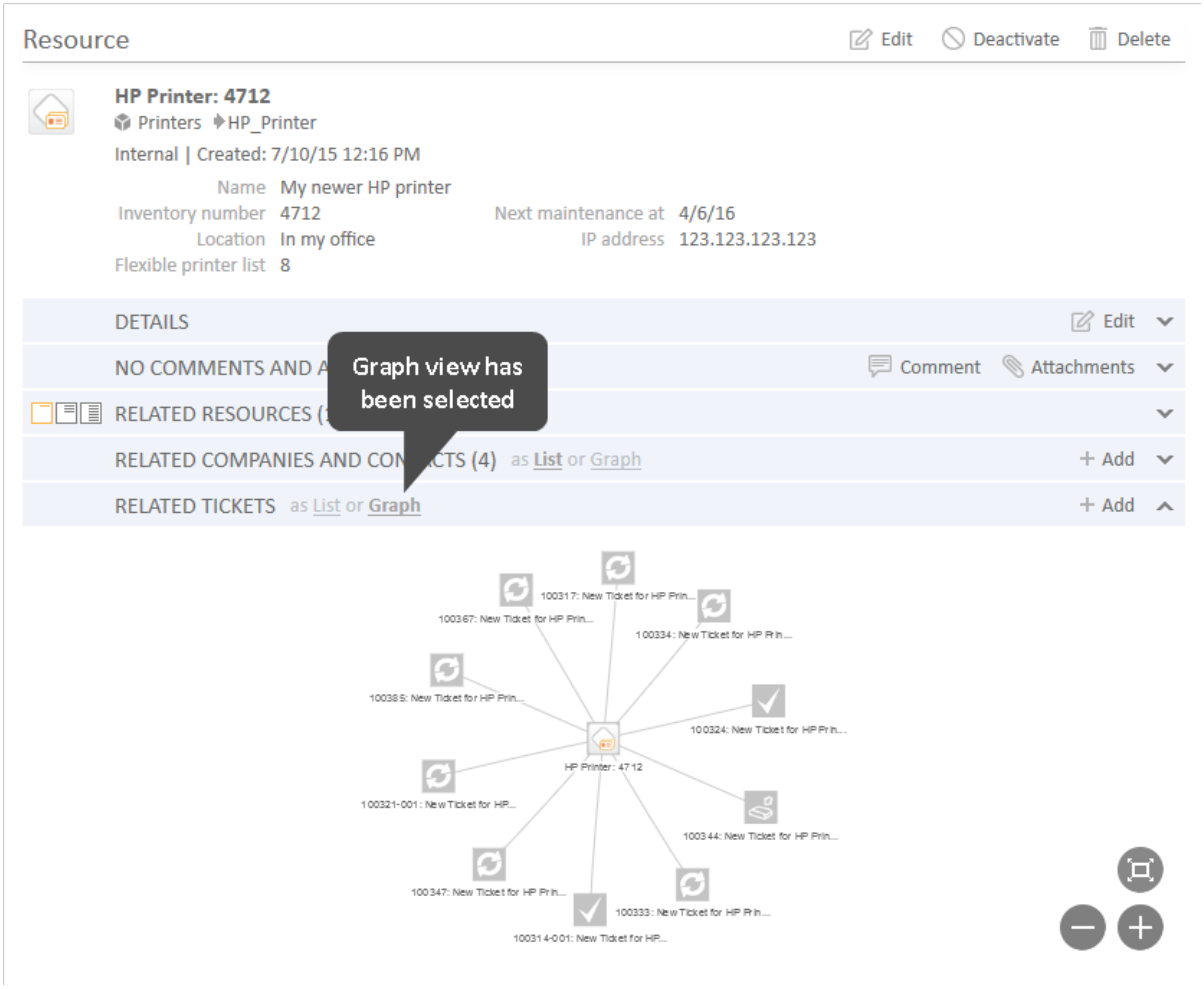


Figure 111: *ConSol CM Web Client - Resource page of a printer: Related tickets section in graph mode*

The graph view has to be activated for each section (see list above) separately. It is configured using page customization attributes. Please see section Page Customization for the Graphical Representation of Relations for a detailed explanation of the configuration.

C.7.2 Configuration of More Relation Displays - Expert Mode

In addition to the standard relations sections in ticket, customer, and resource pages, an additional section can be displayed on each of these pages. In this section, a relations graph can be displayed which is implemented specifically for the system and is based on a script. Please see section Page Customization for the Graphical Representation of Relations for a detailed explanation of the configuration.

C.8 Page Customization

This chapter discusses the following:

C.8.1 General Introduction to Page Customization

In addition to the design of the Web Client GUI in the process of defining ticket fields (see section Ticket Field Administration (Setting Up the Ticket Data Model)), customer fields (see section Customer Field Management and GUI Design for Customer Data), and maybe resource fields (see section CM/Resource Pool - Setting Up the Basic Resource Model) an administrator can configure more GUI layout features and functionality using **page customization**. The configuration is performed by assigning values to *attributes*.

When you log in to the Web Client as an administrator, you see the item *Enable page customization* in the main menu. Depending on the context, i.e., on the page that is currently displayed, the page customization offers different, page- and context-specific functionality.

For example, when you have opened a ticket and start the page customization, you can configure attributes for the ticket in general. When the Ticket Email Editor is open, you can also configure email editor-specific attributes.

In the following sections, the general principle of page customization and all available page customization attributes are described and explained in detail. In all other sections of this *ConSol CM Administrator Manual,* references to this sections will be included where required.

The page customization settings are part of the ConSol CM configuration data. You can export and import them using the Admin Tool, as described in section Deployment (Import/Export).

C.8.2 Page Customization in the Web Client

When you start the page customization mode, the *Page Customization Definition Section* is displayed in the lower half of the GUI. On the right side you see a tree that reflects the structure of the current page. Within the page, every element of the page is marked by a blue border. When you move the mouse over an element, its name is displayed. When you click on this name, the definition section for this element is opened, and the element is marked in the tree. In this way, you can easily identify the component you would like to modify.

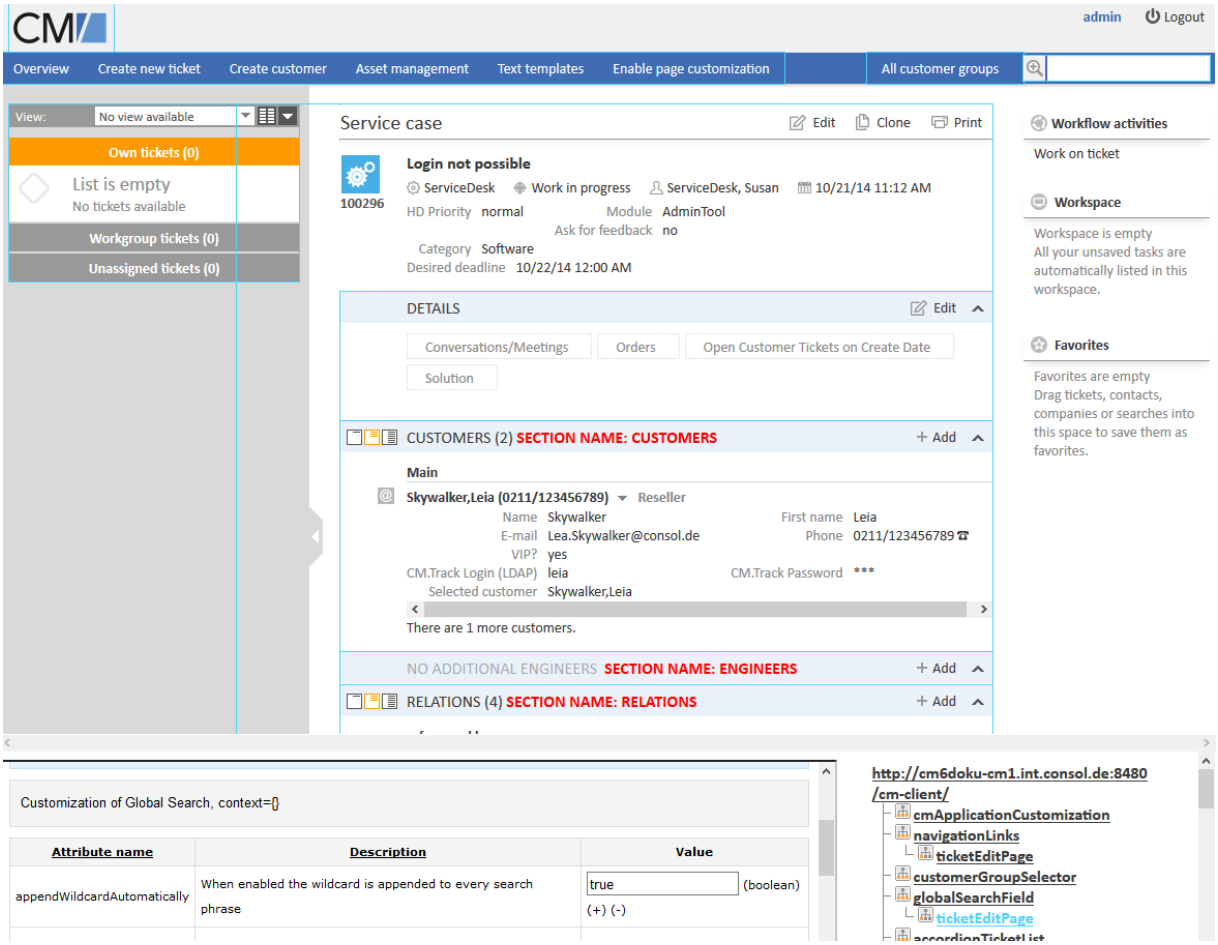


Figure 112: *ConSol CM Web Client - Page Customization Definition Section*

The tree might display the following elements (see next figure). Since the Page Customization Definition Section is rather small you might have to scroll to see all elements. In this example (see figure above), the administrator has opened the *ticketEditPage* (see following paragraphs for details).

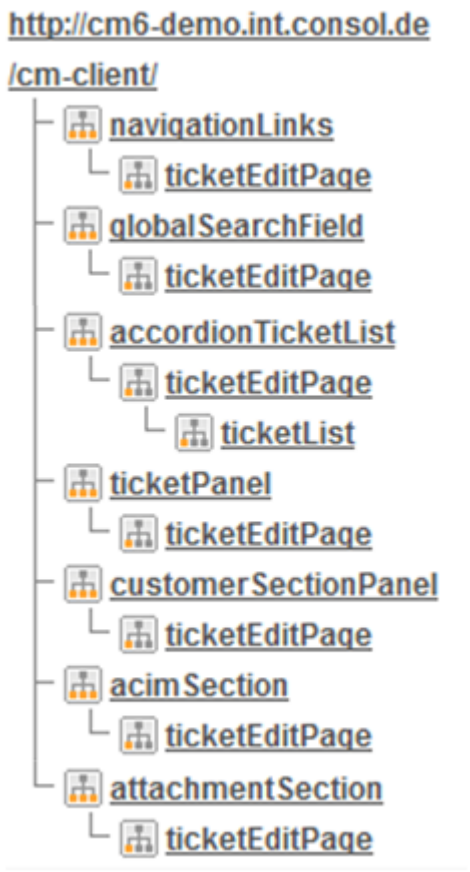


Figure 113: *ConSol CM Web Client - Page Customization tree*

Icon	Description
	Configuration of all elements of this type
	Configuration of this specific element deployed within the identified scope

You can also click on an element name in the tree to open the editor for this element in the left area of the Definition Section, e.g., for the element *navigationLinks* (see following figure).

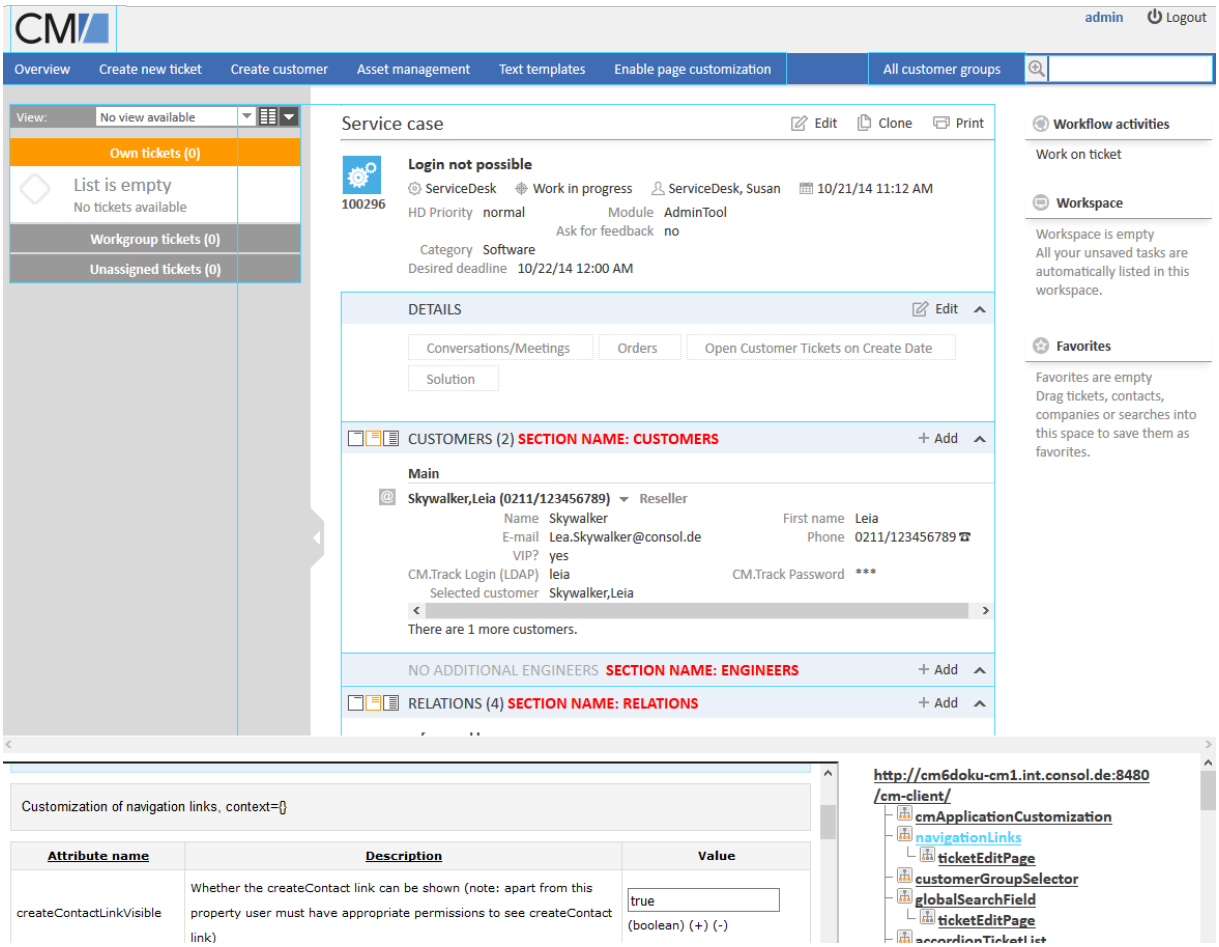


Figure 114: *ConSol CM Web Client - Selected tree element in the definition section*

The entire page is built according to a strictly hierarchical structure and every element is defined by

- a type (mandatory)
- a scope (mandatory)
- a subscope (optional)
- a class (the respective Java class, mandatory)

These are displayed in the blue header section of the editor in the Definition Section when you mark an element either in the tree or in the GUI. Using the page customization, you can decide on which level you want to configure attributes:

- When you work on the *type* level, you define the attributes for all sub-elements of this type, i.e., for all scopes (subscopes).
- When you work on the *scope* level you define the attributes for all (sub-)elements of this scope and all subscopes.
- When you work on *subscope* level you define the attributes for this subscope only.

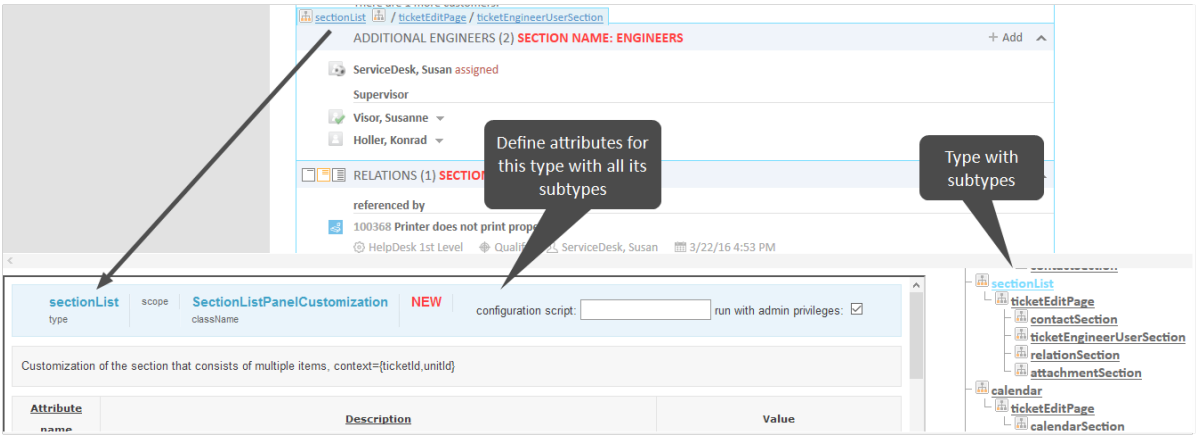


Figure 115: *Selecting the level for the Page Customization, here: type.*

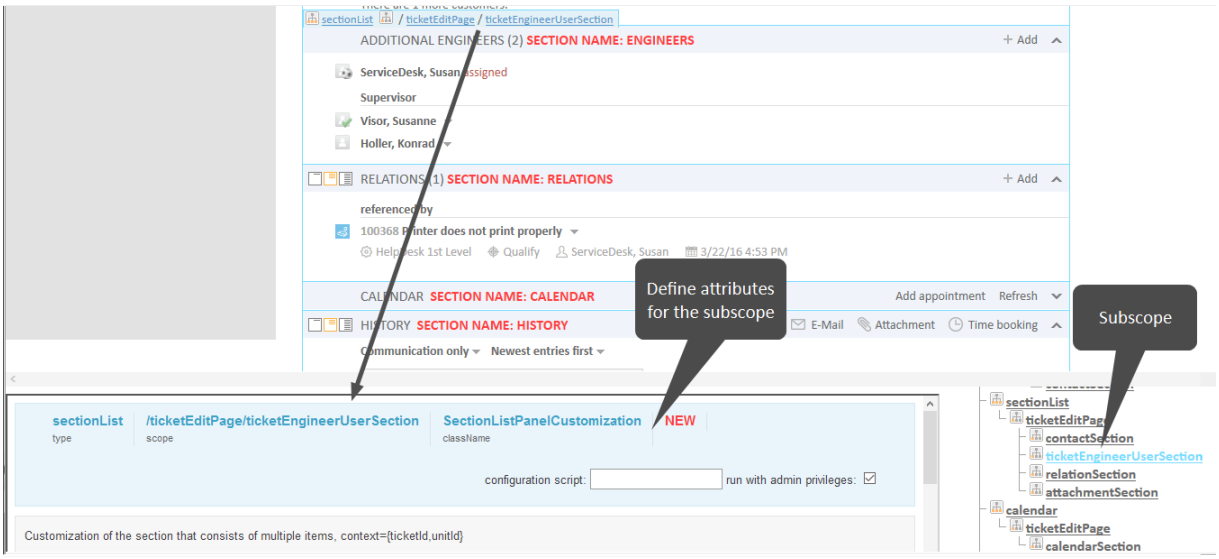


Figure 116: *Selecting the level for the Page Customization, here: subscope.*

Please see the following example for ticket list configuration.

- **Type level:**

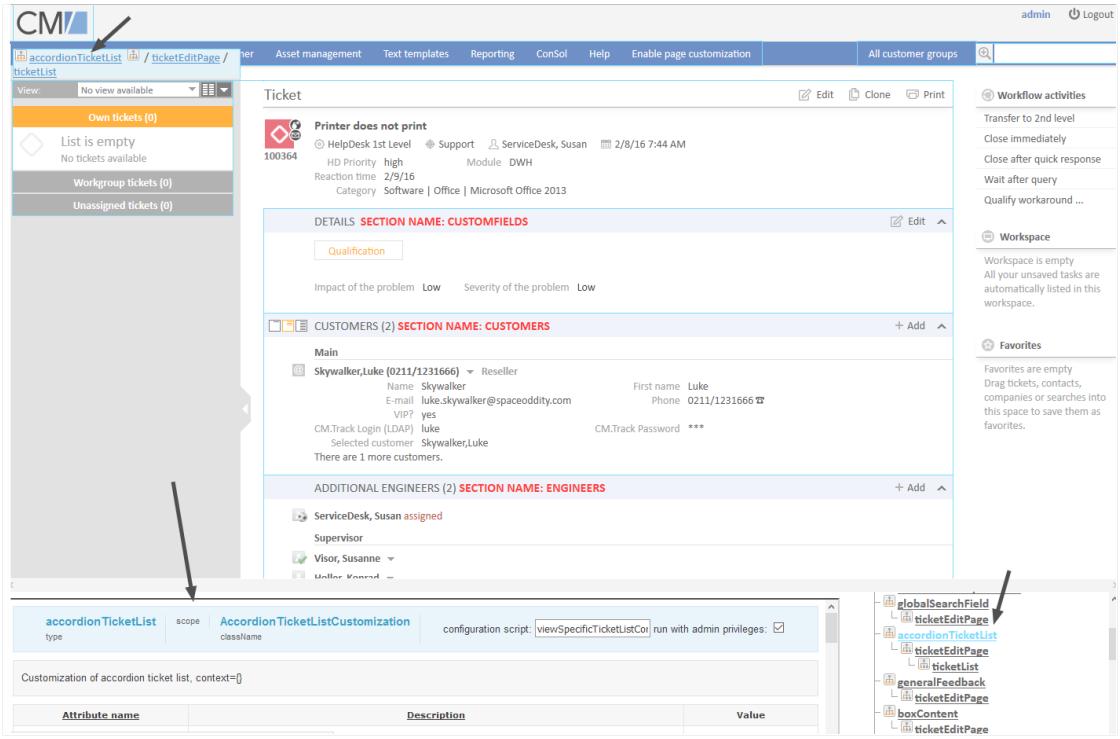


Figure 117: *ConSol CM Web Client - Defining attributes on Type level*

- **Scope level:**

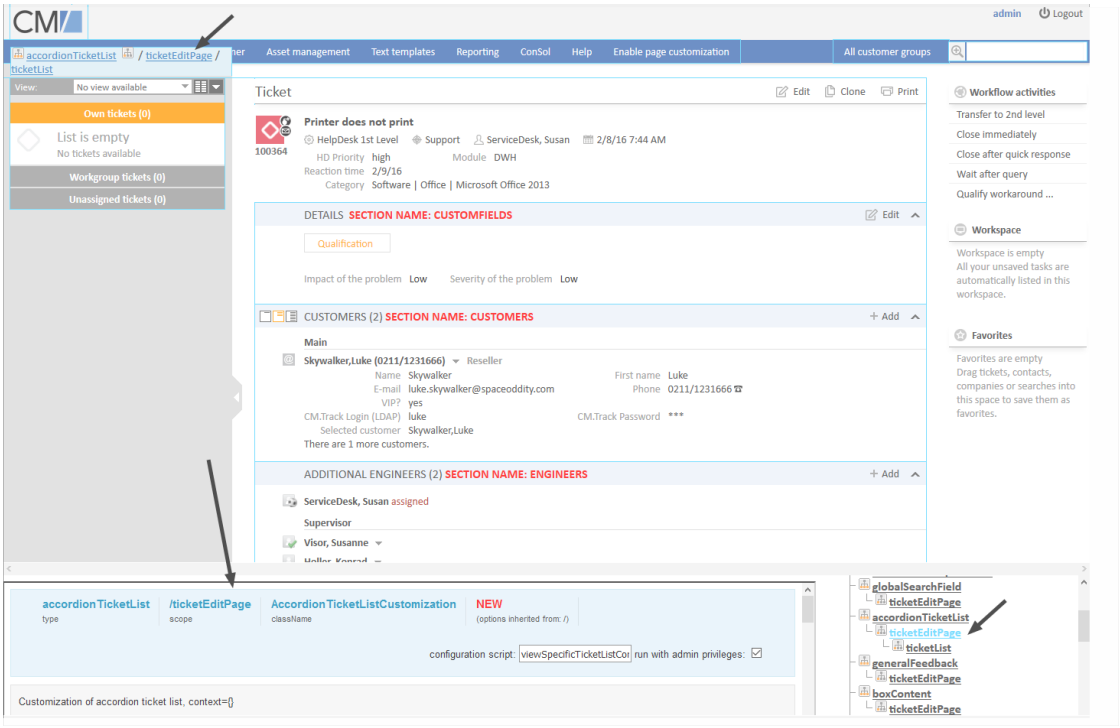


Figure 118: *ConSol CM Web Client - Defining attributes on Scope level*

- **Subscope level:**

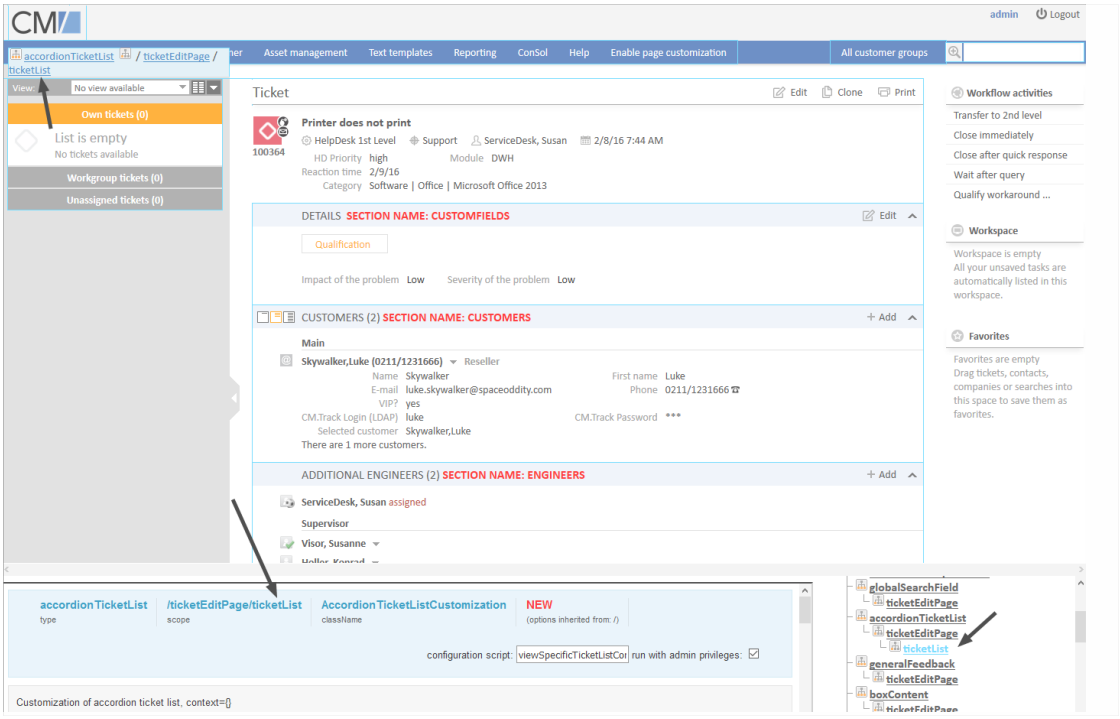


Figure 119: *ConSol CM Web Client - Defining attributes on Subscope level*

For some elements there may be two hierarchical paths, as shown in the following example for the subscope *contactSection*:

- Path 1: customerSectionPanel / ticketEditPage / contactSection
- Path 2: sectionList / ticketEditPage / contactSection

So, in this example, a *ticketEditPage / contactSection* subscope can be configured for two different types: *customerSectionPanel* and *sectionList*.

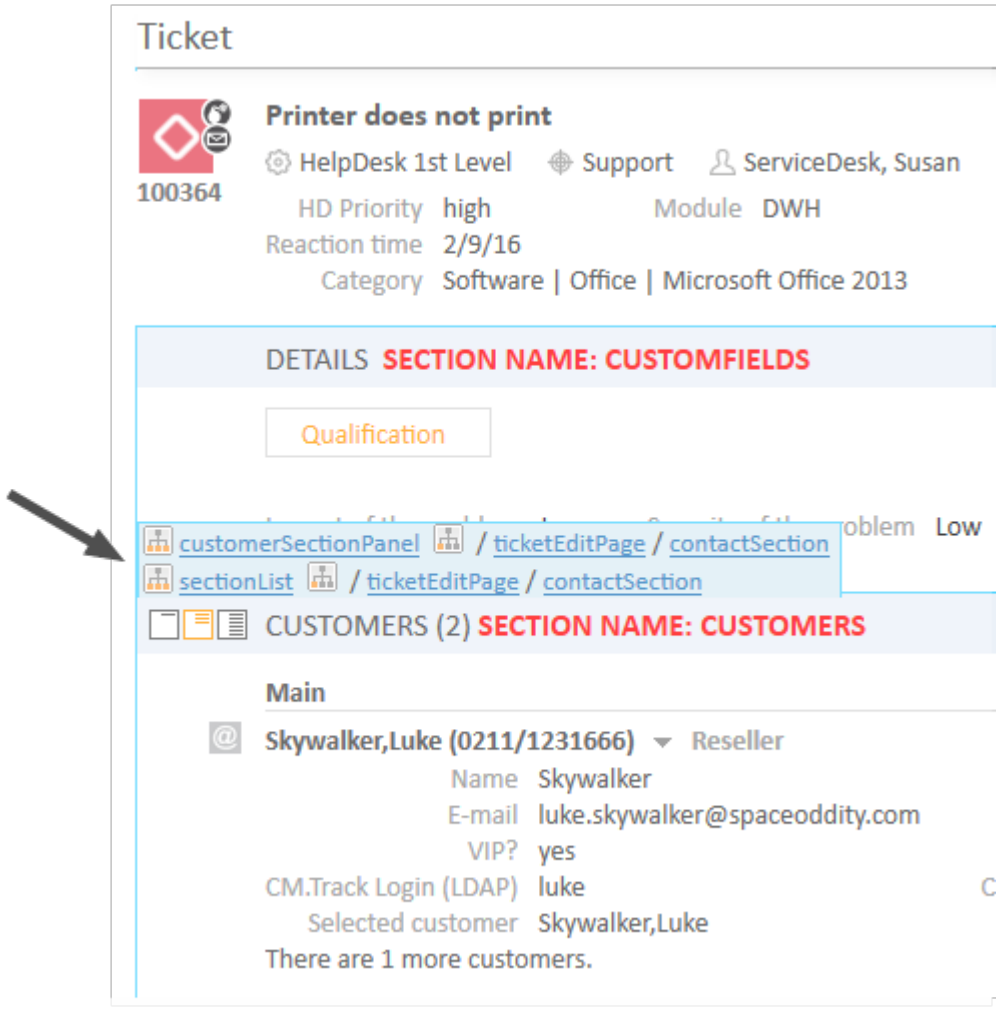


Figure 120: *ConSol CM Web Client - Two paths to the Page Customization element contactSection*

For every element, there is also a configuration script that can dynamically define values. The script is executed in the context of the engineer who is currently logged in. Therefore, the engineer permissions might exert an influence on the script result, e.g., when you use the code to select *all tickets*, the amount of tickets will be different depending on the queue permissions of the current engineer. When you need global access within the script, you can always execute it with admin privileges by ticking the respective checkbox. Within the script, the objects of the context (see following figure: example of *ticketId*) are available.

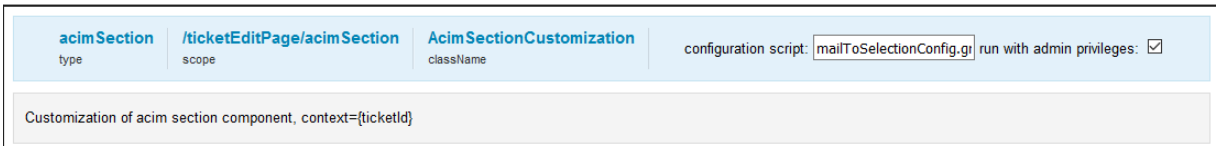


Figure 121: *ConSol CM Web Client - Configuration script for defining values*

For example, if there should be only one email recipient (here: the main customer) for medium and low priority tickets, but emails for a high priority ticket should be sent to all customers of the ticket, the following script can be used:

```
import com.consol.cmas.common.model.ticket.Ticket
import com.consol.cmas.common.model.customfield.enums.EnumValue

Ticket ticket = ticketService.getById(ticketId);
EnumValue value = ticket.get("helpdesk_standard.priority");
if (value != null && "high".equals(value.getName()))
return [mailToSelection: 'contacts'];
return [mailToSelection: 'contact'];
```

The script has to be stored in the *Scripts and Templates* section of the Admin Tool (see section Admin Tool Scripts for details) and its name has to be entered in the field *configuration script*.

The return values define the values for the attributes which can also be defined using the string input fields. Please note the order of all components involved, see section Order and Priorities of Page Customization.

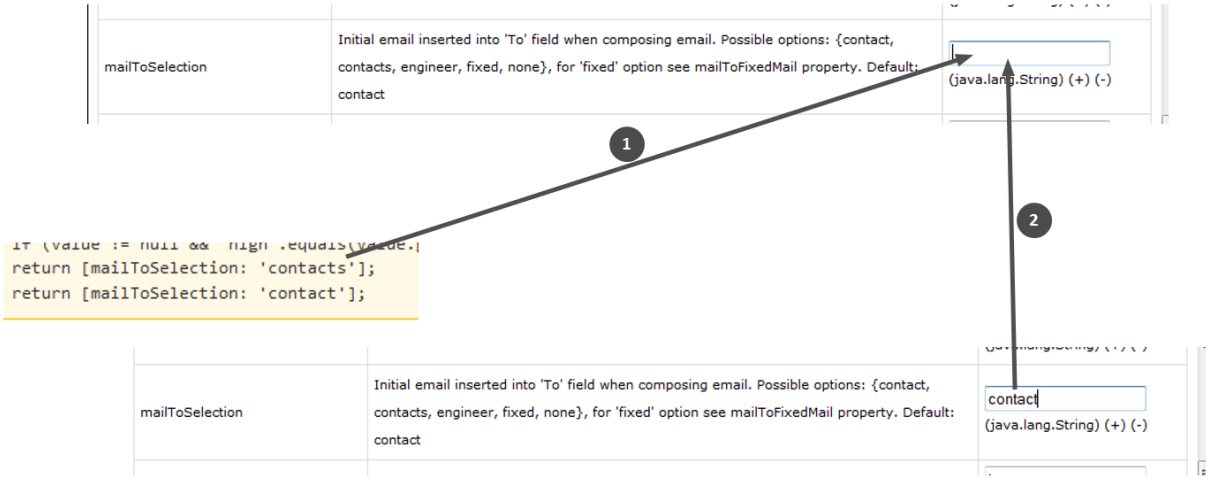


Figure 122: *Two alternatives for setting Page Customization attributes*

- Alternative 1: return attribute via script (1)
- Alternative 2: set attribute in string field (2)

Please note that in scripts Boolean values (“true”/“false”) might have to be returned. In case of ConSol CM Page Customization scripts, a syntax is used where those values have to be returned as strings!

Example: The value of the attribute *emailFeature* is “false”. In the script you will have to write:

```
emailavailable = [emailFeature:'false'] ... return emailavailable
```

C.8.2.1 Buttons for Setting Page Customization Attributes

Three buttons are available:

- **Create**
  Creates a new Page Customization (in the database). Use this to (re-)define attributes. All changes which you have made in the form will be saved.
- **Delete**
  Deletes the entries from the database for this type, scope or subscope, i.e. all attributes available in the form will be set to their original values.
- **Reset**
  Does not commit a database operation. Only reverses the entries which you have made while editing the form and which have not yet been saved.

C.8.3 Order and Priorities of Page Customization

In case more than one value is set for an attribute, the following hierarchy is applied:

1. **Highest priority:** script
2. **Medium priority:** scope definition
3. **Lowest priority:** type definition

Example for the value of *maxHints* in the *GlobalSearchField* on the *ticketEditPage*:

- **Variant A:**
  - script: no value
  - scope definition (GlobalSearchField/ticketEditPage): maxHints = 10
  - type definition (GlobalSearchField): maxHints = 5
  - => maxHints will be 10
- **Variant B:**
  - script: maxHints = 7
  - scope definition (GlobalSearchField/ticketEditPage): no value
  - type definition (GlobalSearchField): maxHints = 5
  - => maxHints will be 7
- **Variant C:**
  - script: no value
  - scope definition (GlobalSearchField/ticketEditPage): no value
  - type definition (GlobalSearchField): maxHints = 5
  - => maxHints will be 5

C.8.4 Page Customization Using Attributes

In the following sections, all configuration attributes for page customization will be explained. A short description is also given for each attribute in the editor section.

C.8.4.1 Possible Pages (Scopes) for Page Customization

The following main scopes are available. i.e., when you have opened the respective page you can configure page customization attributes which are visible only on this page for the given attribute.

companyEditPage

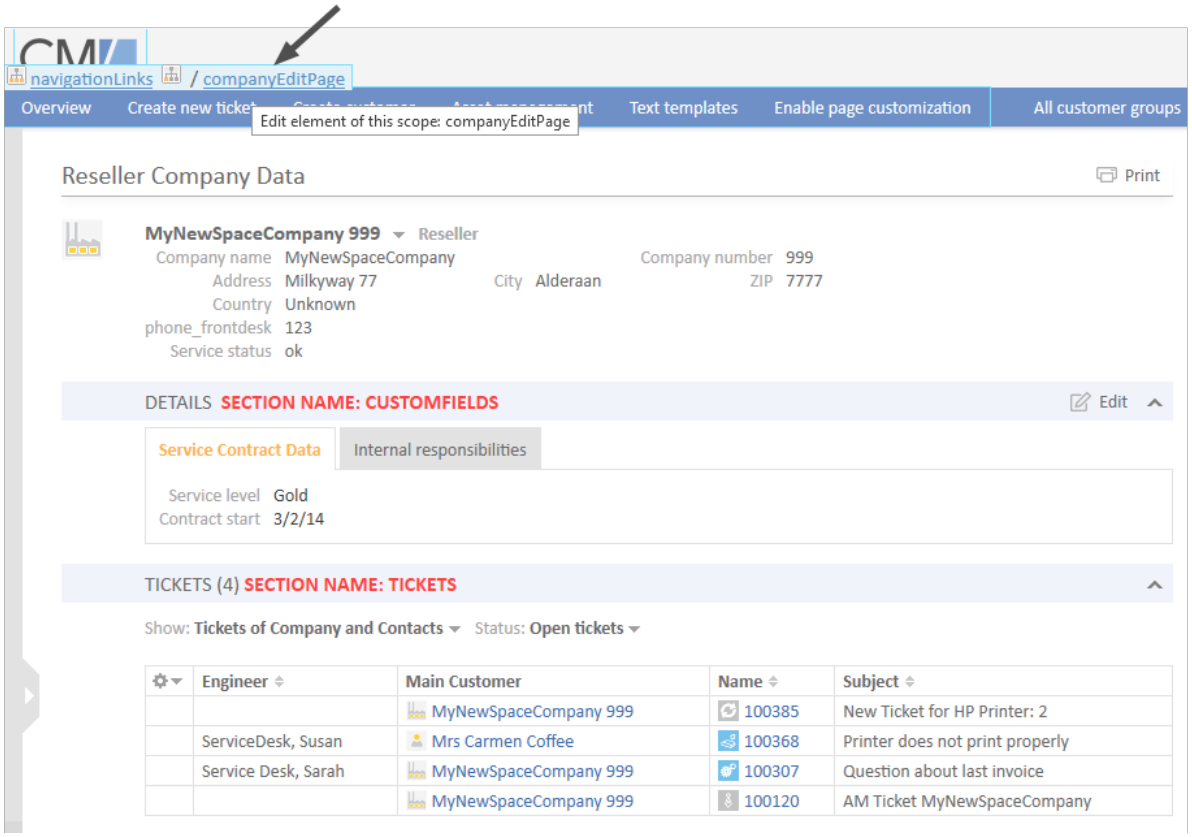


Figure 123: *ConSol CM Web Client - Page Customization for the company page*

contactCreatePage

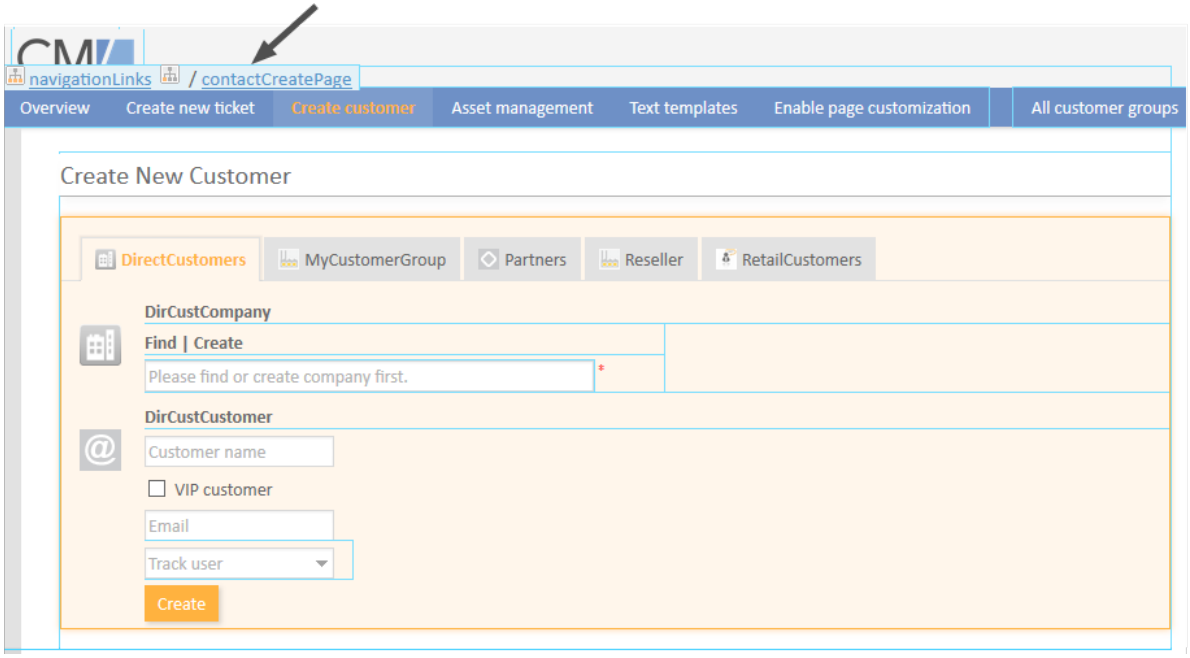


Figure 124: *ConSol CM Web Client - Page Customization for the Create New Customer page*

contactEditPage

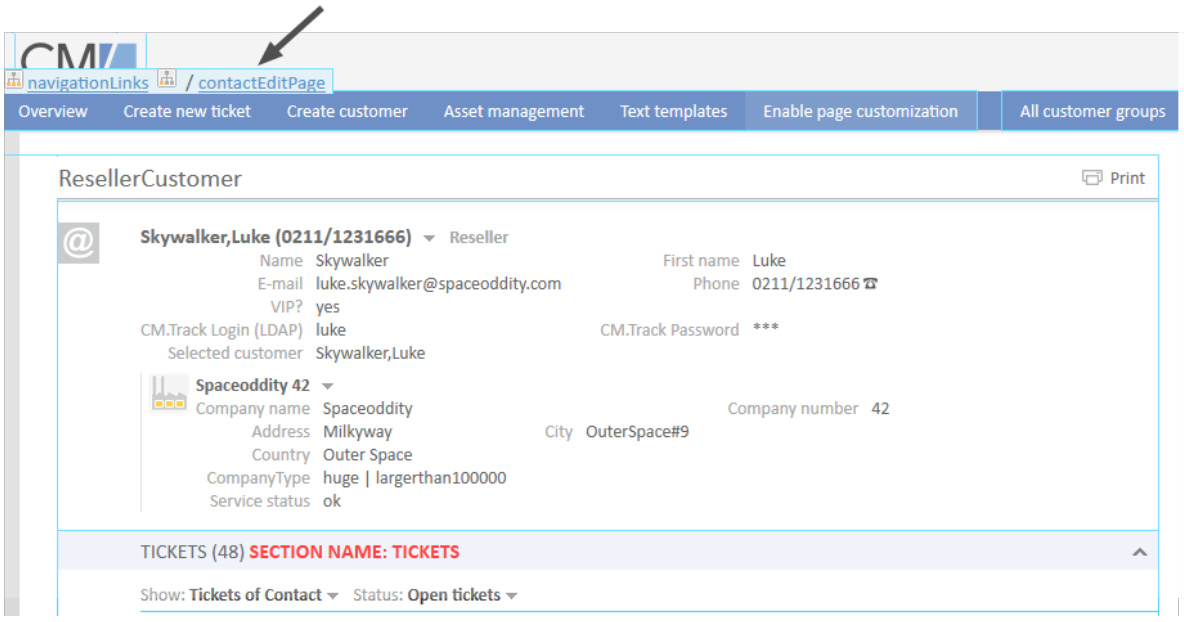


Figure 125: *ConSol CM Web Client - Page Customization for the contact page*

resourceDashboard

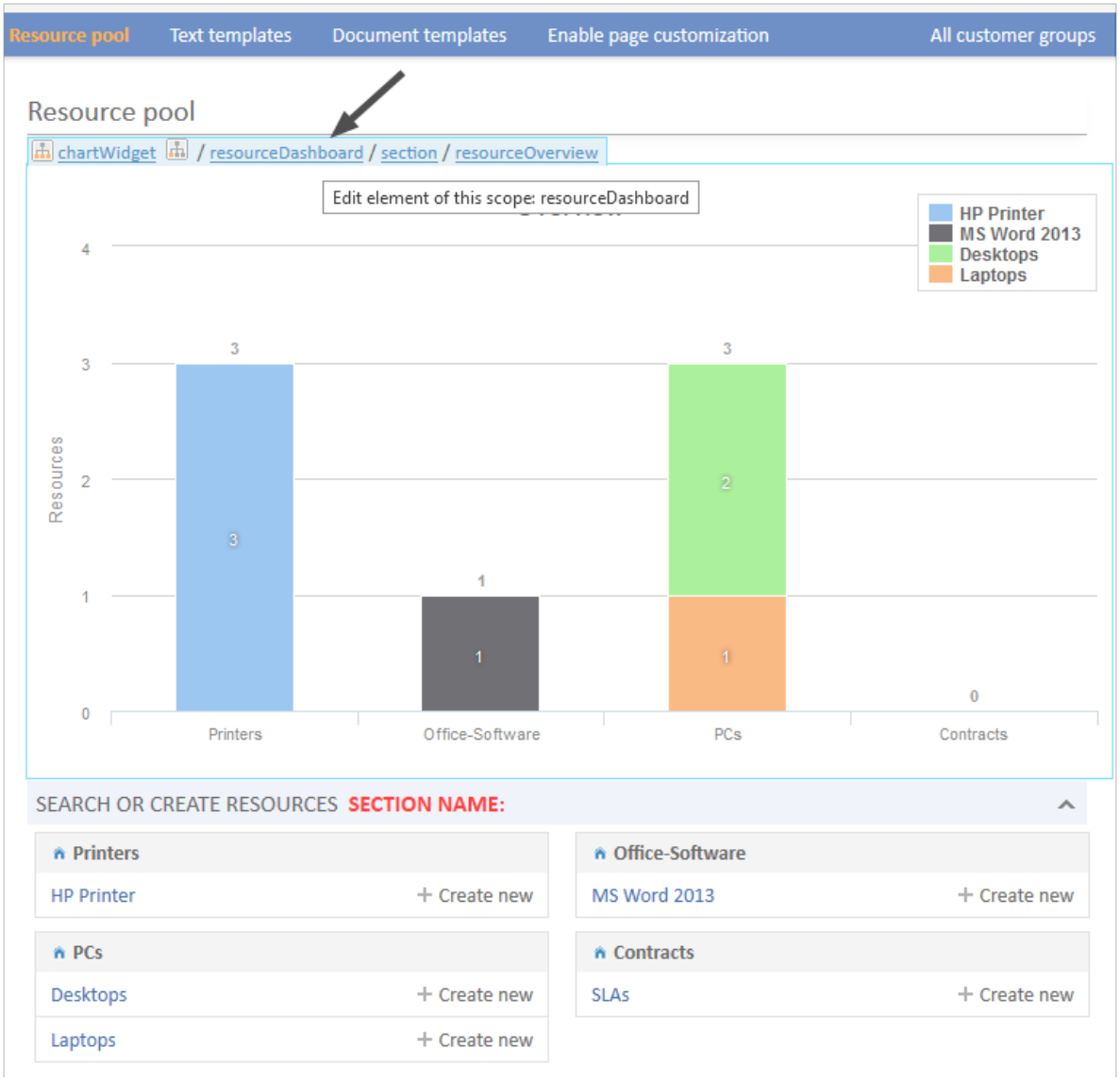


Figure 126: *ConSol CM Web Client - Page Customization for the Resource Pool Dashboard*

resourceType (Resource Type Page)

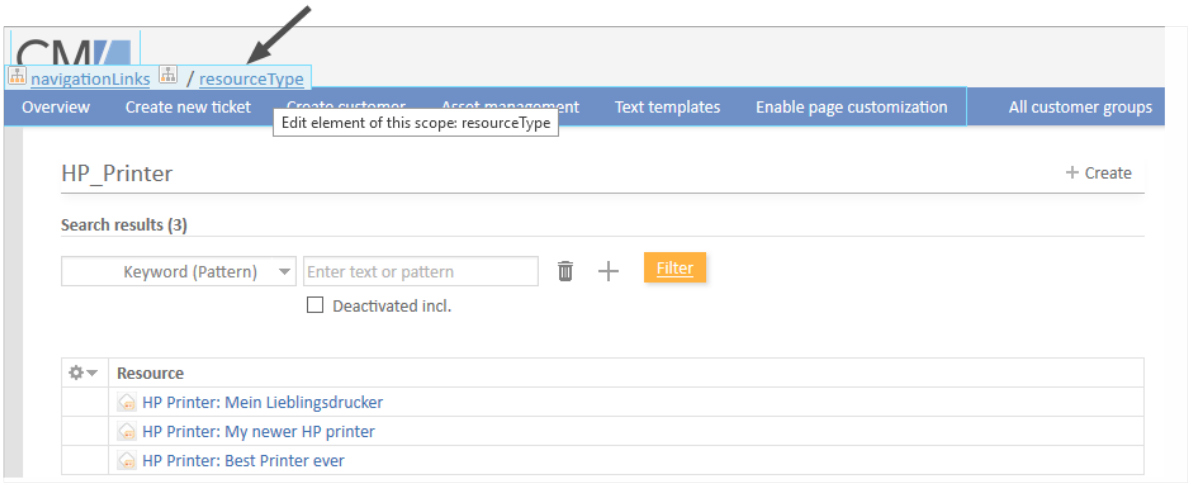


Figure 127: *ConSol CM Web Client - Page Customization for the resource type page*

resource (Resource Page)

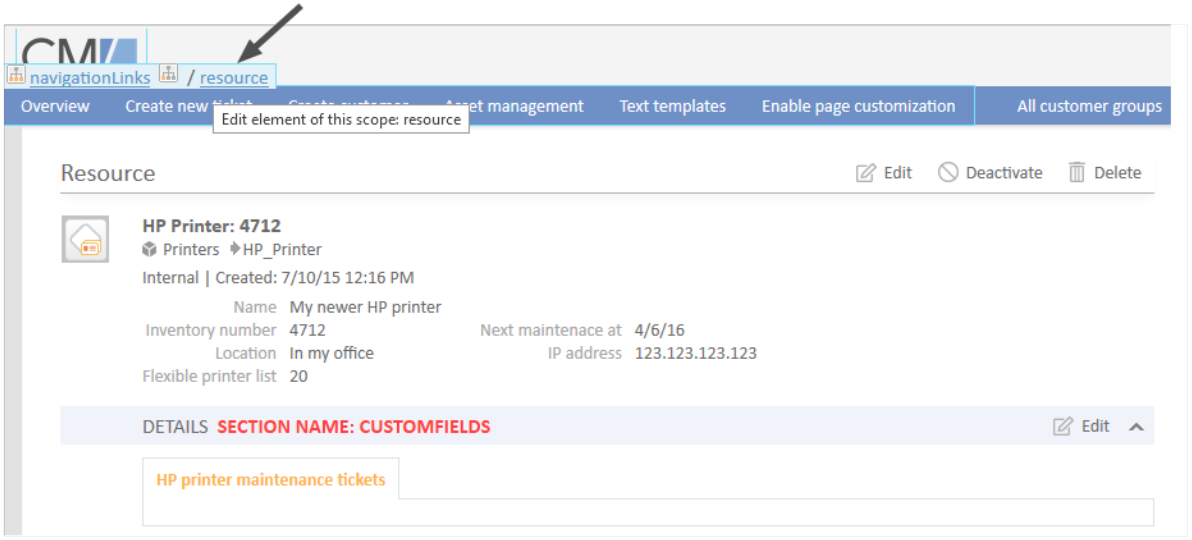


Figure 128: *ConSol CM Web Client - Page Customization for the resource page*

officeTemplatePage (Only When CM/Doc Is Activated)

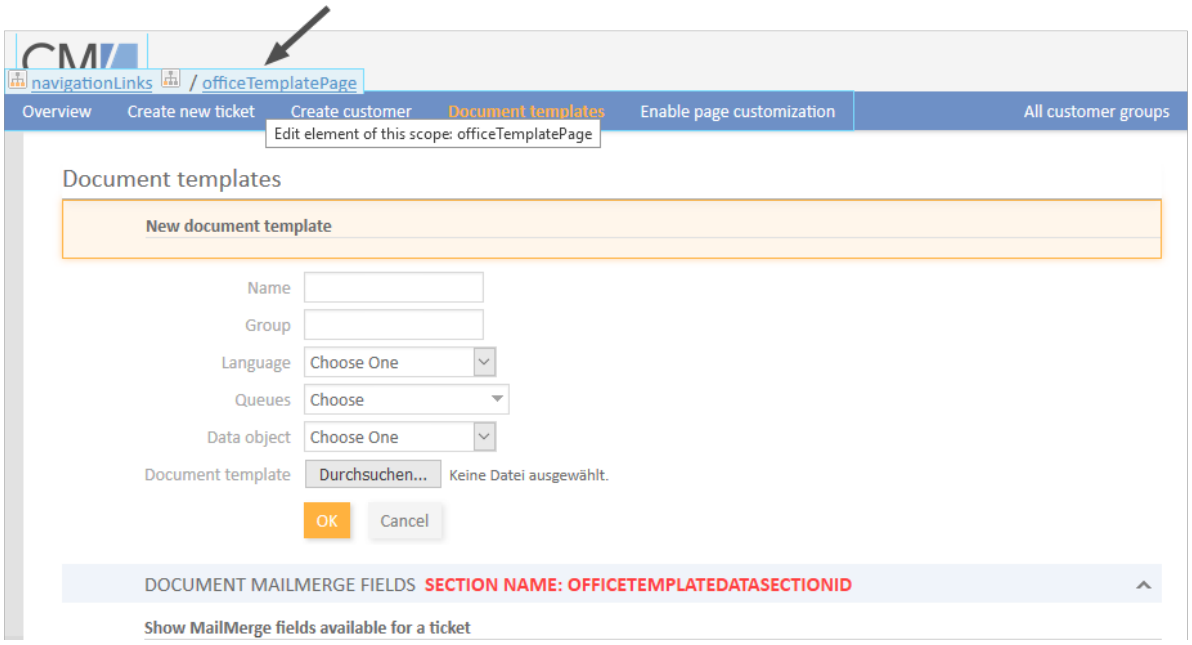


Figure 129: *ConSol CM Web Client - Page Customization for the Document templates page (only when CM/Doc is activated)*

searchDetailPage

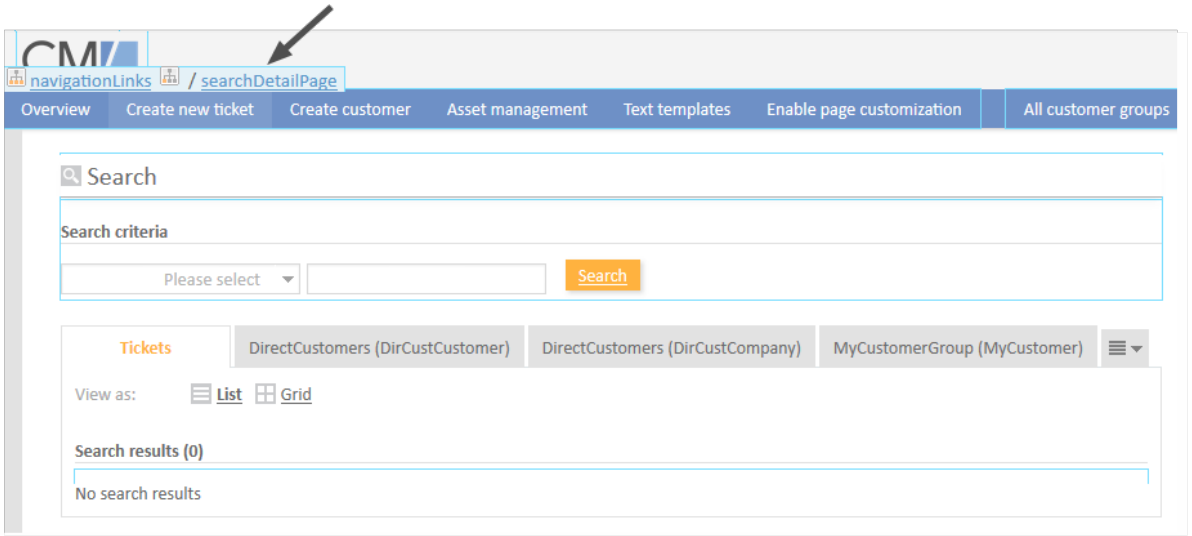


Figure 130: *ConSol CM Web Client - Page Customization for the Detailed Search*

templateViewPage

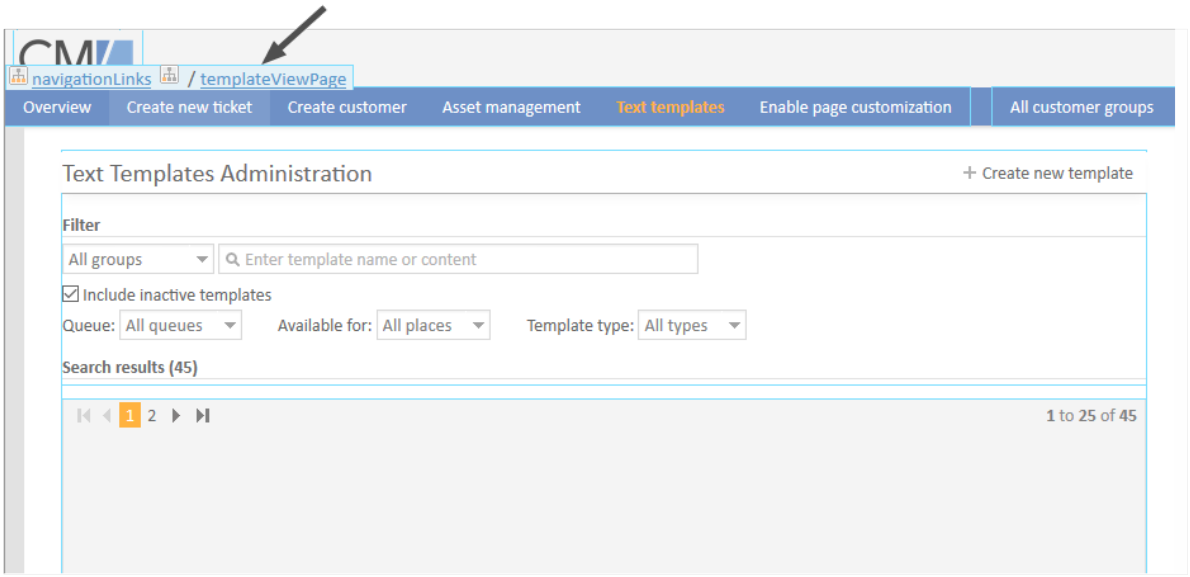


Figure 131: *ConSol CM Web Client - Page Customization for the Text templates page (view mode)*

templateEditPage

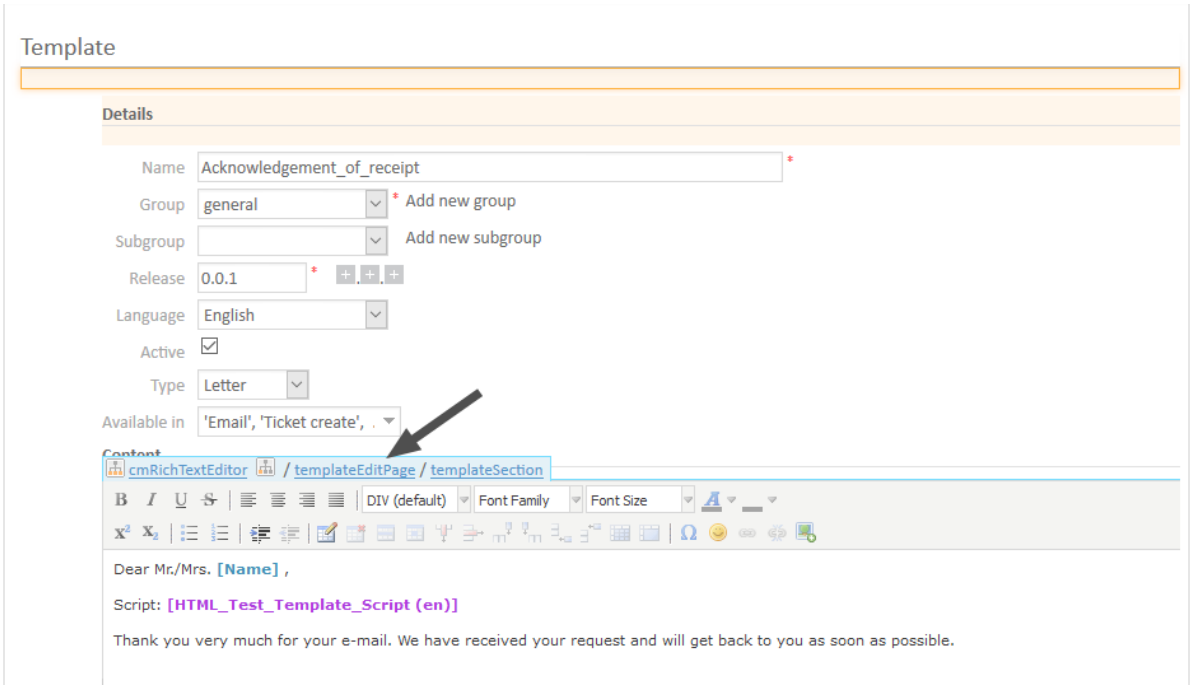


Figure 132: *ConSol CM Web Client - Page Customization for the Text templates page (edit mode)*

ticketCreatePage

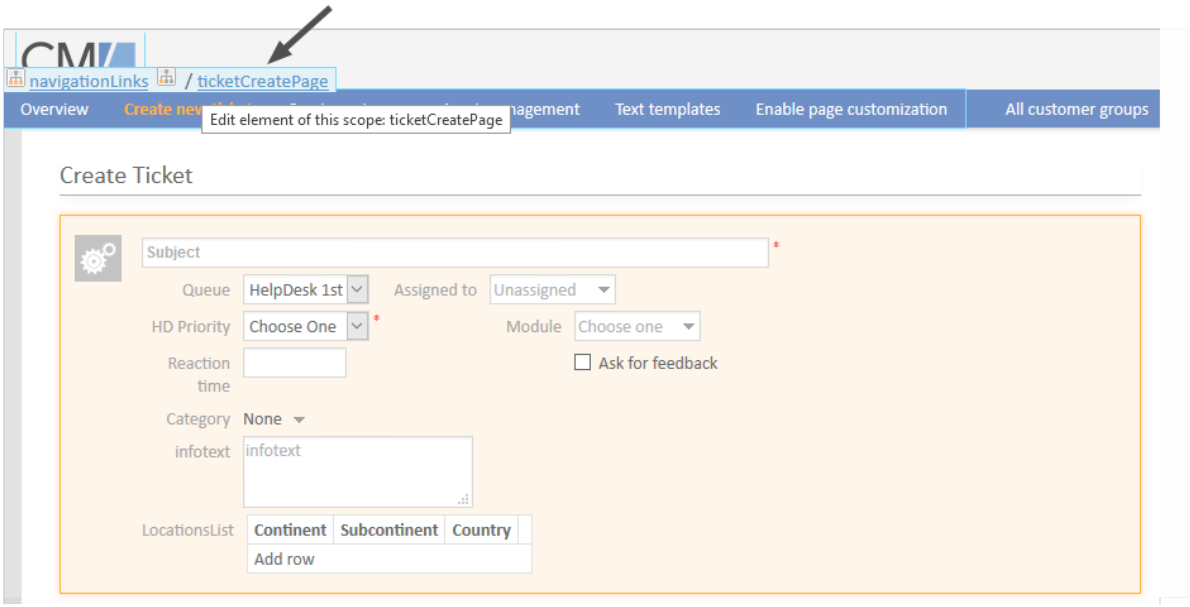


Figure 133: *ConSol CM Web Client - Page Customization for the New ticket page*

ticketEditPage

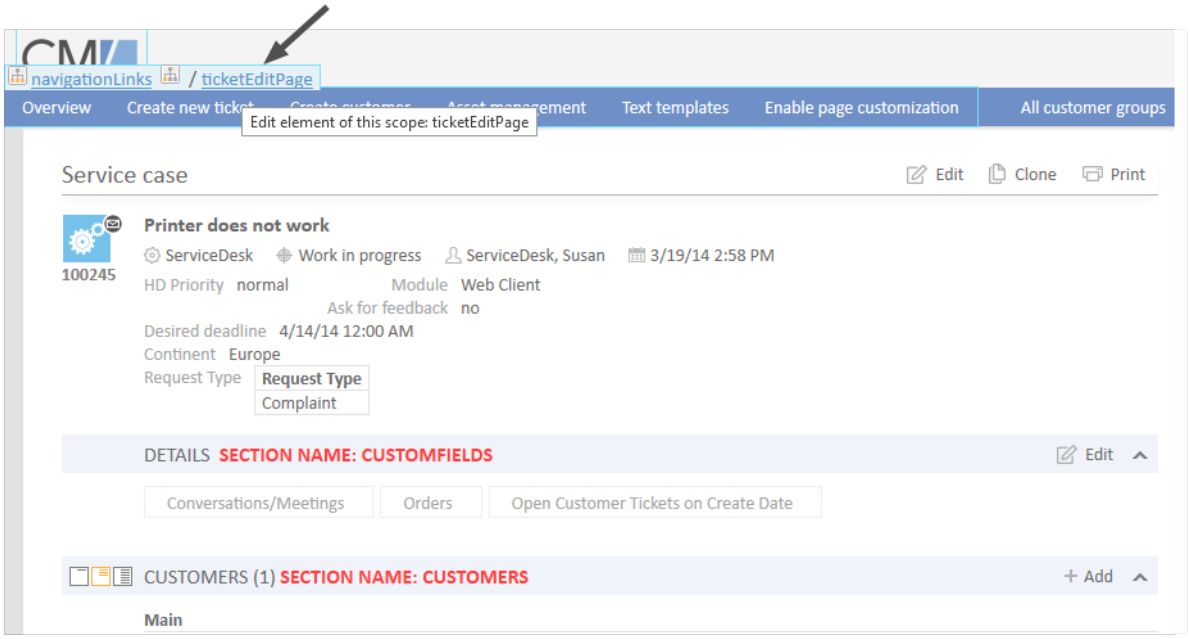


Figure 134: *ConSol CM Web Client - Page Customization for the ticket page*

userProfilePage

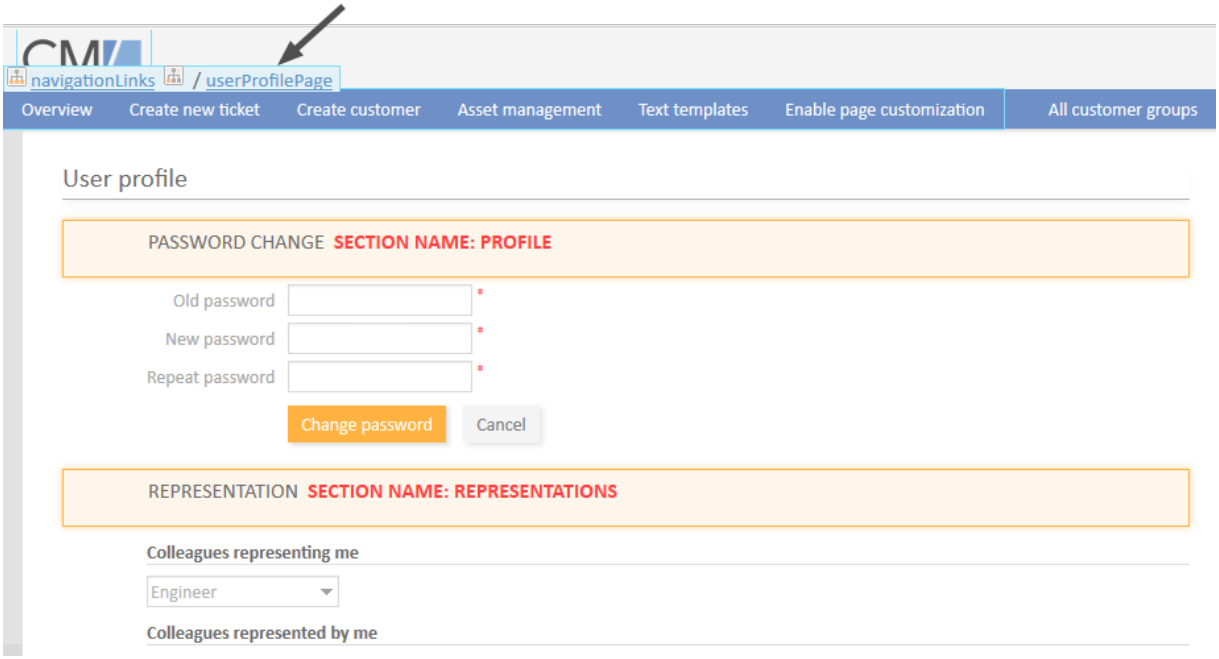


Figure 135: *ConSol CM Web Client - Page Customization for the engineer profile*

welcomePage (Overview Page)

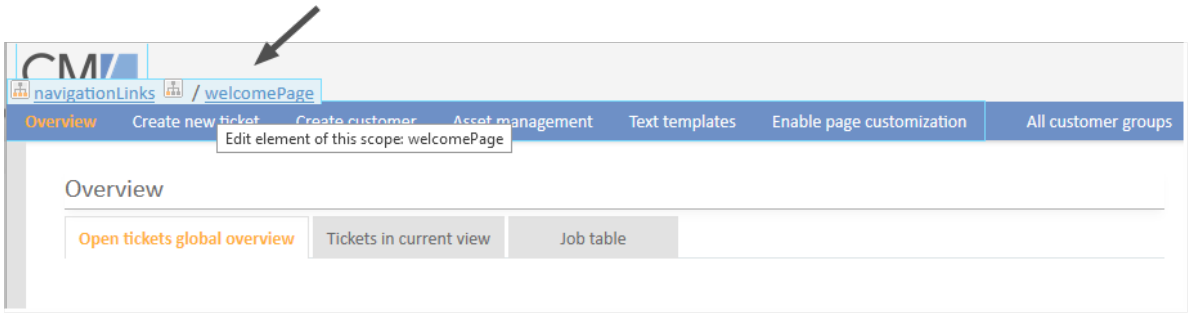


Figure 136: *ConSol CM Web Client - Page Customization for the Welcome page*

For example, for the type *globalSearchField* (see subsequent section) the following page-specific scopes can be configured. That means the behavior of the *globalSearchField* (type) can be configured for each of the following pages (scopes) where it is available:

- globalSearchField/companyEditPage
- globalSearchField/contactCreatePage
- globalSearchField/contactEditPage
- globalSearchField/resource
- globalSearchField/resourceDashboard

- globalSearchField/resourceType
- globalSearchField/searchDetailPage
- globalSearchField/ticketCreatePage
- globalSearchField/ticketEditPage
- globalSearchField/welcomePage

C.8.5 Page Customization Attributes for Types, Scopes and Subscopes, in Alphabetical Order

The following sections contain a list of Page Customization types, scopes and subscopes. The attributes are described on the respective page.

accordionTicketList (Type)

Here you can define attributes for the ticket list. Only one subscope is available: *ticketList*

Attributes:

- **hideEmptyGroups**
  Defines whether empty groups are hidden from the ticket list. If set to “false” (default), all groups are displayed. If set to “true”, empty groups are hidden.

  Note that a ticket list group is not hidden if it has the focus. In this case the group is displayed when the ticket list is loaded, and hidden once the user clicks another group.
- **loadingTicketListMode**
  The mode used to render the ticket list. There are four options to select from:

  1. LAZY_SYNCH (default)
     The waiting indicator is shown in the place of the ticket list while the rest of a page is rendered, then the ticket list is loaded. The main benefit of this approach is the possibility to show/read the main content faster.
  2. LAZY_ASYNCH
     (deprecated since 6.8.2, will be treated as LAZY_SYNCH mode)
     The modification of the LAZY_SYNCH strategy. It does not wait when the page is being rendered but sends the second request and then loads the ticket list. This strategy will load the ticket list faster but may reduce the benefit of having the main page immediately.
  3. INCLUDED
     The ticket list is loaded along with the rest of a page.
  4. LAZY_ASAP
     The waiting indicator is shown in the place of the ticket list while the rest of a page is rendered. The request for the ticket list is sent as soon as two libraries have been loaded. In this approach a request for the ticket list is sent and processed concurrently with the first request. The ticket list will appear much faster on the page.
     (java.lang.String)
- **mainCustomerDescriptionVisible**
  The page customization attribute *accordionTicketList.**mainCustomerDescriptionVisible***={true, false} replaces the annotation `show-contact-in-ticket-list` (which is valid until CM version 6.8). When this value is set to “true”, the customer data of the main customer is displayed in the ticket list representation of the ticket. Default is “true”. (boolean)
- **quickAssignLinkShowsTicketPageFlag**
  Whether the quick assign link (represented by the arrow rendered for each unassigned ticket) opens the assigned ticket immediately (boolean, default is “false”).
- **ticketDataConfigQueueGroupingScript**
  Defines the ticket information display when grouping by queue is selected. See section Configuration of the Ticket List.

- **ticketDataConfigEngineerGroupingScript**
  Defines the ticket information display when grouping by engineer is selected. See section Configuration of the Ticket List.
- **ticketDataConfigCustomGroupingScript**
  Defines the ticket information display when grouping by a ticket field is selected. See section Configuration of the Ticket List.
- **ticketDataConfigNoGroupingScript**
  Defines the ticket information display when no grouping criterion is selected. See section Configuration of the Ticket List.

acimSection (Type or Subscope)

An *ACIM* (activity item) is an entry in the *History* section of a ticket. This can be ...

- a comment
- an email which has been sent from the ticket
- an email which has been received by the ticket
- an attachment
- a time booking entry

An ACIM group is a group of entries which has a distinct date/time stamp. An ACIM item is a single entry within the ACIM group. It has only a time stamp.

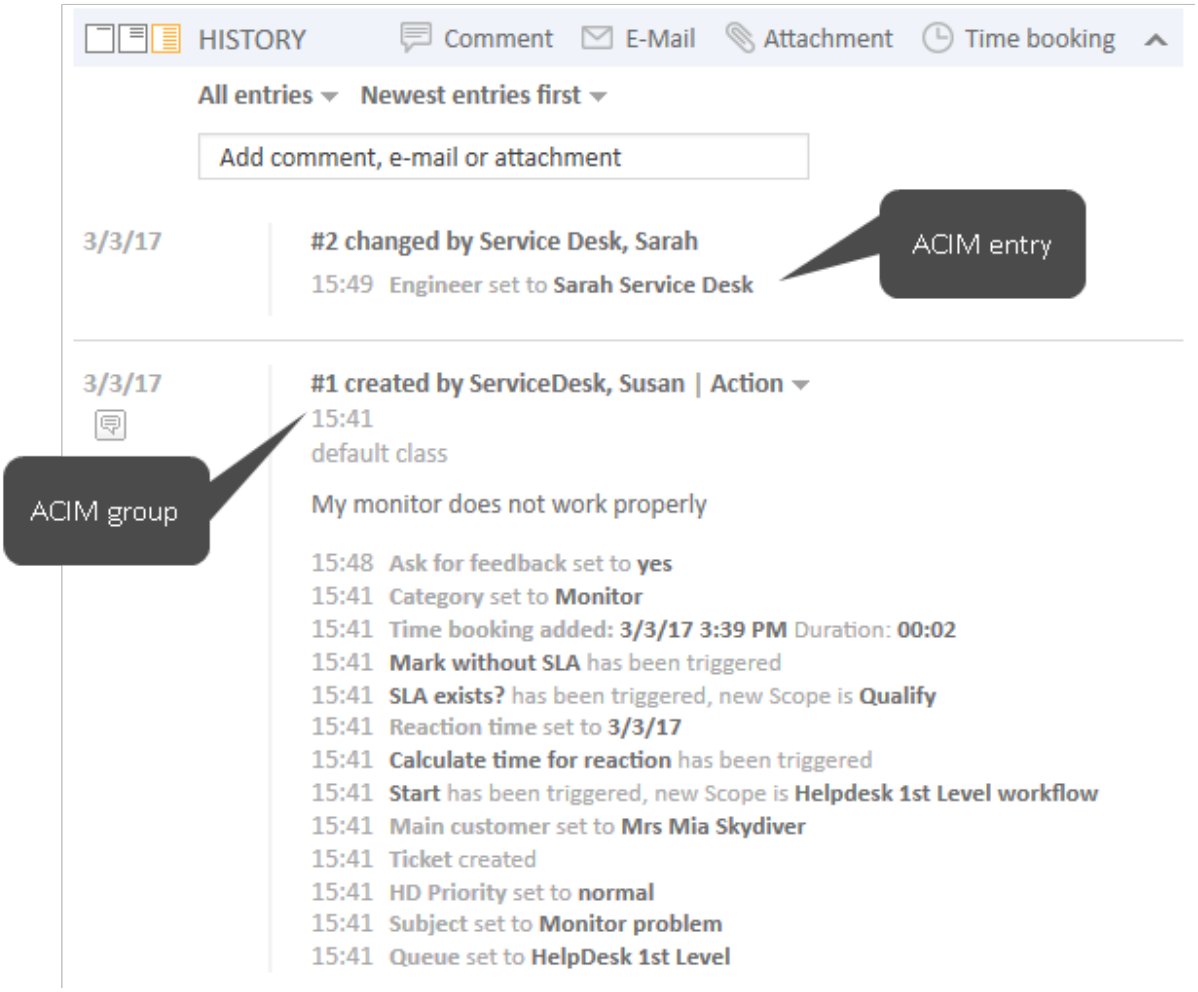


Figure 137: *ConSol CM Web Client - ACIM group and item*

Please make sure that the date format you have entered for one of the following date attributes is correct! If the date format is not correct, the entire page cannot be displayed! The Web Client will not work! Please see the correct date formats in the following table. By using an empty text (' ') as value it is possible to hide the date/time stamp completely.

Letter	Date or Time Component	Examples
G	Era designator	AD
y	Year	1996; 96
M	Month in year	July; Jul; 07
w	Week in year	27
W	Week in month	2
D	Day in year	189
d	Day in month	10
F	Day of week in month	2
E	Day in week	Tuesday; Tue
a	Am/pm marker	PM
H	Hour in day (0-23)	0
k	Hour in day (1-24)	24
K	Hour in am/pm (0-11)	0
h	Hour in am/pm (1-12)	12
m	Minute in hour	30
s	Second in minute	55
S	Millisecond	978
z	Time zone	Pacific Standard Time; PST; GMT-08:00
Z	Time zone RFC 822	-0800

Figure 138: *ConSol CM Web Client - Valid date formats for ACIM date configuration*

Attributes:

- **acimGroupActionEntryDateFormat**
  Date format for group of ACIM without text or email entry (i.e., for automatic actions). If nothing has been entered as pattern, the default one will be used.
  Syntax: dateFormatFirstLevelOfDetails|secondLevel|thirdLevel
  (java.lang.String, default = dd.MM.yyyy HH.mm|dd.MM.yyyy HH.mm|dd.MM.yyyy HH.mm)

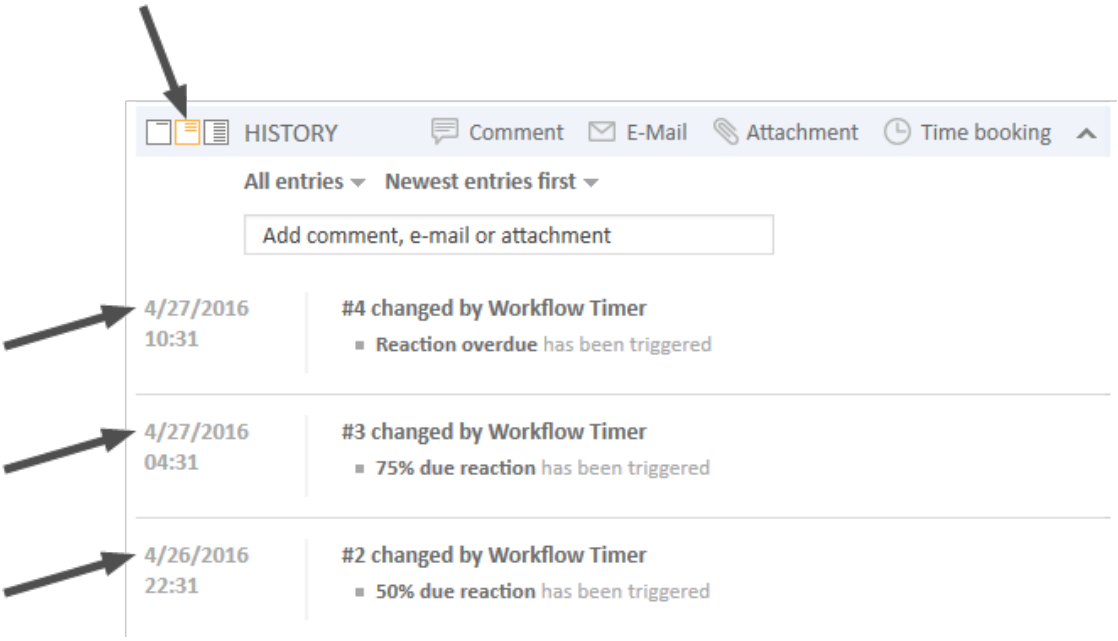


Figure 139: *ConSol CM Web Client - Display for format: dd.MM.yyyy HH.mm|dd.MM.yyyy HH.mm|dd.MM.yyyy HH.mm*

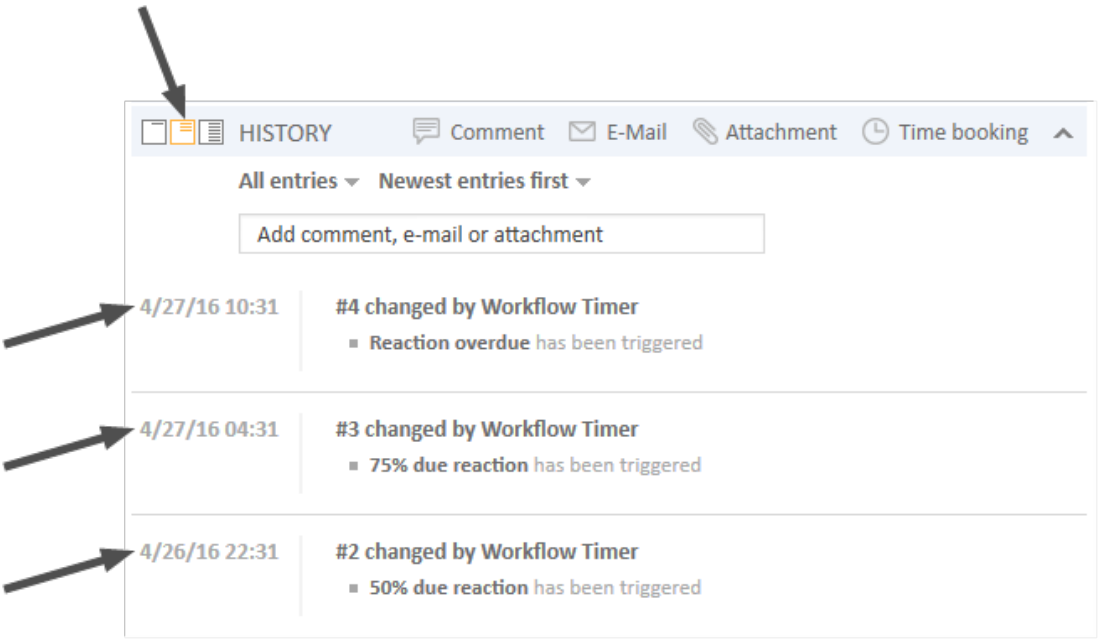


Figure 140: *ConSol CM Web Client - Display for format: dd.MM.yy HH.mm|dd.MM.yy HH.mm|dd.MM.yy HH.mm*

- **acimGroupTextEntryDateFormat**
  Date format for group of ACIM with text, email, or attachment entry. If nothing has been entered as pattern, the default one will be used.
  Syntax: dateFormatFirstLevelOfDetails|secondLevel|thirdLevel
  (java.lang.String, default = dd.MM.yyyy HH.mm|dd.MM.yyyy HH.mm|dd.MM.yyyy HH.mm)

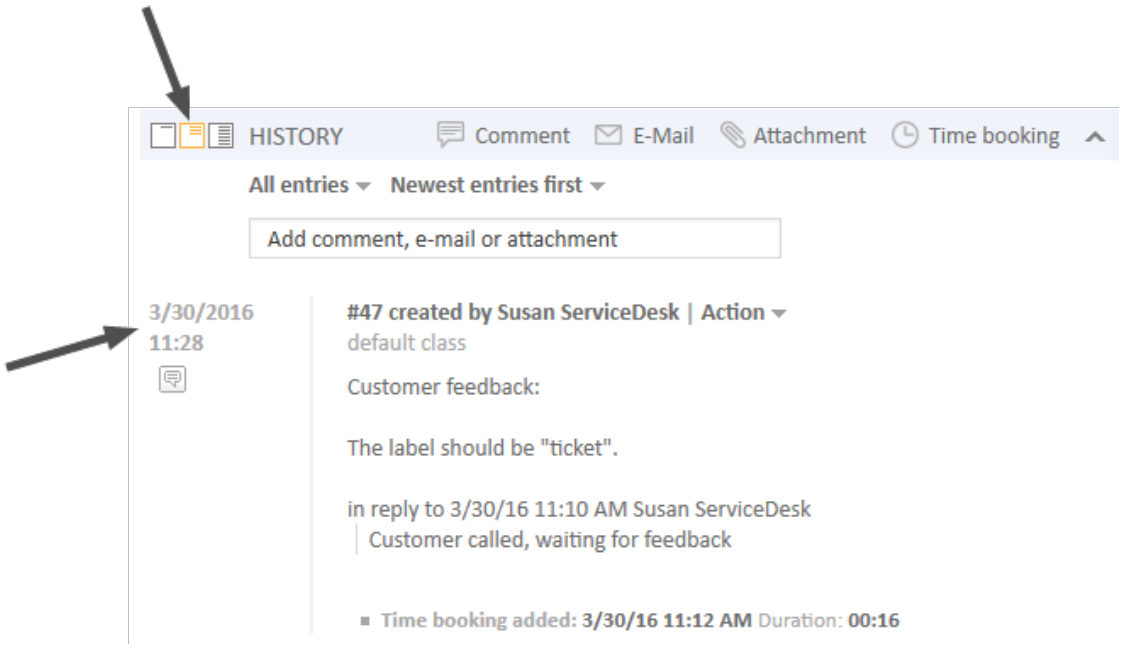


Figure 141: *ConSol CM Web Client - Display for format: dd.MM.yyyy HH.mm|dd.MM.yyyy HH.mm|dd.MM.yyyy HH.mm*

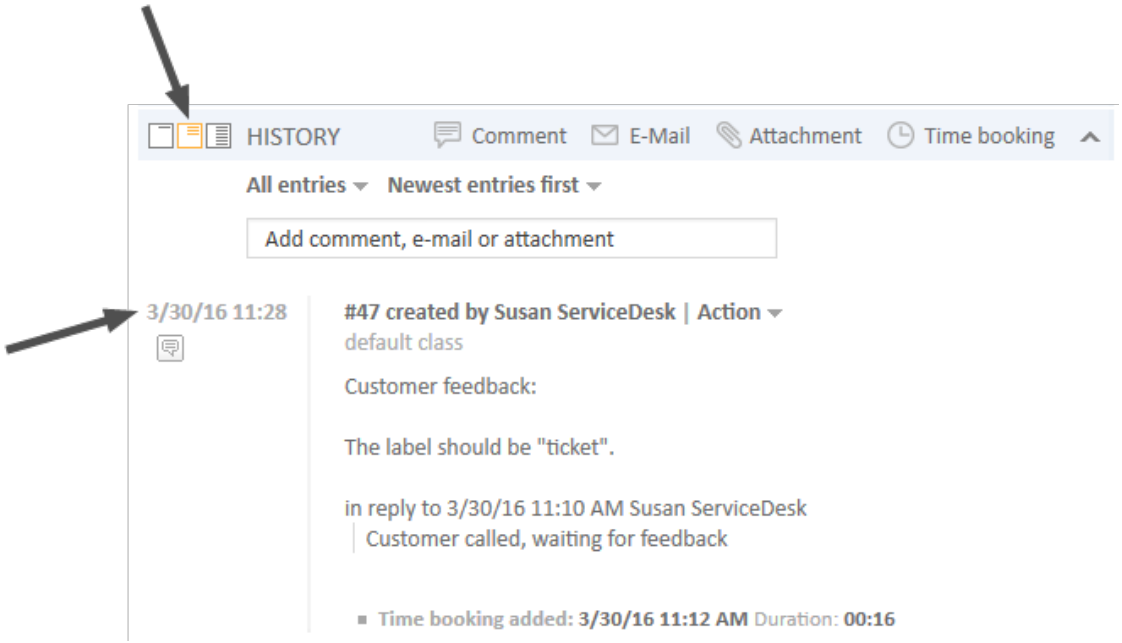


Figure 142: *ConSol CM Web Client - Display for format: dd.MM.yy HH.mm|dd.MM.yy HH.mm|dd.MM.yy HH.mm*

- **acimItemActionEntryDateFormat**
  Date format for item of ACIM entry. If nothing has been entered as pattern, the default one will be used.
  Syntax: dateFormatFirstLevelOfDetails|secondLevel|thirdLevel
  (java.lang.String, default = dd.MM.yyyy HH.mm|dd.MM.yyyy HH.mm|dd.MM.yyyy HH.mm)
- **acimItemTextEntryDateFormat**
  Date format for text or email entry. If nothing has been entered as pattern, the default one will be used.
  Syntax: dateFormatFirstLevelOfDetails|secondLevel|thirdLevel
  (java.lang.String, default = dd.MM.yyyy HH.mm|dd.MM.yyyy HH.mm|dd.MM.yyyy HH.mm)

- **showCloneOption**
  Enables clone option for text ACIM entry (comment or email entry). (boolean)

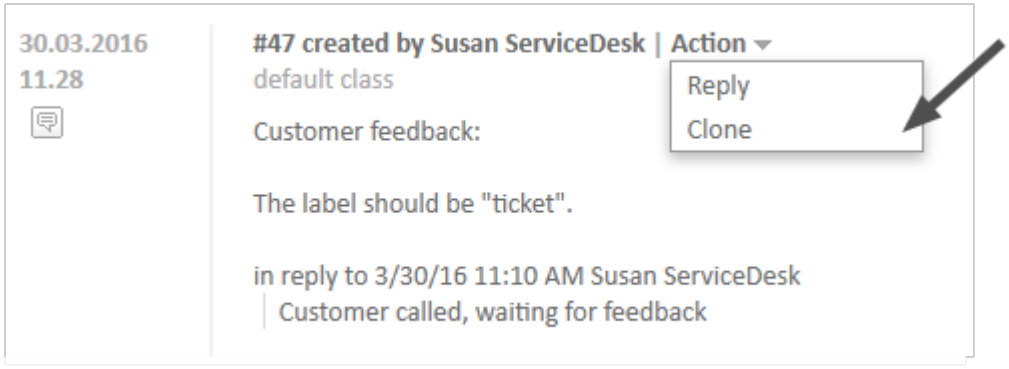


Figure 143: *ConSol CM Web Client - Clone option for text ACIM entry*

- **appendOrReplaceOnClone**
  Works only if clone option is set to “true”. If the editor is opened and already contains some text, you can append or replace its content when the clone of another item is selected simultaneously. Possible values are “append”, “replace”. Default is “append”. (java.lang.String)
- **attachmentDeletionAllowedManuallyUploaded**
  Boolean. Enables the delete functionality (entry in context menu) for ticket attachments which have been uploaded manually as a file. Default “true”.
- **attachmentDeletionAllowedIncomingEmail**
  Boolean. Enables the delete functionality (entry in context menu) for attachments from incoming emails. Default “false”.
- **attachmentDeletionAllowedOutgoingEmail**
  Boolean. Enables the delete functionality (entry in context menu) for attachments from outgoing emails sent from the system. Default “false”

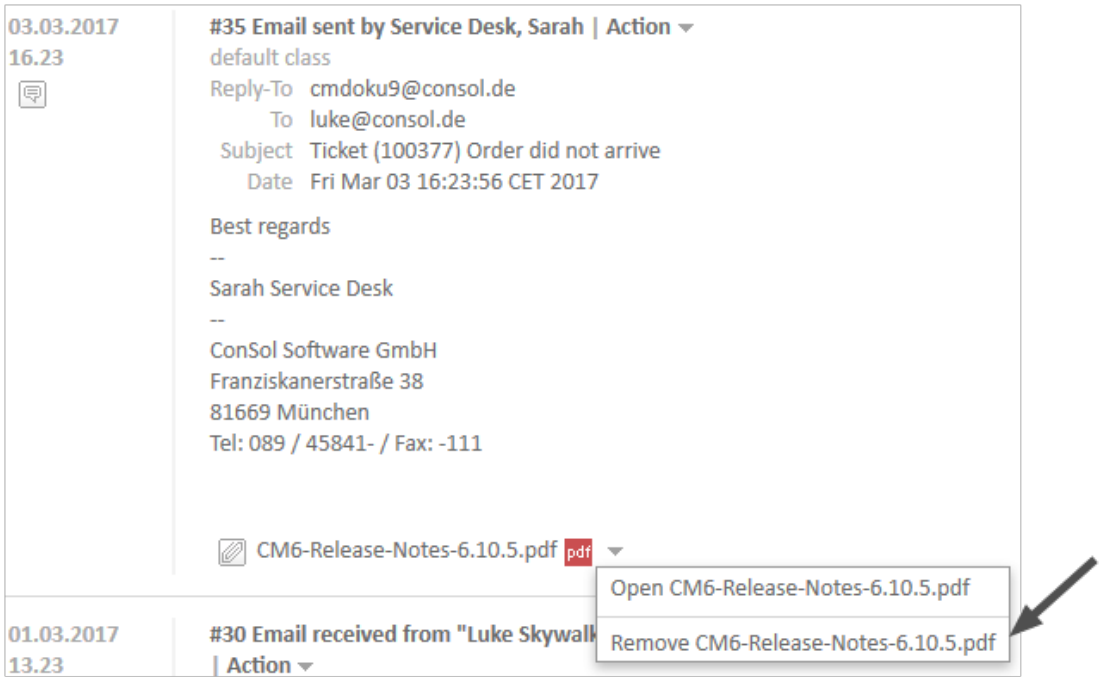


Figure 144: *Excerpt of a ticket page: attachmentDeletionAllowedManuallyUploaded set to true, manually uploaded attachment can be removed/deleted*

- **headHistoryElementsCount**
  Lazy loading - Number of groups in ACIM section that will be loaded from the top of the history. Default number is “0” (= lazy loading switched off). Customization works only when configured by type, not location. If head and tail elements count is “0”, then all history is loaded at once. (int)
- **tailHistoryElementsCount**
  Lazy loading - Number of groups in ACIM section that will be loaded from the bottom of the history. Default number is “0” (= lazy loading switched off). Customization works only when configured by type, not location. If head and tail elements count is “0”, then all history is loaded at once. (int)

The page customization attributes *headHistoryElementsCount* and *tailHistoryElementsCount* are available on three different scope levels:

- *acimSection* type on top level without scope
- *acimSection* type with */ticketEditPage* scope
- *acimSection* type with */ticketEditPage/acimSection* (sub-)scope

A non-zero value activating the lazy loading feature must be set either on the top level without scope alone or on all three levels.

Other scope uses may lead to unwanted behavior.

Example 1: Lazy loading switched off

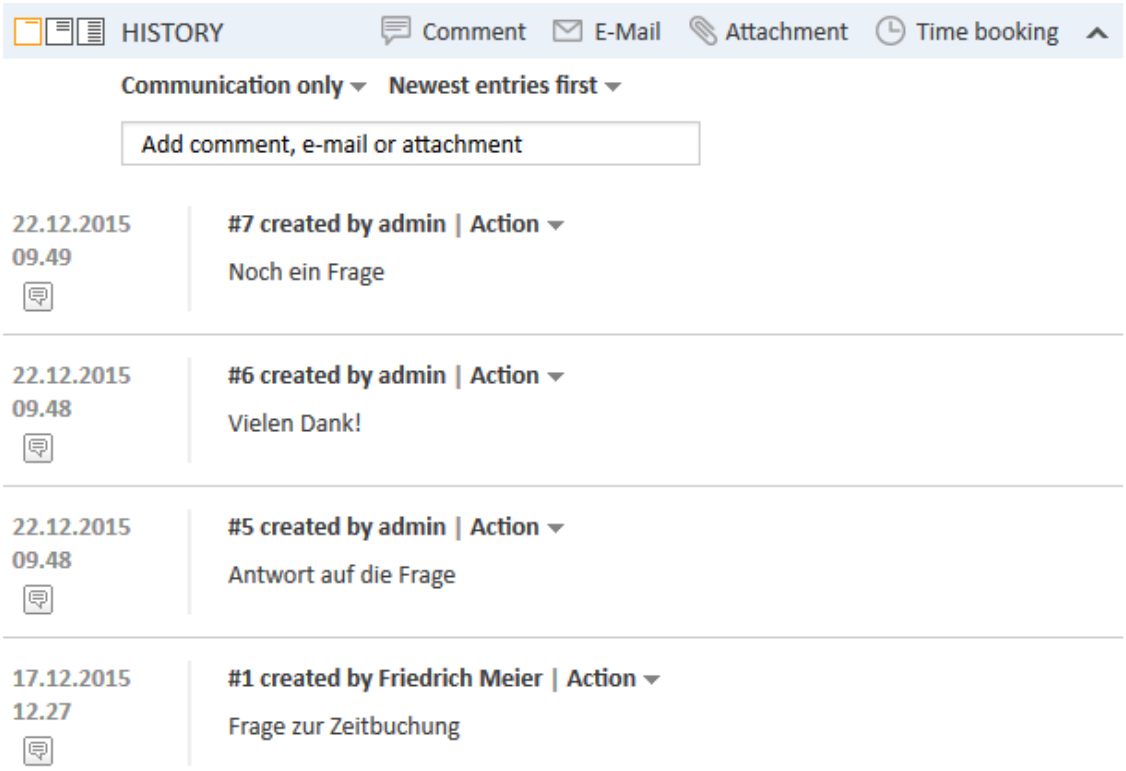


Figure 145: *ConSol CM Web Client - Lazy Loading switched off*

Example 2: headHistoryElementsCount and tailHistoryElementsCount each set to “1”

Figure 146: *ConSol CM Web Client - headHistoryElementsCount and tailHistoryElementsCount set to 1*

- **mailToSelection**
  Initial email address inserted into To field when composing email. Default: “contact”
  (java.lang.String)
  Possible options:
  - **contact**
    The email address of the main contact is copied into the To field when the Ticket Email Editor is opened.
  - **contacts**
    The email addresses of the main contact and all additional contacts are copied into the To field when the Ticket Email Editor is opened.
  - **engineer**
    The email address of the engineer is copied into the To field when the Ticket Email Editor is opened.
  - **fixed**
    for *fixed* option see *mailToFixedMail* attribute.
  - **none**
    The To field is left empty when the Ticket Email Editor is opened.

- **mailToFixedMail**
  Fixed email address used when attribute *mailToSelection* is set to “fixed”. (java.lang.String)
  Example: Email to a fixed email address, *mailToFixedMail* set to “ foo@bar.de”, *mailToSelection* set to “fixed”.

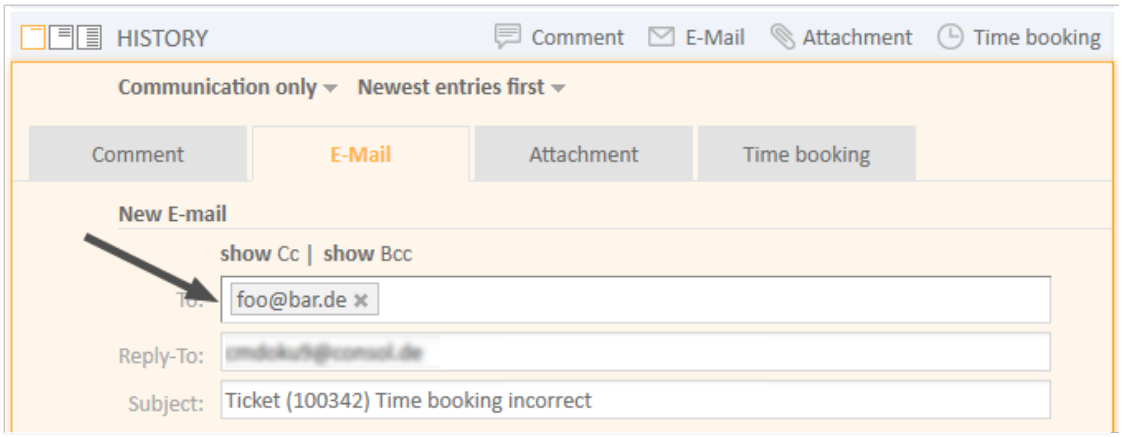


Figure 147: *ConSol CM Web Client - Email to a fixed email address*

- **recordLastUsedAcimTab**
  Boolean. Records last used ACIM tab. i.e., when you open the editor again, the tab (comment/email/attachment/time booking) will be opened that was open and active (i.e. some actions were performed, e.g., writing a comment) when you closed the editor last time.
- **reloadPageIfIE8onAcimSubmit**
  Boolean. Reloads page after a new ACIM submit, only for ***IE8***. This is a workaround for the problem that adding comments/emails may take a long time when using IE8.
- **removeContentOnTabSwitch**
  Clears text area content each time the tab panel in the editor is being switched. (boolean, default = “false”, i.e., when you switch, for example, from the Ticket Email Editor to the Comment Editor, the text you have typed will remain in the editor panel and will not be removed)
- **timeBookingFeature**
  Boolean. Activates or deactivates the time booking support in acimSection (i.e., display the link to time booking and the tab in the editor), default = “true”.

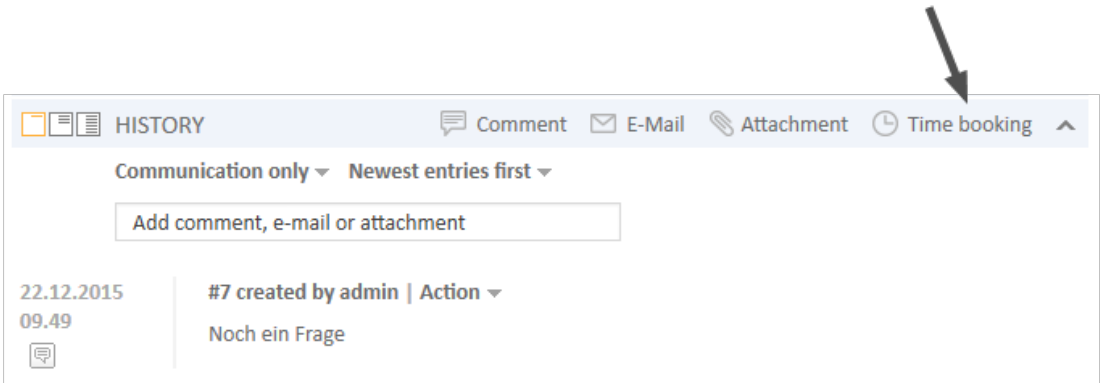


Figure 148: *ConSol CM Web Client - Activate time booking support (timeBookingFeature = true)*

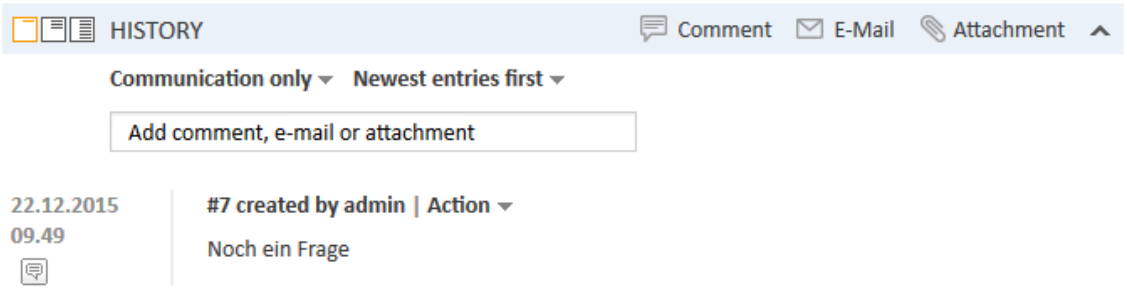


Figure 149: *ConSol CM Web Client - De-activate time booking support (timeBookingFeature = false)*

Please note that

- the value “false” in the *timeBookingFeature* hides the hyperlink from the time booking editor (see figure above). The engineer cannot change to display mode for it.
- the visibility of the time booking section on the *userProfilePage* is configured via the *timeBookingSection* attribute *visible*!

- **extendedViewCharactersLimit**
  Integer. Defines the maximum number of characters which will be displayed in the ticket history for comments in extended view if this level is defined as *Extended (short)* in the class of text which is applied to the entry. Default 350. Please see the example in section Modifying the Display Behavior for the Basic and Extended Level to Optimize Ticket History Display.
- **basicViewCharactersLimit** (formerly **standardViewCharactersLimit**)
  Integer. Defines the maximum number of characters which will be displayed in the ticket history for comments in basic view if this level is defined as *Basic (short)* in the class of text which is applied to the entry. The text is cut off after the maximum number of characters defined in a

page and formatting is removed. Emails will not show any header information. This provides a very concise informative view using very little space. Default 150. Please see the example in section Modifying the Display Behavior for the Basic and Extended Level to Optimize Ticket History Display.

attachmentSection (Type or Subscope)

Attributes:

- **defaultVisibilityFlag**
  The visibility of the attachment section for engineers who have not changed the default visibility yet (for others the last visibility state is used, default = “false”, i.e., attachment section is not displayed). This feature defines only the initial behavior of the system. The engineer can show the attachment section using the *Display* option in the ticket's menu and the Web Client will remember this configuration for this particular engineer.

> Valid until CM version 6.9.3. For newer versions (i.e., 6.9.4 and up) with the new ticket section management, use the *state* attribute in sectionList / ticketEditPage / attachmentSection.

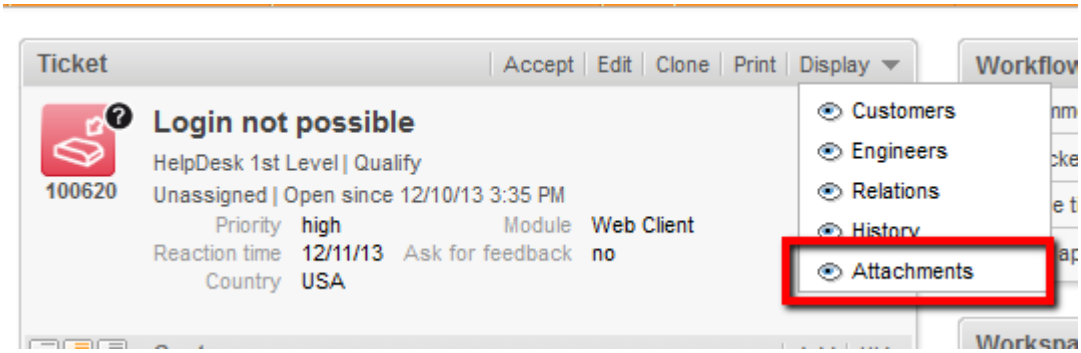


Figure 150: *ConSol CM Web Client - Visibility of Attachment section (defaultVisibilityFlag = true)*

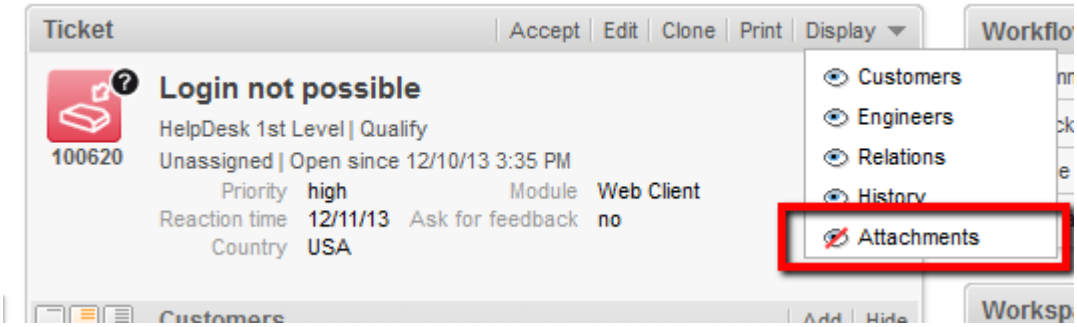


Figure 151: *ConSol CM Web Client - Visibility of Attachment section (defaultVisibilityFlag = false)*

Please note that in ConSol CM version 6.11, one page customization concerning the attachment section has been removed. The details are explained in the following figure.

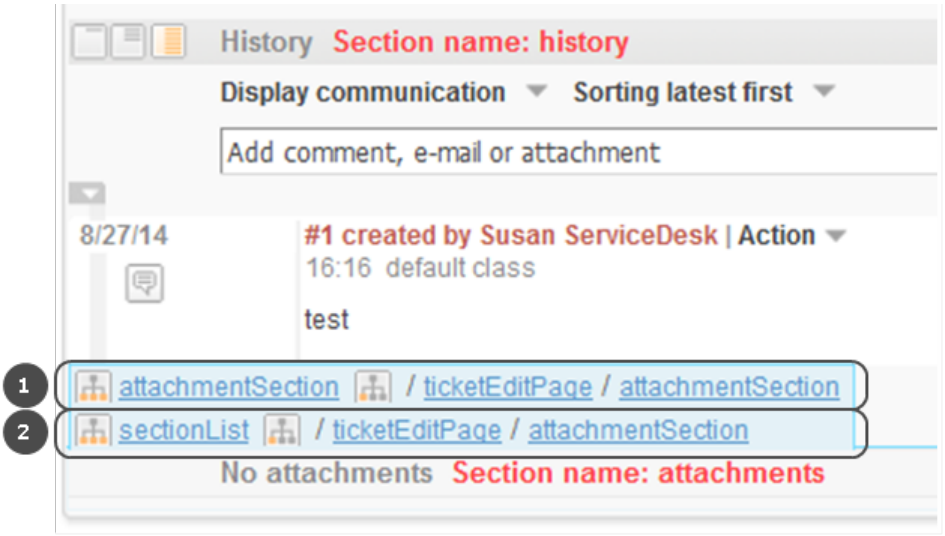


Figure 152: *Web Client in ConSol CM version 6.10.5. The page customization (1) has been removed in version 6.11, the page customization (2) is still available in version 6.11.*

autocomplete

(available e.g., on *userProfilePage*)

Attributes:

- **suffixCharactersToRemove**
  Any occurrences of these characters will be removed from the tail of each search pattern word. (java.lang.String)
  Example: If you search by using patterns, i.e., of the form “<Last name>, <First name>” then the search will not find any results because the word “<Last name>,” is not found in the search index because of the “,” at the end of the word. It is now possible to configure that certain characters should be removed from the tail of the search pattern word. So in this case the “,” will be ignored during the search and (see following figure) the engineer *Holler* is found.

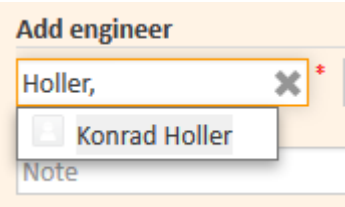


Figure 153: *ConSol CM Web Client - Input in a search field where characters should be ignored in the search*

boxContent

Available for the following scopes:

- ticketEditPage
- userProfilePage

Attributes:

- **order**
  Specify the order of the sections within a ticket in csv format (e.g.: header, history, relations). Default values for standard installation:
  - **ticket create**: customfields, contacts, comment
  - **ticket details**: customfields, contacts, engineers, relations, history, attachments
  - **contact details**: customfields, tickets, additional_details, relations, history
  - **company details**: customfields, tickets, contacts, additional_details, relations, history

For other pages or custom projects check the section names shown in the header or in your ContentBuilder implementation.

This attribute has to be created before it can be changed. Use the *Create* button of the page customization.

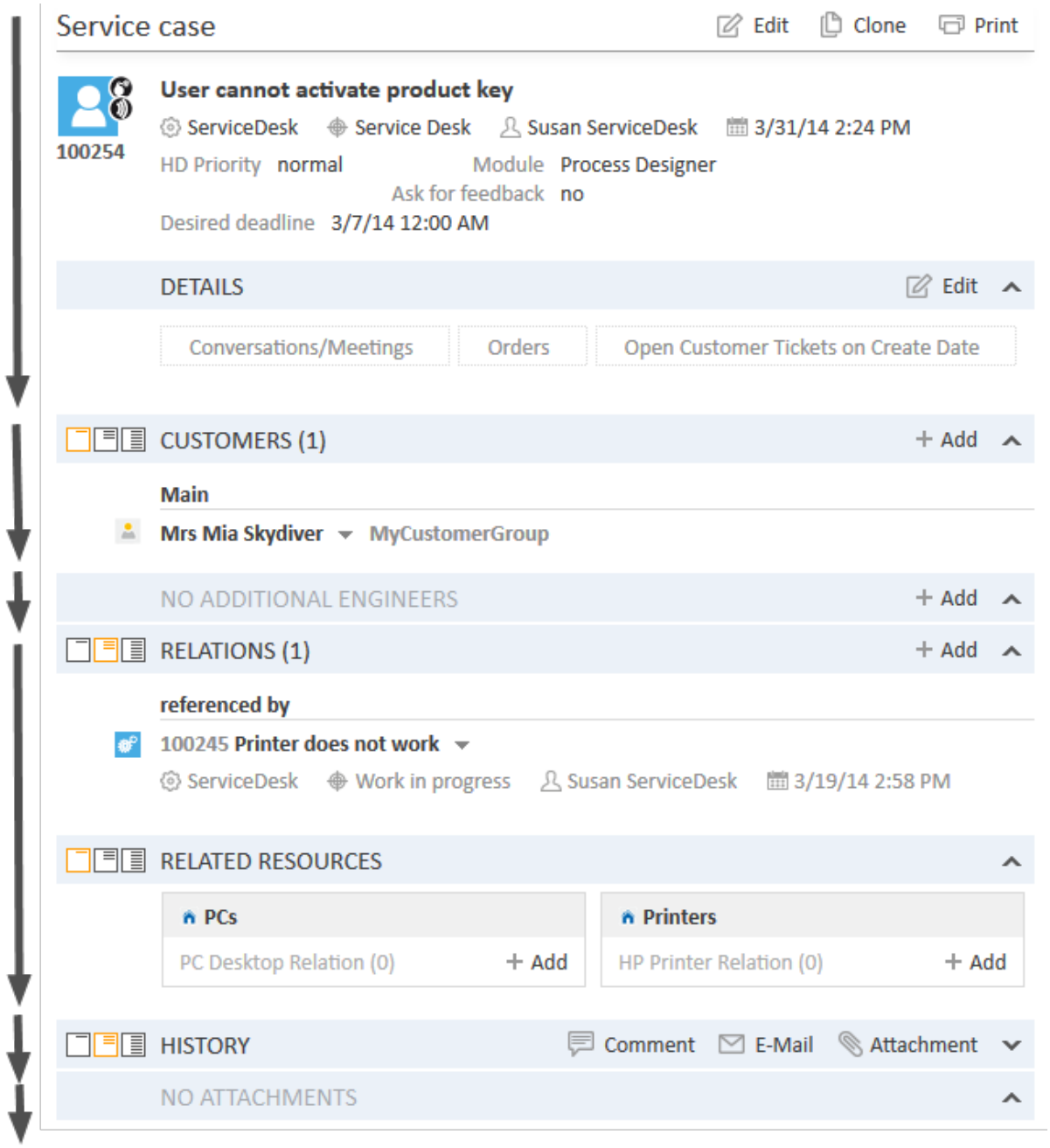


Figure 154: *ConSol CM Web Client - Default value for attribute boxContent / order*

calendarSection

This is used to configure the Calendar section on ticket pages and customer pages. Currently the respective attributes are used to configure the integration of Microsoft Exchange calendar views. Please see section Microsoft Exchange Calendar Integration for a detailed explanation of the feature and a list of all available attributes.

cmApplicationCustomization (Type)

Attributes:

- **activityKeyBindingEnabled**
  Boolean. Enable/disable shortcuts for activities/actions. Starting with version 6.10.1, the engineer can use shortcuts to address workflow activities of the ticket. Default “true”.
- **globalSearchFocusKeyBindingEnabled**
  Boolean. Enable/disable shortcuts for the Quick Search. If this attribute is enabled, it is possible to jump to the Quick Search field by pressing the “F” key. Default “true”.
- **submitACFOnEnterEnabled**
  Boolean. Enable/disable submit ACF form on Enter pressing. Starting with version 6.10.1, the engineer can press ENTER to submit Activity Control Forms. Default “true”.

The following two attributes influence the session timeout for engineers in the Web Client. In order to prevent the session from terminating while an engineer edits important information (and loosing the entered data with the end of the session), the following two attributes can be used to configure the automatic session renewal.

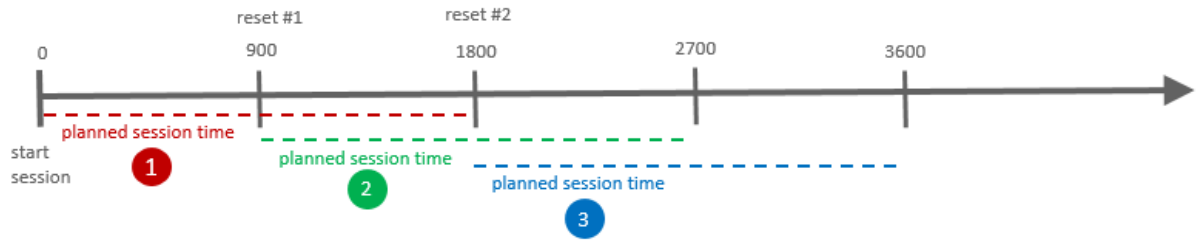


Figure 155: *Principle of renewing engineer session, example with updateTimeServerSessionActivity = 900.*

- **updateTimeServerSessionActivity**
  Integer. Set time in seconds to autoupdate server session activity. Default 900.
- **updateTimeServerSessionActivityEnabled**
  Boolean. Enable/disable autoupdate server session activity. Default “true”. When set to “false”, the session renewal is not active.

The following actions underlie the control of the `updateTimeServerSession`:

- Creating a ticket, customer, or resource
- Creating a comment or an email in a ticket
- Creating a comment for a customer or for a resource
- Editing data fields of a ticket, customer, or resource
- Editing data field groups
- Activity control forms

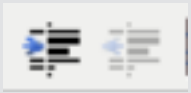
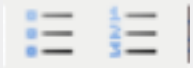
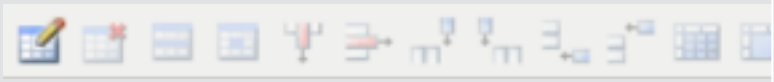
cmRichTextEditor

Available for the following (sub-)scopes:

- acimSection/ticketEditPage
- createTicketAcimSection/ticketCreatePage
- templateSection/templateEditPage (in the Text Template Manager)

Attributes:

- **editorFeatures**
  Additional editor features. By default, all values are set, i.e., the respective menu entries are available. (java.lang.String)
  Possible values:

Value	Icon in the Rich Text Editor	Description
SUB_SUP		Allows sub-script and superscript.
INDENTS		Allows the abil-ity to indent text.
LISTS		Allows inserting lists and list formatting.
TABLES		Allows inserting tables.
INSERT		Allows inserting text elements; for finer control you can use INSERT_ EMOTICON, INSERT_CHAR and INSERT_ IMAGE
INSERT_ EMOTIO-N		Allows inserting emoticons.

Value	Icon in the Rich Text Editor	Description
INSERT_ CHAR		Allows inserting special characters.
INSERT_ IMAGE		Allows inserting images.
INSERT_ LINK		Allows inserting/editing and removing hyperlinks. See explanation below.

INSERT_LINK: Mark the words you would like to mark as hyperlink and click on the button *Insert Link*. This will open a dialog window to enter the target URL, the target (same or other browser window) and an optional link title.

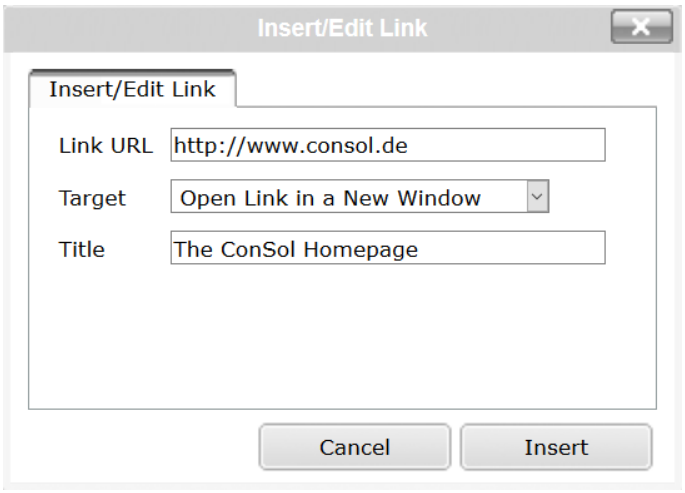


Figure 156: *Insert link dialog window*

The target selection can be used to define, if the link should open in a new window/tab or in the one currently displayed. Valid protocol prefixes for the URL are “http://”, “https://”, “ftp://”, “ftps://”, “file://”, and “mailto:”. When omitting the protocol, the prefix “http://” will be added automatically. For file links a change in the browser configuration often is necessary which is described in section Types of Data Fields, *How can I display a file link?*
This linking functionality is available for text templates as well, so that predefined links can be used within text templates.

- **editorFonts**
  The list of fonts for the editor in form <Font display name>=<Font name in system>.
  Fonts are separated by ';'. You can specify a comma-separated list of possible system names for

each font. (java.lang.String) (default = empty string)
Example: Arial=arial,helvetica,sans-serif;Courier New=courier new,-courier;Verdana=verdana,geneva

- **fontSizeValue**
  This is the default size for the text in the rich text editor. It must be one of the values form the list in the other customization *fontSizeValues*.
- **fontSizeValues**
  This is the list of values shown in the font size selector of the rich text editor. It is a comma separates list of legitimate font size values including their unit like “6pt,10pt,12pt”. The values can be prepended by a label which is shown in the selector instead of the value itself: “tiny-y=6pt,regular size=10pt,large=12pt”.
- **imagePasteEnabled**
  Flag whether direct pasting of images from the clipboard into the editor is enabled. Note that enabling this requires running a Java Applet (boolean, default = “false”). Web browsers (i.e., Internet Explorer, Firefox) might show different behavior concerning display of the images. Available in CM versions up to 6.10.4.x, no longer supported in CM versions 6.10.5.0 and up.

Configuring The Default Font Size of the Rich Text Editor Depending on the Customer Group of the Ticket's Contact

In case the ConSol CM system is used in an environment with different customer groups and each customer group has different requirements concerning the default font in the Rich Text Editor (RTE), this requirement can be met using a configuration script for the page customization type *cmRichTextEditor*, scope *ticketEditPage/acimSection*.

In the script, the customer group has to be retrieved. Depending on the group, a default font is set.

The page customization is set as shown in the next figure.

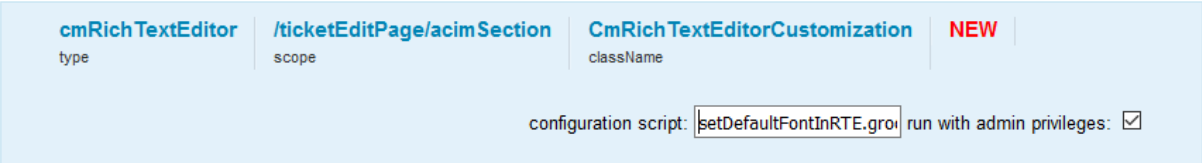


Figure 157: *Setting a configuration script for ticketEditPage/acimSection to configure the default font size in the RTE*

The script has the following code.

```
log.info 'Adapting font size to customer group ...'
ticket = ticketService.getById(ticketId);
def mainContact = ticket.mainContact
def cgName = mainContact.customerGroup.name
switch(cgName) {
   case "DirectCustomers": [fontSizeValue:"14pt"];
   break;
   case "MyCustomerGroup": [fontSizeValue:"13pt"];
   break;
   case "OurPartnerCompanies": [fontSizeValue:"14pt"];
   break;
   case "Reseller": [fontSizeValue:"12pt"];
   break;
   case "RetailCompanies": [fontSizeValue:"10pt"];
   break;
   case "RetailCustomers": [fontSizeValue:"10pt"];
   break;
   default: [fontSizeValue:"8pt"];
   break;
}
```

Code example 2: *Script used in page customization to set the default font for the Rich Text Editor depending on the customer group of the ticket's main contact*

For a ticket of a *Reseller* customer the RTE is opened with the default font of 12 as shown in the following figure.

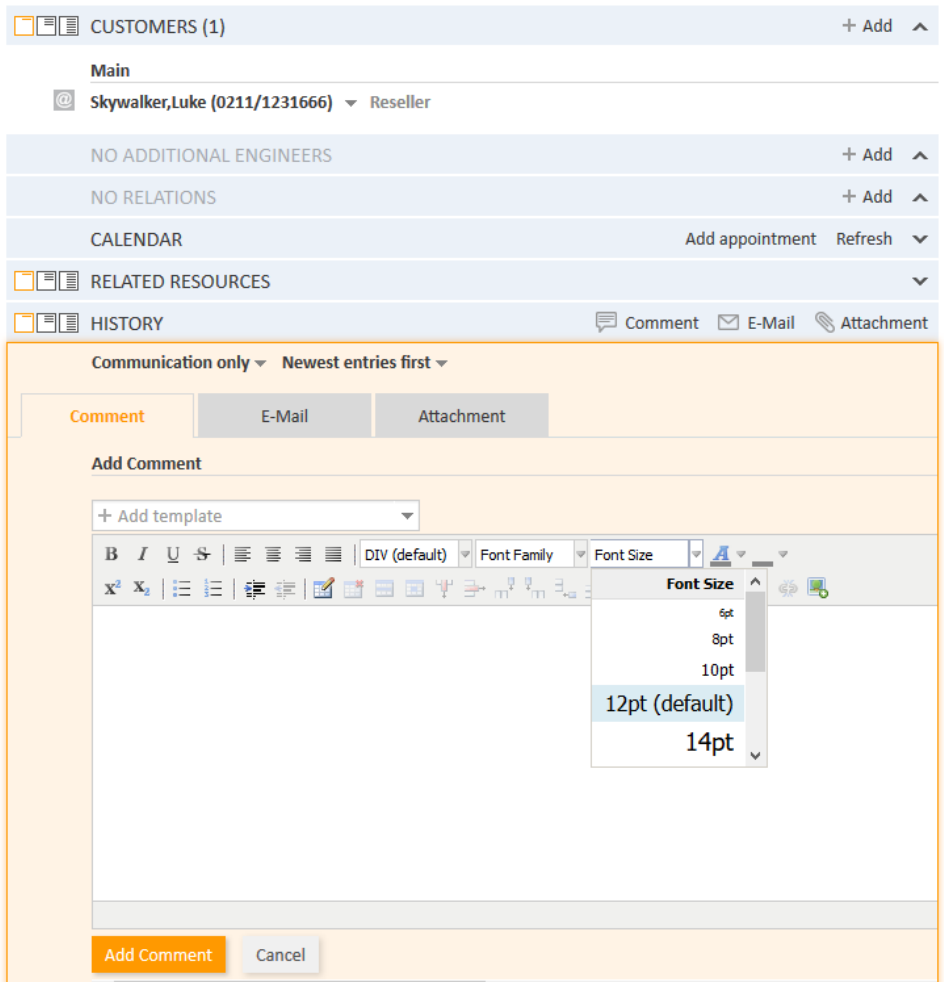


Figure 158: *Rich Text Editor of a ticket of a customer in customer group Reseller: default font set with script to size 12*

customerGroupSelector

Available directly under the root node in the Definition Section.

Attributes:

- **hiddenCustomerGroups**
  List of the technical names of customer groups which should not appear in Customer Groups Filter in the main menu. Comma-separated list of names of customer groups. Default: empty

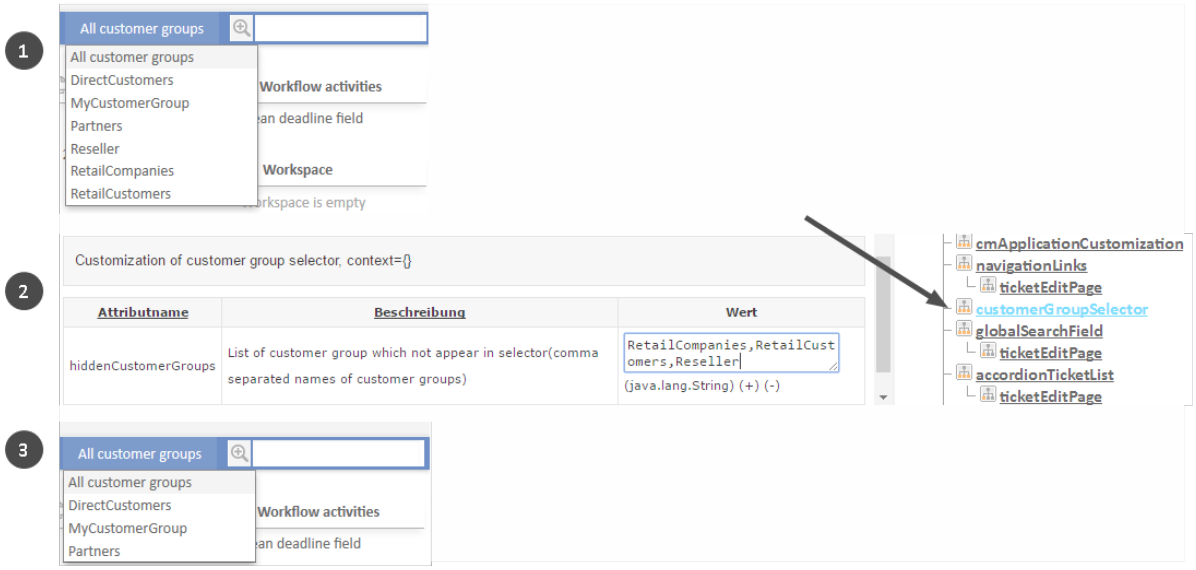


Figure 159: *ConSol CM Web Client - hiddenCustomerGroups*

customerSectionPanel

Here you can define whether the menu item *edit* should be displayed in the context menu for companies. Please note that the engineer must also have the according access rights (see section Role Administration) to edit company data.

Attributes:

- **referencedUnitEditLinkVisible** (Versions 6.9.3.3 and less. Similar, new attribute is *companyEditLinkVisible*)
  Boolean. The visibility of the link for editing referenced units (i.e., contacts or companies) (default is *true*).

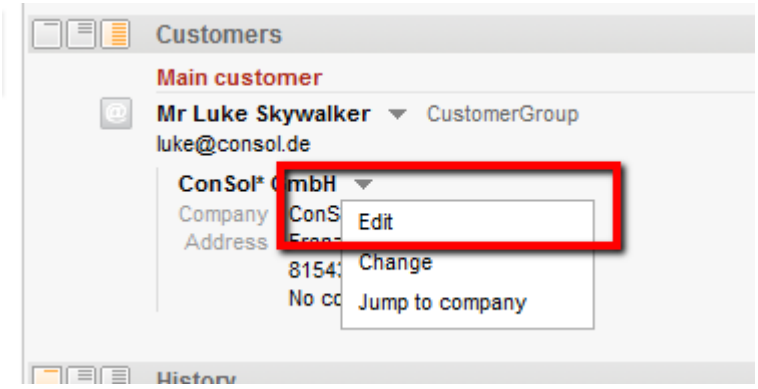


Figure 160: *ConSol CM Web Client - Visibility of Edit link (referencedUnitEditLinkVisible = true)*

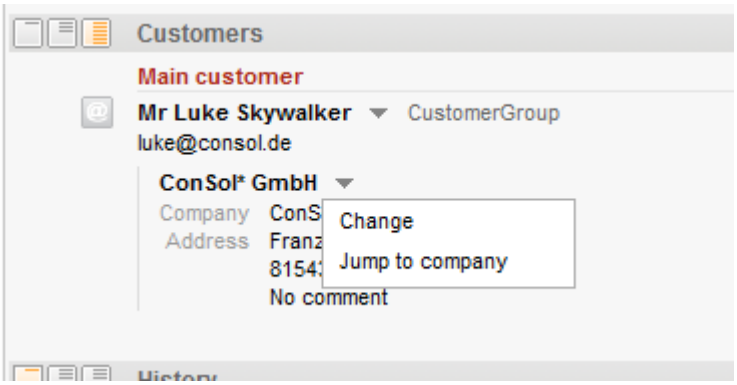


Figure 161: *ConSol CM Web Client - Visibility of Edit link (referencedUnitEditLinkVisible = false)*

- **companyEditLinkVisible** (Versions 6.9.3.4 and up. Older attribute is *referencedUnitEditLinkVisible*)
(available in *customerSectionPanel* in ticket and on *companyEditPage* and *contactEditPage*)
The visibility of the link for editing a company, default is “true”.

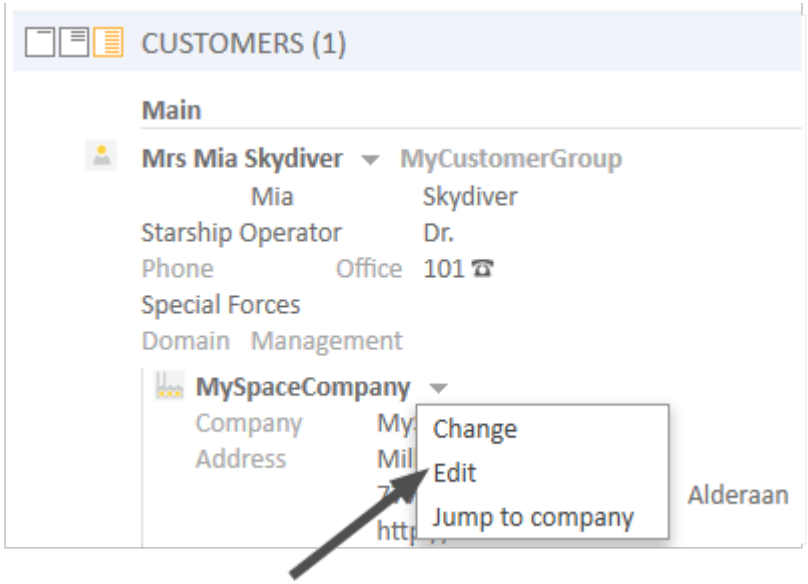


Figure 162: *ConSol CM Web Client - Visibility of company Edit link (companyEditLinkVisible = true)*

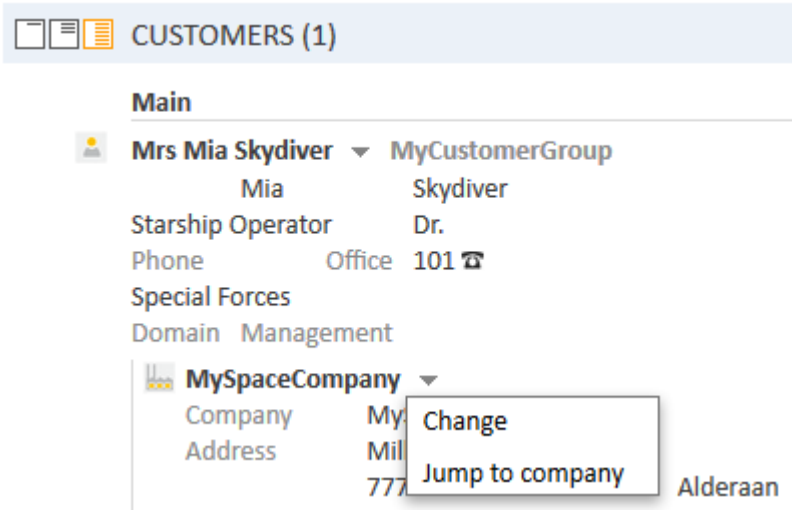


Figure 163: *ConSol CM Web Client - Visibility of company Edit link (companyEditLinkVisible = false)*

- **additionalCustomersSortStrategy**
  The sort order of additional customers for a ticket can be defined using this attribute. If no value is set, the additional customers are sorted in the order they have been added to the ticket.

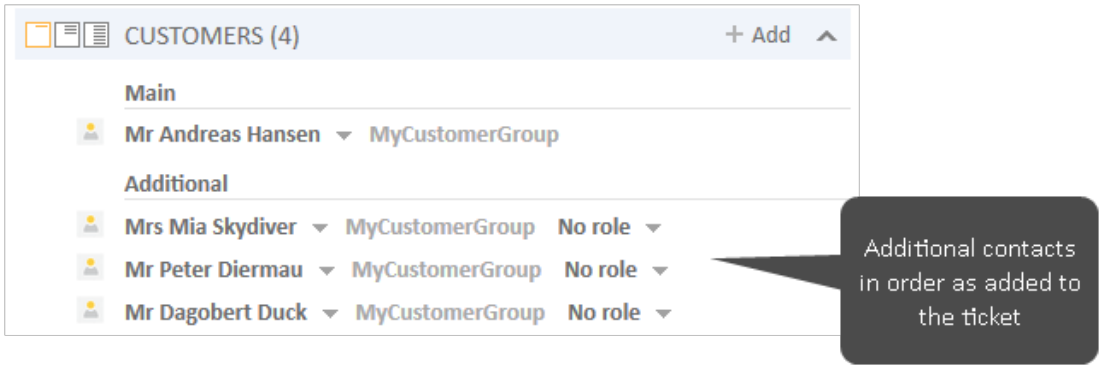


Figure 164: *ConSol CM Web Client - Page customization for sort order of additional customers (1)*

The following values are possible:

- **COMPANY_OF_MAIN_CUSTOMER**
  Customers are sorted by the company description with the company of the main customer first.
- **COMPANY**
  Customers are sorted by the company description.
- **CONTACT**

Customers are sorted by the contact description.

- **ROLE**
  Customers are sorted by customer roles.

Multiple values can be provided as a comma-separated list. The default sort order (no value) works as before: customers are sorted as previously in the order of their addition.

The customer object descriptions used for the display of contacts and companies are taken from the template *<Customer object>.Ticket page*, see section Templates for Customer Data.

In the following example, the additional customers should be ordered by the name of the customer using the value *CONTACT*.

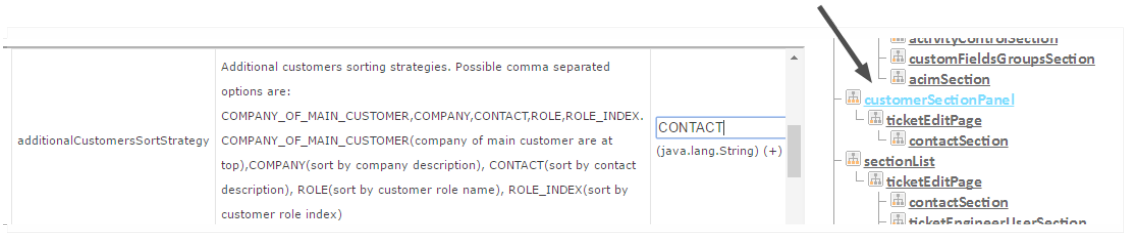


Figure 165: *ConSol CM Web Client - Page customization for sort order of additional customers (2)*

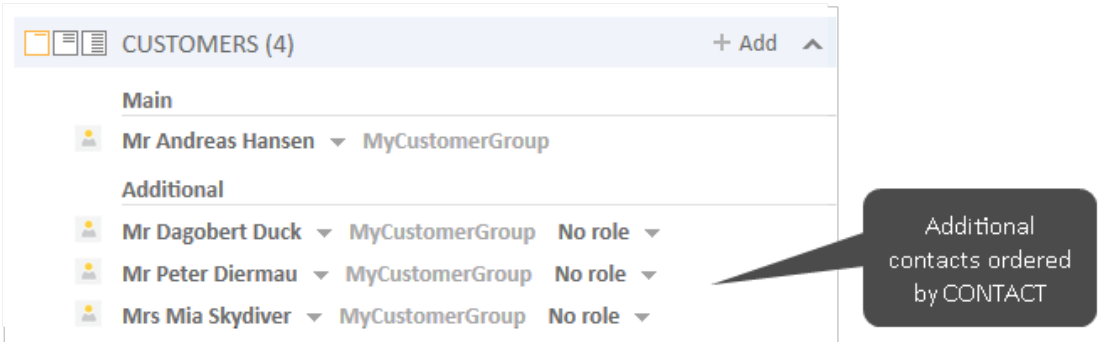


Figure 166: *ConSol CM Web Client - Page customization for sort order of additional customers (3)*

customerTickets

Using the attributes of this type, you can configure the behavior of the ticket list on customer (i.e. contact and/or company) pages.

Attributes:

- **enabled**

  Boolean. Defines if the engineer and queue filter are available in the ticket list on customer (i.e. company and/or contact) pages. Default value: “false”.

This will only apply if the number of tickets in the list exceeds the number which is set for the attribute *compactViewLimit*.

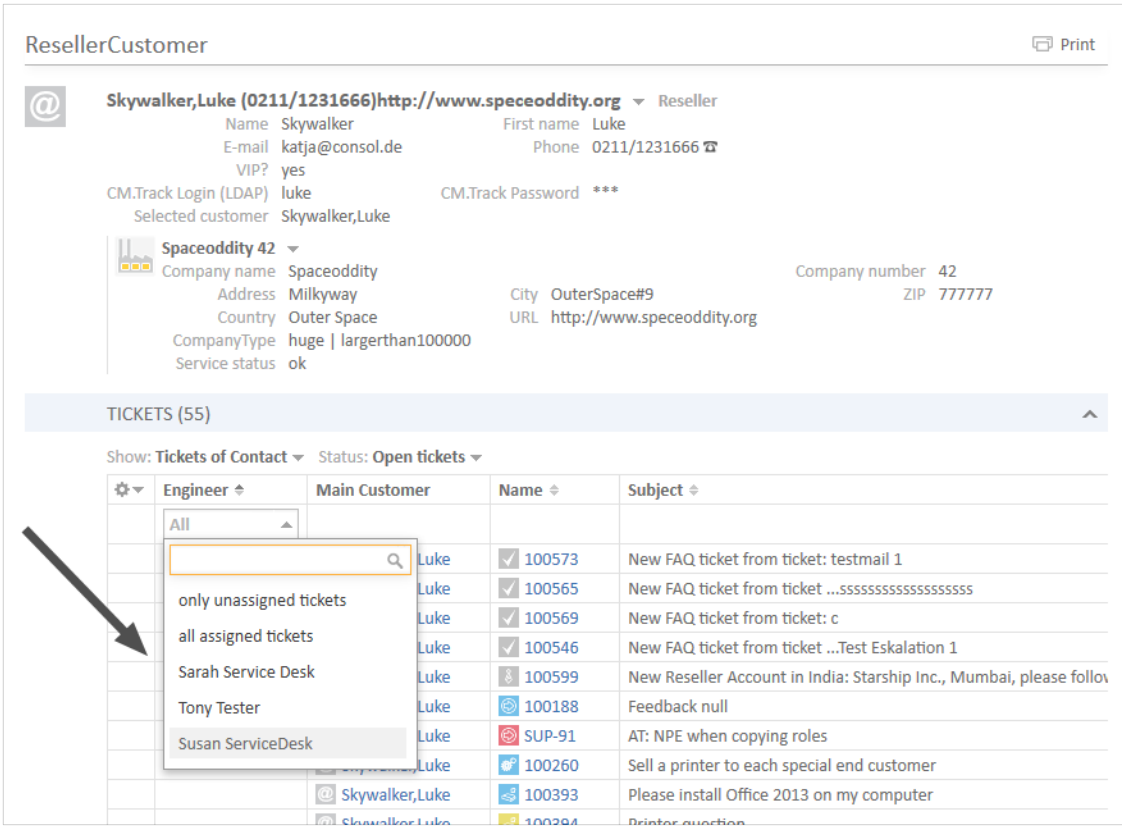


Figure 167: *List of customer tickets on a customer (here: contact) page, engineer and queue filters available*

The filter is applied to all types of customer pages, i.e. to contact and company pages if the attribute *enabled* is set for the type *customerTickets* (for an example see following figure).

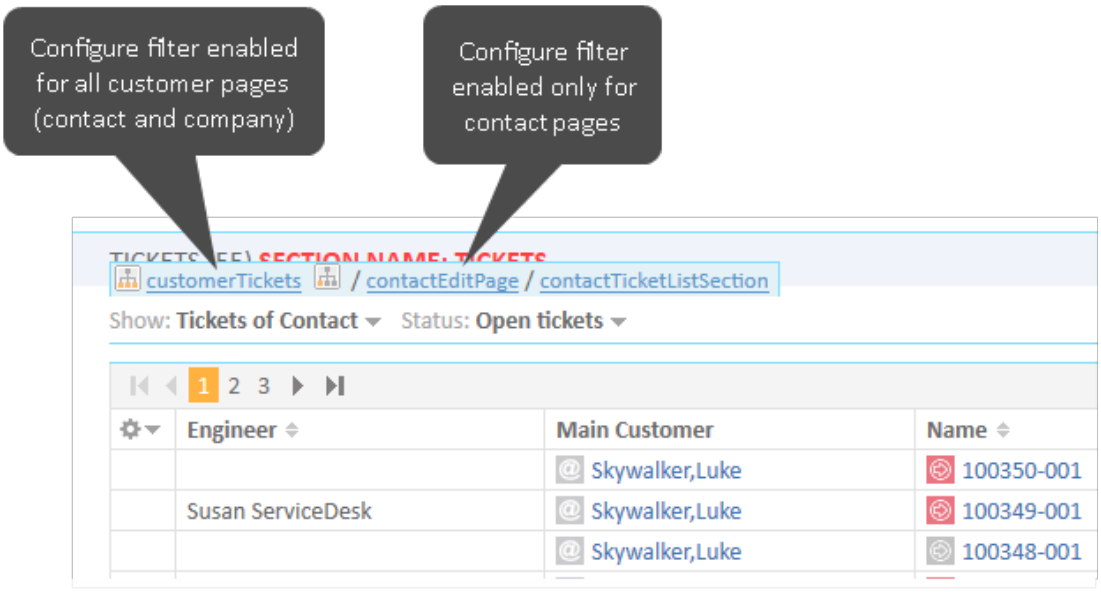


Figure 168: *Page customization type and scopes used to enable engineer and queue filer in customer ticket list*

The filter is applied only to contact pages when the attribute *enabled* is set for the scope *contactEditPage* or *ContactTicketListSection*, and the filter is applied only to company pages when the attribute *enabled* is set for the scope *companyEditPage* or *CompanyTicketsSection*.

> You need to re-index the tickets before using this feature. Otherwise, the filters are empty and a warning is written to the log files.

- **compactViewLimit**

  Integer. If the limit is exceeded the results are presented along with filtering feature. The default limit is “10”.

detailSearch (Type)

Available for the following scopes:

- searchDetailPage

Attributes:

- **criteriaForAllTypeOfContacts**
  Boolean field. When set to “true”, the search will include the main customers and the additional customers of tickets. When set to “false” (default), the search will apply the search criteria only to the main customers of tickets.
- **duplicatedCustomFieldLabelFormat**
  It allows to customize the label format used when several ticket fields have the same localized name. There are four values which can be used to create a unique label:

- fieldName
- groupName
- fieldTechnicalName
- groupTechnicalName

The values “fieldName” and “groupName” are localized.
Default format: ${fieldName} (${groupName}) (java.lang.String)
This helps distinguish different ticket fields or customer fields with the same name, e.g., when they are offered in the drop-down menu in the detail search.
Example:
SalesProcess - Priority
HelpdeskProcess - Priority

- **enableRowSelection**
  Boolean. Switches on the row selection functionality in the result tables, i.e. a checkbox will be available for each line, one checkbox to (de-)select all lines is available. Default value “true”. The checkboxes are only displayed if search actions are defined for the respective object type. The search action script will only be executed for the rows which have been selected (by the engineer) using the checkboxes.

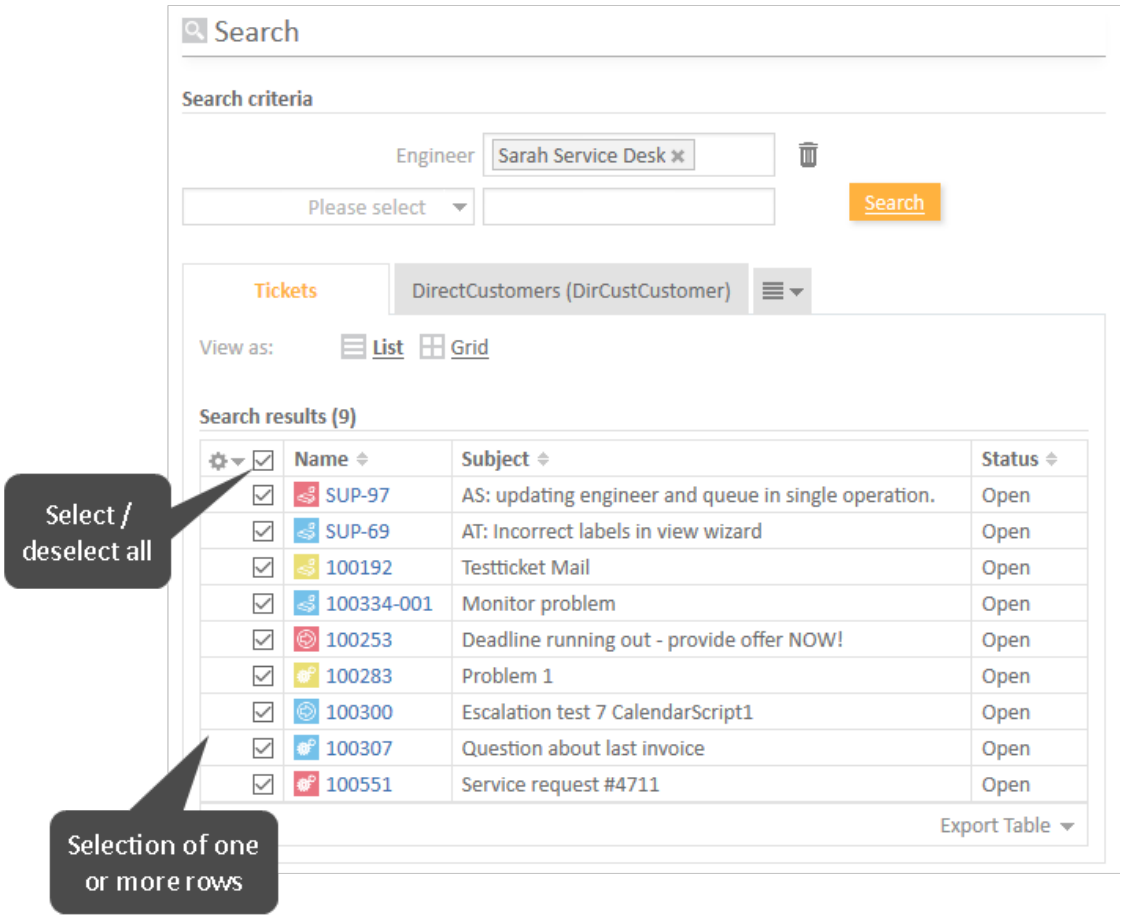


Figure 169: *ConSol CM Web Client - Search result table with detailSearch/searchDetailPage, enableRowSelection = true*

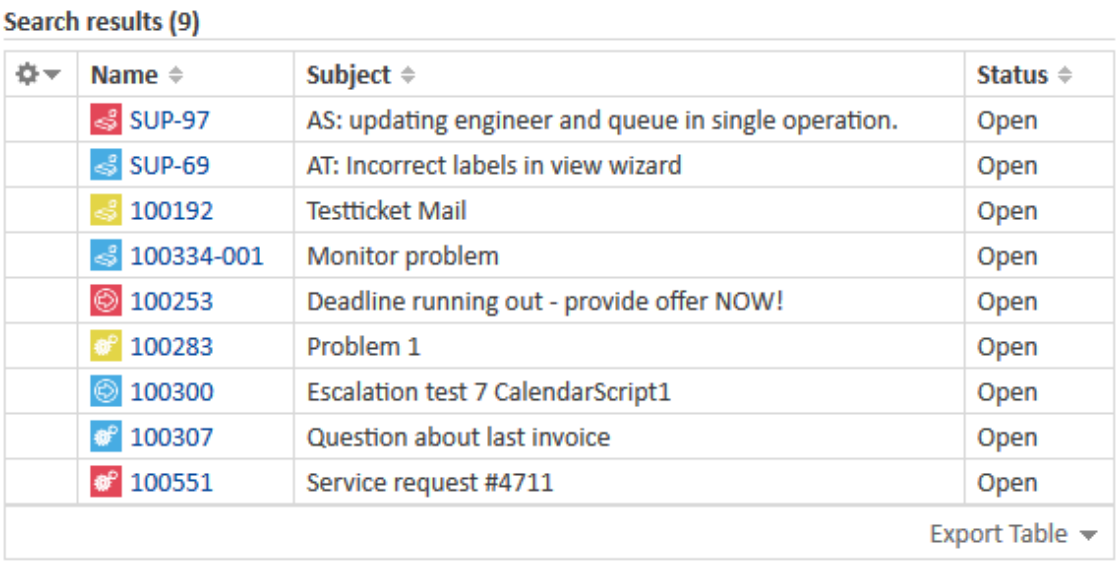


Figure 170: *ConSol CM Web Client - Search result table with detailSearch /searchDetailPage, enableRowSelection = false*

- **enableSearchForDeactivatedEnums**

  Boolean. If “true”, deactivated enum and MLA values will be accessible in the drop-down menu in Detailed Search.

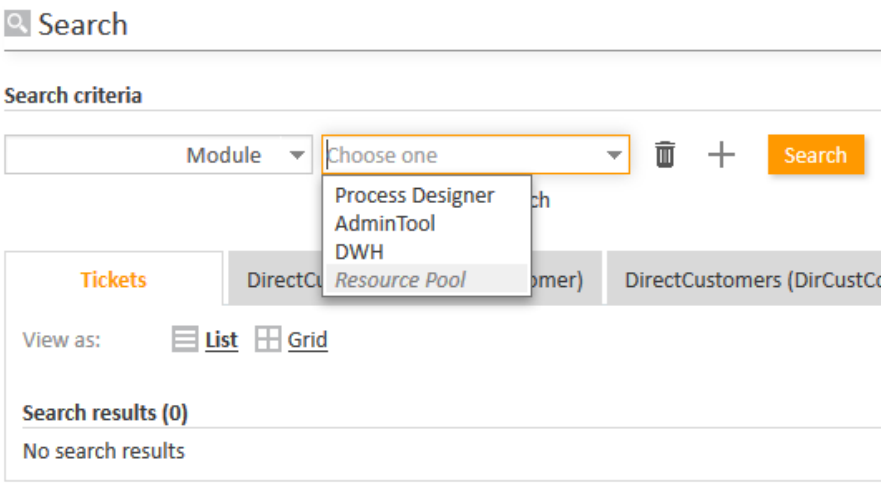


Figure 171: *'enableSearchForDeactivatedEnums' set to true in searchDetailPage: deactivated enum value is displayed*

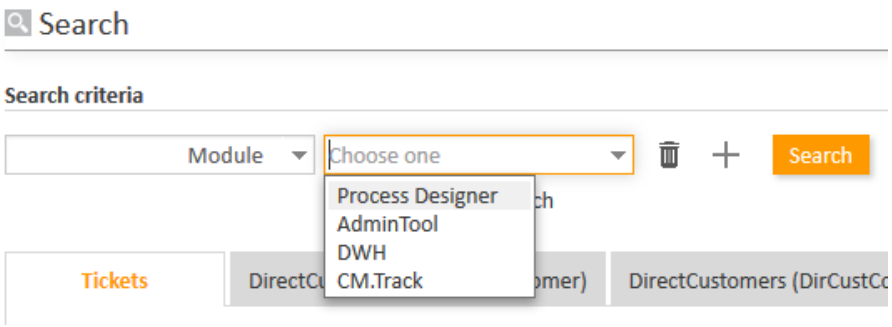


Figure 172: *'enableSearchForDeactivatedEnums' set to false in searchDetailPage: deactivated enum value is not displayed, but next active value is displayed in list*

- **enableSearchForOtherUsersTickets**
  Boolean. Decides if an engineer can search CM for all tickets ("true") or only for tickets which are assigned to himself ("false"). Default "true".
  This attribute has to be set on the (sub-)scope level, i.e., for the (sub-)scope *detailSearch/searchDetailPage*.
  The value of this attribute only influences the layout of the drop-down menu in the Detailed Search (see figures below). It does not have any influence on the results of the Quick Search.

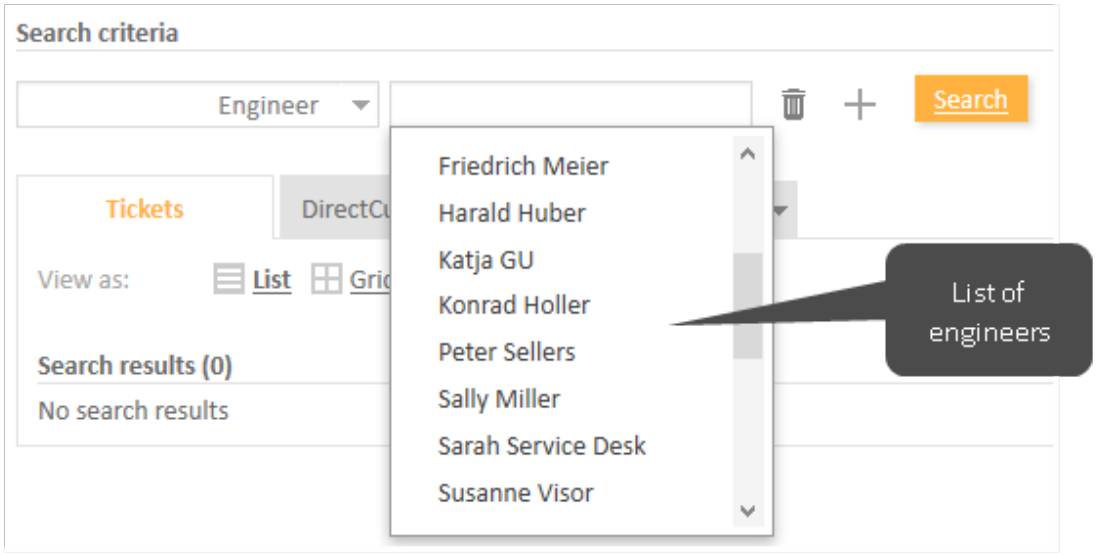


Figure 173: *ConSol CM Web Client - Detailed Search for engineers with enableSearchForOther-UsersTickets = true*

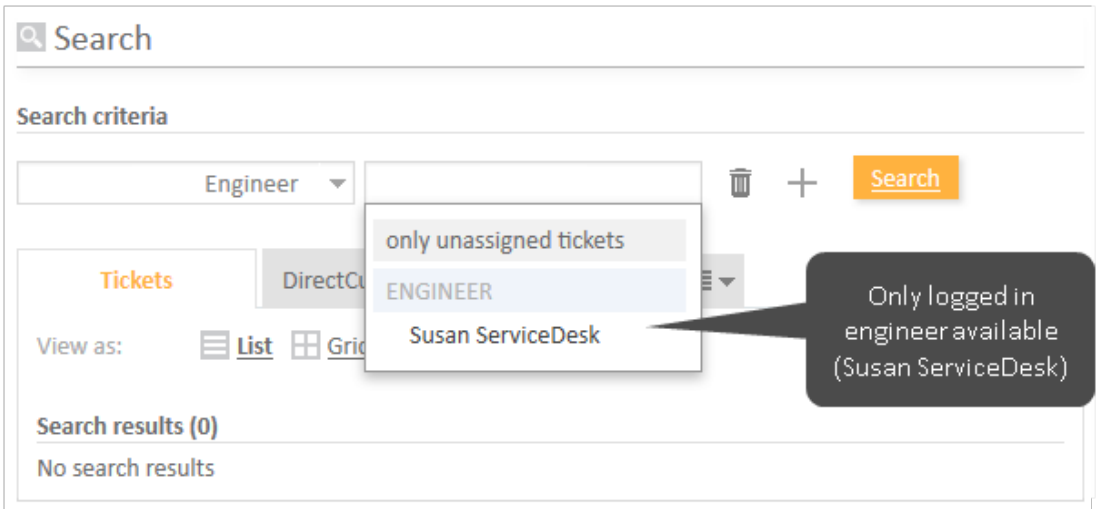


Figure 174: *ConSol CM Web Client - Detailed Search for engineers with enableSearchForOtherUsersTickets = false*

- **excludedUserNames**

  (usually has to be set for searchDetailPage)
  Login names provided here will be filtered out in engineers' criteria selector

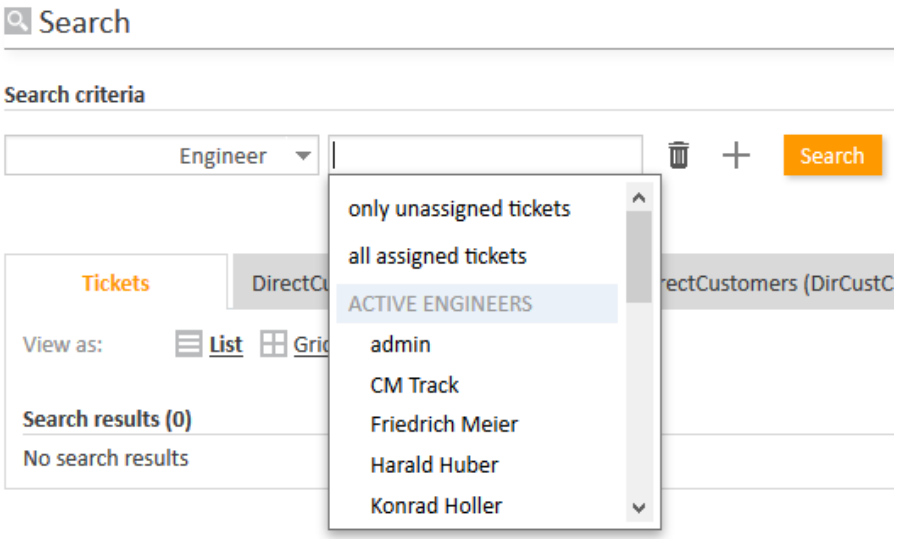


Figure 175: *Drop-down menu in engineer detail search before excluding user names*

Figure 176: *Using the page customization attribute excludedUserNames for two user names*

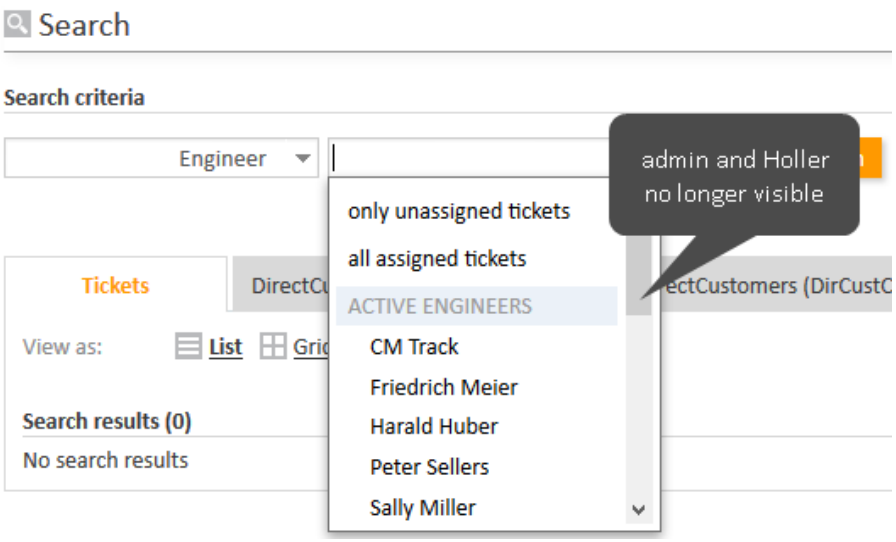


Figure 177: *Drop-down menu in engineer detail search after two user names have been excluded from the list*

- **maxGridTicketsNumber**
  Maximum number of tickets shown in grid view, i.e., in the entire grid, not in one column. The default value is “120”.

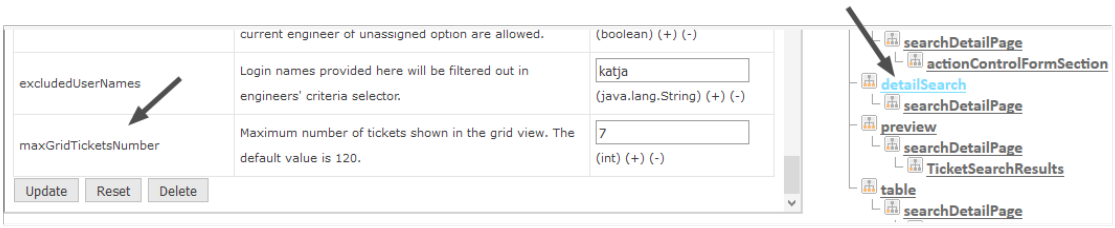


Figure 178: *ConSol CM Web Client - Using page customization to adapt maximum number of tickets shown in grid*

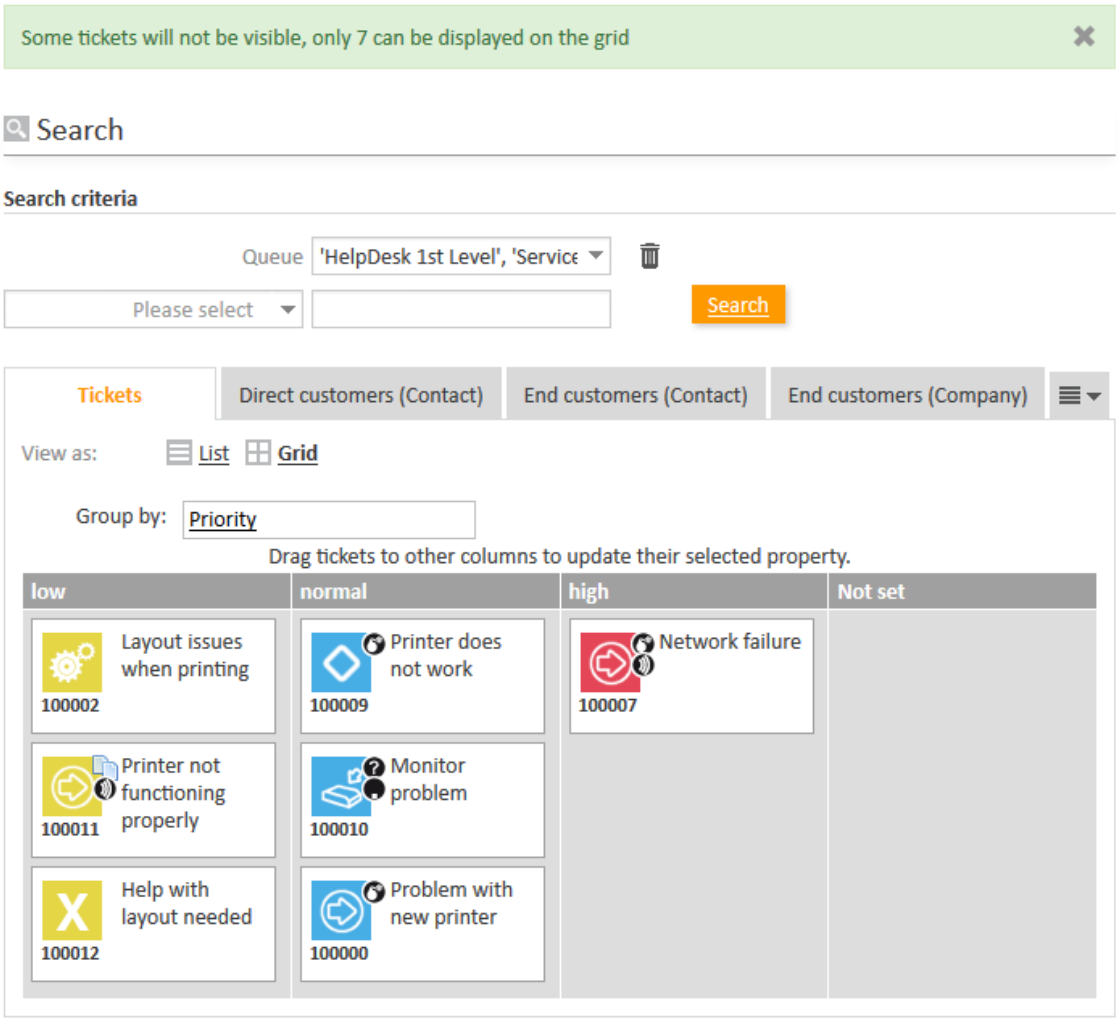


Figure 179: *ConSol CM Web Client - Maximum number in ticket grid with maximum 7*

The maximum number of tickets displayed in the ticket grid view can also be role-specific. For example, an engineer with the role *Teamlead* would see 100 tickets, whereas an engineer with the role *Helpdesk* would see 50 tickets. In the following simple example, all engineers with the role *ServiceDesk* see ten tickets, all others see 5. The configuration uses a script which is defined for the page customization type. The script is stored in the *Script and Template Administration* of the Admin Tool.

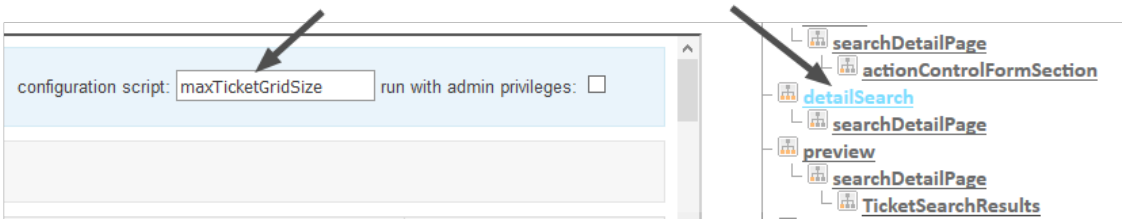


Figure 180: *ConSol CM Web Client - Defining a script for maximum number of tickets in grid view using page customization*

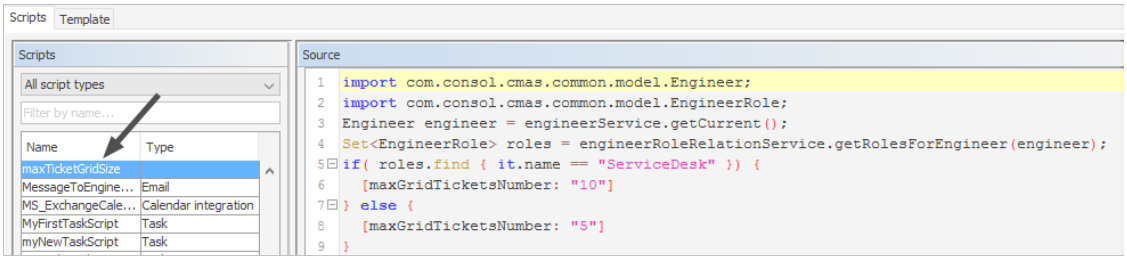


Figure 181: *ConSol CM Admin Tool - System, Scripts and Templates: Script for maximum number of tickets in grid*

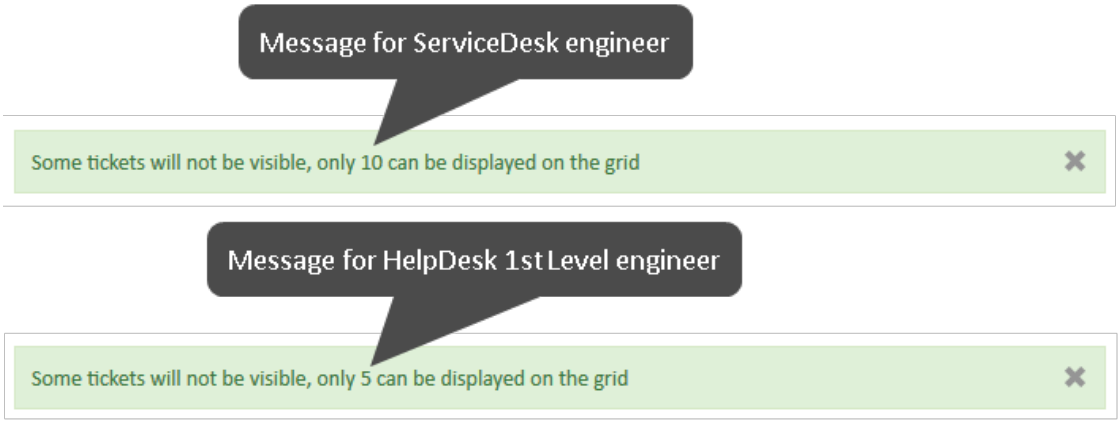


Figure 182: *ConSol CM Web Client - Number of tickets in ticket grid for different engineer roles*

engineerAutocomplete

Available for

- ticketCreatePage
- ticketEditPage
- userProfilePage
- contactEditPage

Attributes:

- **maxHints**
  Defines the maximum number of suggestions which is displayed. When set to “0”, all suggestions are displayed, no limit.
- **suffixCharactersToRemove**
  Occurrence of any of these characters will be removed from the tail of each search pattern word. (string, default: empty)

enumAutocomplete

(available on *ticketEditPage* for enums with annotation `enum type` = “autocomplete”)

Attributes:

- **maxHints**
  Defines the maximum number of suggestions which is displayed. When set to “0”, all suggestions are displayed, with no limit.
- **suffixCharactersToRemove**
  Occurrence of any of these characters will be removed from the tail of each search pattern word. (string, default: empty)

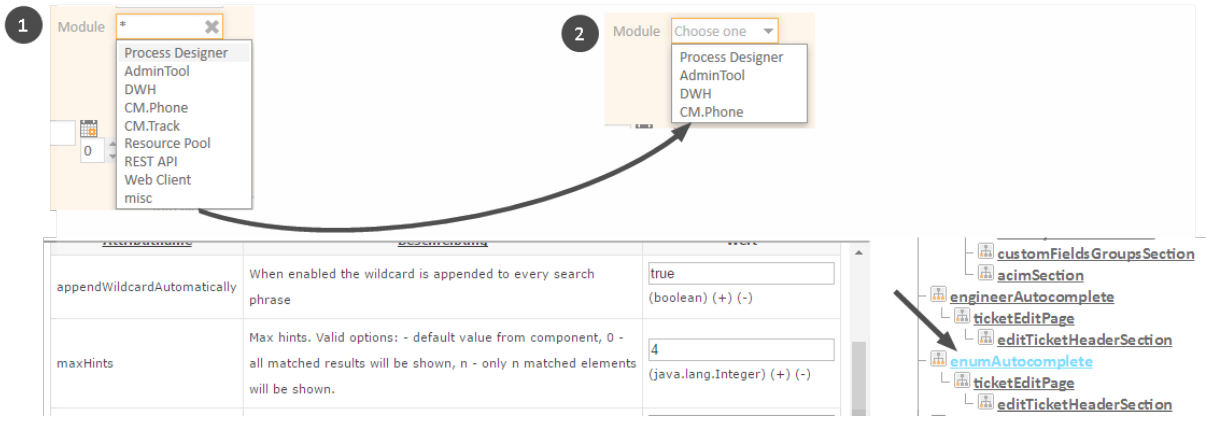


Figure 183: *ConSol CM Web Client - Using enumAutocomplete parameter to reduce number of hints*

generalFeedback

This type is used to manipulate log entries in a clustered environment and to set the timeout for success messages.

Attributes:

- **appendHostInfo**
  Configures whether the cluster node information is shown together with the error message.
- **appendHostInfoSystemProperty**
  Identifies which system property will be used to provide the value shown in *appendHostInfo.* There are two properties which can be used:

- cmas.http.host.port (default)
- cmas.clusternode.id

- **successMessageTimeout**
  Integer. Defines the time in seconds before a success message (e.g. "Ticket XY has been created") in the Web Client will be faded out. If set to 0, message will never fade out automatically but will have to be closed by the engineer. Default “0”.

globalSearchField (Type)

Here you can define the layout and the behavior of the *Quick Search* input field.

Figure 184: *ConSol CM Web Client - Quick Search input field*

Attributes:

- **appendWildcardAutomatically**
  Boolean. When enabled the wildcard ("*") is appended to every search phrase consisting of more than one character or number automatically, i.e., even when you have entered only part of a word it will be found. Default “true”.

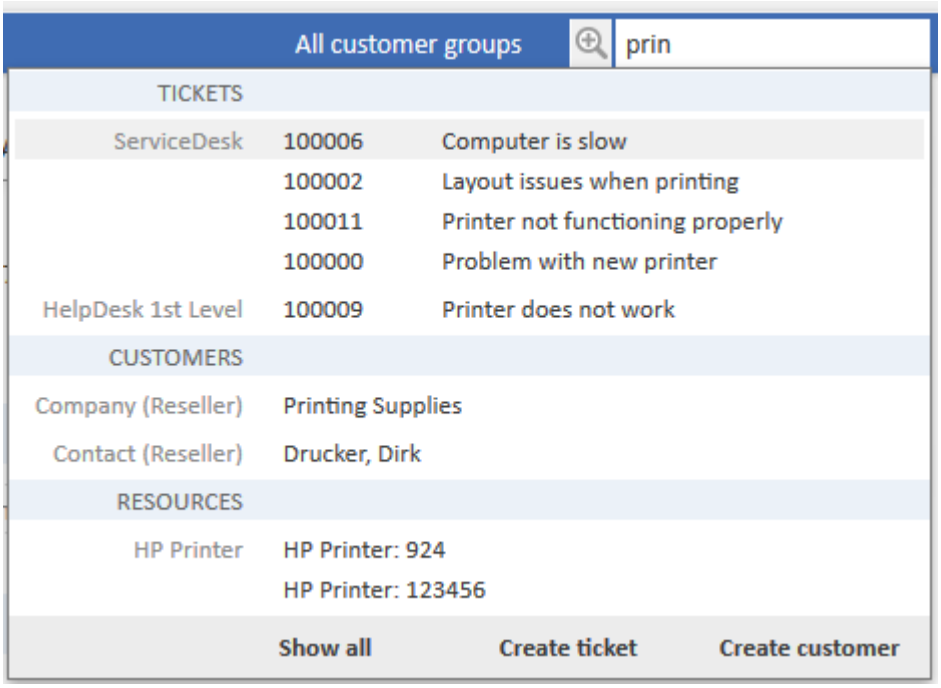


Figure 185: *Results of Quick Search with wildcard appended automatically (appendWildcardAutomatically = true)*

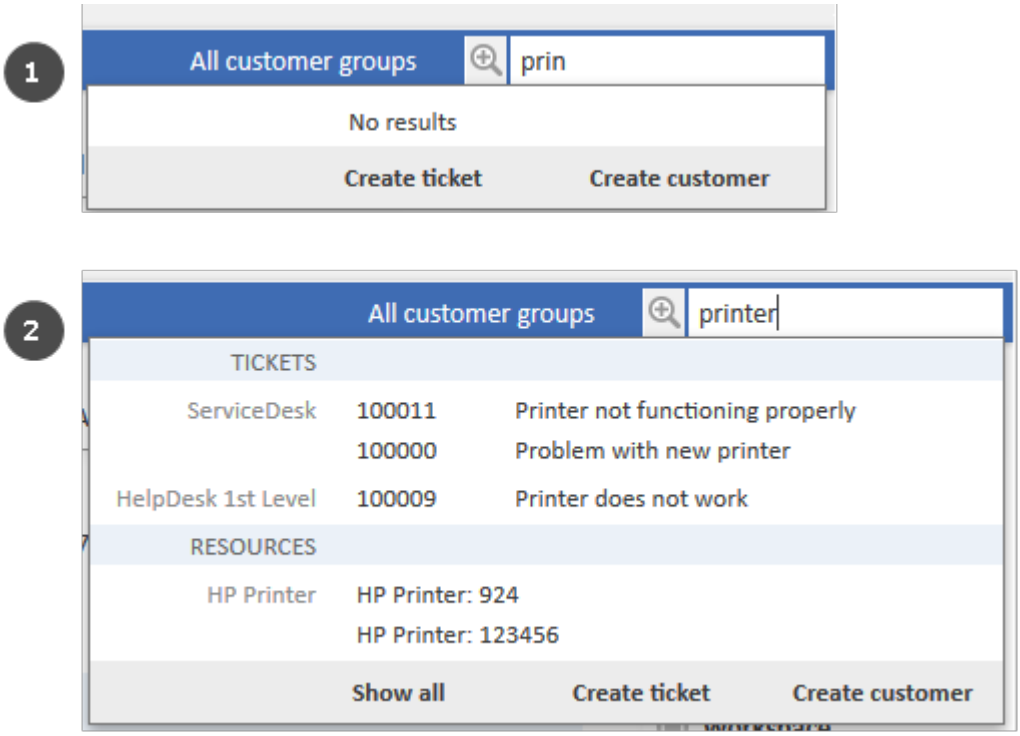


Figure 186: *Results of Quick Search with wild card not appended automatically (appendWildcardAutomatically = false)*

- **maxHints**
  Defines the maximum number of suggestions which is displayed. When set to "0", all suggestions are displayed, with no limit.
- **searchResultItemsOrder**
  Comma-separated values defining order and visibility of search result items. (java.lang.String)
  Possible values: CONTACT, COMPANY, TICKET
- **suffixCharactersToRemove**
  Occurrence of any of these characters will be removed from the tail of each search pattern word. (java.lang.String)

mailTemplate (Type)

(available on *ticketEditPage*/*acimSection*)

Attributes:

- **addingManyTemplateEnabled**
  Boolean. Makes it possible to compose an email by using more than one template.
  Obsolete with CM version 6.10.5.0, since starting with this CM version, a new Text Template Manager has been implemented which offers (besides other features) to insert more than one template into an email in the Ticket Email Editor or in a comment in the Ticket Comment Editor.

- **emailDisplayMode**
  String. Defines the format of the email address which is displayed in the To/Cc/From/Bcc fields in the Ticket Email Editor. Possible values: “FULL” (name and email address), “NAME”, “EMAIL”. Default “FULL”.
- **engineerPersonalMailsIncluded**
  Boolean. Enable appending personal email feature. If the value of *engineerPersonalMailsIncluded* is “false”, new email addresses typed in the *To*, *Cc*, *Bcc* fields are not remembered and not present as suggestions next time.
- **includeTextBlocksInEmailTemplate**
  Boolean. Whether text blocks from email template will be included by default.
- **mailBodyLockedAfterTemplateSelection**
  Boolean. Indicates whether email body will be locked after template selection.
- **mailEncryptionAvailable**
  Boolean. Makes email encryption option available.
- **mailSelectionComponentWidth**
  The width of the email selection component (in pixels). (java.lang.Integer)
- **mailTemplateSortStrategy**
  Email template list sorting strategies. (java.lang.String)
  Default value: USAGE, NAME. Possible comma-separated options are: USAGE, NAME
- **maxElementLength**
  The maximum length of a single element. If variable's value is set to “0”, elements will not be trimmed. (java.lang.Integer)
- **minMailInputLength**
  Minimum input length that triggers email suggestion drop-down when engineer starts typing email recipients. Integer. Default “1”.
- **quotingFeature**
  Activate the quoting function in email content. (boolean)
- **selectionComponentWidth**
  The width of the mail/comment selection component (in pixels). Integer.
- **showBcc**
  Boolean. Defines if the *Bcc* field in the Ticket Email Editor is displayed. Default “true”.
- **showCc**
  Boolean. Defines if the *Cc* field in the Ticket Email Editor is displayed. Default “true”.
- **showReplyTo**
  Boolean. Defines if the *ReplyTo* field in the Ticket Email Editor is displayed. Default “true”.

> Please note that the attribute *showReplyTo* is not the only parameter which influences ConSol CM behavior concerning display and use of the email Reply-To address! See section Scripts of Type Email for a detailed explanation!

- **showUniqueEmails**
  Boolean. Results in autocomplete email fields will be compared by email, but only first email will be used in results. If set to “true” then results will be compared by whole email description.
- **templateSortStrategy**
  String. Template list sorting strategies. Default value is: USAGE, NAME. Possible comma separated options are: USAGE, NAME. Default USAGE,NAME.

markersLibrary (Type)

This page customization type is available on the following page:

- templateEditPage (in the Text Template Manager)

Attributes:

- **excludedFields**
  The list of excluded ticket fields, e.g.: helpdesk.module,sales.priority. Fields which are listed here will not be offered in the *Select binding* menu any longer. In huge environments, it might save loading time of the templateEditPage to exclude some fields. However, it has to be absolutely clear that they will never be needed for template assignment or binding!
  The attribute can be set on type level or for the subscope templateEditPage/templateSection/bindings.

navigationLinks (Type)

Here you can define the display for several hyperlinks that are displayed in the main menu of the Web Client.

Attributes:

- **createContactLinkVisible**
  Whether the *createContact* (*Create customer*) link (menu item in the main menu) can be shown (boolean, default value is “true”).

  > In addition to this attribute being set to “true”, engineers must have appropriate permissions to see the *Create customer* menu item.

- **createTicketLinkVisible**
  Whether the *createTicket* link (menu item in the main menu) can be shown (boolean, default value is “true”).

  > In addition to this attribute being set to “true”, engineers must have appropriate permissions to see the *Create ticket* menu item.

- **externalLinks**
  External main menu items which will be appended to the main menu. This attribute may configure more than one external link (the order is significant, no characters between the link statements).
  Format (compatible with wiki): [http://link description] or [http://link description]@target

This attribute may be used to integrate hyperlinks to the company's web site, to a reporting application, to a help page or to any other valid URL.
Example: [http://www.consol.com ConSol][http://www.somewhere.com Somewhere]@

> Links are opened in a new tab by default. If the link should be opened in the same tab, add @ after the link, e.g. [http://www.consol.com ConSol]@

- **manageTemplateLinkVisible**
  Whether the menu item *Text template* (for the start of the ConSol CM Text Template Manager) can be shown (boolean, default value is "true") in the main menu. See also section The ConSol CM Text Template Manager.

  > In addition to this attribute being set to "true", engineers must have appropriate permissions (*Global Permissions - Write template*) to see the menu item *Text templates*.

- **officeTemplateLinkVisible**
  Whether the menu item *Document templates* (for the start of the Document Template Manager) can be shown (boolean, default value is "true") in the main menu.

  > In addition to this attribute being set to "true", engineers must have appropriate permissions (*Global Permissions - Write template*) to see the menu item *Document templates*. CM/Doc has to be enabled in the system (see section CM/Doc).

- **overviewLinkVisible**
  Defines whether the menu item *Overview* can be shown in the main menu.

preview (Type)

The attributes define the behavior of the CM Web Client concerning the preview feature for result tables where tickets are listed.

Available on

- the Search Detail Page (tab *Tickets*): preview**/searchDetailPage/TicketSearchResults**
- the resource page (ticket section): preview**/resource/ticketRelationsSection/ResourceTicketRelationSearchResults**

Set the attributes for the subscope printed in bold font.

Attributes:

- **enabled**
  Boolean. Enable / disable the inline preview feature. Default "true".
- **previewHeight**
  Integer. Height of the attachment preview in pixels. Default "200".

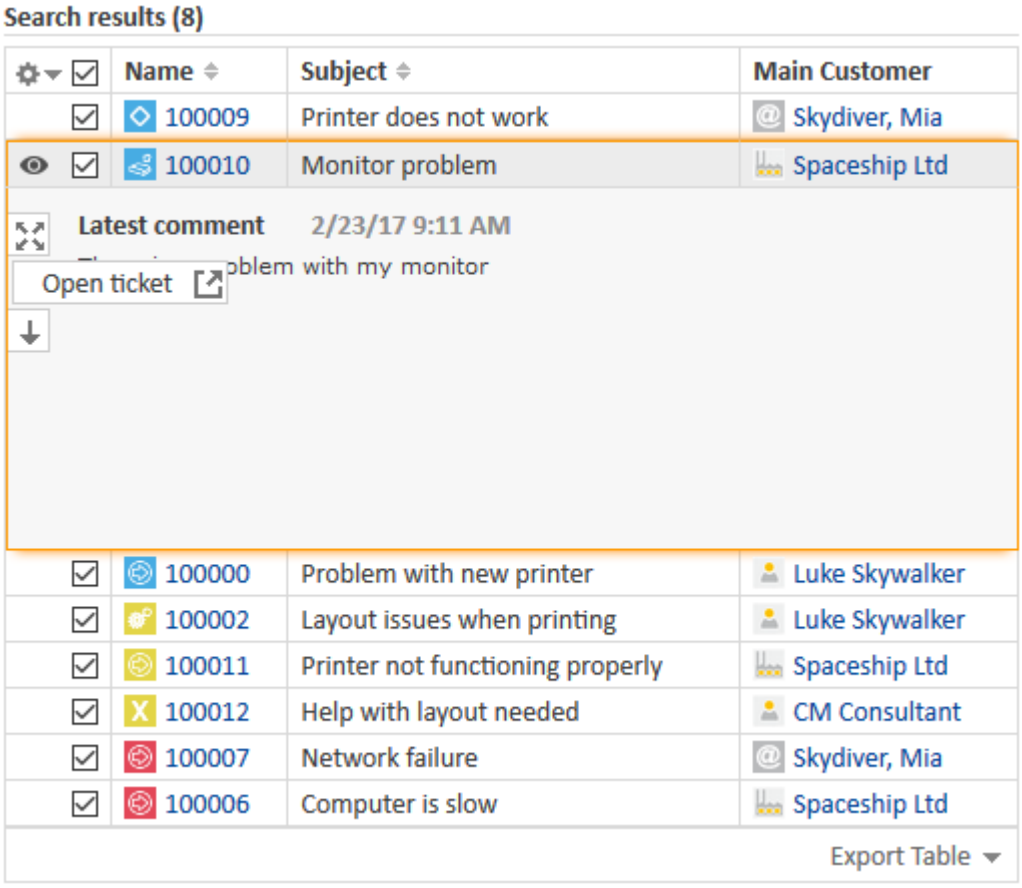


Figure 187: *ConSol CM Web Client - Preview with /searchDetailPage/TicketSearchResults, enabled = true*

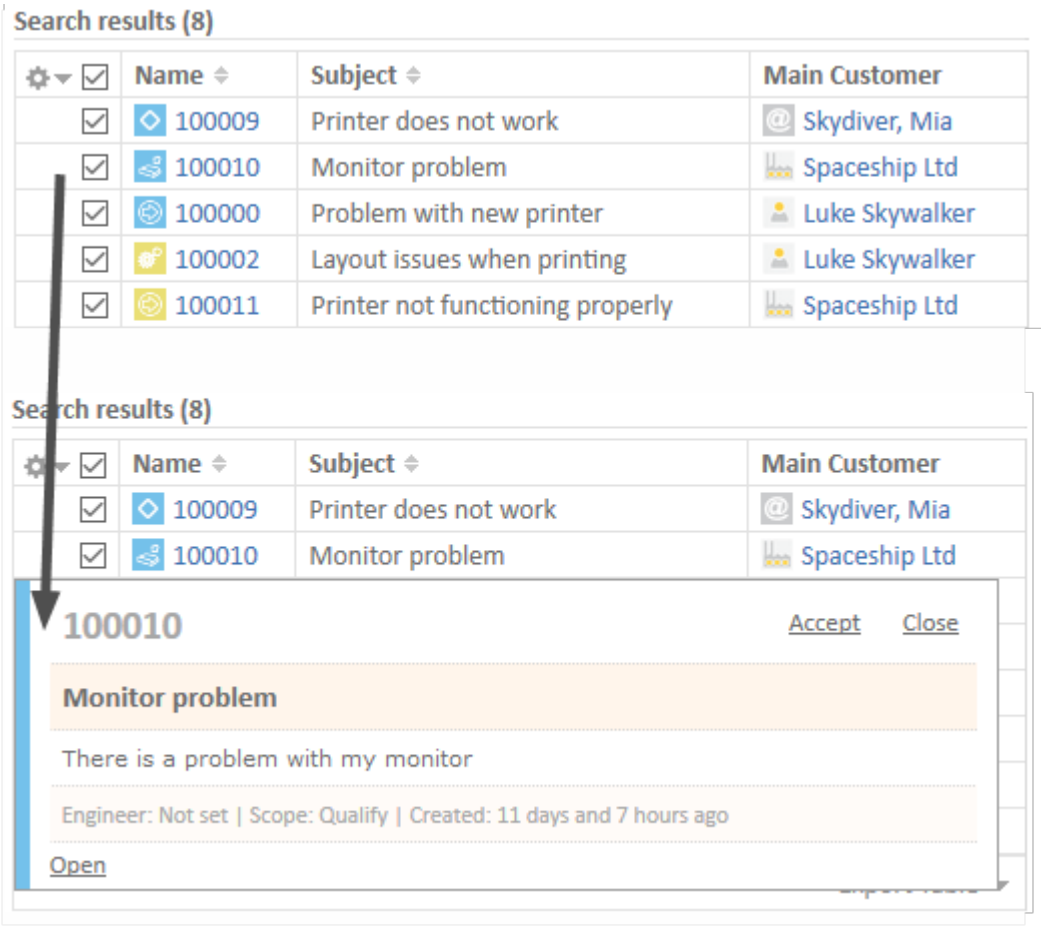


Figure 188: *ConSol CM Web Client - Preview with /searchDetailPage/TicketSearchResults, enabled = false*

For a detailed description of the preview functionality, please refer to the *ConSol CM User Manual*.

resourceRelations

Available e.g., on ticketEditPage.

Attributes:

- **counterVisible**
  Whether the counter should be shown in the section header. The default value is “true”.
- **defaultRelationSortStrategy**
  Additional resource relation sorting strategies. Possible options are: CREATION_DATE, DESCRIPTION, Default CREATION_DATE.
- **state**
  The visibility mode of the section, possible values are [expanded, collapsed, collapsed_and_pre-load, hidden], default: “expanded”.

resourceRelationsPanel

Available, e.g., on the TicketEditPage.

Attributes:

- **preserveOrder**
  Whether to preserve order of elements. By default elements will be placed in an optimal position based on available vertical space. Default is “false”.

resourceTypes

Available, e.g., on the Resource Dashboard

Attributes:

- **preserveOrder**
  Whether to preserve the order of elements. By default elements will be placed in an optimal position based on available vertical space. Default is “false”.

section (Type)

Available in the following scopes:

- ticketEditPage
- contactEditPage
- companyEditPage
- userProfilePage

Attributes:

- **state**
  The visibility mode of the section. A detailed explanation is provided under *sectionList*

sectionList (Type)

Available in the following scopes:

- ticketEditPage
- contactEditPage
- companyEditPage

Attributes:

- **counterVisible**
  Whether the counter should be shown in the section header. The default value is “true”.

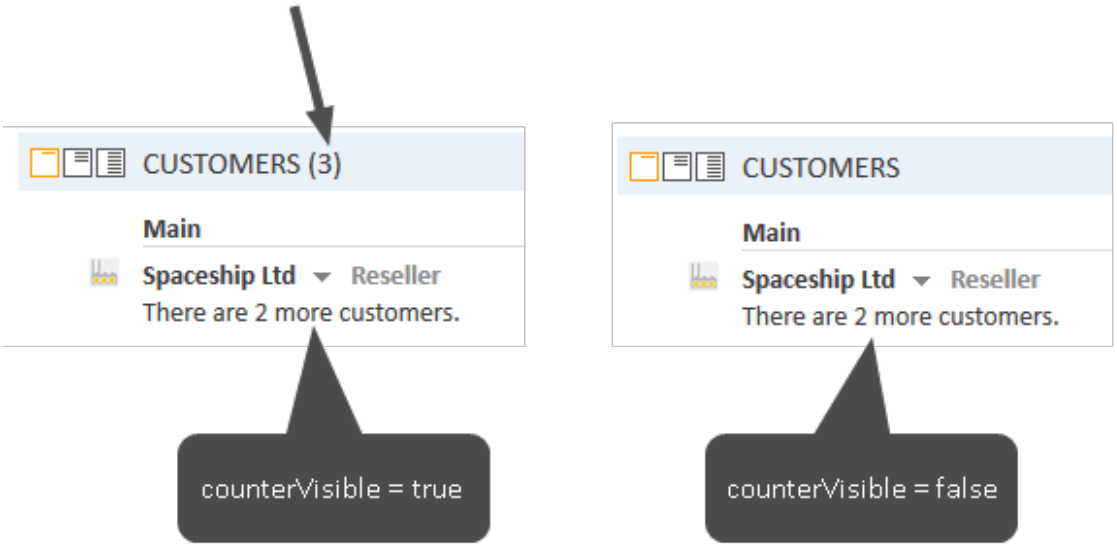


Figure 189: *ConSol CM Web Client - Attribute counterVisible*

- **state**
  The visibility mode of the section. Possible values are [expanded, collapsed, collapsed_and_preload, hidden]. The default value is “expanded”. This mode defines the initial visibility mode after start of a session. The engineer can change the visibility of a section afterwards, but this will not be saved. When the next session is started (at the next login), the initial visibility mode will be applied again.
  - **expanded**
    Default, data are shown initially
  - **collapsed**
    Data are not shown initially and will be loaded only on demand, can provide some performance improvement
  - **collapsed_and_preload**
    Data are not shown initially, but will be loaded
  - **hidden**
    The section is completely hidden and cannot be made visible. Can only be reversed using tools like the application server admin console.

> **Attribute HIDDEN will hide all customer data!**
>
> When the attribute *state* = “hidden” has been set for a section/scope, this scope will not be displayed in the page customization tree and thus will not be available in the page customization anymore. You can reverse this setting only using tools which directly access the Java beans, e.g., ***JConsole*** for JBoss. So please think twice before applying this value.

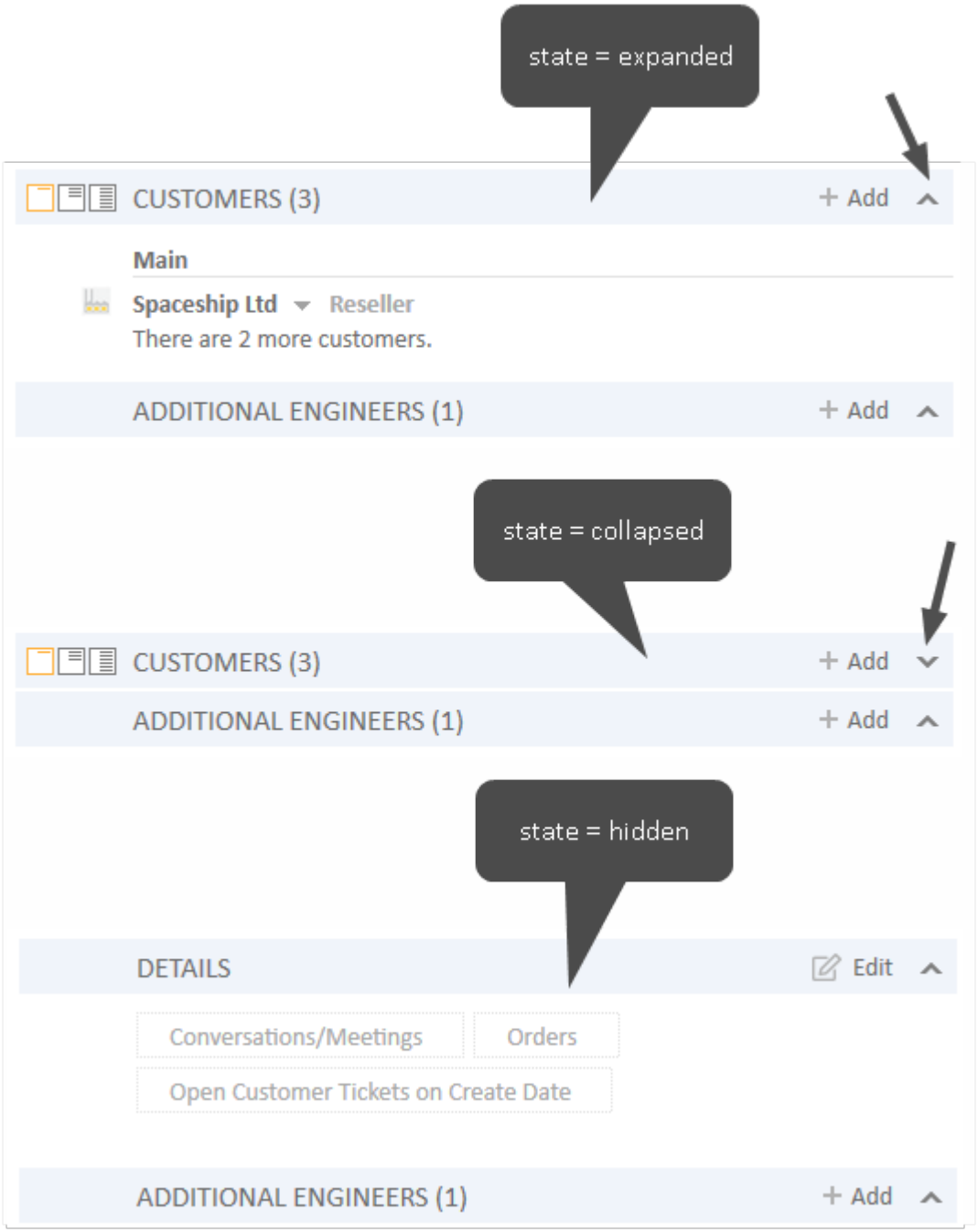


Figure 190: *ConSol CM Web Client - Attribute state (example: Customers section of a ticket)*

table

This is used to configure the csv export functionality in search result pages. Please refer to section CSV Export of Search Results for an explanation.

template

Available on all pages which feature the Rich Text Editor with text template insertion. The format is always defined on the lowest scope for which the attribute value is available.

Attributes:

- **templateNumberFormat**
  Used to define the format of numeric variables, i.e., of types integer (number) or decimal (fixed-point number) for the placeholders (markers) in text templates.
  Example:

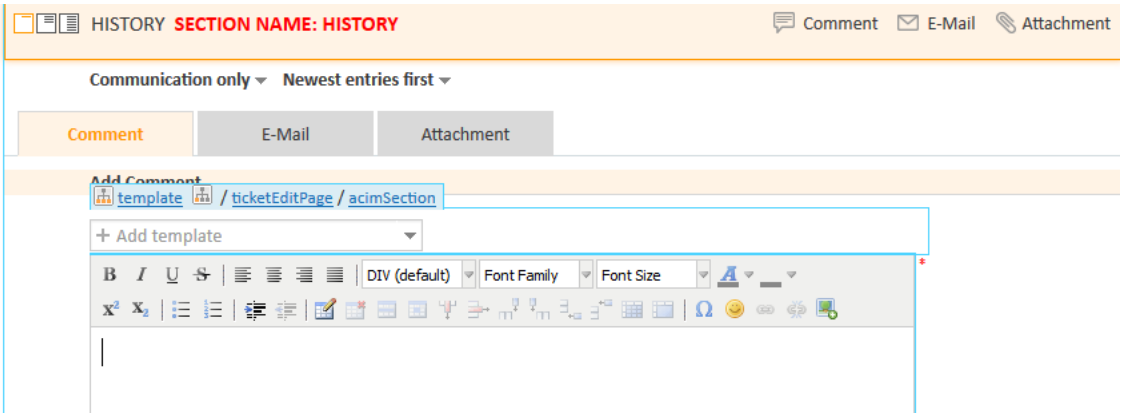


Figure 191: *Page Customization enabled in Web Client to set templateNumberFormat. Use acimSection!*

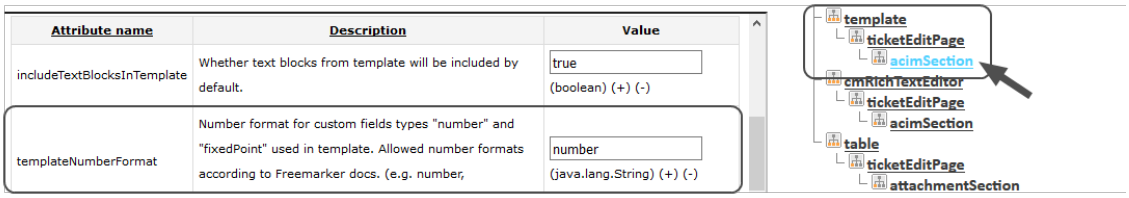


Figure 192: *Page Customization attribute templateNumberFormat in acimSection*

The currently shown value “number” represents the default format and it can be used to return to the default after having a dedicated format configuration previously. The format definition syntax is explained in the public Java documentation for the class DecimalFormat.

The most important elements are:

- “0” - shows a digit in this place always, will show “0”, i f no digit is in this position.
- “#” - shows only, if the number has a digit in this position.
- “.” - decimal separator
- “,” - internal grouping separator for integer/decimal positions

For example “00000.#####” as attribute value will always show five digits before the decimal separator and show up to five digits after the decimal separator, depending on how many are present for decimal numbers. It will always display integers in five digit notation. This effect can be observed in the illustration below. The first number in blue is an integer, the second one is a decimal number with three decimal digits.

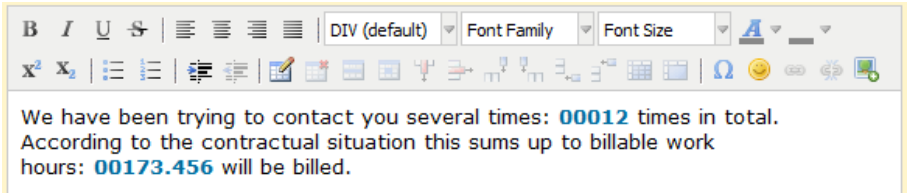


Figure 193: *Text template with numbers in the Text Template Manager*

ticketList (Subscope)

see accordionTicketList (Type).

ticketPanel

Attributes:

- **scrollSpeed**
  Scroll speed in milliseconds. The attribute is used to determine the speed of page scrolling when you click on a handler on the right side of the main page section.

The value will determine how long the animation will run. Typical values: 200, 600, 1000 ... (higher value means slower) (java.lang.String, default = “200”). E.g., 200 means that the scroll to the bottom/top of the page will take 200ms.

- **topBottomPageButtonVisible**
  Whether *go to top and bottom page* button is visible (boolean, default is “false”).
  - **topBottomPageButtonVisible = false:**

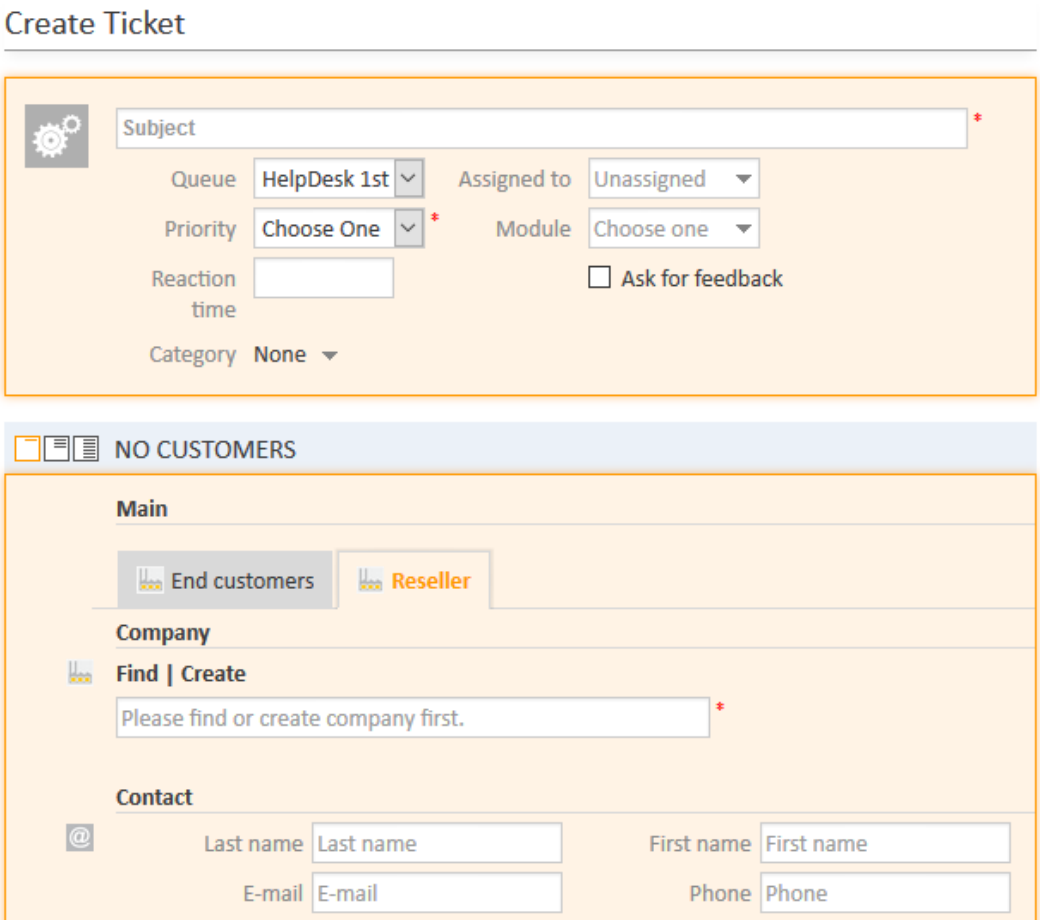


Figure 194: *ConSol CM Web Client - Scroll button not visible (topBottomPageButtonVisible = false)*

- **topBottomPageButtonVisible = true:**

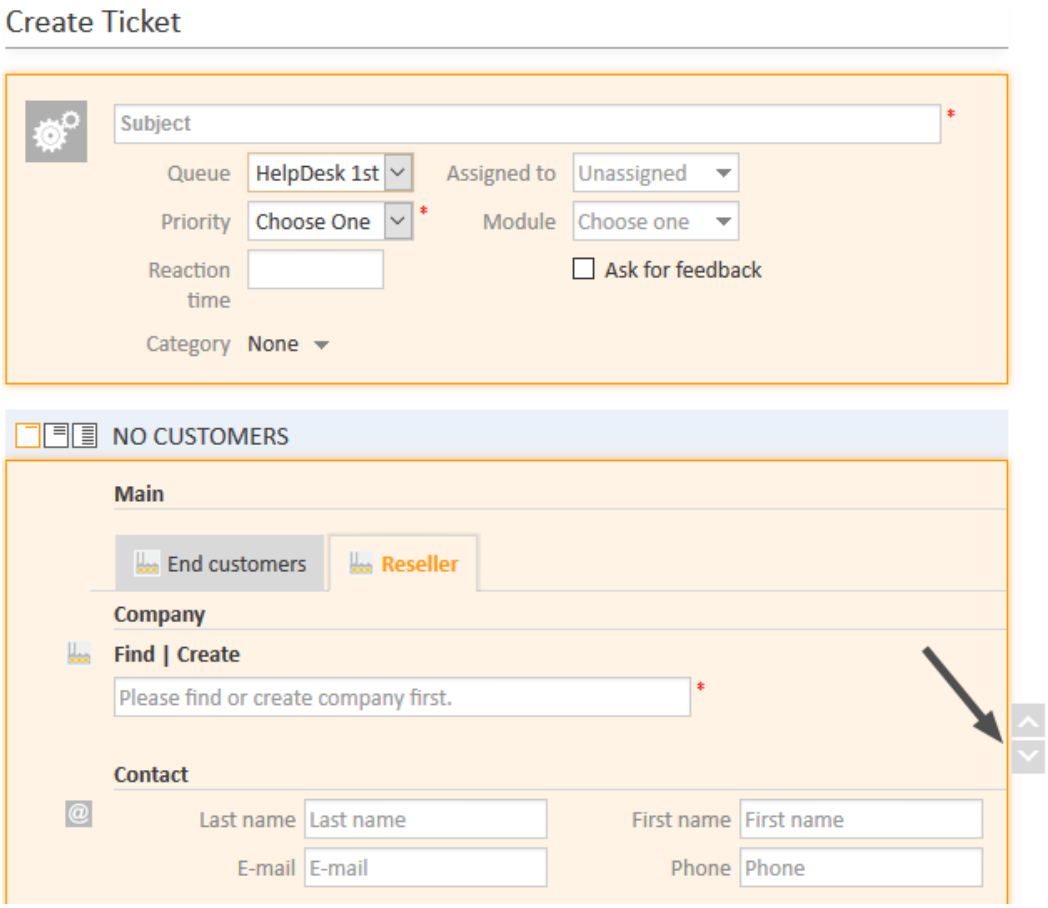


Figure 195: *ConSol CM Web Client - Scroll button visible (topBottomPageButtonVisible = true)*

TicketRelation

Available in the following scopes:

- TicketRelationSection (e.g., on the resource page)

Attributes:

- **compactViewLimit**
  If the limit is exceeded, the table is presented along with a filtering feature. The default limit is "10".
- **defaultRelationSortStrategy**
  Additional resource relation sorting strategies. Possible options are: CREATION_DATE (default), FIRST_SORTABLE_COLUMN (sort by the first column which is sortable)

ticketsAutocomplete

(available on the relations addition form)

Attributes:

- **maxHints**
  Defines the maximum number of suggestions to display. When set to "0", all suggestions are displayed, with no limit.

ticketsBookingAutocomplete

(available on the time booking addition form of the *userProfilePage*)

Attributes:

- **maxHints**
  Defines the maximum number of suggestions to display. When set to "0", all suggestions are displayed, with no limit.

timeBookingSection

(available e.g., on *userProfilePage*)

Attributes:

- **visible** (up to CM version 6.9.3.x)
  Boolean. The visibility of the time booking section on the *userProfilePage*. (default value = "true")
  Please keep in mind that the visibility of the time booking section on the ticket page is configured via the acimSection, attribute *timeBookingFeature*!
- **state** (CM version 6.10.4 and up)
  Possible values are:
  - **expanded**
    Default, data are shown initially
  - **collapsed**
    Data are not shown initially and will be loaded only on demand.
  - **collapsed_and_preload**
    Data are not shown initially, but will be loaded.
  - **hidden**
    The section is completely hidden and cannot be made visible). Can only be reversed using tools like the application server admin console.

    Attribute **hidden** will hide the time booking section permanently.

    When the attribute *state* = "hidden" has been set for a section/scope, this scope will not be displayed in the page customization tree and thus will not be available in the page customization anymore. You can reverse this setting only using tools which directly access the Java beans, e.g., ***JConsole*** for JBoss. So please think twice before applying this value.

unitAutocomplete

(available on the customer addition and creation forms)

Attributes:

- **maxHints**
  Defines the maximum number of suggestions which is displayed. When set to “0”, all suggestions are displayed, with no limit.

unitFormPanel

Available on:

- contactCreatePage
- ticketEditPage
- contactEditPage, e.g., unitFormPanel / ticketCreatePage / contactSection

Attributes:

- **maxSuggestions**
  This refers to the customer section which is displayed when you create a new ticket. Here, suggestions for customers are displayed if matching hits are found in the database. The number of suggestions which are displayed can be configured using this attribute.

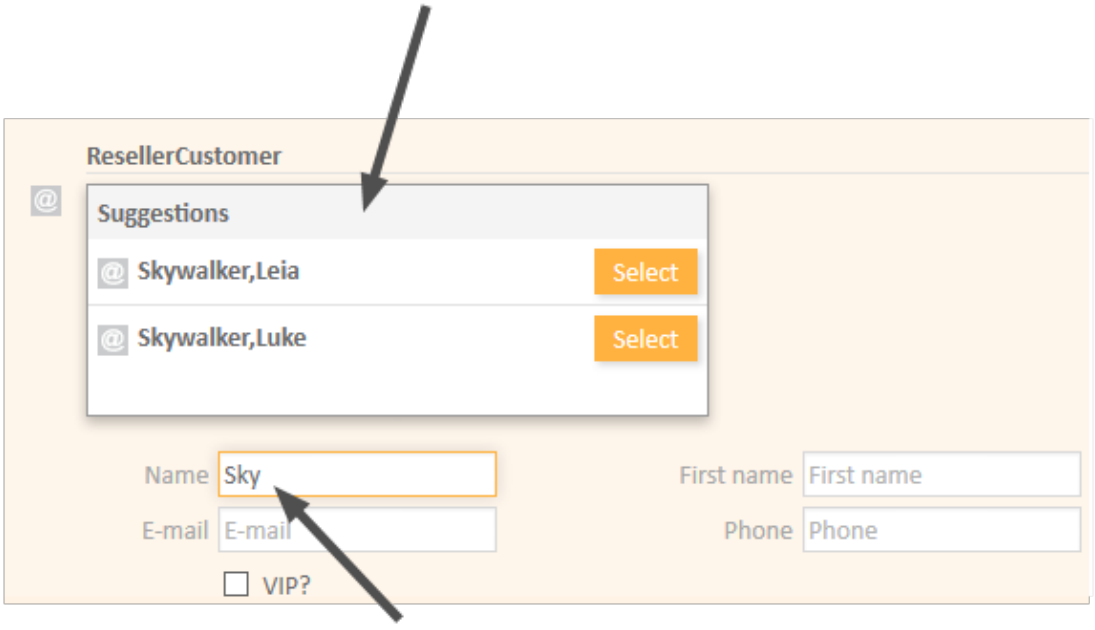


Figure 196: *ConSol CM Web Client - Suggestions for the ticket contact*

unitRelationSection (Type UnitSection)

Available on:

- contactEditPage
- companyEditPage

Attributes:

- **compactViewLimit**
  If the limit is exceeded, the relations are presented along a with filtering feature. The default limit is “10”.

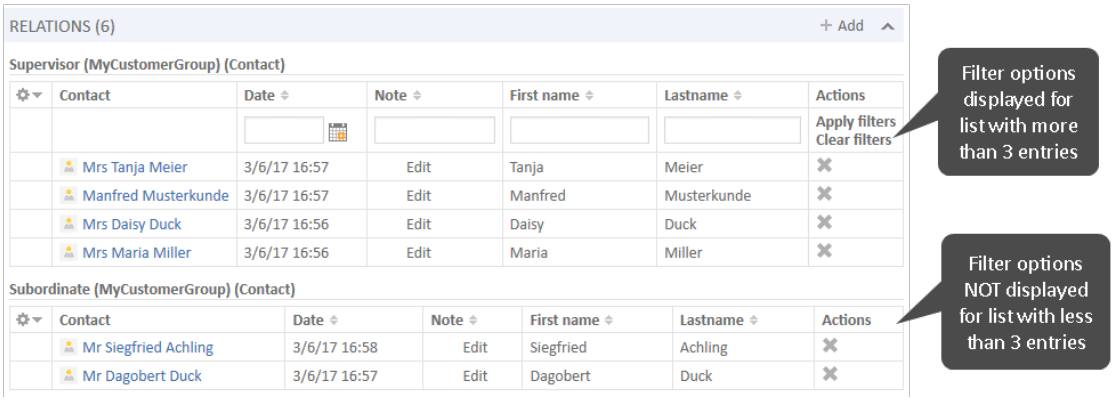


Figure 197: *ConSol CM Web Client - Effect of attribute compactViewLimit = 3 (contactEditPage / unitRelationSection)*

- **numberOfRelations**
  The default number of relations to display. The default value is “10”.
- **unitPreviewLayout**
  The unit preview configuration, this is used for the box which is displayed when you click on the name of a customer or company in the relation section. Similar to the box which is displayed when you click on a ticket name in a list which contains tickets.
  A JSON object has to be returned, as demonstrated in the next code example.

```
{ "layout": [ ["firstname", "lastname", "lastname"], ["email", "email", null]
 ]}
```

Code example 3: *JSon object for customer data format in box preview*

For different unit definitions in the FlexCDM use the following syntax:

```
{"unit_definition_1": "<json description>", "unit_definition_2": "component_
 name_used_to_display_preview"}.
```

Code example 4: *Example value for customer data preview box layout*

In JSON we set the configuration for a particular unit definition. The JSON can contain a unit preview or just the name of a component (spring bean) used to render the preview.

UnitResourceRelation

Available in unitRelationsSection (e.g., on the resource page).

Attributes:

- **compactViewLimit**
  If the limit is exceeded, the relations are presented along with a filtering feature. The default limit is 10.
- **defaultRelationSortStrategy**
  Additional resource relation sorting strategies. Possible options are: CREATION_DATE (default), FIRST_SORTABLE_COLUMN (sort by the first column which is sortable)
- **numberOfRelations**
  The default number of relations to display. The default value is “10”.

unitSearch

(available on the *ticketCreatePage* in the company section)

Attributes:

- **aidLevel**
  Beginner-friendly help level:
  - NONE
  - BASIC (wider search field with more descriptive text)
  - EXTENDED (as in BASIC plus additional help icon with tool tip)

  (java.lang.String, default value = “BASIC”)

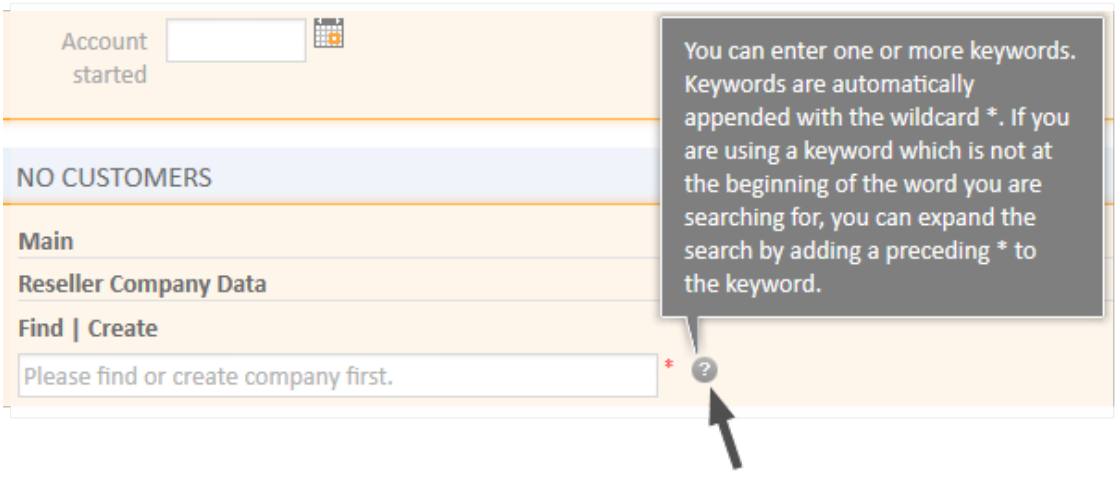


Figure 198: *ConSol CM Web Client - EXTENDED aidLevel in the unit search*

unitSearchHeader

(available on *ticketCreatePage* in the company section)

Attributes:

- **companyCreateLinkVisible**
  Boolean. The visibility of the link for referenced company creation.

viewDiscriminatorsSection (Type)

(available, e.g., on *userProfilePage*)

Attributes:

- **visibilityFlag**
  Boolean. The visibility of the *View criteria* section (section for attribute settings for dynamic views on *userProfilePage*). (default value = “true”)

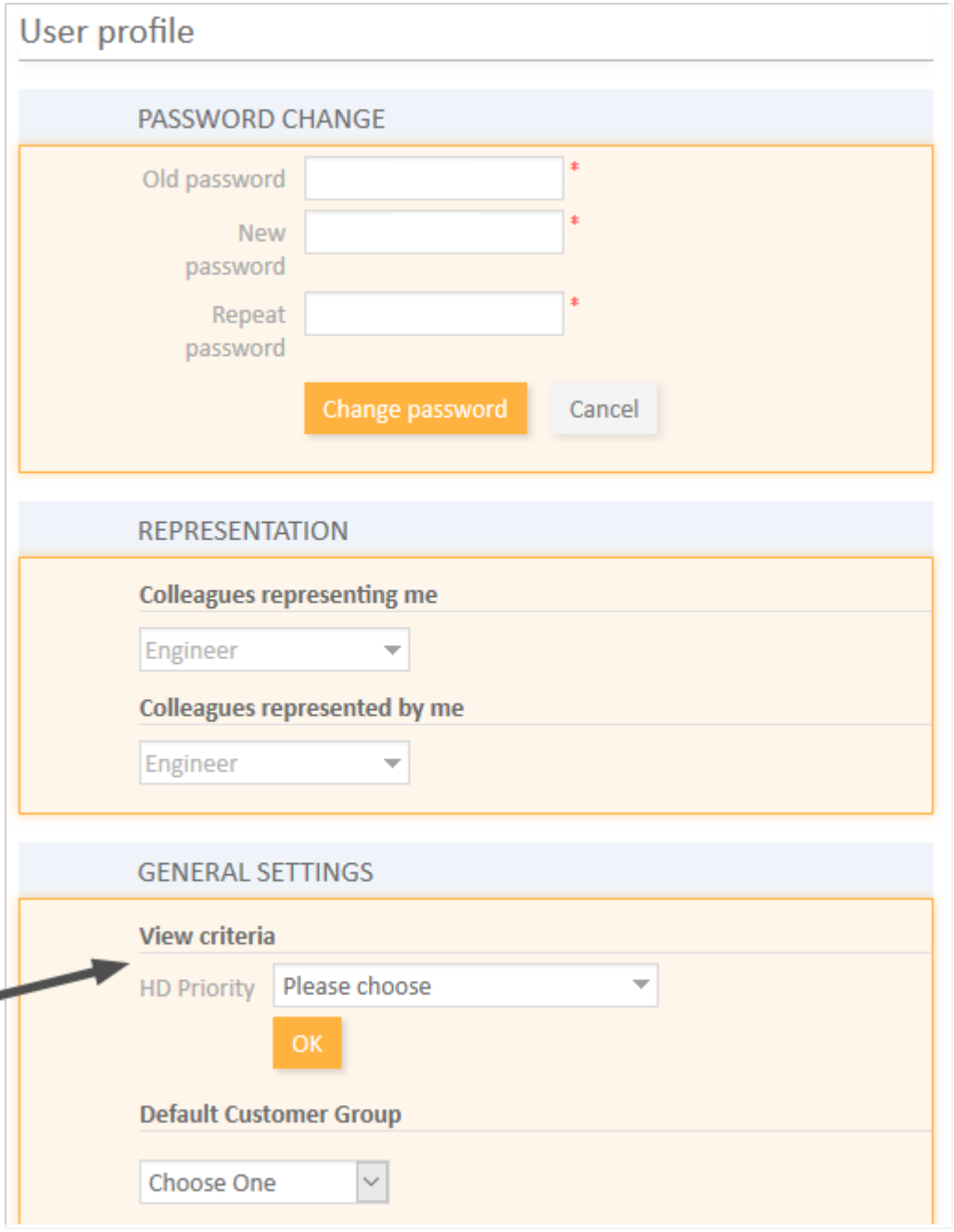


Figure 199: *ConSol CM Web Client - Visibility of View criteria section (visibilityFlag=true)*

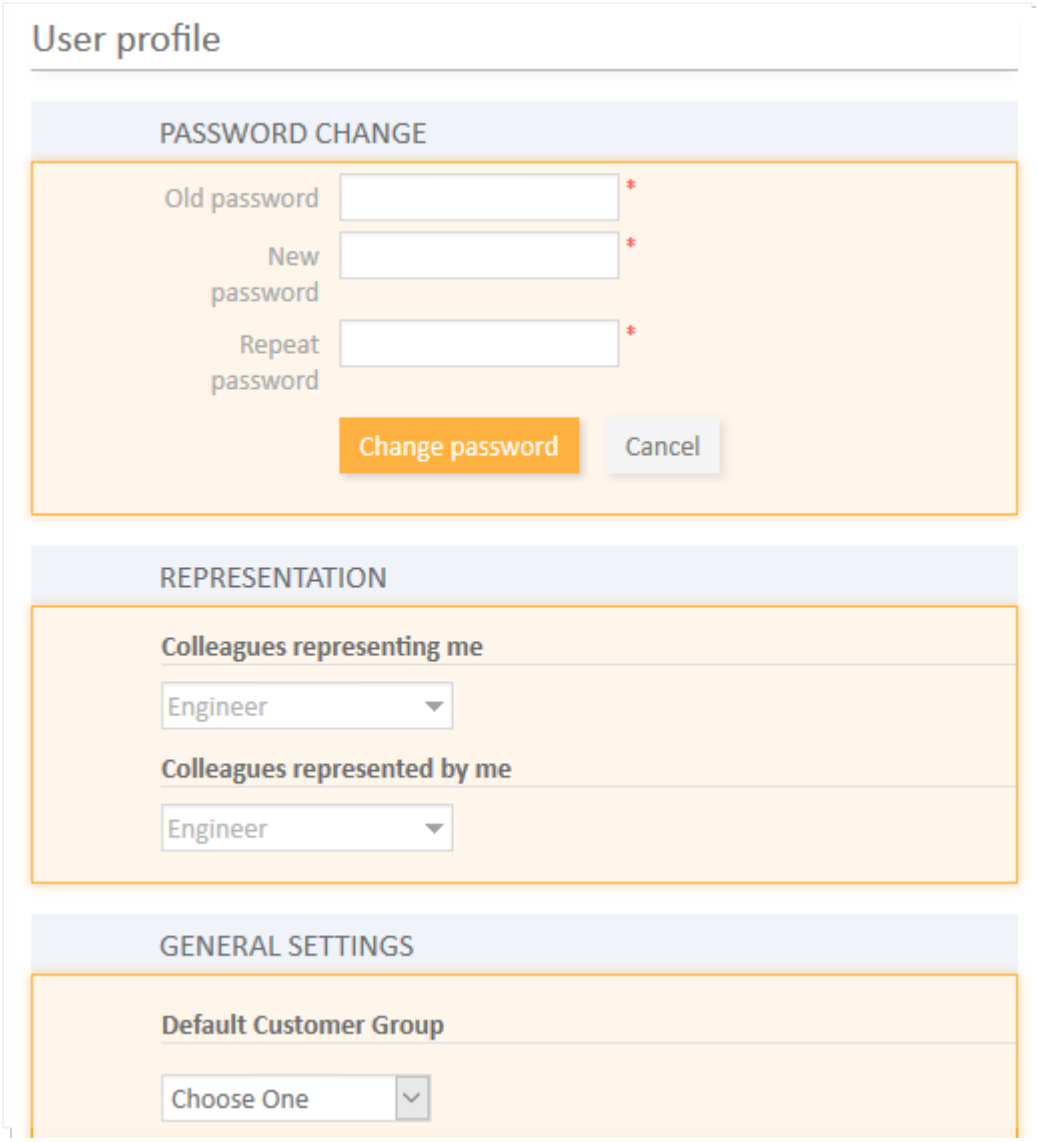


Figure 200: *ConSol CM Web Client - Visibility of View criteria section (visibilityFlag=false)*

welcomePage

The most important configuration on the *welcomePage* (Overview page) is the Page Customization for the Web Client Dashboard.

C.8.6 Page Customization for the Web Client Dashboard

C.8.6.1 Introduction

The **Web Client Dashboard** is configured using Page Customization.

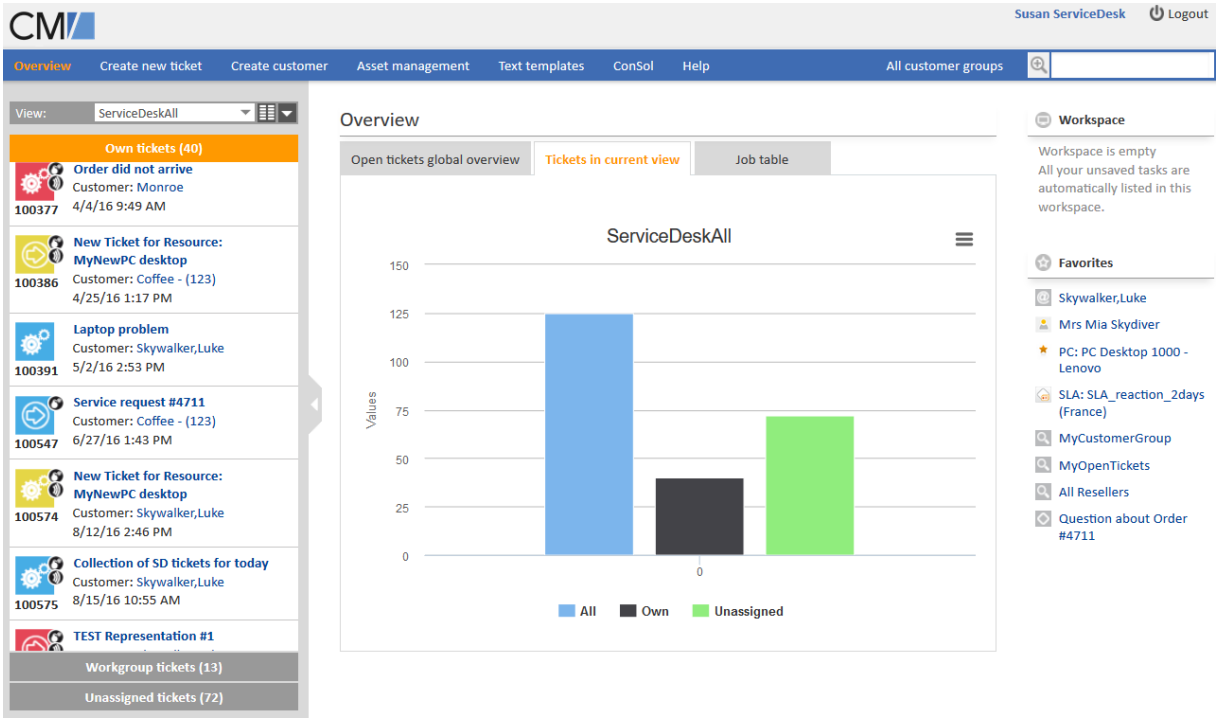


Figure 201: *ConSol CM Web Client - Web Client Dashboard with several tabs, open tab with chart widget*

Log in as *admin*, open the *Overview* page and select *Enable page customization* in the main menu. The following (Dashboard-relevant) elements can be configured here:

1. ***widgetsGrid / welcomePage***
   Here, the Web Client Dashboard can be switched on or off. If a correct value is entered in the *layout* field, the Dashboard is displayed (**Attention:** If the value is not correct, the Web Client will not start!). If the field is empty, no Dashboard is shown.
   The following configuration can be made here:
   a. the Dashboard layout, i.e., the layout of the grid on which the Dashboard is based (see section Definition of the Overall Dashboard Layout, this comprises:
      i. the widgets which should be displayed
      ii. the order and organization of those widgets on the Dashboard page

2. ***chartWidget / welcomePage*** (only available if chart widgets are present)
    a. the definition/layout for all chart widgets in the *chartWidget* subtree
    b. each widget is represented by one node which has the name of the widget, e.g., *chartWidget / welcomePage / ticketsInView*
    c. a new chart widget is added when its name has been added in the *layout* value
    d. attributes can be defined for all chart widgets on the level *chartWidget* or *chartWidget / welcomePage* or they can be configured for one chart widget individually using the values of the attributes for the chart widget, e.g., *chartWidget / welcomePage / ticketsInView*
    e. Chart widget attributes are explained in detail in section Attributes for Chart Widgets.
3. ***tableWidget / welcomePage*** (only available if table widgets are present)
    a. the definition of all table widgets in the *tableWidget* subtree
    b. each widget is represented by one node which has the name of the widget, e.g., *tableWidget / welcomePage / ticketsOverview*
    c. a new table widget is added when its name has been added in the *layout* value
    d. attributes can be defined for all table widgets on the level *tableWidget* or *tableWidget / welcomePage* or they can be configured for one table widget individually using the values of the attributes for the table widget, e.g., *tableWidget / welcomePage / ticketsOverview*.
    e. Table widget attributes are explained in detail in section Attributes for Table Widgets.
4. ***kpiWidget / welcomePage*** (only available if KPI widgets are present)
    a. the definition of all KPI widgets in the *kpiWidget* subtree
    b. each widget is represented by one node which has the name of the widget, e.g., *kpiWidget / welcomePage / ticketsGlobalOverview*
    c. a new KPI widget is added when its name has been added in the *layout* value
    d. attributes can be defined for all KPI widgets on the level *kpiWidget* or *kpiWidget / welcomePage* or they can be configured for one KPI widget individually using the values of the attributes for the KPI widget, e.g., *kpiWidget / welcomePage / ticketsGlobalOverview*.
    e. KPI widget attributes are explained in detail in section Attributes for KPI Widgets.
5. **recentlyVisitedWidget / welcomePage/**(only available if recentlyVisitedWidget has been configured using the layout option)
    a. The configuration of the appearance of the recentlyVisitedWidget using attributes.
    b. The attributes of the recentlyVisitedWidget are explained in detail in section Attributes and Settings for the recentlyVisitedWidget.
6. **recentChangesWidget / welcomePage** (only available if recentChangesWidget has been configured using the layout option)

a. The configuration of the appearance of the recentChangesWidget using attributes.

b. The attributes of the recentChangesWidget are explained in detail in section Attributes for the recentChangesWidget.

As explained before in section Page Customization each of the four elements is represented by a subtree in the page customization tree. The following figure provides an example page customization tree with subtrees relevant for the Web Client Dashboard. A detailed explanation is provided in the following sections.

Please note that when you work with a tabbed dashboard, only the page customization subtree(s) of the active (i.e., currently opened) tab is/are displayed.

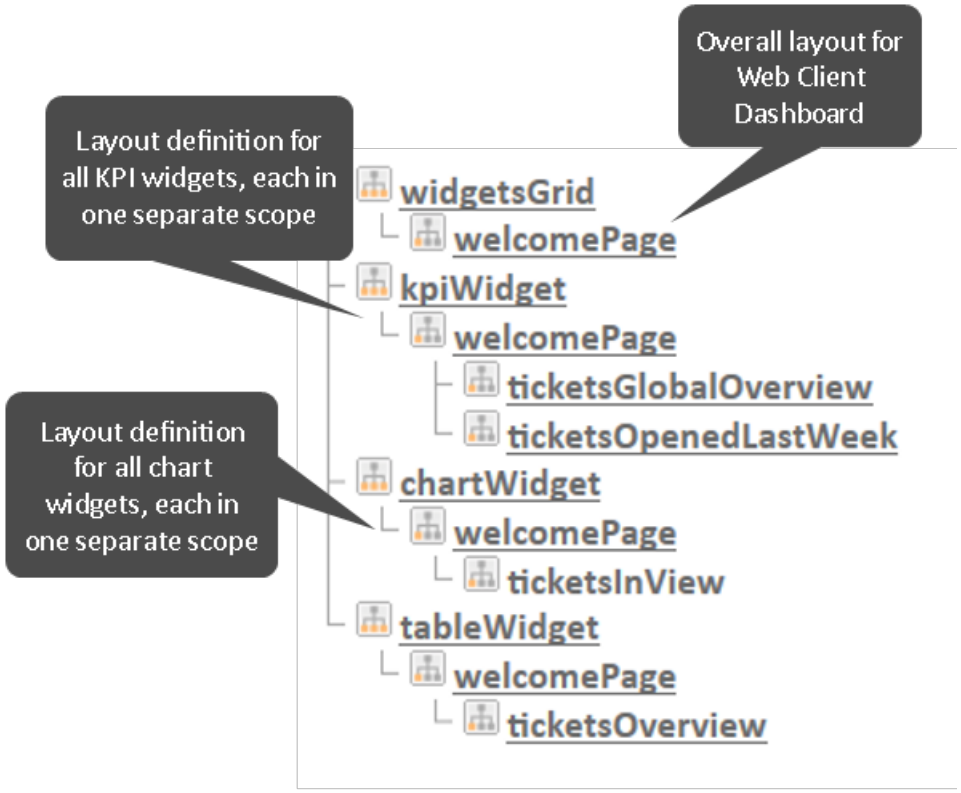


Figure 202: *ConSol CM Web Client - Page customization subtrees for Web Client Dashboard layout*

C.8.6.2 Definition of the Overall Dashboard Layout

The overall layout of the entire Web Client Dashboard is defined using the page customization attribute *widgetsGrid/welcomePage*.

Variant 1: All Widgets Displayed on One Page

Attributes:

- **layout**
  This defines the layout of the entire Dashboard on the *welcomePage* based on the following principles:
  - A widget is described by its name and its type, separated by a colon, e.g., 'ticketsInView:Chart'. The name for a specific widget must be unique.
  - The type of a widget is *Chart* or *Table*. The type has to be indicated only at the first appearance of the widgets name, afterwards, it can be omitted, e.g., [ticketsInView:Chart, ticketsInView, ticketsInView].
  - Each row of the Dashboard grid is represented as an array of elements: [x,y,z]. A new widget object will be added to the page customization tree automatically when it is added as value in the *layout* attribute, e.g., when the value *has been* [ticketsInView:Chart, ticketsInView] and the value is now [ticketsInView:Chart, ticketsInView, myTickets:Table], another table widget named *myTickets* will appear in the page customization tree (see figure above) and on the Dashboard. In the same way, widgets can be removed from the Dashboard - just remove the name and type of the widget in the *layout* value. After saving and reloading the page all layout changes are available in the tree for further configuration.
  - The grid starts with the upper left corner (0,0) and it is built up row after row, e.g., a *layout* value with two pairs of [ ] brackets will represent two rows as shown in the figure and code shown below.
  - *null* is a reserved key word for an empty cell.
  - The widget can occupy multiple adjacent rows and columns.
  - The Dashboard can be completely disabled in removing the value from the attribute *layout*.

The following examples show two possible layout configurations and the respective display in the Web Client.

Example 1

Three widgets are displayed, two charts and one table.

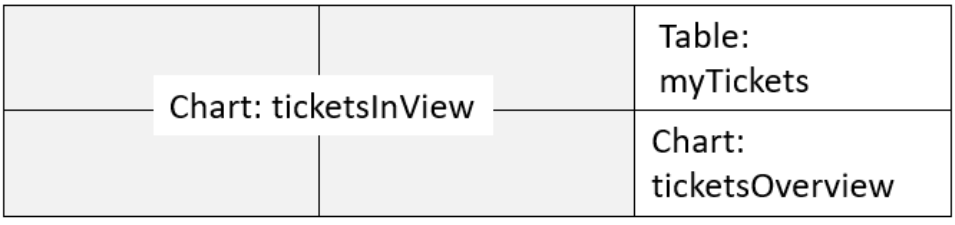


Figure 203: *Organization of an example grid of a welcomePage*

The value of the respective *layout* attribute would be:

```
[ticketsInView:Chart, ticketsInView, myTickets:Table], [ticketsInView,
 ticketsInView, ticketsOverview:Chart]
```

Example 2

Four widgets are displayed, two KPI widgets, one chart and one table.

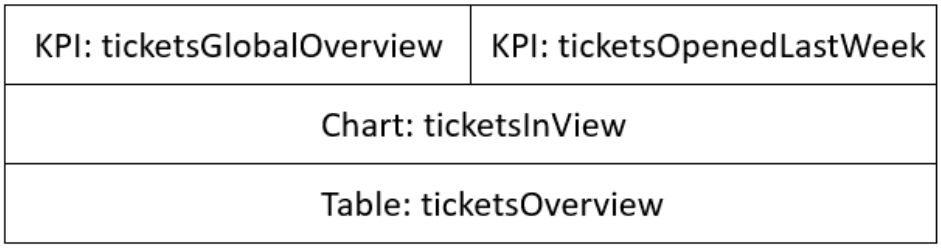


Figure 204: *Organization of an example grid of a welcomePage with KPI widgets*

The value of the respective *layout* attribute would be:

```
[ticketsGlobalOverview:KPI,ticketsOpenedLastWeek:KPI],[ticketsInView:Chart],
 [ticketsOverview:Table]
```

The Dashboard page in the Web Client looks as demonstrated in the following figure.

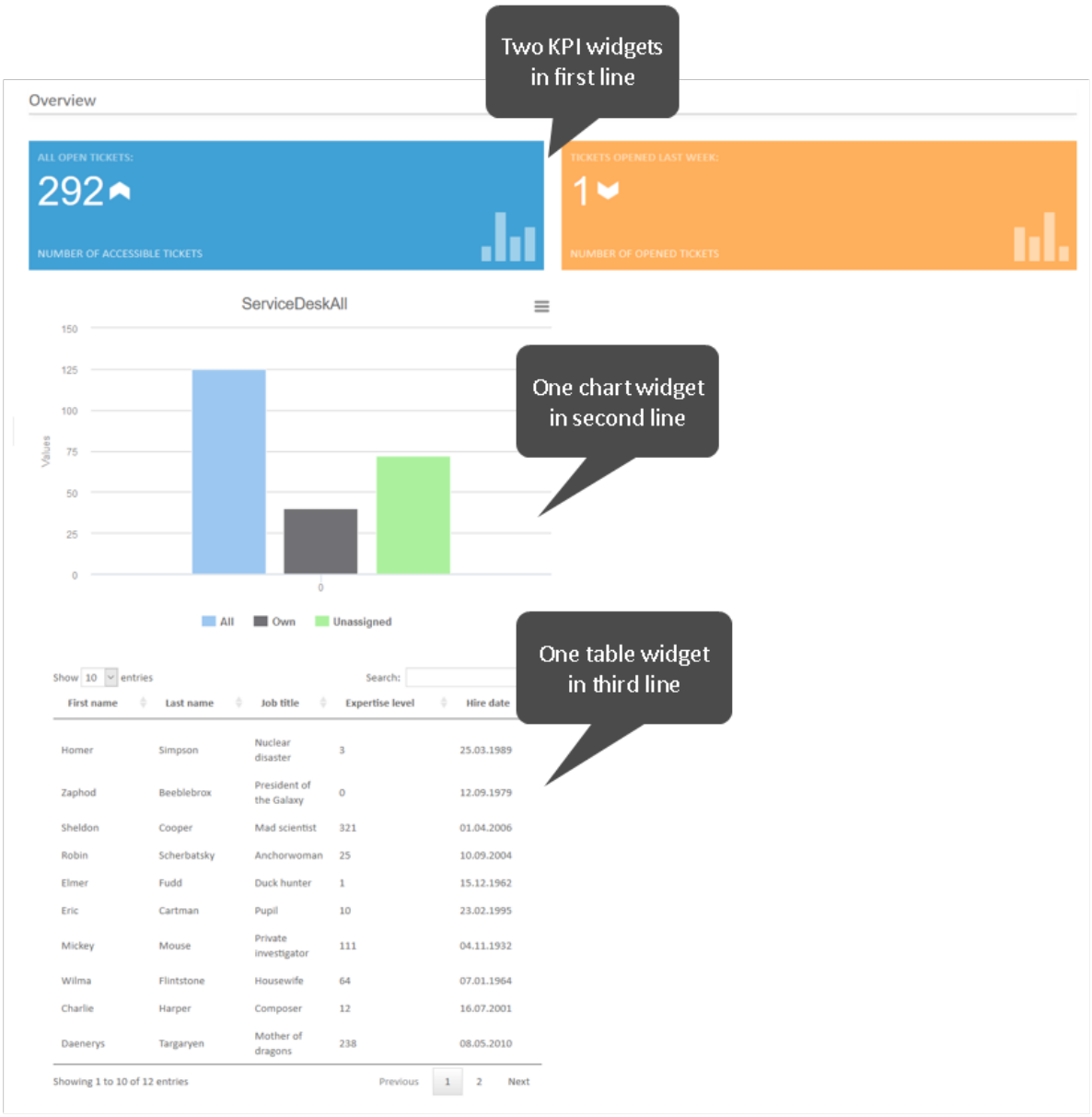


Figure 205: *ConSol CM Web Client - Dashboard with four widgets of different types*

Variant 2: Tabbed Widgets

The widgets can also be displayed on tabs. In this case, each widget is displayed on a separate tab. The layout attribute is used for this configuration.

Variant 2.1: Using Explicit Headers

The following notation for the value of the *layout* attribute sets an explicit header for each tab, i.e. the browser locale is not taken into consideration.

```
[tabName:'Tickets in current view',widgets:[ticketsInView:Chart]],[tabName:'Job
 table',widgets:[ticketsOverview:Table]]
```

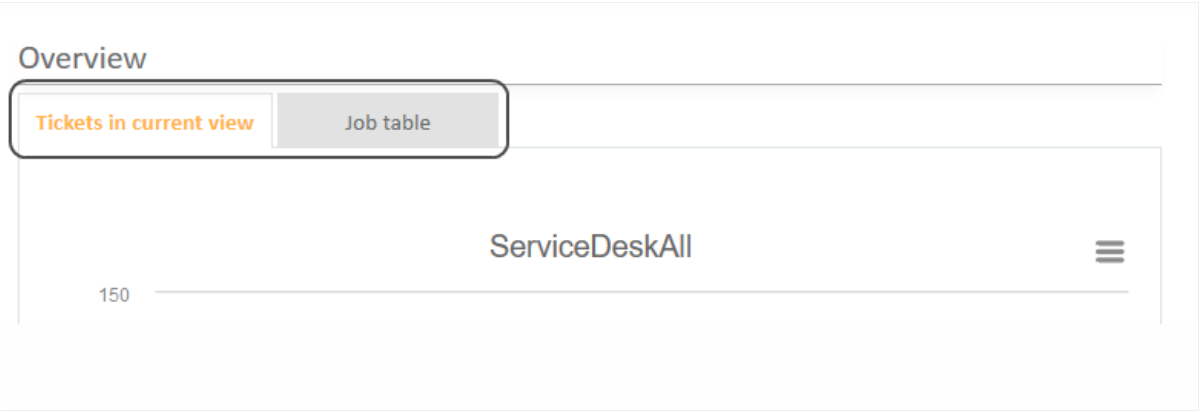


Figure 206: *Setting explicit headers as tab titles*

Variant 2.2: Using Localized Headers

The following notation for the value of the *layout* attribute sets localized headers, i.e., the header depends on the browser locale. In the example, English and German headers are specified. If the browser locale is not set, the default CM locale is used.

Localized Headers, Example 1

```
[tabName:'Tickets in current view',i18n:{en:'Tickets in current view',de:'Tickets
 in der aktuellen Sicht'},widgets:[ticketsInView:Chart]],[tabName:'Job table',i18n:
 {en:'Job table',de:'Tabelle Jobs'},widgets:[ticketsOverview:Table]]
```

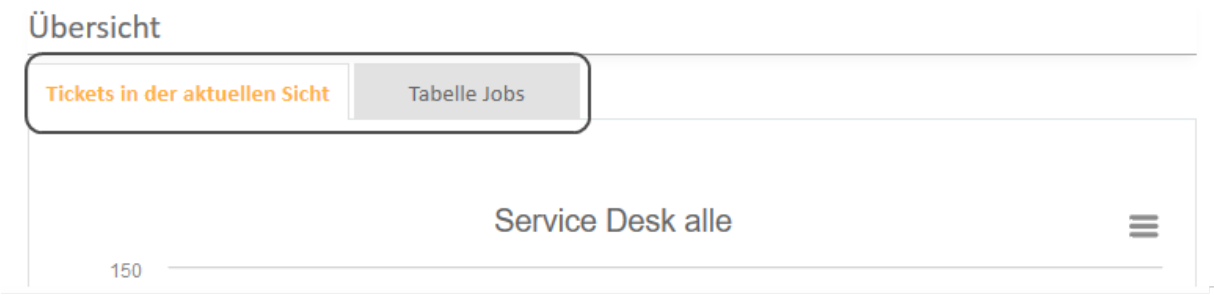


Figure 207: *Using localized headers (in the given example en and de, displayed: de)*

Localized Headers, Example 2

An example of a rather complex dashboard layout with three tabs is shown in the following code and figure.

Value for the layout attribute in *widgetsGrid/welcomePage*:

```
[tabName:'Open tickets global overview',i18n:{en:'Open tickets global
 overview',de:'Übersicht offene Tickets'},widgets:
 [ticketsGlobalOverview:KPI,ticketsOpenedLastWeek:KPI]],[tabName:'Tickets in
 current view',i18n:{en:'Tickets in current view',de:'Tickets in der aktuellen
 Sicht'},widgets:[ticketsInView:Chart]],[tabName:'Job table',i18n:{en:'Job
 table',de:'Tabelle Jobs'},widgets:[ticketsOverview:Table]]
```

Display in the Web Client, with EN browser locale:

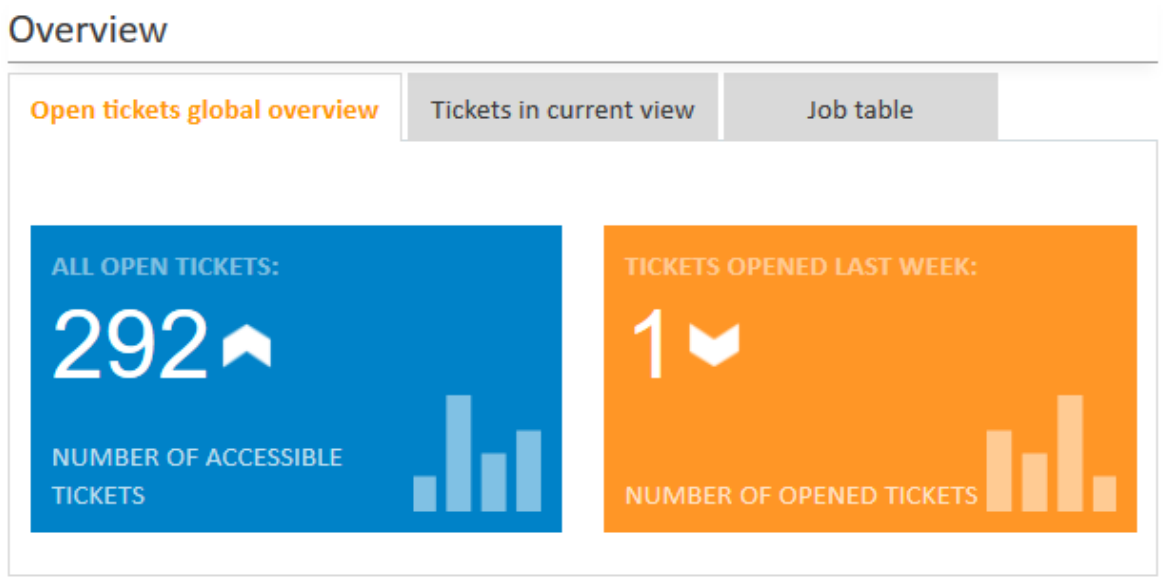


Figure 208: *ConSol CM Web Client: tabbed dashboard with two KPI widgets on the first tab*

C.8.6.3 Configuration of Widgets

Configuration Script for Widgets

Each widget has a configuration script. This script is a Groovy script which is stored in the *Scripts* section of the Admin Tool and is referenced by its name. The scripts has to be of type *Widget*.This type has been introduced in ConSol CM version 6.11. The value used in previous versions, *Page customization*, will still work. Select the widget in the PCDS and enter the script name.

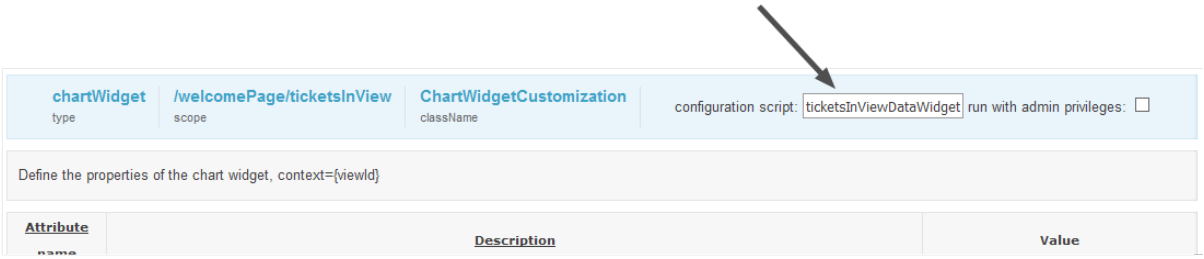


Figure 209: *ConSol CM Web Client - Script definition for a chart widget*

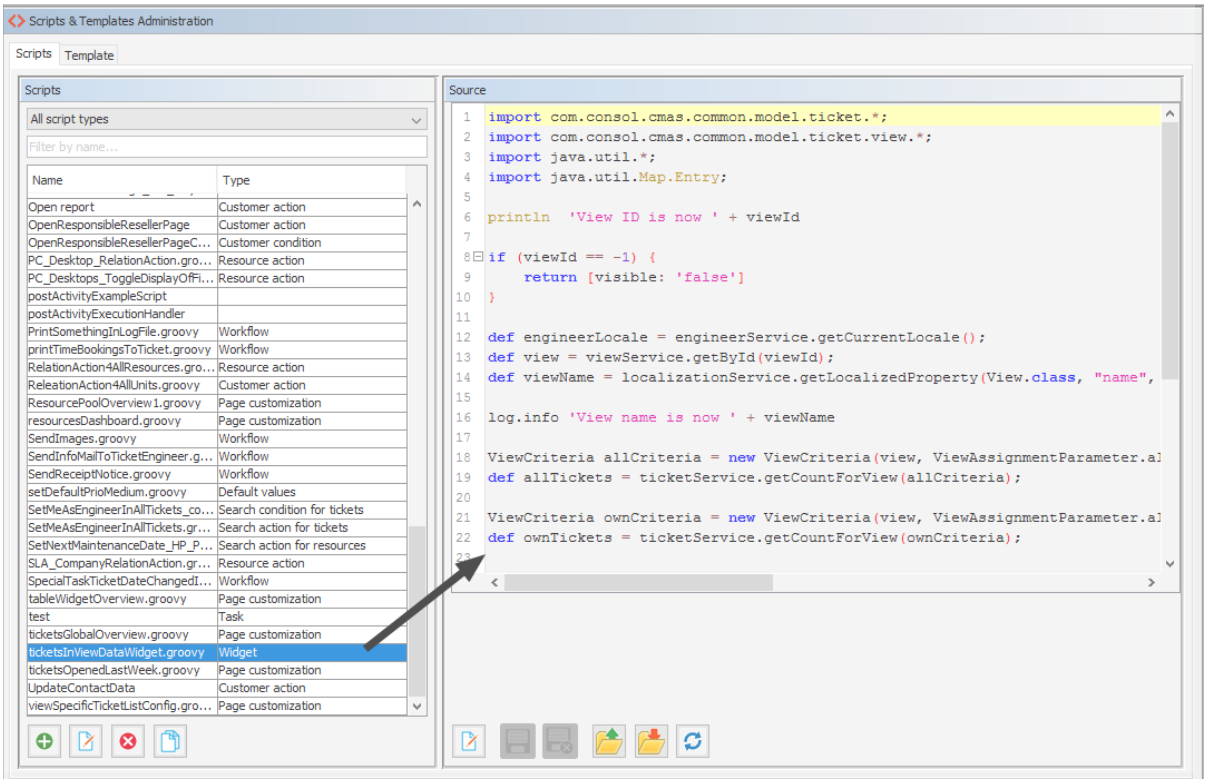


Figure 210: *ConSol CM Admin Tool - Admin Tool script for a widget of the Web Client Dashboard*

The configuration script of a widget is the place where the statements are defined which retrieve the required data from the CM system and where the widget layout is defined. The execution of this groovy script is a core part of the customization. The script must return a map of variables which correspond to the defined widget properties.

⚠ An **incorrect script** does not provide a data structure which can be displayed by the Web Client Dashboard. Since the Dashboard is displayed on the Overview page which is the start page, **the Web Client will not start** in those cases! Disable or comment out the script to start the Web Client again.

⚠ The script overwrites the configuration data provided in the page customization. The values are **not** merged! The script will thus override any widget attribute value set in the graphical customization, so please make sure that the desired attribute is not set within the script if you want to use the graphical page customization for attribute setting.

A **script** which is associated with a widget is usually executed with user (= engineer) permissions, e.g., the standard chart widget shows a graphical representation of the selected view. However, sometimes values have to be used which are not available in the engineer context, e.g., escalated tickets (of all engineers) in a certain queue. In order to execute a script with admin permissions, select the check box *run with admin privileges*. Please keep in mind that the results of the Java or Groovy methods which retrieve the data will vary depending on the context. For example, the method `ticketService.getAll()` will return only tickets for which the current engineer has at least read permissions, but will return all tickets in the system when executed as admin.

The **chart** representation in the Web Client Dashboard is based on the Highcharts library. Thus, for chart widgets, the Admin Tool script has to return a HashMap containing the attributes which should be set (see return statement in the code example below which uses the attributes *series*, *visible*, *chart*, *title*, *tooltip*, and *localization*). A detailed explanation of all attributes and the respective hyperlinks is provided in the section Attributes for Chart Widgets.

The **table** representation in the Web Client Dashboard is based on the Datatables library. Thus, for table widgets, the Admin Tool script has to return a HashMap containing the attributes which should be set. Please see section Attributes for Table Widgets.

The **KPI** representation in the Web Client Dashboard is based on the `jquery-kpiwidget` library. For KPI widgets, the Admin Tool script has to return a HashMap containing the attributed which define the KPI representation. For a detailed explanation, please refer to the section Attributes for KPI Widgets.

⚠ Very complex scripts can decrease system performance!

The following example shows the script `ticketsInViewDataWidget.groovy` which is provided with a standard ConSol CM distribution.

```
import com.consol.cmas.common.model.ticket.*;
import com.consol.cmas.common.model.ticket.view.*;
import java.util.*;
import java.util.Map.Entry;

if (viewId == -1) {
   return [visible: 'false']
}

def engineerLocale = engineerService.getCurrentLocale()
def view = viewService.getById(viewId)
def viewName = localizationService.getLocalizedProperty(View.class, "name", viewId,
 engineerLocale)
```

```
ViewCriteria allCriteria = new ViewCriteria(view,
   ViewAssignmentParameter.allTickets(),
   ViewGroupParameter.allTickets(),
   new ViewOrderParameter())


def allTickets = ticketService.getCountForView(allCriteria)


ViewCriteria ownCriteria = new ViewCriteria(view,
   ViewAssignmentParameter.allTickets(engineerService.getCurrent()),
   ViewGroupParameter.onlyOwnTickets(),
   new ViewOrderParameter())

def ownTickets = ticketService.getCountForView(ownCriteria)

ViewCriteria unassignedCriteria = new ViewCriteria(view,
   ViewAssignmentParameter.allUnassignedTickets(),
   ViewGroupParameter.onlyUnassignedTickets(),
   new ViewOrderParameter())

def unassignedTickets = ticketService.getCountForView(unassignedCriteria)


def data = []

data.add("{name: _('all'), data:[${allTickets}]}" as String)
data.add("{name: _('own'), data:[${ownTickets}]}"as String)
data.add("{name: _('unassigned'), data:[${unassignedTickets}]}"as String)


return [series: "[${data.join(',')}]" as String,
   visible: 'true',
   chart: "{type: 'column'}", title: "{text: '${viewName}'}" as String,
   tooltip:"{headerFormat:''}" ,
   localization:"de: {all:'Alle',own:'Eigene',unassigned:'Unzugewiesene'},"+ "en:
    {all:'All', own:'Own', unassigned: 'Unassigned'}"];
```

Code example 5: *Standard ConSol CM chart widget script ticketsInViewDataWidget.groovy*

The following chart is defined by the script above. For a detailed explanation, please refer to the section Example of a Chart Widget.

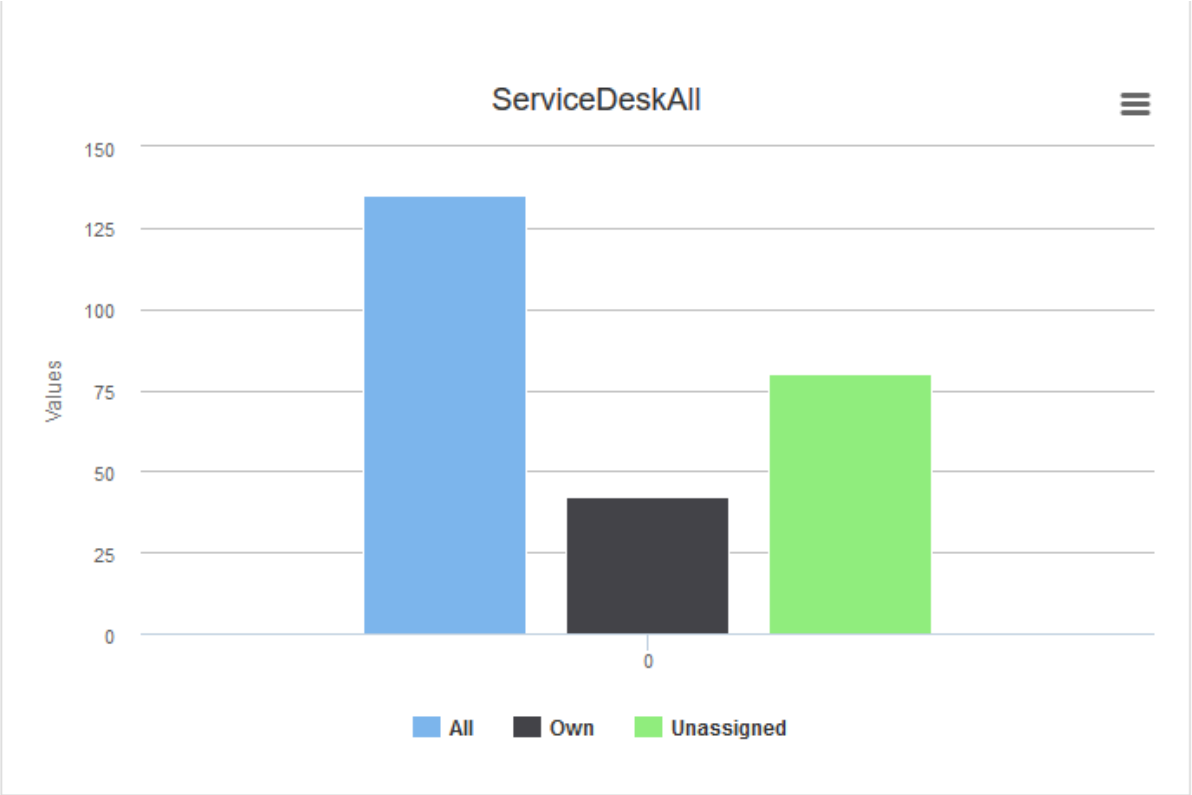


Figure 211: *ConSol CM Web Client - Example chart widget*

C.8.6.4 Configuration Attributes for Widgets

All definitions which can be used for a widget can be set using the attributes and values of the page customization. There are three types of attributes:

- general attributes (available for each widget type)
- attributes for chart widgets
- attributes for table widgets

Please keep in mind that an attribute which is set within the Admin Tool script of a widget always overwrites the respective attribute which has been set as an attribute!

Example: For the chart widget `ticketsInView`, the attribute *visibility* has been set to "true". The Admin Tool script which is associated with the widget (`ticketsInViewDataWidget.groovy`) contains the statement:

```
return [visible: 'false']
```

In this case the widget will not be displayed!

General Attributes

Those attributes are valid for all widgets and can be set in two locations of the page customization tree:

- widgetsGrid
- widgetsGrid/welcomePage

Attributes:

- **layout**
  See General Attributes.
- **refreshOnViewChange**
  Indicates whether the Dashboard should be refreshed when the engineer changes the view, i.e., uses the drop-down list *View* to select another view for the ticket list, default value is “true”
- **visible**
  Defines if the widget is displayed, “true” or “false”.

Examples for Visibility Configuration

Example 1: The chart should not be displayed.

```
return [visible: 'false']
```

Code example 6: *Visibility switched off (set in Admin Tool script)*

Example 2: The chart should only be displayed if the selected view is *service_customer*and the engineer has the *consultant* role.

```
view = viewService.getById(view_id) // view_id is passed in context
if (!view.getName().equals("service_customer"))
{
   return {"visible": false}
}

def role = roleService.getById('consultant');
def engineer = engineerService.getById(engineer_id);
if(!getRolesForEngineer(engineer).contains(role))
{
   return {"visible": false}
}
```

Code example 7: *Visibility depending on engineer role (set within Admin Tool script)*

Localization of String Values of Attributes

- **localization**

  Localized values, i.e., "de: {subject:'Thema', yes:'Ja'}, en: {subject:'Subject', yes:'Yes'}".

The *localization* attribute can provide the localized values for all string attributes used in the widget script.

For example (see example with graphical user interface below), you need values for the attributes *title* and *footer*, than you can set three attributes:

- attribute: *title*, value _('titlestring')
- attribute *footer*, value _('footerstring')
- attribute *localization* which provides localized values for both. Value de: {titlestring:'Offene Tickets: ',footerstring:'Anzahl bearbeitbare Tickets'}, en: {titlestring:'Open tickets: ', footerstring:'Number of accessible tickets'}

You can set values for attributes in the Admin Tool script, as shown in the following script for the three strings *all*, *own*, and *unassigned*.

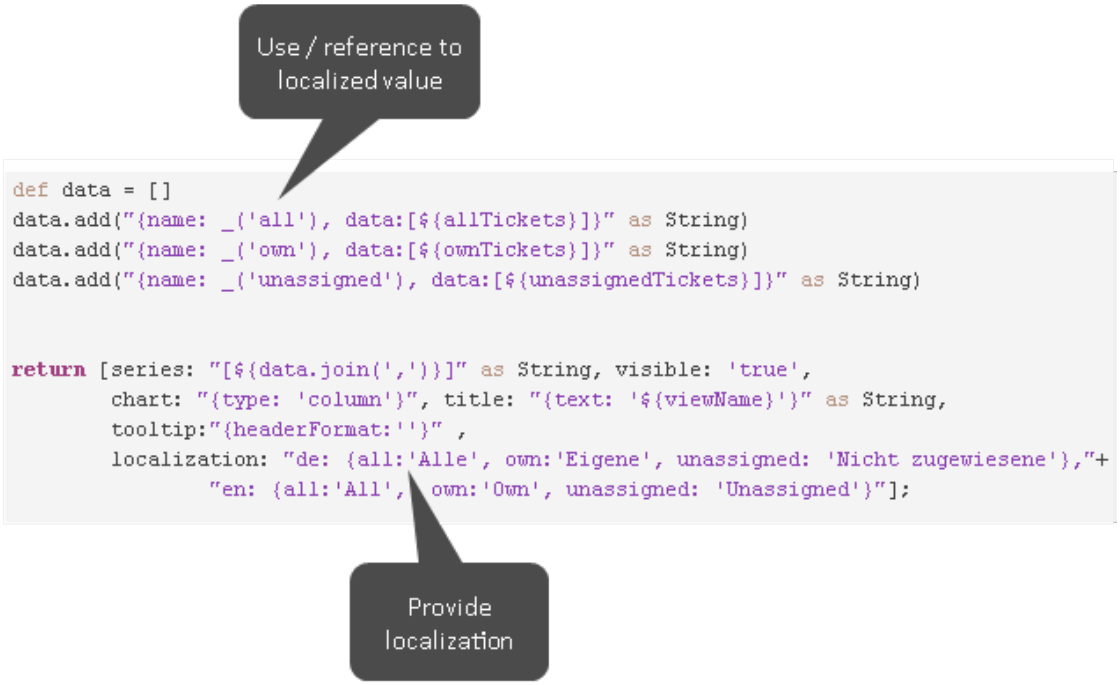


Figure 212: *Localizing page customization attributes in an Admin Tool script*

Please note that it does not matter if you provide the values for the attributes using a script or by entering the values using the graphical user interface for the page customization! You might also use the following way of localizing values (an example of a KPI widget):

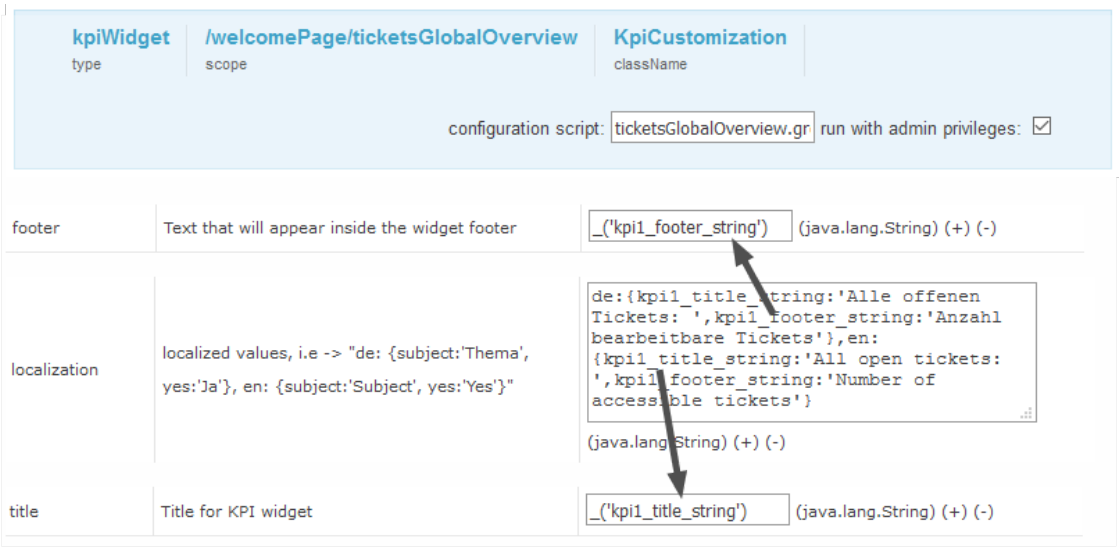


Figure 213: *Localizing string page customization attributes using the graphical user interface*

Attributes for Chart Widgets

Chart widgets use the Highcharts library. All attributes are JSON objects.

The attributes which can be set comprise:

- General attributes like visibility.
- The basic configuration options of the Highcharts library. Their values ...
  - can be set using the page customization attributes.
  - can be set using the Admin Tool script which is associated with the chart widget, see section above. The attributes have to be returned as a HashMap.
  - can be left empty.

Figure 214: *Highcharts configuration options*

General attributes:

- **developmentMode**
  Enables the development mode for the widget. If this attribute is set to “true”, the widget is displayed with a red border and a JSLint validation is performed on the widget script. If there are errors in the widget script, the error messages are displayed instead of the widget.
- **localization**
  Allows to set localized (i.e., country-specific) values for the strings used as values in attributes. Please see section Localization of String Values of Attributes for a detailed explanation.
- **visible**
  Defines if the widget is displayed, “true” or “false”.

Highchart-specific attributes:

- **chart**
  Options regarding the chart area and plot area as well as general chart options (http://api.highcharts.com/highcharts#chart). Example:

```
chart = "type:'column', pltShadow:false, backgroundColor:'#4dc245', height:
300";"items: [{html:'sometext', style: { left: '100px'; }}]"
```

Code example 8: *Chart object*

- **colors**
  An array containing the default colors for the chart's series. When all colors are used, new colors are pulled from the start again. Defaults to: http://api.highcharts.com/highcharts#colors
- **credits**
  Highchart, by default, puts a credits label in the lower right corner of the chart. This can be changed using these options: http://api.highcharts.com/highcharts#credits
- **drilldown**
  Options for drill down, the concept of inspecting increasingly high resolution data through clicking on chart items like columns or pie slices (http://api.highcharts.com/highcharts#drilldown).
- **exporting**
  Options for the exporting module (http://api.highcharts.com/highcharts#exporting).
- **global**
  Global options that don't apply to each chart (http://api.highcharts.com/highcharts#global). Can only be set for a type, i.e., for *chartWidget* or *tableWidget*, not for scopes or single widgets!
- **labels**
  HTML labels that can be positioned anywhere in the chart area (http://api.highcharts.com/highcharts#labels).

```
labels = "items: [{html:'sometext', style: { left: '100px'; }}]"
```

  Code example 9: *Labels object*

- **lang**
  Language object. The language object is global and it can't be set on each chart initiation (http://api.highcharts.com/highcharts#lang). Can only be set for a type, i.e., for *chartWidget* or *tableWidget*, not for scopes or single widgets!
- **legend**
  The legend is a box containing a symbol and name for each series item or point item in the chart (http://api.highcharts.com/highcharts#legend).
- **loading**
  The loading options control the appearance of the loading screen that covers the plot area on chart operations (http://api.highcharts.com/highcharts#loading)
- **localization**
  Localized values, i.e., “de: {subject:'Thema', yes:'Ja'}, en: {subject:'Subject', yes:'Yes'}”.
- **navigation**
  A collection of options for buttons and menus appearing in the exporting module (http://api.highcharts.com/highcharts#navigation).
- **noData**
  Options for displaying a message like "No data to display" (http://api.highcharts.com/highcharts#noData).

- **pane**
  Applies only to polar charts and angular gauges. This configuration object holds general options for the combined X and Y axes set (http://api.highcharts.com/highcharts#pane).
- **plotOptions**
  The plotOptions is a wrapper object for config objects for each series type (http://api.highcharts.com/highcharts#plotOptions).
- **series**
  The actual series to append to the chart (http://api.highcharts.com/highcharts#series).
- **subtitle**
  The chart's subtitle (http://api.highcharts.com/highcharts#subtitle).
- **title**
  The chart's main title (http://api.highcharts.com/highcharts#title).
- **tooltip**
  Options for the tool tip that appears when the user's mouse hovers over a series or point (http://api.highcharts.com/highcharts#tooltip).
- **visible**
  Indicates whether the widget is shown.
- **xAxis**
  The X axis or category axis (http://api.highcharts.com/highcharts#xAxis).
- **yAxis**
  The Y axis or value axis (http://api.highcharts.com/highcharts#yAxis).

Example of a Chart Widget

The following example shows the widget *TicketsInView* and explains the logic of the associated Admin Tool script `ticketsInViewDataWidget.groovy`. For the entire script, please see the code block above (Configuration Script for Widgets). Here, the lines of code are set in relation to the GUI elements which they configure.

Figure 215: *Chart widget example with script code*

Funnel Charts

With charts of type *funnel*, funnel-like representations can be implemented as often used in sales funnels.

A funnel chart is implemented in the same way as other charts, only the sub-attribute *type* of the chart attribute has to be set to “funnel”. The following example shows a sales funnel chart implementation where fixed numbers are used in the script. Of course, in a real life implementation, these figures will have to be replaced by figures which are created dynamically from the current content of the database.

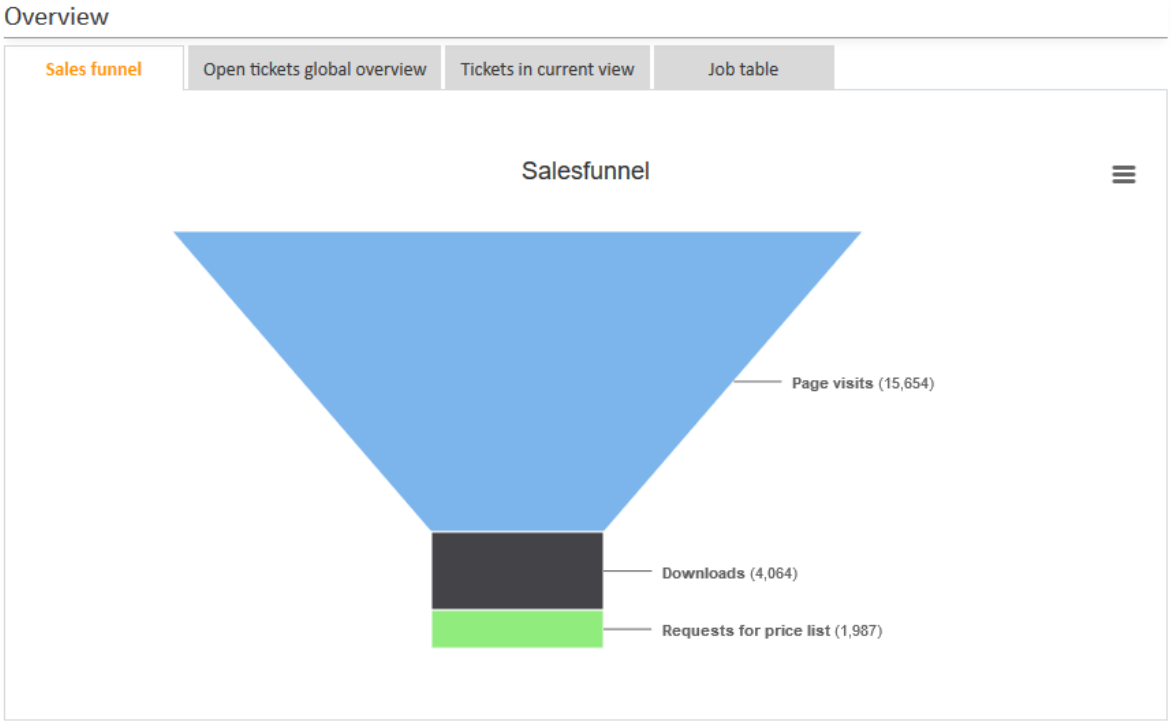


Figure 216: *ConSol CM Web Client - Sales funnel widget*

The following script is used:

```
return [title: "{text: _('title')}",
   chart: "{type: 'funnel', \
      marginLeft: '50', \
      marginRight: '150'}",
   plotOptions: "{series:{ height: '90%', \
      width: '85%', \
      neckWidth: '20%', \
      neckHeight: '20%', \
      dataLabels: {enabled:'true', \
      format: '<b>{point.name}</b> ({point.y:,.0f})', \
      softConnector: 'true'}}}",
   visible: "true",
   series: "[{name: _('users'), \
      data: [[_('visits') , 15654], \
      [_('downloads'), 4064], \
      [_('requests') , 1987]]}]",
   localization: "de: {title: 'Vertriebstrichter', \
      users: 'Individuelle Benutzer', \
      visits: 'Seitenaufrufe', \
      downloads: 'Downloads', \
      requests: 'Anfrage Preisliste'}, \
   en: {title: 'Salesfunnel', \
      users: 'Unique users', \
      visits: 'Page visits', \
      downloads: 'Downloads', \
      requests: 'Requests for price list'}"
]
```

Code example 10: *Groovy script for the implementation of a sales funnel chart on the Web Client Dashboard (with fixed numbers as example)*

Attributes for Table Widgets

Table widgets use the Datatables library.

The attributes can be set in the page customization or they can be set in the associated Admin Tool script. Please keep in mind that the script parameters always overwrite the respective attributes.

Attributes comprise:

- General attributes
- Datatables-specific attributes

General attributes:

- **developmentMode**
  Enables the development mode for the widget. If this attribute is set to “true”, the widget is displayed with a red border and a JSLint validation is performed on the widget script. If there are errors in the widget script, the error messages are displayed instead of the widget.
- **localization**
  Localized values, i.e., “de: {subject:'Thema', yes:'Ja'}, en: {subject:'Subject', yes:'Yes'}”.

- **visible**
  Defines whether the widget is displayed, “true” or “false”.

Datatables-specific attributes:

- **columns**
  Options which you can apply to the columns objects (http://data-tables.net/reference/option/#Columns).
- **data**
  Data (http://datatables.net/reference/option/#Data)
- **localization**
  Localized values, i.e., “de: {subject:'Thema', yes:'Ja'}, en: {subject:'Subject', yes:'Yes'}”. Please see section Localization of String Values of Attributes for a detailed explanation.
- **options**
  Options (http://datatables.net/reference/option/)
- **visible**
  Indicates whether the widget is shown.

Example of a Table Widget

The following demonstrates the basic principle of the implementation of a table widget based on the Datatables library.

```
// provide some dummy data for display
def rawdata = [
   [firstname:'Homer' , lastname:'Simpson' , title:'Nuclear disaster' , level:'3' ,
    hired:'25.03.1989'],
   [firstname:'Zaphod' , lastname:'Beeblebrox' , title:'President of the Galaxy',
    level:'0' , hired:'12.09.1979'],
   [firstname:'Sheldon' , lastname:'Cooper' , title:'Mad scientist' , level:'321',
    hired:'01.04.2006'],
   [firstname:'Robin' , lastname:'Scherbatsky', title:'Anchorwoman' , level:'25' ,
    hired:'10.09.2004'],
   [firstname:'Elmer' , lastname:'Fudd' , title:'Duck hunter' , level:'1' ,
    hired:'15.12.1962'],
   [firstname:'Eric' , lastname:'Cartman' , title:'Pupil' , level:'10' ,
    hired:'23.02.1995'],
   [firstname:'Mickey' , lastname:'Mouse' , title:'Private investigator' ,
    level:'111', hired:'04.11.1932'],
   [firstname:'Wilma' , lastname:'Flintstone' , title:'Housewife' , level:'64' ,
    hired:'07.01.1964'],
   [firstname:'Charlie' , lastname:'Harper' , title:'Composer' , level:'12' ,
    hired:'16.07.2001'],
   [firstname:'Daenerys', lastname:'Targaryen' , title:'Mother of dragons' ,
    level:'238', hired:'08.05.2010'],
   [firstname:'Lara' , lastname:'Croft' , title:'Tomb Raider' , level:'239',
    hired:'10.12.1991'],
   [firstname:'Henry' , lastname:'Jones' , title:'Archeologist' , level:'109',
    hired:'08.06.1942']
]
```

```
// prepare the data for display
def tabledata = []

rawdata.each { element ->
   tabledata.add("""
      {'firstname': '${element['firstname']}',
      'lastname' : '${element['lastname']}' ,
      'jobtitle' : '${element['title']}' ,
      'expertise': '${element['level']}' ,
      'hiredate' : '${element['hired']}' }
   """)
}

// return the table information including the data
return [
   "columns": """[
      {title: 'First name' , data: 'firstname'},
      {title: 'Last name' , data: 'lastname' },
      {title: 'Job title' , data: 'jobtitle' },
      {title: 'Expertise level', data: 'expertise'},
      {title: 'Hire date' , data: 'hiredate' }
      ]""",
   "options": """{
      'order': []
      }""",
   "data": "[${tabledata.join(",")}]" as String
]
```

Code example 11: *Admin Tool script for a table widget*

In the Web Client, the table is displayed as follows (all other widgets have been set to invisible).

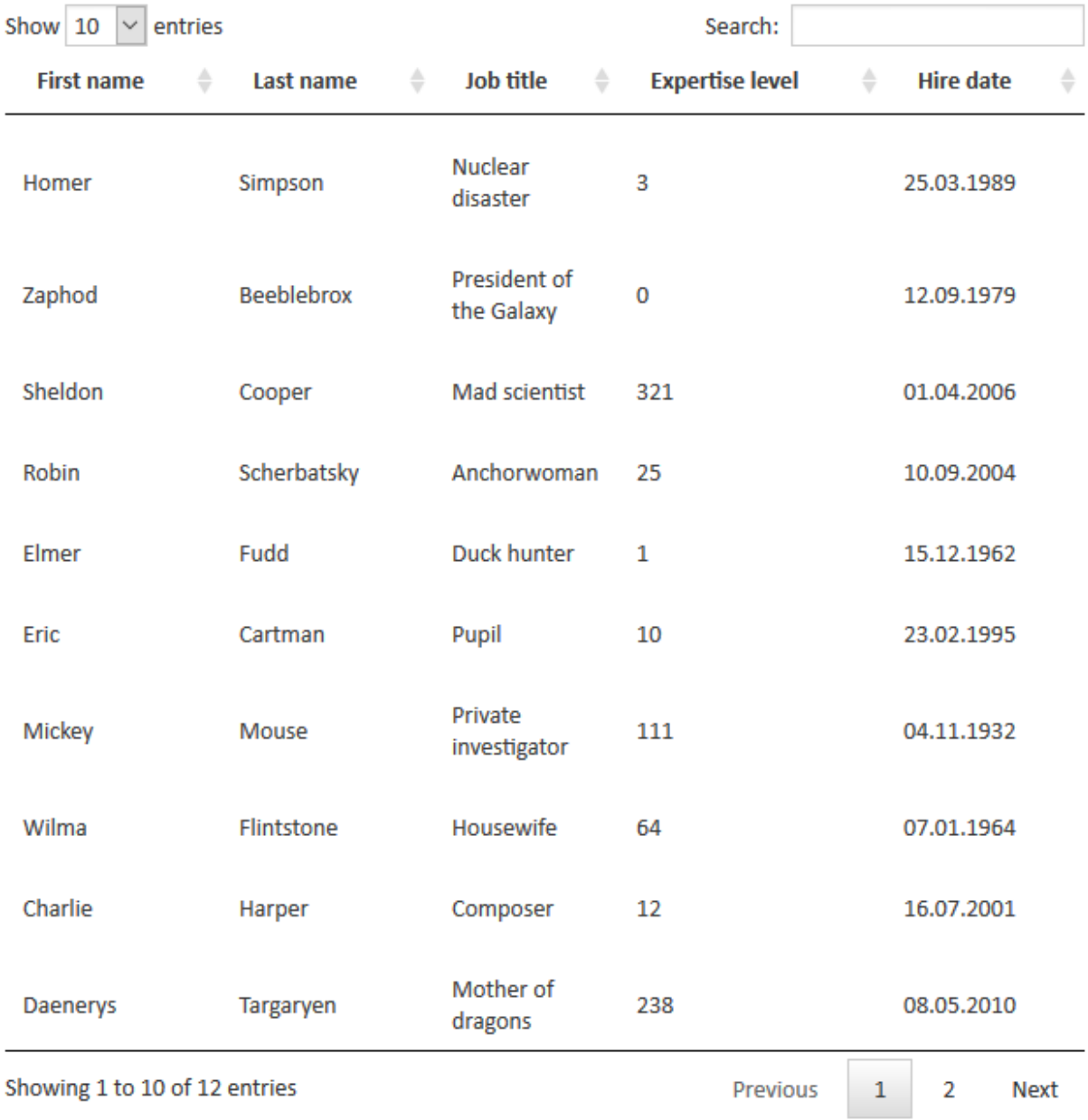


Figure 217: *ConSol CM Web Client - Table widget on the Web Client Dashboard*

Attributes for KPI Widgets

KPI widgets can make use of the following attributes:

- **color**
  Background color for the rectangle widget area, for example “#A0B0C0”.
- **footer**
  String to show at the bottom of the widget (shown in the left example as *Number of accessible tickets*), can be localized, see section Localization of String Values of Attributes for details.

- **height**
  Widget rectangle height in pixels (integer value).
- **localization**
  Localization definition for string values displayed, see section Localization of String Values of Attributes for details.
- **maxValueForSize**
  The numeric widget value above which the font size will be reduced in order to fit the value in the line inside the rectangle, default is “100000”.
- **symbol**
  Character symbol or string to show as prefix for the numeric value.
- **title**
  String to show in the top line of the widget (shown in the left example as *Open tickets*), can be localized, see section Localization of String Values of Attributes for details.
- **trend**
  Identifier for the trending symbol to be shown after the numeric value (see the left example), valid identifiers are “up”, “down”, and “flat”.
- **value**
  The numeric value to be shown, normally it is not wanted that this is a fixed value, but it should be dynamically resolved. This, however, can only be achieved by a script, see below for details. Just entering a number here without using a script will always display this number only.
- **visible**
  A boolean value determining, if the widget should be rendered or not, identical to other widget types.

The following code provides an example of a KPI widget script:

```
import com.consol.cmas.common.model.ticket.*
import java.util.*
import com.consol.cmas.common.model.DateRange
TicketCriteria crt = new TicketCriteria()
def to_date = new Date()
def from_date = to_date - 7
def range = new DateRange(from_date,to_date)
crt.setCreationDateRange(range)
// crt.setStatus(TicketCriteria.Status.OPEN)
ticketcount = ticketService.getByCriteria(crt).size()
switch (ticketcount) {
   case 0..25:
      trendline = 'down'
      break
   case 26..50:
      trendline = 'flat'
      break
   default:
      trendline = 'up'
}
return[value: ticketcount as String, trend: trendline, visible: 'true']
```

Code example 12: *Admin Tool script for a KPI widget which calculates the tickets which have been opened during the last week*

The widget will be displayed in the Web Client as shown in the following figure.

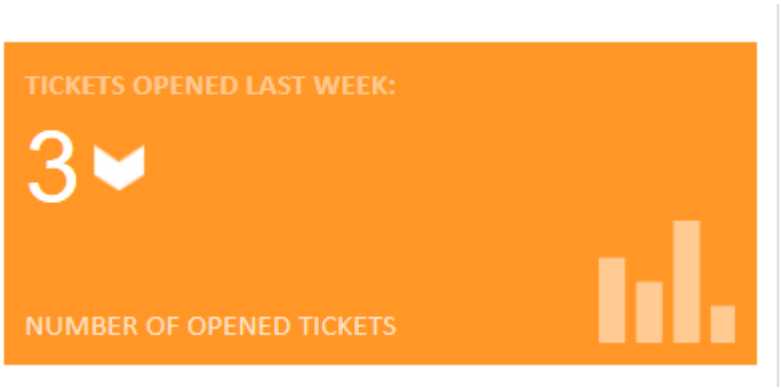


Figure 218: *ConSol CM Web Client - KPI widget on Dashboard page*

Attributes and Settings for the recentlyVisitedWidget

Page Customization Attributes

- **excludedCustomerGroups**: String. A comma-separated list of customer groups identified by their technical names. The named customer groups and their members will be excluded from the listing in the widget.

- **excludedItemTypes**: String. A comma-separated list of base object types which will be excluded. The types are identified by the following keywords:
  - TICKET: All tickets will be excluded from the listing in the widget.
  - UNIT: All customers (companies and contacts) will be excluded from the listing in the widget.
  - RESOURCE: All resources will be excluded from the listing in the widget.
- **excludedQueues**: String. A comma-separated list of queues identified by their technical names. The named queues and their tickets will be excluded from the listing in the widget.
- **excludedResources**: String. A comma-separated list of resource types identified by their technical names. The named resource types and their resources will be excluded from the listing in the widget.
- **localization**: Currently not used.
- **maxItems**: Integer. the maximum number of items shown in the widget main view.
- **title**: String. The title string shown in the header bar, localization of this label is not available yet.
- **visible**: Boolean. The widget visibility, must be set to “true” to have the widget on the welcome page, setting it to “false” hides the widget and all its contents.

Further Settings Using CM System Properties

The CM system behavior concerning data gathering for the `recentlyVisitedWidget` can be configured using CM system properties.

Switch Data Gathering On/Off

Depending on the use of the `recentlyVisitedWidget`, the data gathering which is the basis for the widget's display can be switched on or off. If the widget is not displayed, it does not make sense to have the data gathering thread running in the background using system resources for nothing. In this case, you can switch off the data gathering process using the following CM system property:

- cmas-core-server, recent.items.persistence.enabled

Do not forget to switch it on (again), if the widget should be displayed!

Configure Data Gathering Behavior

If the data gathering has been switched on, i.e., if `cmas-core-server`, `recent.items.persistence.enabled` is set to “true”, the system behavior regarding data gathering can be configured using the following CM system properties:

- recent.items.max.per.engineer
- recent.items.cleanup.interval.minutes
- recent.items.cleanup.cluster.node.id

Attributes for the recentChangesWidget

Page Customization Attributes

The list of the "excludedHistoryTypes" also shows which actions qualify as `recent change`.

- **excludedHistoryTypes**: String. The history entry types which should be excluded from the listing as a change. The value is entered as a comma-separated list of the following entry type identificators:
  - "OPEN": Ticket has been created.
  - "REOPEN": Ticket has been re-opened.
  - "UPDATE": Ticket has been updated.
  - "SUBJECT_CHANGE": Ticket subject has been updated.
  - "ENGINEER_CHANGE": Engineer associated has been changed.
  - "TICKET_USER_ADDED": An additional engineer has been added.
  - "TICKET_USER_REMOVED": An additional engineer has been removed.
  - "TICKET_USER_MOVED": An additional engineer function has been changed from one function to another.
  - "QUEUE_CHANGE": Ticket's queue has been changed.
  - "ACTIVITY_CHANGE": Ticket status has changed.
  - "TRANSFERED_TO_ACTIVITY": Ticket has been transferred to another activity.
  - "CONTENT_ADD": New ticket content has been added.
  - "ATTACHMENT_DELETE": Ticket attachment has been deleted.
  - "ATTACHMENT_UNDO_DELETION": Ticket attachment deletion has been made undone.
  - "EMAIL_ATTACHMENT_DELETE": An attachment was deleted from an email.
  - "EMAIL_ATTACHMENT_UNDO_DELETION": An email attachment deletion has been made undone.
  - "BOOKING": A time booking has been added.
  - "CONTACT_ADD": A customer has been added to the ticket.
  - "CONTACT_REMOVE": A customer has been removed from the ticket.
  - "CONTACTROLE_SWAP": The customer role in the ticket has been changed.
  - "CONTACTROLE_UPDATE": The customer role in the ticket has been updated.
  - "TRANSFERRED_TO_CONTACT": The ticket has been transferred to a new contact.
  - "TICKETRELATION_CREATE": A ticket relation has been created.
  - "TICKETRELATION_REMOVE": A ticket relation has been removed.
  - "TICKETRELATION_UPDATE": A ticket relation has been updated.
  - "TICKETRELATION_MOVE": A ticket relation has been changed from one relation type to

another

- “RESOURCE_RELATION_CREATED”: A resource relation has been created.
- “RESOURCE_RELATION_UPDATED”: A resource relation has been updated.
- “RESOURCE_RELATION_REMOVED”: A resource relation has been removed.
- “RESOURCE_ON_THE_FLY_RELATION_CREATED”: An on-the-fly resource relation has been created.
- “RESOURCE_ON_THE_FLY_RELATION_UPDATED”: An on-the-fly resource relation has been updated.
- “RESOURCE_ON_THE_FLY_RELATION_REMOVED”: An on-the-fly resource relation has been removed.

- **excludedQueues**: String. A comma-separated list of queues identified by their technical names. The named queues and their tickets will be excluded from the listing in the widget.
- **localization**: Currently not used.
- **maxItems**: Integer. The maximum number of items shown in the widget's main view.
- **title**: String. The title string shown in the header bar, localization of this label is not available yet.
- **visible**: Boolean. Widget visibility, must be set to “true” to have the widget on the welcome page, setting it to “false” hides the widget and all its contents.

Example of a Composed Dashboard

The following two figures provide an example of a composed dashboard.

Figure 219: *ConSol CM Web Client - Example of composed dashboard, top part*

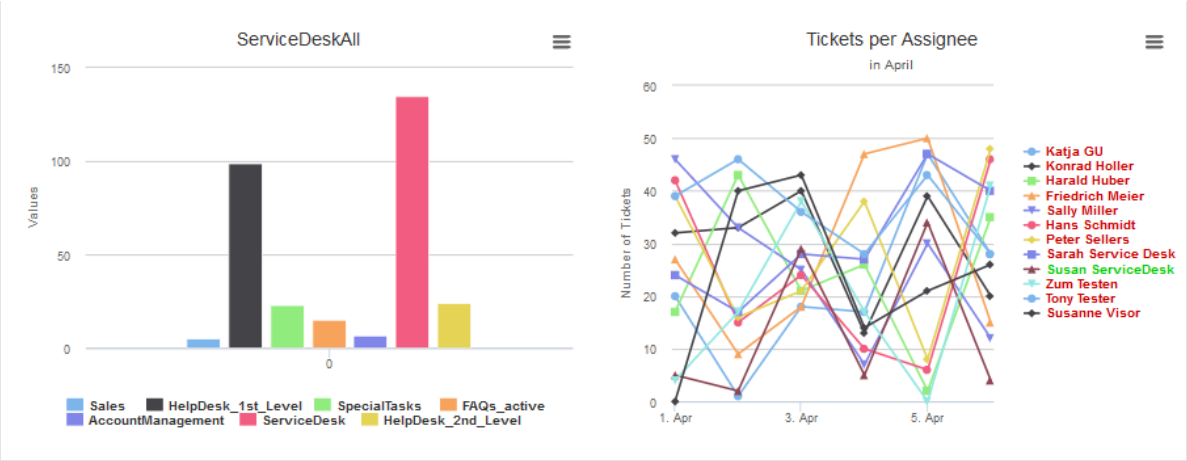


Figure 220: *ConSol CM Web Client - Example of composed dashboard, bottom part*

The page customization attribute *widgetsGrid / welcomePage / layout* is set as follows:

```
[tabName:'Sales funnel',i18n:{en:'Sales funnel',de:'Vertriebstrichter'},widgets:
 [SalesFunnel:chart]],[tabName:'Open tickets global overview',i18n:{en:'Open
 tickets global overview',de:'Übersicht offene Tickets'},widgets:
 [ticketsGlobalOverview:KPI,ticketsOpenedLastWeek:KPI]],[tabName:'Tickets in
 current view',i18n:{en:'Tickets in current view',de:'Tickets in der aktuellen
 Sicht'},widgets:[ticketsInView:Chart]],[tabName:'Job table',i18n:{en:'Job
 table',de:'Tabelle Jobs'},widgets:[[ticketsOverview:Table,pieChartExample:Chart],
 [stackedBarColumnExample:Chart,stackedBarColumnExample:Chart],
 [chartWidgetExample:Chart,lineChartExample:Chart]]]
```

Code example 13: *Layout attribute for composed dashboard*

Please note that the page customization tree is always shown for the active tab!

In the following example, this means, only the Job table subtree is displayed. The *Sales funnel*, *Open tickets global overview*, and *Tickets in current view* subtrees are not displayed.

The page customization attributes for the tab *Job table* look like this:

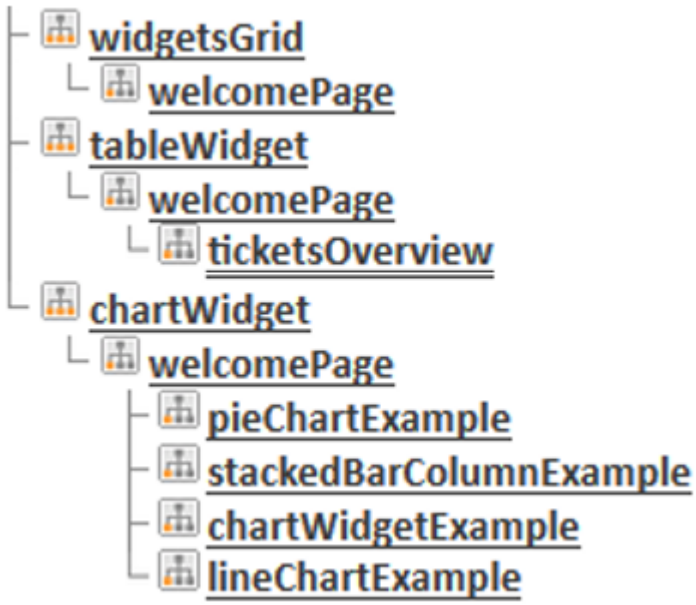


Figure 221: *ConSol CM Web Client - Example page customization attributes subtree*

All widgets have a an associated Admin Tool script each. The following figure shows the configuration of the tableWidget `ticketsOverview`.

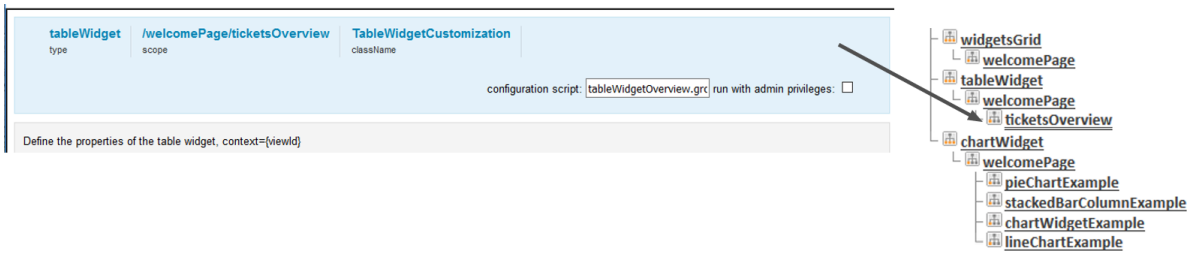


Figure 222: *ConSol CM Web Client - Configuration of example table widget*

The Admin Tool script, named `tableWidgetOverview.groovy`, for the table widget (named `ticketsOverview`):

```
// provide some demo data for display
def rawdata = [
[firstname:'Homer' , lastname:'Simpson' , title:'Nuclear disaster' , level:'3' ,
 hired:'25.03.1989'],
```

```
[firstname:'Zaphod' , lastname:'Beeblebrox' , title:'President of the Galaxy',
 level:'0' , hired:'12.09.1979'],
[firstname:'Sheldon' , lastname:'Cooper' , title:'Mad scientist' , level:'321',
 hired:'01.04.2006'],
[firstname:'Robin' , lastname:'Scherbatsky', title:'Anchorwoman' , level:'25' ,
 hired:'10.09.2004'],
[firstname:'Elmer' , lastname:'Fudd' , title:'Duck hunter' , level:'1' ,
 hired:'15.12.1962'],
[firstname:'Eric' , lastname:'Cartman' , title:'Pupil' , level:'10' ,
 hired:'23.02.1995'],
[firstname:'Mickey' , lastname:'Mouse' , title:'Private investigator' ,
 level:'111', hired:'04.11.1932'],
[firstname:'Wilma' , lastname:'Flintstone' , title:'Housewife' , level:'64' ,
 hired:'07.01.1964'],
[firstname:'Charlie' , lastname:'Harper' , title:'Composer' , level:'12' ,
 hired:'16.07.2001'],
[firstname:'Daenerys', lastname:'Targaryen' , title:'Mother of dragons' ,
 level:'238', hired:'08.05.2010'],
[firstname:'Lara' , lastname:'Croft' , title:'Tomb Raider' , level:'239',
 hired:'10.12.1991'],
[firstname:'Henry' , lastname:'Jones' , title:'Archeologist' , level:'109',
 hired:'08.06.1942']
]

// prepare the data for display
def tabledata = []

rawdata.each { element ->
   tabledata.add("""
      {'firstname': '${element['firstname']}',
      'lastname' : '${element['lastname']}' ,
      'jobtitle' : '${element['title']}' ,
      'expertise': '${element['level']}' ,
      'hiredate' : '${element['hired']}' }
      """)
   }


// return the table information including the data
return [
"columns": """[
   {title: 'First name' , data: 'firstname'},
   {title: 'Last name' , data: 'lastname' },
   {title: 'Job title' , data: 'jobtitle' },
   {title: 'Expertise level', data: 'expertise'},
   {title: 'Hire date' , data: 'hiredate' }
   ]""",
"options": """{
   'order': []
   }""",
"data": "[${tabledata.join(",")}]" as String
]
```

Code example 14: *Admin Tool script associated with table widget*

The first chart widget, a pie chart named `pieChartExample`, also has an associated Admin Tool script (named `pieChartExample.groovy`):

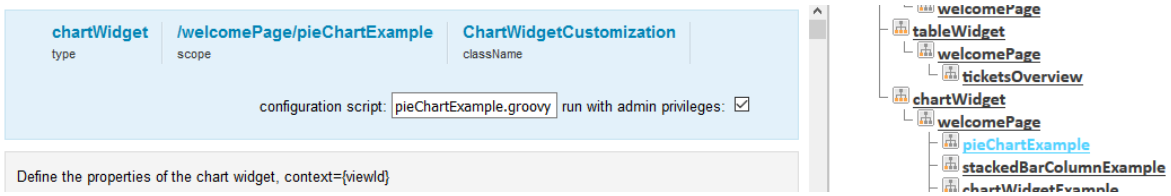


Figure 223: *ConSol CM Web Client - Configuration of example chart widget*

The Admin Tool script for the first chart widget (upper right, named `pieChartExample`):

```
def kpiGroups = ['Ok', 'Warning', 'Caution']
def escalationColor = [ ok: "#95E546", warning: "#E5E546", danger: "#E54646"]
def data = []
def name = [:]
def size = [:]
def percent = [:]
percent[kpiGroups[0]] = Math.round(Math.random() * 50)
percent[kpiGroups[1]] = Math.round(Math.random() * 50)
percent[kpiGroups[2]] = 100 - percent[kpiGroups[1]] - percent[kpiGroups[0]]
def prio = [:]
prio[kpiGroups[0]] = "39-A"
prio[kpiGroups[1]] = "40-B"
prio[kpiGroups[2]] = "41-C%2B"
for(def kpiGroup : kpiGroups){
   name[kpiGroup] = kpiGroup
   size[kpiGroup] = percent[kpiGroup] * 3
   sliced = kpiGroup.equals(kpiGroups[1]) ? ", sliced: true, selected: true" : ""
   def link = '/cm-client/search?c_queue=1'
   link += "&sales_standard:priority=" + prio[kpiGroup]
   link += "&c_status=OPEN"
   def dataSet = "{name: '${name[kpiGroup]} (${percent[kpiGroup]}%)', y: ${size
    [kpiGroup]} ${sliced}, link: '${link}'}" as String
   data.add(dataSet)
   //println dataSet
}
def chartData = [
   series: "[{name:'Escalation-Type', data: [${data.join(',')}]}]" as String,
   visible: 'true',
   chart: """{
   type: 'pie',
   }""",
   title: "{text: 'Escalation Status'}" as String,
   tooltip:"{headerFormat:'{point.name}', pointFormat: '{point.name}:
    {point.y}<br/>'}",
   plotOptions: """{
      pie: {
         allowPointSelect: true,
         cursor: 'pointer',
         dataLabels: {
```

```
        enabled: false
        },
      showInLegend: true
      },
      series: {
        cursor: 'pointer',
        point: {
          events: {
            click:
            function () {
              window.open(this.link,'_blank')
            }
          }
        }
      }
   }""" as String,
   colors: "['${escalationColor['ok']}', '${escalationColor['warning']}',
    '${escalationColor['danger']}']" as String
]
//println chartData
return chartData
```

Code example 15: *Admin Tool script associated with pie chart widget*

The same principle as shown for the first two widgets (which represent the first line in the dashboard tab) is implemented for the other widgets as well. The second line of the dashboard is filled completely by the chart `stackedBarColumnExample`, and in the third line, two chart widgets are placed: `chartWidgetExample` and `lineChartExample`.

For each chart, the chart-specific attributes can be defined, e.g., *title* or *subtitle*. These attributes can be defined in the script and included in the returned data structure, or they can be defined using the graphical interface. The attributes for chart widgets are explained in section Attributes for Chart Widgets, the attributes for table widgets are explained in section Attributes for Table Widgets.

C.8.6.5 Print Functionality in the Web Client Dashboard

Starting with version 6.9.4.2, ConSol CM offers print functionality for Chart Widgets in the Web Client Dashboard. The *Print* button opens the print dialog of the browser.

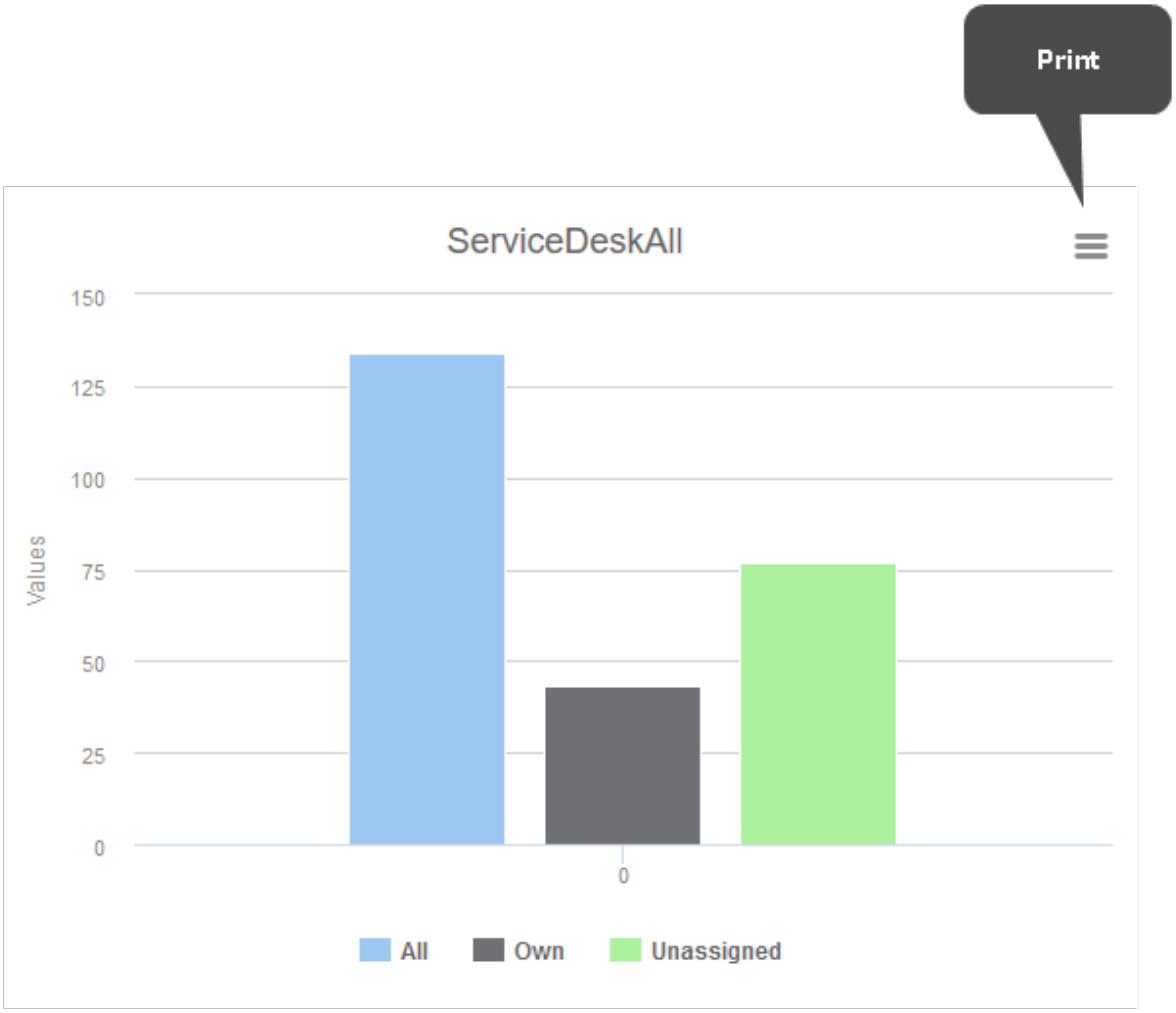


Figure 224: *ConSol CM Web Client - Print button in Web Client Dashboard*

In order to disable the print functionality, i.e., to hide the button, set the page customization attribute *exporting* to the value “enabled:false”.

C.8.6.6 3D Rendering for Graphics (Chart Widgets)

You can use 3D rendering for chart widgets. The following library is used: http://api.highcharts.com/highcharts#chart.options3d.

For more information on the 3D implementation and concepts of the solution see http://www.highcharts.com/docs/chart-concepts/3d-charts.

⚠ Usage of this kind of rendering puts a rather heavy performance load on the browser. Please be sure that the client environments are capable of displaying these 3D charts when implementing them!

An example script for showing the default chart in a 3D view is listed below. It sets the 3D options in the return value of the script.

```
import com.consol.cmas.common.model.ticket.*;
import com.consol.cmas.common.model.ticket.view.*;
import java.util.*;
import java.util.Map.Entry;


if (viewId == -1) {
   return [visible: 'false']
}
def engineerLocale = engineerService.getCurrentLocale()
def view = viewService.getById(viewId)
def viewName = localizationService.getLocalizedProperty(View.class, "name", viewId,
 engineerLocale)


ViewCriteria allCriteria = new ViewCriteria(view,
   ViewAssignmentParameter.allTickets(),
   ViewGroupParameter.allTickets(),
   new ViewOrderParameter())
def allTickets = ticketService.getIdsByView(allCriteria)


ViewCriteria ownCriteria = new ViewCriteria(view,
   ViewAssignmentParameter.allTickets(engineerService.getCurrent()),
   ViewGroupParameter.onlyOwnTickets(),
   new ViewOrderParameter())
def ownTickets = ticketService.getIdsByView(ownCriteria)


ViewCriteria unassignedCriteria = new ViewCriteria(view,
   ViewAssignmentParameter.allUnassignedTickets(),
   ViewGroupParameter.onlyUnassignedTickets(),
   new ViewOrderParameter())
def unassignedTickets = ticketService.getIdsByView(unassignedCriteria)


def data = []
data.add("{name: _('all'), data:[${allTickets.size()}]}" as String)
data.add("{name: _('own'), data:[${ownTickets.size()}]}" as String)
data.add("{name: _('unassigned'), data:[${unassignedTickets.size()}]}" as String)


return [series: "[${data.join(',')}]" as String,
   visible: 'true',
   chart: "{type: 'column',
      options3d: {enabled: 'true', alpha: '15', beta: '15', depth: '50',
         viewDistance: '25'}}",
      plotOptions: "{column: {depth: '25'}}",
      title: "{text: '${viewName}'}" as String,
      tooltip:"{headerFormat:''}" ,
```

```
      localization: "de: {all:'Alle', own:'Eigene', unassigned:'Nicht
       zugewiesene'},"
         + "en: {all:'All', own:'Own', unassigned: 'Unassigned'}"];
```

Code example 16: *Show the default chart in 3D view*

C.8.6.7 Drilldown Functionality for Graphics (Chart Widgets)

ConSol CM offers basic drilldown functionality for charts in the Dashboard. The page customization attribute *drilldown* can be used for general settings. The following library is used: http://api.highcharts.com/highcharts#drilldown.

However, to be reasonably used by this functionality, the data script should be extended, too. The data in the return script needs to be extended with the data shown in the drilldown. These additional data must be referenced with the data they extend. For a description of the concepts see the highcharts documentation: http://www.highcharts.com/docs/chart-concepts/drilldown.

The effect is that a second level of data can be shown in the same chart when clicking on a subset. In the left screenshot the columns are clickable and the column name labels are links. After clicking on either a column or a link, a detail view of this subset is shown, illustrated by the right screenshot.

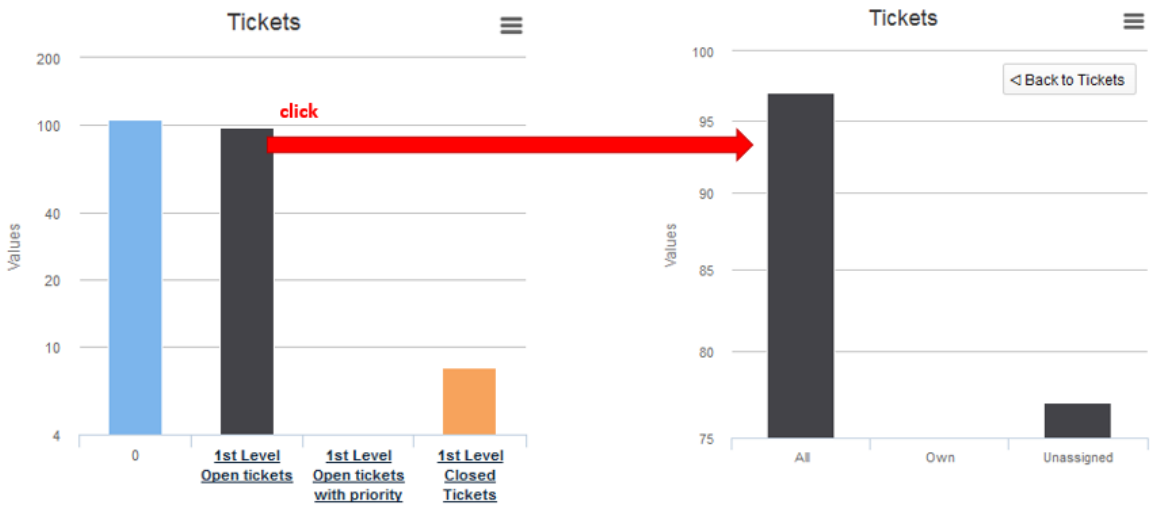


Figure 225: *ConSol CM Web Client - Drilldown functionality in chart widgets*

The color of the columns in the detail view is the color of the selected subset in the main view. The detail view also displays a back button on the upper right side of the chart. The following data script dynamically provides the data for this drilldown.

```
import com.consol.cmas.common.model.ticket.*;
import com.consol.cmas.common.model.ticket.view.*;
import java.util.*;
import java.util.Map.Entry;
```

```
if (viewId == -1) {
   return [visible: 'false']
}

def engineerLocale = engineerService.getCurrentLocale()

def views = []

def seriesdata = []
def drilldownseries = []
def allTicketsCounter = 0
views = viewService.getByEngineer(engineerService.getCurrent())

for (view in views){
   def viewName = localizationService.getLocalizedProperty(View.class,
      "name",
      view.getId(),
      engineerLocale)

   ViewCriteria allCriteria = new ViewCriteria(view,
      ViewAssignmentParameter.allTickets(),
      ViewGroupParameter.allTickets(),
      new ViewOrderParameter())
   def allTickets = ticketService.getIdsByView(allCriteria)
   ViewCriteria ownCriteria = new ViewCriteria(view,
      ViewAssignmentParameter.allTickets(engineerService.getCurrent()),
      ViewGroupParameter.onlyOwnTickets(),
      new ViewOrderParameter())
   def ownTickets = ticketService.getIdsByView(ownCriteria)

   ViewCriteria unassignedCriteria = new ViewCriteria(view,
      ViewAssignmentParameter.allUnassignedTickets(),
      ViewGroupParameter.onlyUnassignedTickets(),
      new ViewOrderParameter())
   def unassignedTickets = ticketService.getIdsByView(unassignedCriteria)

   seriesdata.add("{name: '${viewName}',
      y: ${allTickets.size()},
      drilldown: '${view.getName()}'}")

   def data = []
   data.add("['All', ${allTickets.size()}]")
   data.add("['Own', ${ownTickets.size()}]")
   data.add("['Unassigned', ${unassignedTickets.size()}]")

   drilldownseries.add("{id: '${view.getName()}',
      data:[${data.join(',')}]}" as String)

   allTicketsCounter += allTickets.size()
}

return [series: "[{name: 'Tickets', colorByPoint: true,
```

```
   data: [${allTicketsCounter}, ${seriesdata.join(',')}]}]" as String,
   drilldown: "{series: [${drilldownseries.join(',')}]}" as String,
   visible: 'true',
   chart: "{type: 'column'}",
   title: "{text: 'Tickets'}",
   xAxis: "{type: 'category'}",
   yAxis: "{type: 'linear'}",
   legend: "{enabled: false}"
];
```

Code example 17: *Drilldown functionality for chart widget*

C.8.7 Page Customization for the Graphical Representation of Relations

The representation of relations as graphs is based on page customization. Two modes are available:

The **standard mode** allows to configure the graph display for relation sections of tickets and on customer and resource pages. The graph display is then offered as link *Graph* and the engineer can switch between list and graph mode. For a detailed explanation, see section Page Customizations for the Visualization of Relations, Standard.

The **expert mode** allows to fade-in a new section on ticket, customer, or resource pages where an entirely custom-specific graph, based on a script, is displayed. For a detailed explanation, see section Page Customizations for the Visualization of Relations, Expert.

The **basic configuration principle** is identical in both modes and is explained in section Configuration of the Graph Display of Relations .

C.8.7.1 Page Customizations for the Visualization of Relations, Standard

Introduction

In all sections which represent relations (see section Configuration of the Graphical Display in Standard Mode), it is possible to configure the ConSol CM Web Client to offer - besides the list view - a graph view of the relations. This is done using page customization.

You can only switch on the graph view and use the default settings, this is a very simple step. Or you can go ahead and configure a more complex configuration.

The following example shows how to configure the display of relation graphs in a ticket page. Since CM/Resource Pool is active, the example comprises the configuration of ticket-resource relations as graph. The same principle as shown for a

- ticket page

applies to

- a company page
- a contact page
- a resource page (available if CM/Resource Pool is active)

Page Customization Example (Standard): contactSection in Ticket Page

Start with Page Customization of Contact Section in Ticket Page

In a ticket page, there are three sections where relations can be displayed.

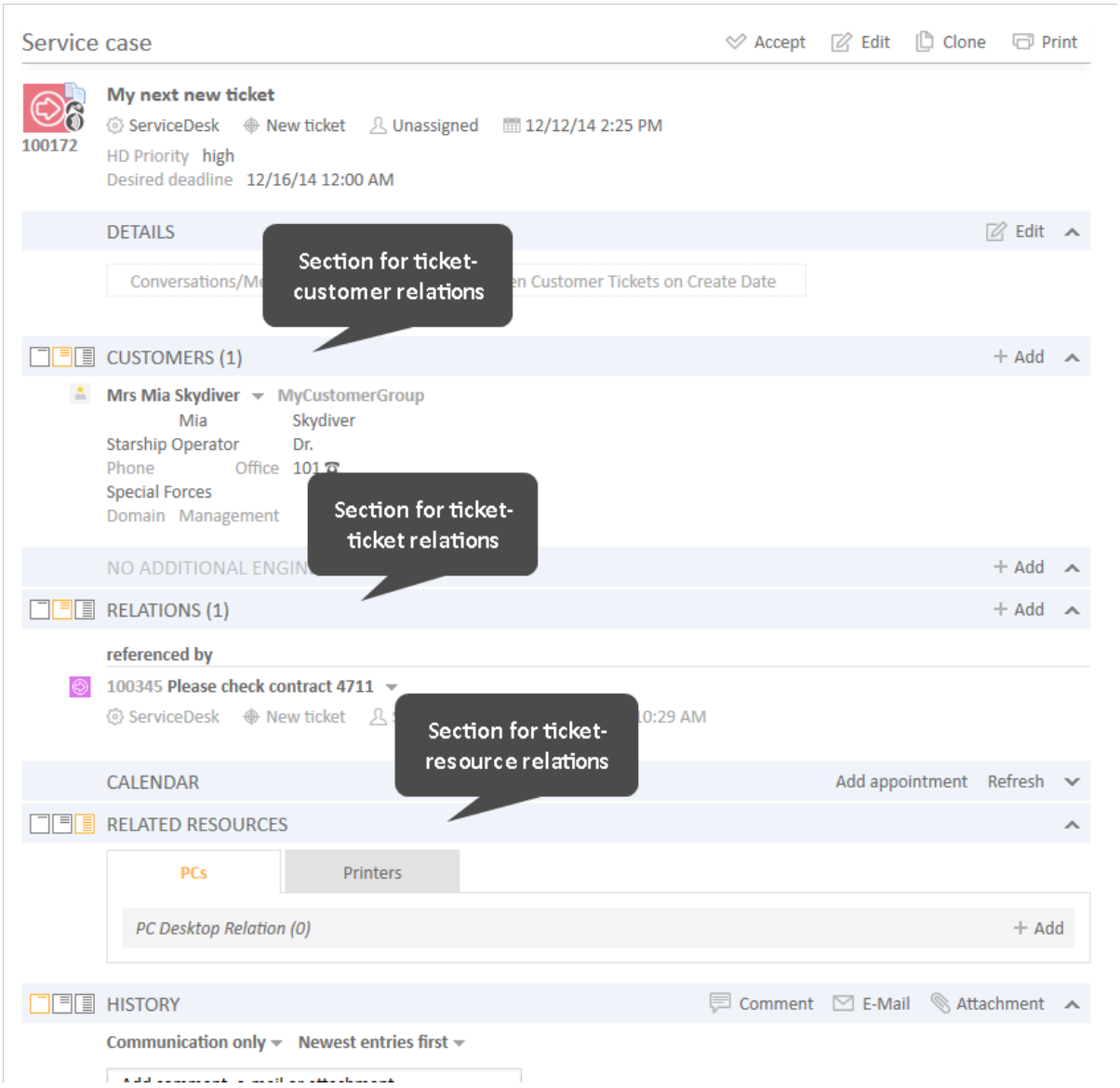


Figure 226: *ConSol CM Web Client - Ticket page: Sections where relations are displayed*

For each of the sections, the graph display can be configured separately. For example, start with the customer relations section. Log in as engineer with administrator privileges and *Enable page Customization* (main menu). For the customer relations sections several page customization scopes and subscopes are offered. Use the section-specific subscope to configure the graph. For the customer relations section, this is the *contactSection* subscope.

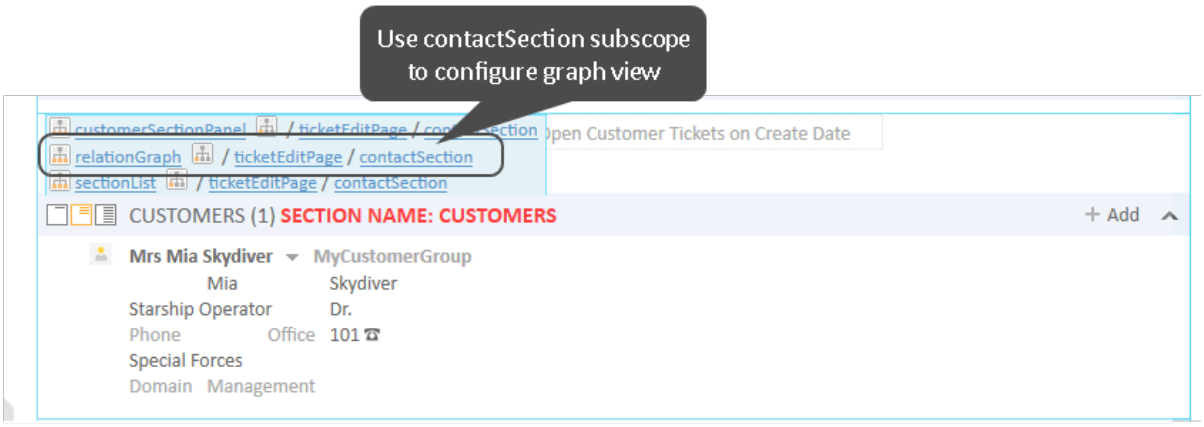


Figure 227: *ConSol CM Web Client - Selection of page customization scope for relationGraph in contactSection*

You can also select the subscope *contactSection* section in the Page Customization tree. In either case, the *relationGraph* configuration page for the contact section is opened (see following figure).

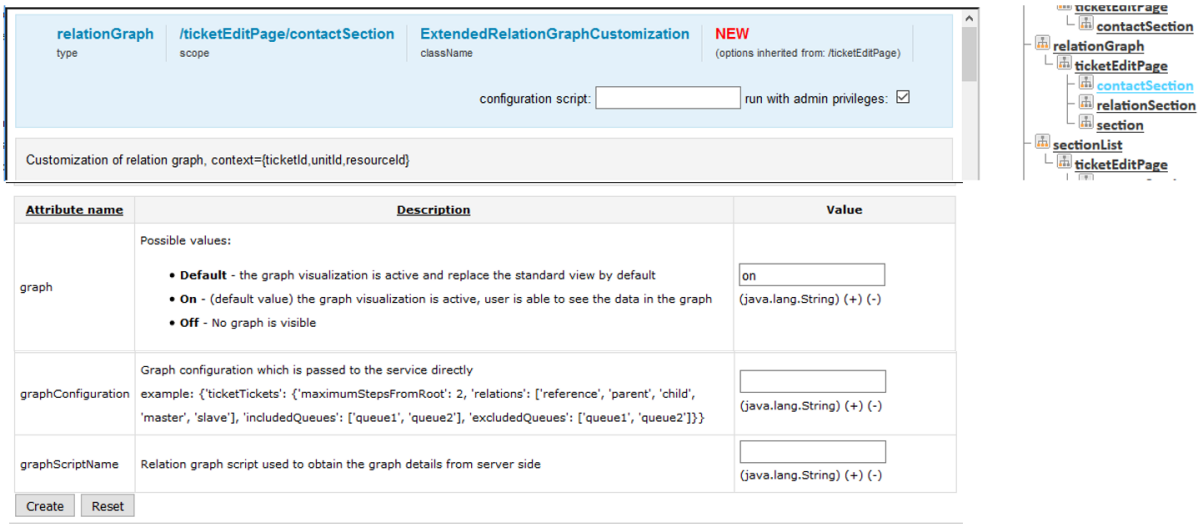


Figure 228: *Page customization of contact section in ticket edit page*

Attributes for Page Customization of Standard Relations Graph View

The following attributes are available:

- **graph** - String. Possible values:
  - Default - the graph visualization is active and replace the standard view by default
  - On - (default value in new installations) the graph visualization is active, user is able to see the data in the graph
  - Off - no graph is visible

- **graphConfiguration** - String, JSON statement

  With this JSON statement, you can configure the appearance of the graph. A default value is defined implicitly in the system, so when you just switch on the graph display, you do not have to define a *graphConfiguration*. But you can use it, if the default configuration does not cover the required display. For a detailed explanation, please see section Configuring the Graph Display of Relations Using JSON Statements.

- **graphScriptName** - String

  Refers to a script which is stored in the Admin Tool *Scripts and Templates* section. In the script, the relations graph can be built based on complex information. For example, it is possible to include data from other CM objects and/or from other systems which might provide CM-relevant data. The script has to be of script type *Relation graph* and always has to return an object of class `RelationGraph`. For a detailed explanation of this type of scripts, please refer to section Configuring the Graph Display of Relations Using Scripts.

Please provide either a *graphConfiguration* or a *graphScriptName*. In case, both are provided, the script is the one which will be used. The *graphConfiguration* is not taken into account in this case.

Simple Example Using *graphConfiguration*

The following example configuration is used for a first explanation of how the *graphConfiguration* attribute works.

In the *contactSection* (see figures above), the following values are set for the attributes:

- *graph*: “on”
- *graphConfiguration*:

```
{"ticketCustomers" :
 {"maximumStepsFromRoot" : 2,
  "relations":["main","default","end customer"],
  "includedCustomerGroups":["Reseller","MyCustomerGroup"]}
 }
```

This results in the following graph display:

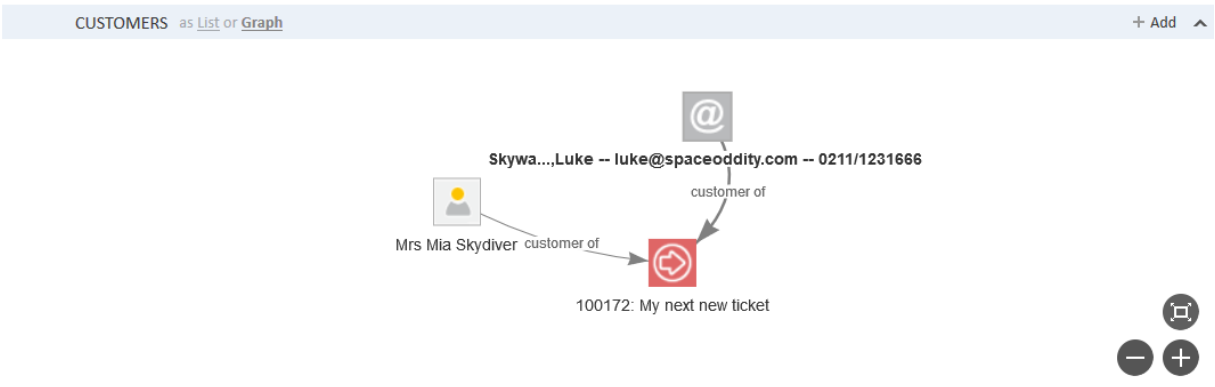


Figure 229: *ConSol CM Web Client - Ticket page: Graph display of customer relations to the ticket*

If you change the *graphConfiguration* as follows

```
{"ticketCustomers" :
 {"maximumStepsFromRoot" : 2,
  "relations":["main","default","end customer"],
  "includedCustomerGroups":["DirectCustomers","MyCustomerGroup"]}
 }
```

the following graph is displayed:

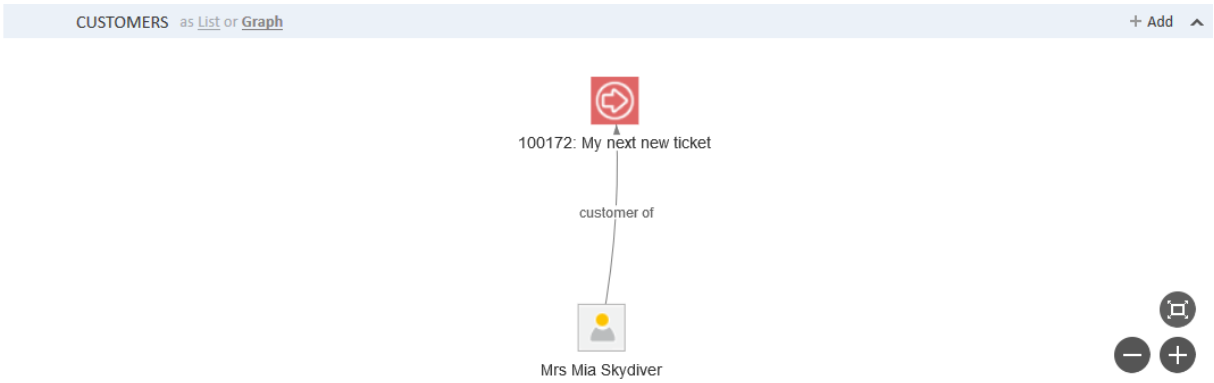


Figure 230: *ConSol CM Web Client - Ticket page: Graph display of customer relations to the ticket, variant*

The customer (contact) Luke Skywalker is in the customer group *Reseller*, and when the customer group *Reseller* has not been explicitly included, this customer is not displayed in the graph.

This simple example hopefully helps you understand the general principle of the *configurationGraph*. A detailed explanation of all parameters is provided in section Configuring the Graph Display of Relations Using JSON Statements.

Simple Example Using *graphScript*

A *graphScript* always has to return an object of class `RelationGraph`.

The following simple script will show only ticket-customer relations and could be used in the customer section of the ticket page. In this case, it would be easier to work with the *configurationGraph*, however, we want to show you how a script works ...

```
import com.consol.cmas.common.model.relation.*

//create criteria using java api
def relationCriteria = new RelationCriteria()
   .withTicketCustomers(new RelationCriteria.TicketCustomers()
   .withMaximumStepsFromRoot(2));
def ticket = ticketService.getById(ticketId)
def relationGraph = relationService.load(ticket, relationCriteria);

return relationGraph;
```

Code example 18: *Simple graphScript*

The following graph is displayed:

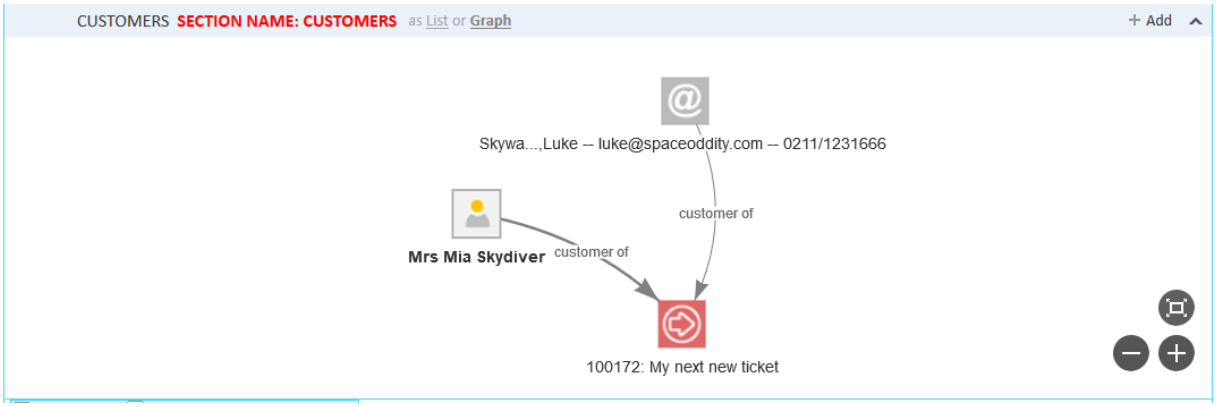


Figure 231: *ConSol CM Web Client - Ticket page: Graph display in the customers section based on the script shown above*

If the layout should be changed, the script could be modified as follows.

```
import com.consol.cmas.common.model.relation.*
//create criteria using java api
def relationCriteria = new RelationCriteria()
   .withTicketCustomers(new RelationCriteria.TicketCustomers()
   .withMaximumStepsFromRoot(2));
def ticket = ticketService.getById(ticketId)
def relationGraph = relationService.load(ticket, relationCriteria);
for (RelationNode node : relationGraph.getNodes()) {
   node.withProperty("font", [size:10,color:"blue"]) //change default property
      .withProperty("image", null) //remove default property
      .withProperty("shape", "diamond") //remove default property
   }
return relationGraph;
```

Code example 19: *Simple graphScript with simple node layout definition*

The following graph is displayed:

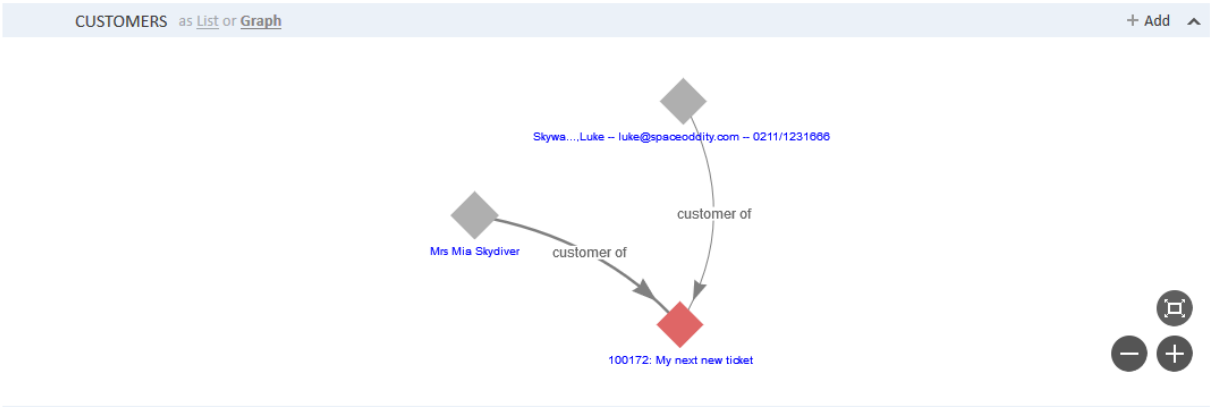


Figure 232: *ConSol CM Web Client - Ticket page: Slightly modified graph in customer relations section, based on the script shown above*

In the script shown above, only the appearance of the nodes has been changed. The appearance of the edges also can be modified. Furthermore, there are numerous configuration parameters which can be set. For a detailed explanation, please refer to section Configuring the Graph Display of Relations Using Scripts.

C.8.7.2 Page Customizations for the Visualization of Relations, Expert

Introduction

The Standard configuration of graph displays for relations (see section Page Customizations for the Visualization of Relations, Standard) offers the possibility to configure graph displays in relations sections of ticket, customer, and resource pages, i.e., it extends the functionality of sections which are already present in the respective pages and adds the option to have the graph view displayed.

In addition to that, a whole new section can be included in

- ticket pages
- contact pages
- company pages
- resource pages (if CM/Resource Pool is active)

This section is hidden in standard installations and can be used to display an individual relation graph which is completely system-specific. The graph is based on a script. There are no other possibilities to configure the content and appearance of the graph! It has to be done in an Admin Tool script.

Please note that that means:

You, as a script developer, are free to include everything which is required, since a script can make use of the entire ConSol CM Java/Groovy API.

You, as a script developer, are fully responsible for the content which is displayed.

The following example will show you how to implement a system-specific relations graph in the resource page. The same principle of cause applies to ticket, contact, and company pages.

Page Customization Example (Expert): New Section in Resource Page

The following steps have to be performed in order to implement a system-specific relations graph in the resource page:

- Step 1: Make the required section visible in the page
- Step 2: Configure the page customization attributes for the newly displayed section
- Step 3: Write the script
- Step 4: Check the result

Step 1: Make the required section visible in the page

The section which has to be displayed is named *customRelationGraphSection*. Since the section is not yet displayed, you cannot select it in page customization mode directly on the page, but you have to use the Page Customization tree.

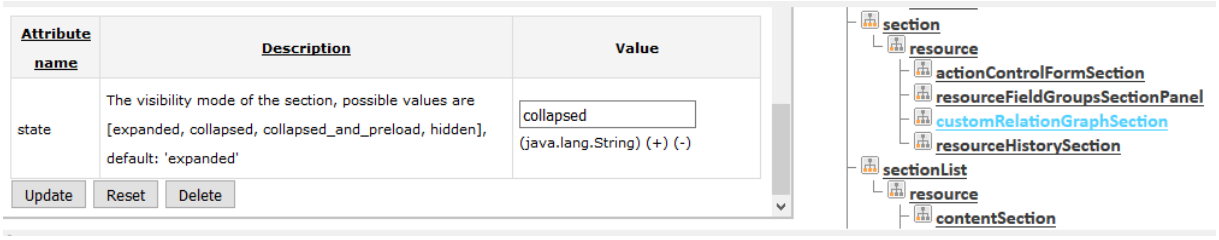


Figure 233: *One set of page customization attributes for the customRelationGraphSection*

Set the attribute *state* to either “collapsed” or “expanded”. Then the section will be available on the page, in the example on the resource page. The default name/header of the new section is *Custom relation graph*. The section is displayed above the history section. No graph is displayed yet - if you look at the section in the Web Client, an error will be displayed, because there is no script yet which defines the respective graph.

Step 2: Configure the page customization attributes for the newly displayed section

In order to configure the content and appearance of the graph, you first have to set the page customization attributes for the new section. Select the section either on the page or in the Page Customization tree.

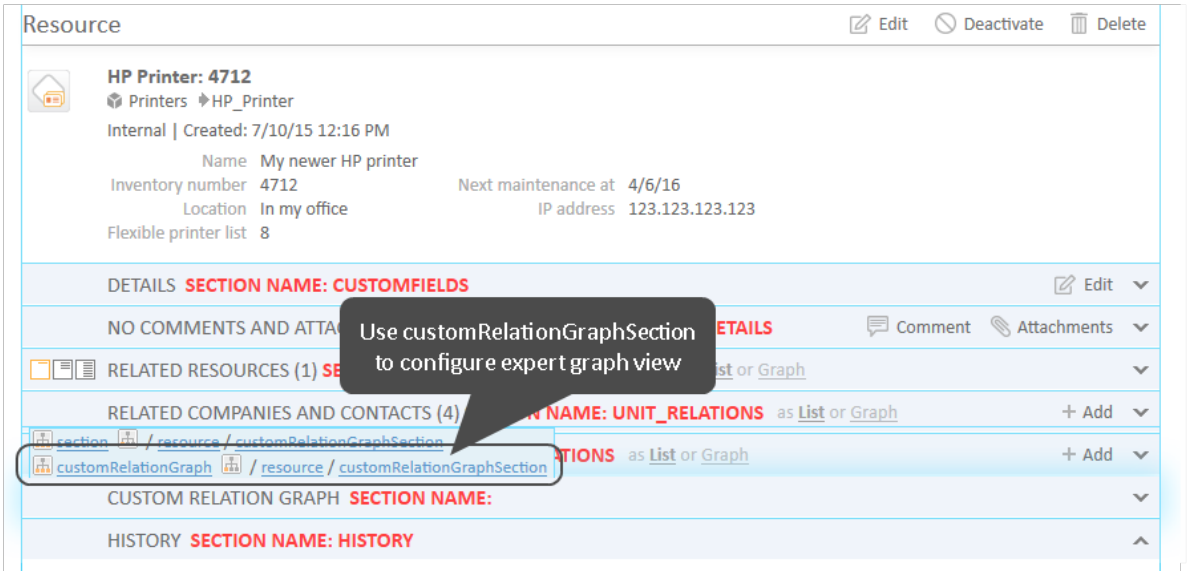


Figure 234: *ConSol CM Web Client - Selection of page customization scope for customRelationGraph in resource page*

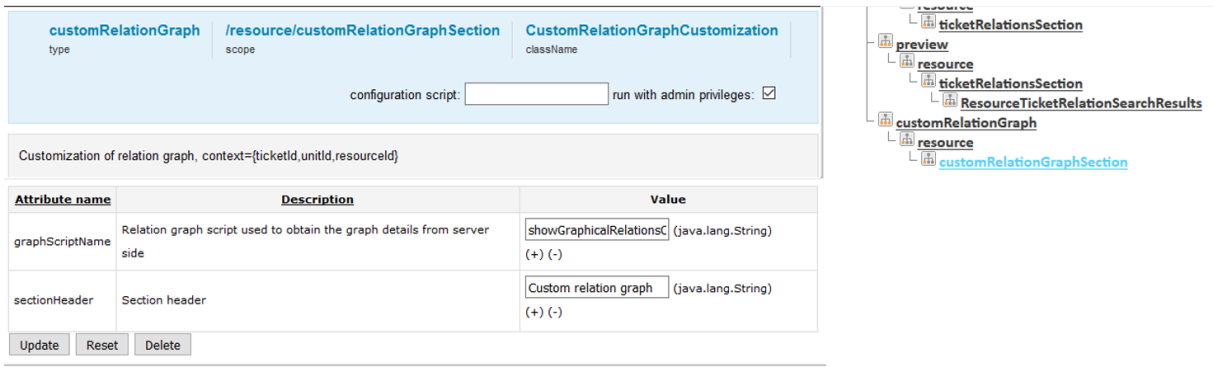


Figure 235: *Page customization tree and attributes for customRelationGraphSection*

Two attributes are available:

- **graphScriptName** - String. Refers to a script which is stored in the Admin Tool *Scripts and Templates* section. In the script, the relations graph can be built based on complex information. For example, it is possible to include data from other CM objects and/or from other systems which might provide CM-relevant data. The script has to be of script type *Relation graph* and always has to return an object of class `RelationGraph`. For a detailed explanation of this type of scripts, please refer to section Configuring the Graph Display of Relations Using Scripts.
- **sectionHeader** - String. Will be displayed in the Web Client as header of the section. Use any String.

Step 3: Write the script

The *graphScriptName* has to refer to a script of type *Relation graph* which is stored in the *Scripts and Templates* section of the Admin Tool. For a detailed explanation of this type of scripts, please refer to section Configuring the Graph Display of Relations Using Scripts.

In the current example, the following values were set for the attributes:

- sectionHeader = “This device within the entire infrastructure”
- graphScriptName = “showGraphicalRelationsOfResource2.groovy”

The following script is used:

```
import com.consol.cmas.common.model.relation.*
//create criteria using java api
def relationCriteria = new RelationCriteria()
   .withResourceTickets(new RelationCriteria.ResourceTickets()
   .withMaximumStepsFromRoot(2))
   .withTicketResources(new RelationCriteria.TicketResources()
   .withMaximumStepsFromRoot(2))
   .withResourceUnits(new RelationCriteria.ResourceUnits()
   .withMaximumStepsFromRoot(2))
   .withResourceResources(new RelationCriteria.ResourceResources()
   .withMaximumStepsFromRoot(2));


//variables available in script: ticketId, unitId, resourceId (depending on page)
def resource = resourceService.getById(resourceId)
def relationGraph = relationService.load(resource, relationCriteria);
//modify graph nodes, edges and visjs property
for (RelationNode node : relationGraph.getNodes()) {
   node.withProperty("font", [size:9,color:"black"]) //change default property
}
for (RelationEdge edge : relationGraph.getEdges()) {
   edge.withProperty("font", [size:12, color:"red"]); //change default property
}
relationGraph.withProperty("layout", [improvedLayout:true]) //change default
 property
return relationGraph;
```

Code example 20: *Script for customRelationGraph , showGraphicalRelationsOfResource2.groovy*

Step 4: Check the result

The following graph is displayed in the Web Client:

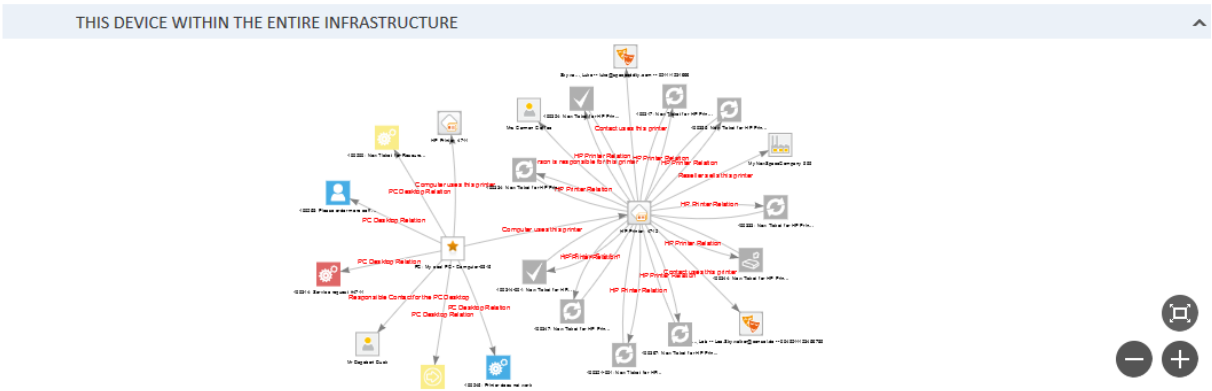


Figure 236: *ConSol CM Web Client - Display of expert relation graph in the resource page*

C.8.7.3 Configuration of the Graph Display of Relations

Introduction to Configuration of the Graph Display of Relations

The graph display of relations is based on vis.js, namely on the Network library. Refer to the vis.js Network detailed documentation to get to know all available configuration parameters. In the following sections, the most important parameters will be explained using some examples.

As explained in section Attributes for Page Customization of Standard Relations Graph View, either a JSON statement or a script can be used to define the layout of the graph.

Configuring the Graph Display of Relations Using JSON Statements

Introduction

There are two ways of providing a JSON statement for configuring a relation graph:

- as value of a page customization attribute:
  provide the statement as value of the page customization attribute `graphConfiguration`, see section Attributes for Page Customization of Standard Relations Graph View
- in a *graphScript*
  provide the statement as JSON string which is then used by the class.method `RelationCriteria.parseJson(JSONstring)`

The first case is the common way of configuring standard relation graphs. The second can be used in special cases, however, usually other methods would be better in this case.

> Please note that only the scope can be defined in a JSON statement! If you want to modify the graph layout (e.g., color of edges, font style), you have to work with scripts, see section Configuring the Graph Display of Relations Using Scripts.

Syntax for the Configuration of the Graph Display

In the JSON statement, the **scope** of the graph can be defined.

For **tickets**, you can define the following parameters:

- **For ticket-ticket relations:**

```
"ticketTickets" : {
 "maximumStepsFromRoot" : 2,
 "relations" : [ "reference", "parent", "child", "master", "slave" ],
 "includedQueues" : [ "queue1", "queue2" ],
 "excludedQueues" :[ "queue1", "queue2" ]
 }
```

Use either `includedQueues` or `excludedQueues`!

The following relation types are possible:

- “reference”
  ticket references the current ticket
- “parent”
  ticket is parent ticket of the current ticket
- “child”
  ticket is child ticket of the current ticket
- “master”
  ticket is master ticket of the current ticket
- “slave”
  ticket is slave ticket of the current ticket

- **For ticket-customer relations:**

```
"ticketCustomers" : {
 "maximumStepsFromRoot" : 2,
 "relations" : [ "main", "default", "contactTicketRoleName" ],
 "includedCustomerGroups" : [ "customerGroup1", "customerGroup2" ],
 "excludedCustomerGroups" : [ "customerGroup1", "customerGroup2" ]
 }
```

Use either `includedCustomerGroups` or `excludedCustomerGroups`!

The following relation types are possible:

- “main”
  customer is the main customer of the ticket
- “default”
  customer is an additional customer without role
- the name of a customer role

- **For ticket-resource relations:**

```
"ticketResources" : {
 "maximumStepsFromRoot" : 2,
 "relations" : [ "related", "resourceRelationDefinition" ],
 "includedResourceTypes" : [ "resourceType1", "resourceType2" ],
 "excludedResourceTypes" : [ "resourceType1", "resourceType2" ]
 }
```

For **customers = units (i.e., contacts and companies)**, you can define the following parameters:

- **For customer-ticket relations:**

```
"customerTickets" : {
 "maximumStepsFromRoot" : 2,
 "relations" : [ "main", "default", "contactTicketRoleName" ],
 "includedQueues" : [ "queue1", "queue2" ],
 "excludedQueues" : [ "queue1", "queue2" ]
 }
```

The following relation types are possible:

- "main"
  the customer is the main customer of the ticket
- "default"
  customer is an additional customer without role)
- the name of a customer role

- **For unit-unit relations:**

```
"unitUnits" : {
 "maximumStepsFromRoot" : 2,
 "relations" : [ "unitRelationDefinition1", "unitRelationDefinition2" ],
 "includedCustomerGroups" : [ "customerGroup1", "customerGroup2" ],
 "excludedCustomerGroups" : [ "customerGroup1", "customerGroup2" ]
 }
```

Use either `includedCustomerGroups` or `excludedCustomerGroups`!

For directional unit-unit relations, it is possible to specify the direction, i.e., whether the objects linked in the graph should be source or target of the relation. The syntax is as follows.

1. The linked customers are the source of the relation and the root node is the target:

```
"relations" : [ "unitRelationDefinition:source"]
```

2. The linked customers are the target of the relation and the root node is the source:

```
"relations" : ["unitRelationDefinition:target"]
```

- **For contact-company relations:**

```
"contactCompany" : {
 "maximumStepsFromRoot" : 2
 }
```

- **For company-contact relations:**

```
"companyContacts" : {
 "maximumStepsFromRoot" : 2
 }
```

- **For unit-resource relations:**

```
"unitResources" : {
 "maximumStepsFromRoot" : 2,
 "relations" : [ "related", "resourceRelationDefinition" ],
 "includedResourceTypes" : [ "resourceType1", "resourceType2" ],
 "excludedResourceTypes" : [ "resourceType1", "resourceType2" ]
 }
```

Use either `includedResourceTypes` or `excludedResourceTypes`!

For resources, you can define the following parameters:

- **For resource-resource relations:**

```
"resourceResources" : {
 "maximumStepsFromRoot" : 2,
 "relations" : [ "related", "resourceRelationDefinition" ],
 "includedResourceTypes" : [ "resourceType1","resourceType2" ],
 "excludedResourceTypes" : [ "resourceType1", "resourceType2" ]
 }
```

Use either `includedResourceTypes` or `excludedResourceTypes`!

For directional resource-resource relations, it is possible to specify the direction, i.e., whether the objects linked in the graph should be source or target of the relation. The syntax is as follows:

1. The linked resources are the source of the relation and the root node is the target:

```
"relations" : ["resourceRelationDefinition:source"]
```

2. The linked resources are the target of the relation and the root node is the source:

```
"relations" : ["resourceRelationDefinition:target"]
```

- **For resource-unit relations:**

```
resourceUnits" : {
 "maximumStepsFromRoot" : 2,
 "relations" : [ "related", "resourceRelationDefinition" ],
 "includedCustomerGroups" : [ "customerGroup1", "customerGroup2" ],
 "excludedCustomerGroups" : [ "customerGroup1", "customerGroup2" ]
 }
```

Use either `includedCustomerGroups` or `excludedCustomerGroups`!

Configuring the Graph Display of Relations Using Scripts

Scripts might be used in two locations to define graph displays of relations:

- in the standard display mode as value of the `graphScriptName` attribute
- in the customized expert display mode as value of the `graphScriptName` attribute

> Please note that in a script, every relation can be included into the display! So the person who implements the script is responsible for the graph display! Please make sure the correct relations are displayed, especially for scripts in the standard sections. For example, it might not make much sense to display ticket-resource relations in a customer section of a ticket, even if it is technically possible. If complex networks of relations of different types should be displayed, we recommend to place those in a customized / expert relations graph and set an informative header for this section!

A script which defines the graph display of relations

- has to be of type “Relation graph”
- has to return an object of class `RelationGraph`

- can define
  - the scope of the graph
  - the layout of the graph
- can make use of the following objects which are implicitly present in the script, depending on the location of the script
  - ticketId (in ticket pages)
  - unitId (in unit pages)
  - resourceId (in resource pages)

In order to define the **scope** of a relation graph in a script, use the class `RelationCriteria` and the static classes which represent the different flavors of `RelationCriteria`, e.g., `RelationCriteria.ResourceTickets` (see ConSol CM Java/Groovy API documentation!)

Example script:

```
import com.consol.cmas.common.model.relation.*
//create criteria using java api
def relationCriteria = new RelationCriteria()
   .withResourceTickets(new RelationCriteria.ResourceTickets()
   .withMaximumStepsFromRoot(2))
   .withResourceUnits(new RelationCriteria.ResourceUnits()
   .withMaximumStepsFromRoot(2))
   .withResourceResources(new RelationCriteria.ResourceResources()
   .withMaximumStepsFromRoot(2));
def resource = resourceService.getById(resourceId)
def relationGraph = relationService.load(resource, relationCriteria);
return relationGraph;
```

Display:

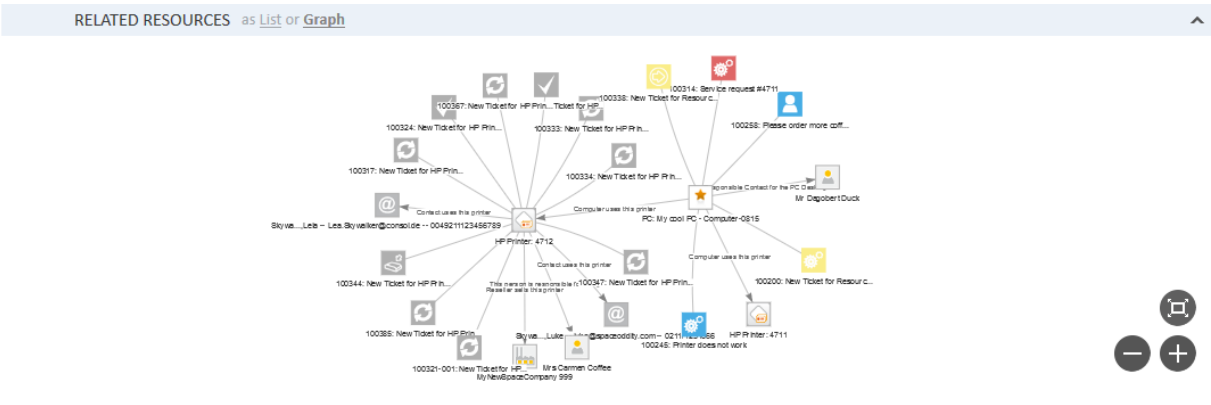


Figure 237: *ConSol CM Web Client - Resource page: Graph display*

In order to define the **layout** of the graph, use `RelationNode` and `RelationEdge` objects.

Example script (script as above, extended with layout configuration):

```
import com.consol.cmas.common.model.relation.*
//create criteria using java api
def relationCriteria = new RelationCriteria()
   .withResourceTickets(new RelationCriteria.ResourceTickets()
   .withMaximumStepsFromRoot(2))
   .withResourceUnits(new RelationCriteria.ResourceUnits()
   .withMaximumStepsFromRoot(2))
   .withResourceResources(new RelationCriteria.ResourceResources()
   .withMaximumStepsFromRoot(2));


def resource = resourceService.getById(resourceId)
def relationGraph = relationService.load(resource, relationCriteria);
//define layout:
for (RelationNode node : relationGraph.getNodes()) {
   node.withProperty("font", [size:12,color:"blue"]) //change default property
      .withProperty("image", null) //remove default property
      .withProperty("shape", "diamond") //remove default property
      .withProperty("margin", 5); //add property
}

for (RelationEdge edge : relationGraph.getEdges()) {
   edge.withProperty("arrows", null) //remove default property
      .withProperty("font", [size:16, color:"red"]); //change default property
}
relationGraph.withProperty("layout", [hierarchical:[direction:"UD"]])

return relationGraph;
```

Display:

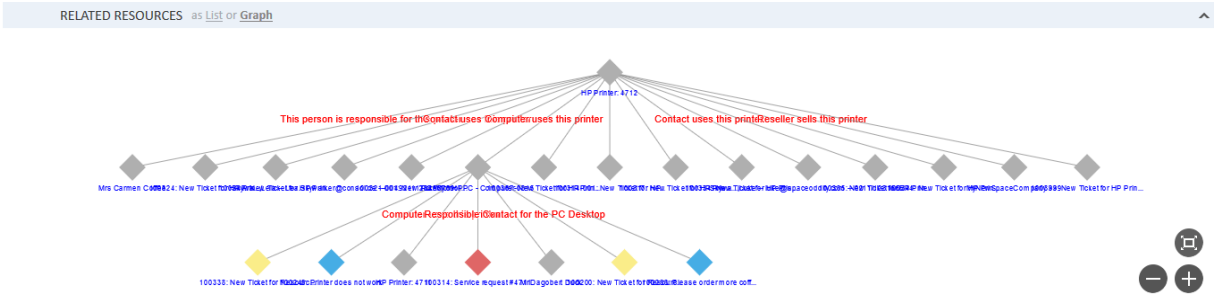


Figure 238: *ConSol CM Web Client - Resource page: Graph display with modified layout options*

The font size has been set to 12 pt, the font color has been changed to blue, the shape has been changed to diamond, and the layout has been changed to a hierarchical display.

This example gives an impression of how the layout of relation graphs can be modified. Use this principle (with the help of the vis.js Network documentation to adapt the relation graphs in your CM system to your company's requirements.

C.9 Design and Configuration of REST-based ConSol CM Client GUIs

C.9.1 Introduction

In order to provide a comfortable configuration of the user interface for REST clients, the Admin Tool in CM versions 6.10.7.0 and up (in 6.11, since version 6.11.0.5) contains the navigation group *Clients*.

It can be used to configure the user interface of

- CM/Track V2 (currently available)
- CM/Phone (will be available in upcoming versions)
- CM/Mobile (will be available in upcoming versions)
- customer-specific CM REST clients

Configuration in this context means:

- layout of the page
- localization for various languages

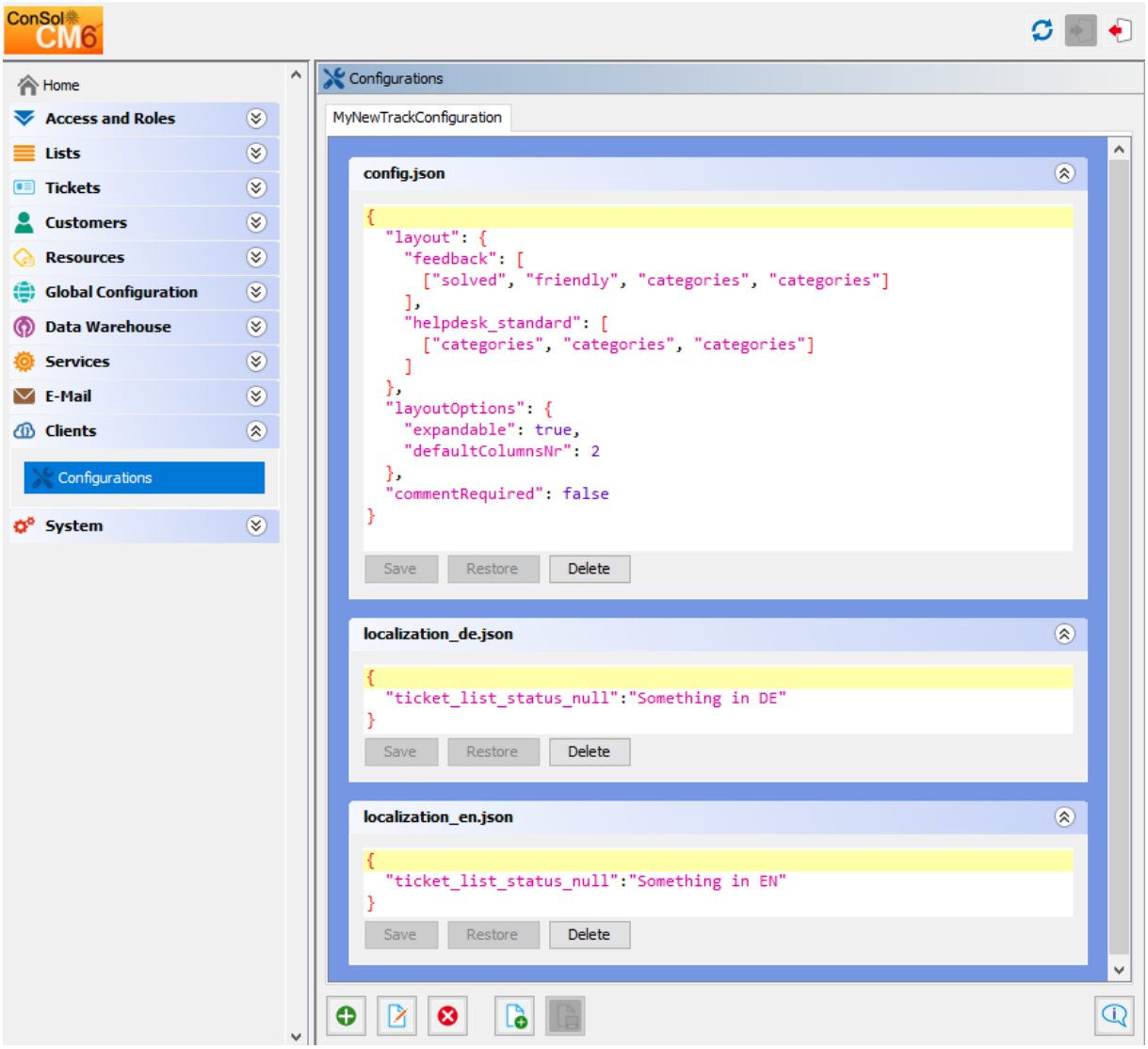


Figure 239: *ConSol CM Admin Tool - REST client configuration (example)*

The configuration panel features a tab for each existing configuration, labeled with the configuration name. A configuration consists of a number of files commonly in the JSON (JavaScript Object Notation) format, which will be provided to the client upon request. In the tab, the files are shown each one after another to be edited. On the bottom of the panel are buttons to organize the configurations. These can be used to achieve the tasks addressing a configuration as a whole:

- **Create**
  new configuration opens a dialog to name the configuration and choose a template to base it on.
- **Edit**
  configuration opens a dialog to rename the selected configuration and choose the template.
- **Delete**
  configuration removes the configuration as a whole and its tab will not be present anymore.

- **Create new file to the current configuration**
  opens dialog to set the file name and extends the configuration with an empty file with this name.
- **Save**
  all files saves all unsaved changes in all files in the currently selected configuration.
- **Examples**
  opens a window with examples from templates with each template showing in a tab. Currently there is only one template, named *track*, available for selection of the client type in the *Create* and *Edit* dialogs as well as in the *Examples* window. The following example shows a newly created configuration, based on the example track configuration.

Each file section offers file-oriented functions. The respective buttons are located below the file content editing section:

- **Save**
  stores the changes made to the file to deliver them to the client.
- **Restore**
  sets the file content back to the state after the last time it was saved.
- **Delete**
  removes the whole file from the configuration.

C.9.2 Configuration Principle

The configuration which is relevant for the client depends on the URL the client has called. The specific configuration for the client is delivered by the Angular app within ConSol CM.

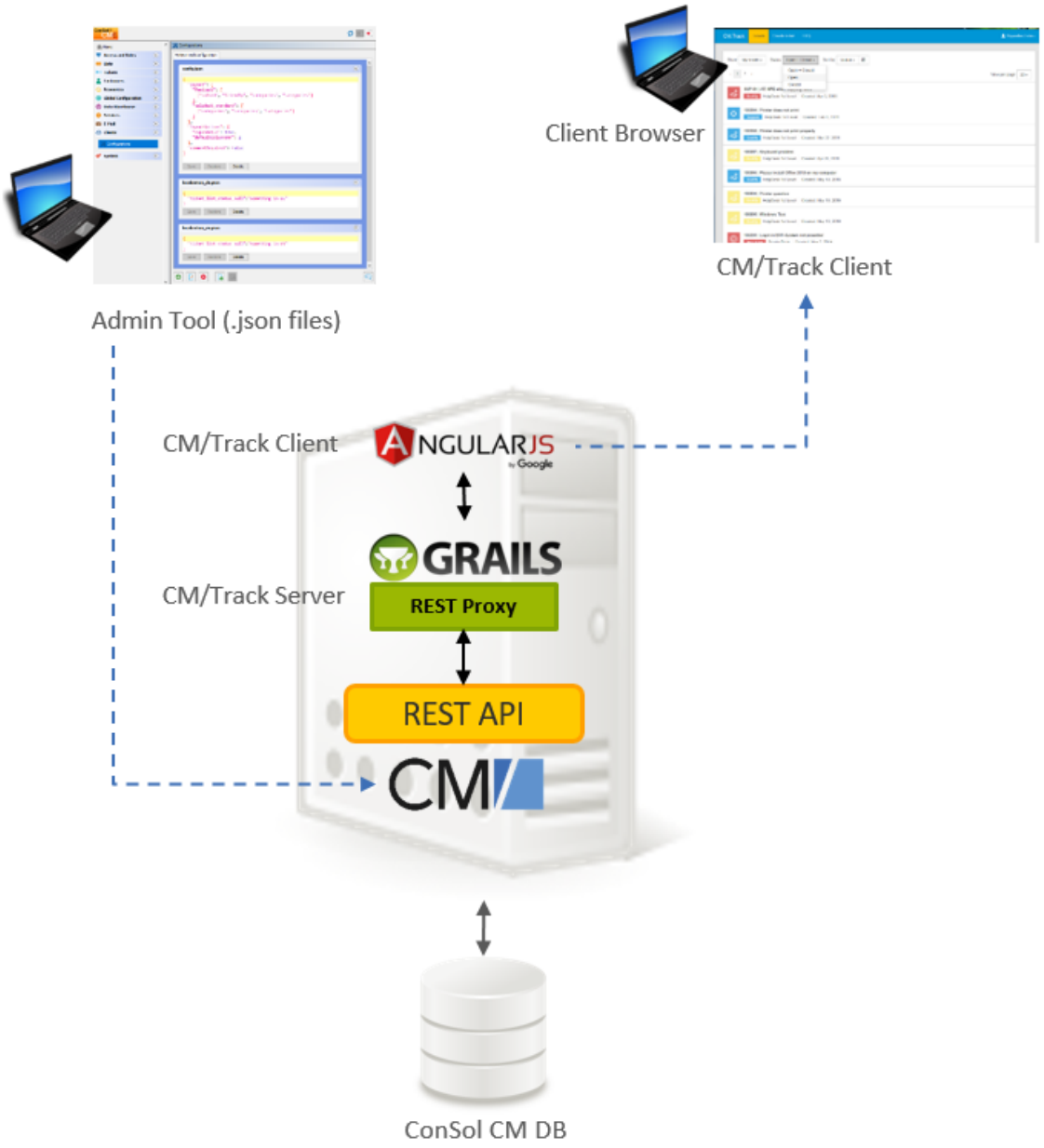


Figure 240: *Principle of REST Client GUI configuration with the Admin Tool*

The mapping between the specific domain and the required configuration is configured using the following CM system property:

`cmas-restapi-core, domain.map.for.client.config.<configName>`

The value has to be the domain and (if required) target port of the respective CM/Track V2 instance.

Example: `cmas-restapi-core, domain.map.for.client.config.trackConfig1` = “myserver:8080/track”

If two CM/Track instances should work with the same configuration, use the following syntax: `cmas-restapi-core, domain.map.for.client.config.<configName>` = <List of domains-with-ports>

Example: `cmas-restapi-core, domain.map.for.client.config.trackConfig2` = “myOtherServer:8080/track , myNewServer:8180/track”

The CM system property has to be added manually to the system configuration!

C.9.3 Configuration of Specific Pages in CM/Track

The configuration of specific pages in CM/Track is done using configuration and localization files.

The file `config.json` is used to configure aspects of CM/Track after the user logged in. It includes both layout settings for the pages to create and display tickets, and settings for the welcome page. The localizations for the non-public pages of CM/Track can be found in the so called localization files, e.g. `localization_en.json` or `localization_de.json`.

The file `public.json` is used to configure the login page. It includes both settings and localizations.

C.9.3.1 Configuration of the Ticket Create and Ticket Display Page

The configuration of the layout of the Ticket Create Page and Ticket Display Page in CM/Track is based on the following files:

- config.json
- localization file (e.g., `localization_en` or `localization_de`)

> Please note that each ticket field which should be displayed in CM/Track has to be configured using the annotations concerning customer exposure! Only fields which are customer-exposed will be displayed! Please read section CM/Track V2: Data Availability For Customers for a detailed explanation.

config.json

In this file, the layout/order of the ticket fields is defined, and general layout and configuration parameters are set.

The `layout` object has the following structure:

```
"layout": {
   "<Ticket field group name 1>": [
      [<List of names of ticket fields in 1st line],
      [<List of names of ticket fields in 2nd line],

      [<List of names of ticket fields in n'th line],
   ],
   <Ticket field group name 2>": [
      [<List of names of ticket fields in 1st line],
      [<List of names of ticket fields in 2nd line],

      [<List of names of ticket fields in n'th line]
   ]
}
```

Rules for writing a JSON layout object:

- each line (array) which contains ticket fields must have the same number of elements, use placeholders for empty spaces (see info box below!)

- use unique placeholder names (see info box below!)
- do not use comments within the JSON statement

Info about placeholders:

Please note that any placeholder for an empty position must be unique! You can use either an empty string (only once!) or placeholder strings (e.g., "x1"). A placeholder is treated like a real data field, i.e., in case two empty positions next to each other in one line are required, this can be set like "x1","x1" , or "x1","x2". The same placeholder in different lines where the fields cannot be connected will not work. Null as a placeholder will not work.

Example:

```
"layout": {
   "helpdesk_standard": [
      ["module", "categories","categories"],
      ["reactiontime","priority",""]
   ]
},
```

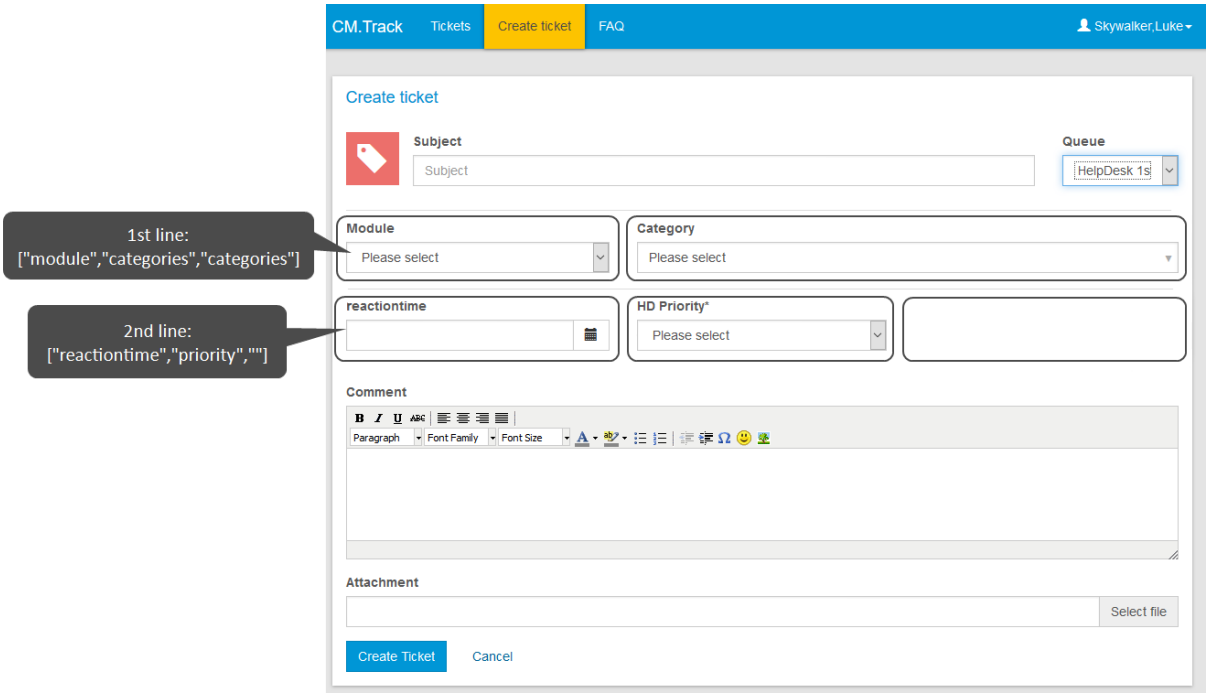


Figure 241: *CM/Track Client - Display of the ticket fields of the example layout shown above*

The **layout options** are set as follows:

```
"layoutOptions": {
   "expandable": false,
   "defaultColumnsNr": 3
}
```

- `expandable` (see example below)
  - "true": Ticket fields which are not explicitly mentioned in the JSON layout object but which are annotated as customer exposed will be displayed.
  - "false": Only ticket fields which are explicitly mentioned in the JSON layout object are displayed.
- `defaultColumns`: This is only used if `expandable` = "true" and a ticket field group is not explicitly layouted. Then the `defaultColumns` are used for display.

If a **comment** is required to create a ticket, this is defined using the following object

```
commentRequired (Boolean)
```

Example:

Only the first two lines are layouted, but more fields (e.g., *infotext*) are displayed, because expandable is "true".

```
{
   "layout": {
      "helpdesk_standard": [
         ["module", "categories","categories"],
         ["reactiontime","priority",""]
      ]
   },
   "layoutOptions": {
      "expandable": true,
      "defaultColumnsNr": 2
   },
   "commentRequired": false
}
```

Code example 21: *Example config.json file*

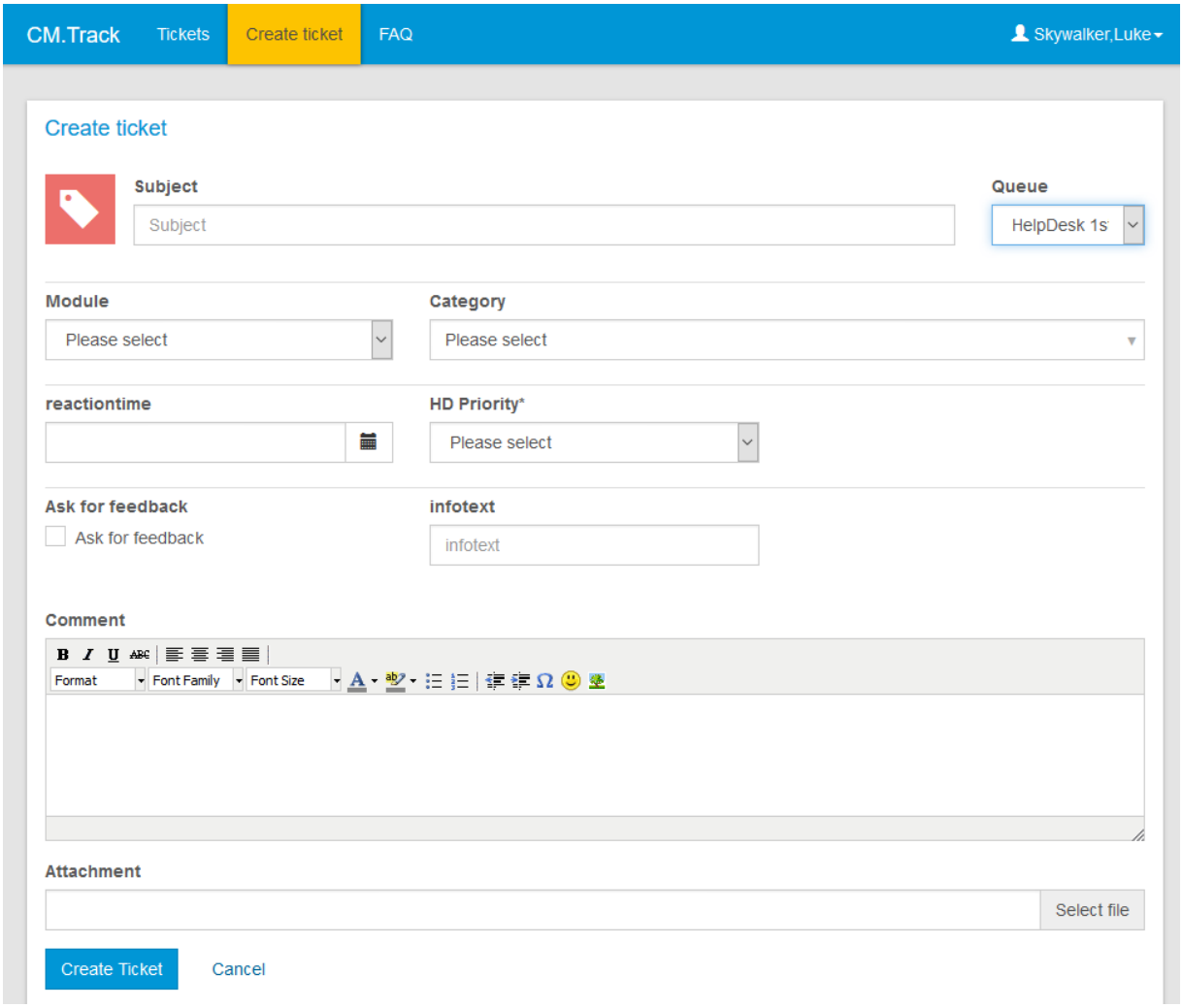


Figure 242: *CM/Track layout using layout of code example above*

Same example using `expandable` = “false”: only the explicitly layouted ticket fields will be displayed.

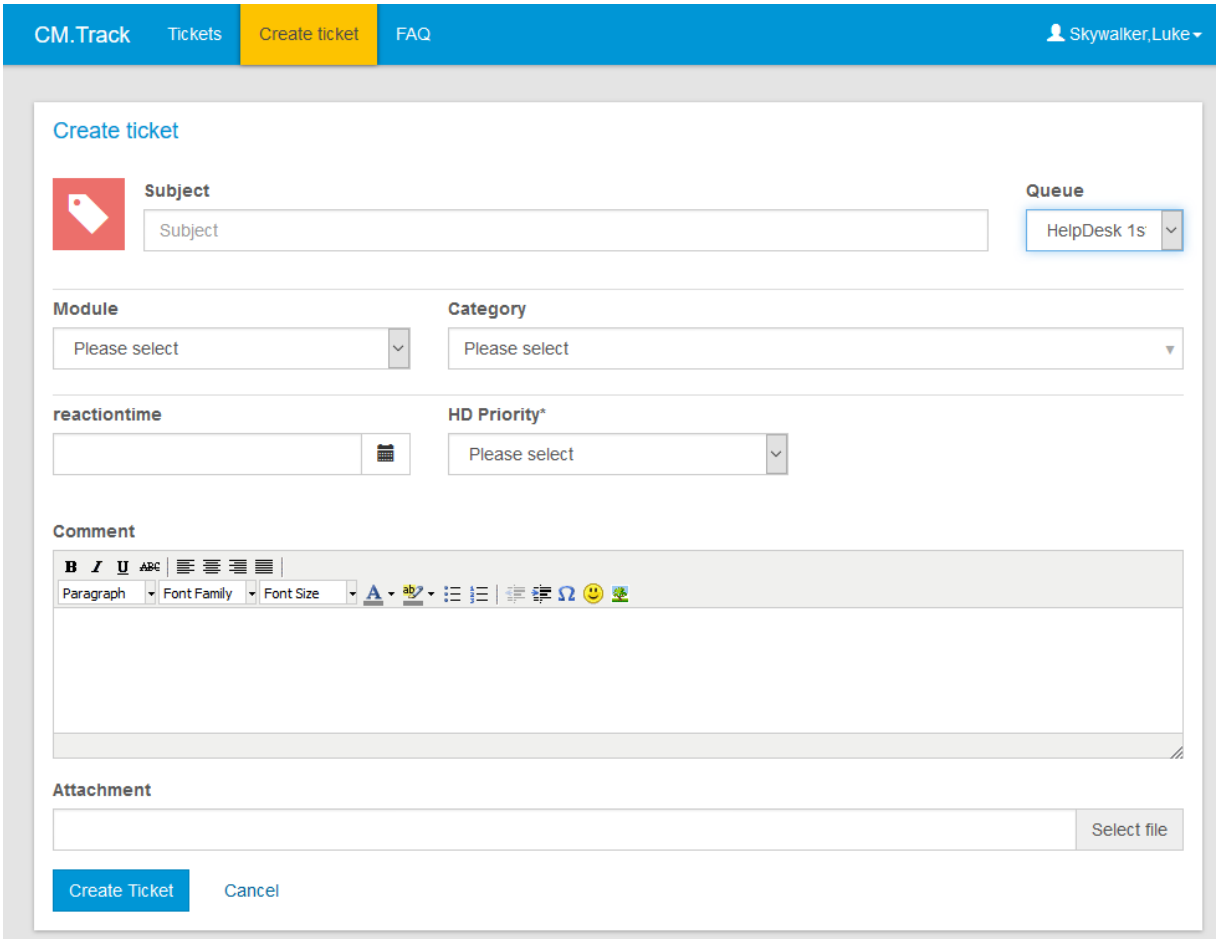


Figure 243: *CM/Track layout using layout of code example above, but expandable = false*

Localization File

The localization files, e.g., `localization_en.json` and `localization_de.json`, contain simple mappings of properties to values. The values are the terms which should be displayed in the GUI in the respective language. The following two examples each show an excerpt from an English and a German localization file.

```
{
   "edit":"Edit",
   "search":"Search",
   "view":"View",
   "delete":"Delete",
   "filter":"Filter",
   "add":"Add",
   "save":"Save",
   "cancel":"Cancel",
   "true":"true",
   "false":"false",
   "previous":"Previous",
```

```
   "next":"Next",
   "validation_required" : "This field is required",
   "validation_password" : "Please enter a valid password",
   "validation_email" : "Please enter a valid e-mail",
   "validation_pattern": "Invalid input format",
   "validation_date" : "Please enter a valid date",
   "validation_time" : "Please enter a valid time",
   "validation_datetime" : "Please enter a valid date and time",
   "validation_datetime_local" : "Please enter a valid date and time",
   "validation_number" : "This field must be numeric",
   "validation_color" : "Please enter a valid color",
   "validation_range" : "Please enter a valid range",
   "validation_month" : "Please enter a valid month",
   "validation_url" : "Please enter a valid URL",
   "validation_file" : "Invalid file",
   "validation_minlength" : "Please use at least {{ minlength }} characters",
   "validation_maxlength" : "Please do not exceed {{ maxlength }} characters",
   "validation_min_time" : "The time provided should be after {{ min |date:
    \"HH:MM\" }}",
   "validation_max_time" : "The time provided should be before {{ min |date:
    \"HH:MM\" }}",
   "validation_ngFabMatch" : "The {{ type ===\"password\"? \"passwords\" :
    \"values\" }} should match",
   "validation_min_date" : "The date provided should be after
    {{min|date:\"dd.MM.yy\"}}",
   "validation_max_date" : "The date provided should be before {{max |date:
    \"dd.MM.yy\"}}",
   "validation_min_number" : "The number provided should be at least {{min}}",
   "validation_max_number" : "The number provided should be at max {{max}}",
   "validation_fixedprecision" : "Illegal format. The integer part should have
    {{integer}} digits, the fraction part {{fractional}} digits",
   "default_error_message" : "Sorry, it looks like something went wrong. An error
    has occurred. Please refresh your browser and try again later. If the error
    occurs another time, please contact your administrator.",
   "cf_value_true" : "Yes",
   "cf_value_false" : "No",
   "session_expired" : "Session has expired",
   "access_denied_error" : "You do not have permission to view this object. Please
    select a different object.",
   "attachment_too_large" : "Sorry, the attachment size exceeds the configured
    limits for attachments. Please shrink the attachment.",
   "datepicker_current" : "Today",
   "datepicker_clear" : "Clear",
   "datepicker_close" : "Done",
   "datepicker_Sunday": "Sunday"
}
```

Code example 22: *Excerpt from a localization_en.json file*

```
{
   "edit":"Bearbeiten",
```

```
"search":"Suchen",
"view":"Anzeigen",
"delete":"Löschen",
"publish":"Veröffentlichen",
"filter":"Filtern",
"add":"Hinzufügen",
"save":"Speichern",
"cancel":"Abbrechen",
"true":"Ja",
"false":"Nein",
"previous":"Vorheriges",
"next":"Nächstes",
"validation_required" : "Dieses Feld ist ein Pflichtfeld",
"validation_password" : "Geben Sie ein gültiges Passwort ein",
"validation_email" : "Geben Sie eine gültige E-Mail-Adresse ein",
"validation_pattern": "Ungültiges Eingabeformat",
"validation_date" : "Geben Sie ein gültiges Datum ein",
"validation_time" : "Geben Sie eine gültige Zeit ein",
"validation_datetime" : "Geben Sie ein gültiges Datum und eine gültige Zeit
 ein",
"validation_datetime_local" : "Geben Sie ein gültiges Datum und eine gültige
 Zeit ein",
"validation_number" : "Dieses Feld muss numerisch sein",
"validation_color" : "Geben Sie eine gültige Farbe ein",
"validation_range" : "Geben Sie einen gültigen Bereich ein",
"validation_month" : "Geben Sie einen gültigen Monat ein",
"validation_url" : "Geben Sie eine gültige URL ein",
"validation_file" : "Ungültige Datei",
"validation_minlength" : "Verwenden Sie mindestens {{ minlength }} Zeichen",
"validation_maxlength" : "Verwenden Sie höchstens {{ maxlength }} Zeichen",
"validation_min_time" : "Die angegebene Zeit sollte nach {{ min |date: \"HH:MM\"
 }} liegen",
"validation_max_time" : "Die angegebene Zeit sollte vor {{ min |date: \"HH:MM\"
 }} liegen",
"validation_ngFabMatch" : "Die {{ type ===\"password\"? \"passwords\" :
 \"values\" }} müssen übereinstimmen",
"validation_min_date" : "Das angegebene Datum sollte nach dem
 {{min|date:\"dd.MM.yy\"}} liegen",
"validation_max_date" : "Das angegebene Datum sollte vor dem {{max |date:
 \"dd.MM.yy\"}} liegen",
"validation_min_number" : "Die angegebene Zahl sollte mindestens {{min}} sein",
"validation_max_number" : "Die angegebene Zahl sollte höchstens {{max}} sein",
"validation_fixedprecision" : "Ungültiges Format. Der ganzzahlige Teil sollte
 {{integer}} Stellen haben und der Dezimalteil {{fractional}} Stellen",
"cf_value_true": "Ja",
"cf_value_false": "Nein",
"default_error_message" : "Entschuldigung, es scheint etwas schiefgelaufen zu
 sein. Ein Fehler ist aufgetreten. Bitte aktualisieren Sie Ihren Browser und
 versuchen es erneut. Sollte der Fehler nochmals auftreten, kontaktieren Sie
 bitte Ihren Administrator.",
"session_expired" : "Die Sitzung ist abgelaufen",
```

```
   "access_denied_error" : "Sie haben keine Zugriffsrechte auf dieses Objekt. Bitte
    wählen Sie ein anderes Objekt aus.",
   "attachment_too_large" : "Leider überschreitet der Anhang die maximale Größe.
    Bitte verkleinern Sie die Datei.",
   "datepicker_current" : "Heute",
   "datepicker_clear" : "Löschen",
   "datepicker_close" : "Fertig",
   "datepicker_Sunday": "Sonntag"
}
```

Code example 23: *Excerpt from a localization_de.json file*

C.9.3.2 Configuration of the Welcome Page

The welcome page is displayed after the user has logged in to CM/Track. It provides the following elements:

- a customizable welcome message
- a customizable sub-header
- a search field
- colored boxes with links to create tickets in the queues for which the user has the required permissions. If the number of queues is greater than four, then the fourth box is a placeholder offering a list with all remaining queues. Each regular box shows a button *Create*, which directly leads to the ticket create page for this particular queue.

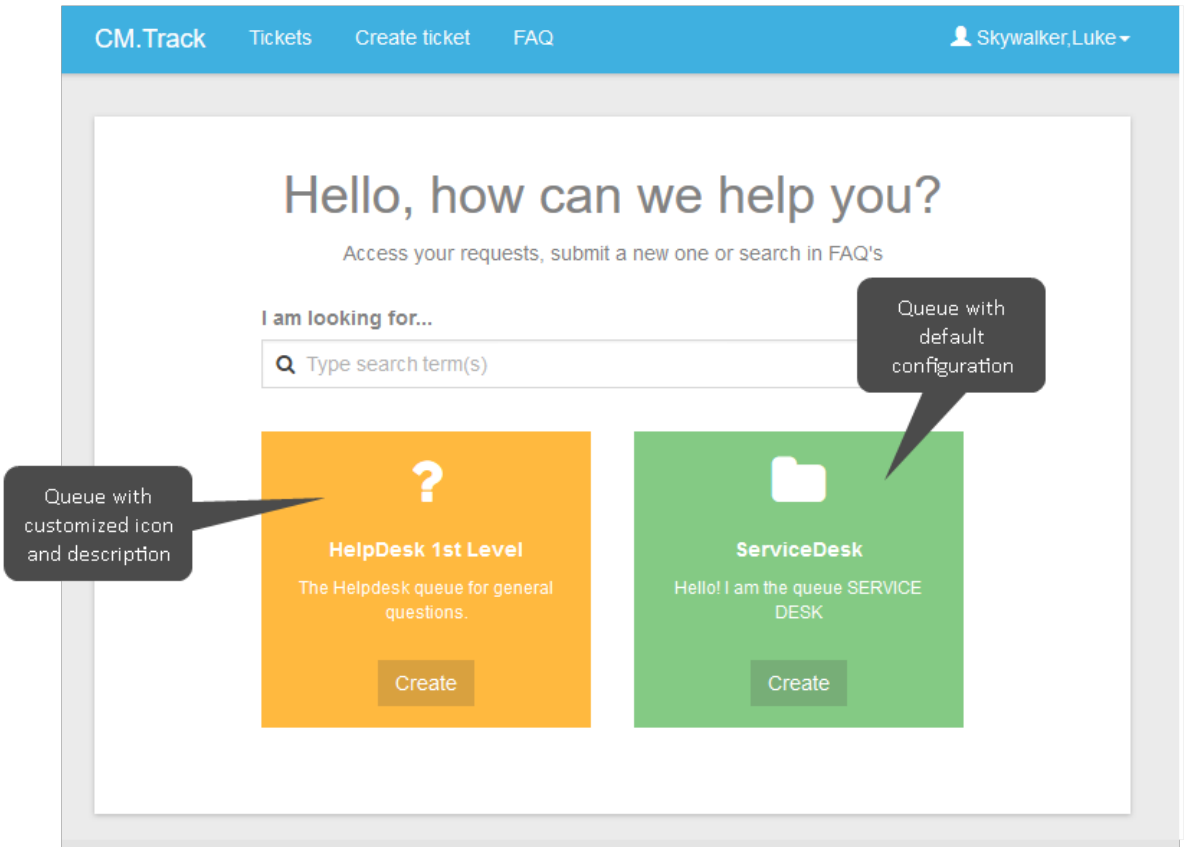


Figure 244: *Welcome page in CM/Track*

In order to configure the welcome page of your customer portal, CM/Track, you have to adapt the `config.json` file.

If you have updated the CM system from a version minor than CM 6.11.1.1, you have to add the welcome page section manually into the `config.json` file. In CM versions 6.11.1.1 and up, an example welcome page section is part of the template *TrackV2*.

The following code shows an example `config.json` which contains a welcome page section.

Figure 245: *Example config.json file with welcome page section*

The following parameters are available:

- `@name`: technical name of the queue and therefore: reference to the queue
- `description`: description of the queue, can either be a fixed string without localization or a key from the `localization_en.json` file or another localization file (as in the example above: *custom_HD1L_description* is a key which has to be present in the localization files).
- `icon`: name of an icon from the FontAwesome icon set (see http://fontawesome.io/icons/). Use the following syntax: fa-<name of the icon>.

If no explicit configuration is provided, the default values (description of the queue in the Admin Tool and folder icon) are used. All the queues for which the user has permissions to create a ticket are displayed.

The localization for the welcome page beyond the queue configuration is done by a small set of keys in the localization files for each language configured, e.g. `localization_en.json`. These keys have to be added to existing configurations. Only new (as of 6.11.1.1) configurations based on the template *TrackV2* will include the following keys per default.

```
"welcome_header" : "Hello, how can we help you?" ,
"welcome_subheader" : "Access your requests, submit a new one or search in FAQ's" ,
"welcome_search_label" : "Type search term(s)" ,
"searchbar_lbl" : "I am looking for..." ,
"queue_action" : "Create" ,
"jump_to_queues" : "More" ,
"custom_HD1_description": "The Helpdesk queue for general questions."
```

Code example 24: *Excerpt from localization_en.json showing the default values of the text displayed on the welcome page and the custom label created for the queue description*

The above configuration results in the above welcome page for a user with access to two queues. Note that both queues are displayed although only one queue has been configured explicitly.

C.9.3.3 Configuration of the Login Page or Other Public Pages

To customize a public CM/Track page (the sign in page, the password change pages) you need to add a `public.json` file to your track clients configuration. The `public.json` file contains localizations and the setting regarding the password reset functionality. The localizations are all to be found in the JSON object `signin`, `i18n` (the latter stands for internationalization).

Example:

```
{
   "signin": {
      "i18n" : {
         "en" : {
            "nav_track_brand":"CM/Track",
            "password_change_header": "Change password",
            "password_change_description": "You would like to change your password.
             Please fill the required fields.",
            "password_change_old": "Old password",
            "password_change_new": "New password",
            "password_change_confirm": "Confirm password",
            "password_change_submit": "Change password",
            "password_change_cancel": "Cancel",
            "password_change_changed": "Your password was successfully changed!",
            "password_old_not_match": "Password does not match the old password",
            "password_confirm_not_match": "Password does not match the confirmed
             password",
            "password_reset_header": "Change your password?",
            "password_reset_header_change": "Change password",
            "password_reset_hint": "Enter your user name and we\"ll help you reset
             your password. ",
            "password_reset_no_user": "User does not exist",
```

```
      "password_reset_mail_or_login": "User name",
      "password_reset_new": "New password",
      "password_reset_confirm": "Confirm password",
      "password_reset_set_new": "Set new password",
      "password_reset_submit": "Continue",
      "password_reset_cancel": "Cancel",
      "password_reset_sent": "An email was sent to your account. Please follow
       the instructions in the email.",
      "password_reset_changed": "Your password was successfully changed!",
      "password_confirm_not_match": "Password does not match the confirmed
       password",
      "reset_code_no_match": "Reset code does not match to any user",
      "reset_code_expired": "Reset code expired",
      "signin_header": "Please sign in",
      "signin_submit": "Sign in",
      "signin_forgotYourPassword": "Do not remember your password?",
      "signin_dontHaveAccountYet": "Don\"t have an account yet?",
      "signin_register_link": "Please register here",
      "signin_username_label": "Username",
      "signin_username_placeholder": "Username",
      "signin_password_label": "Password",
      "signin_password_placeholder": "Password",
      "signin_rememberme_label": "Remember me",
      "signin_error_signin_header": "Login failed",
      "signin_error_signin_text": "Please recheck your username and password
       and try again",
      "signin_error_signout_header": "Logout operation failed",
      "signin_error_signout_text": "Server responded with an error message
       while trying to signout. Please try again later.",
      "signin_error_forbidden_header": "The requested operation is forbidden",
      "signin_error_forbidden_text": "You\"re not allowed to perform the
       requested operation",
      "signin_error_serverfailed_header": "Unexpected server error",
      "signin_error_serverfailed_text": "An internal server error occurred and
       your request couldn't be completed. Please try again",
      "signin_error_unauthorized_header": "Unauthorized access",
      "signin_error_unauthorized_text": "",
      "signin_success_signout_header": "You successfully signed out",
      "signin_success_signout_text": ""
   },
   "de" : {
      "nav_track_brand": "CM/Track",
      "password_change_header": "Passwort ändern",
      "password_change_description": "Sie möchten Ihr Passwort ändern. Füllen
       Sie dazu bitte die erforderlichen Felder aus.",
      "password_change_old": "Altes Passwort",
      "password_change_new": "Neues Passwort",
      "password_change_confirm": "Passwort bestätigen",
      "password_change_submit": "Passwort ändern",
      "password_change_cancel": "Abbrechen",
      "password_change_changed": "Ihr Passwort wurde erfolgreich geändert!",
      "password_old_not_match": "Kennwort stimmt nicht mit altem Kennwort
       überein",
```

```
            "password_confirm_not_match": "Passwortbestätigung stimmt nicht
             überein",
            "password_reset_header": "Passwort vergessen?",
            "password_reset_header_change": "Passwort ändern",
            "password_reset_hint": "Geben Sie Ihren Benutzernamen ein, um Ihr
             Passwort zurückzusetzen. ",
            "password_reset_no_user": "Benutzer nicht vorhanden",
            "password_reset_mail_or_login": "Benutzername",
            "password_reset_new": "Neues Passwort",
            "password_reset_confirm": "Passwort bestätigen",
            "password_reset_set_new": "Neues Passwort festlegen",
            "password_reset_submit": "Weiter",
            "password_reset_cancel": "Abbrechen",
            "password_reset_sent": "Es wurde eine E-Mail an Ihr Konto gesendet.
             Bitte folgen Sie den Anweisungen in der E-Mail.",
            "password_reset_changed": "Ihr Passwort wurde erfolgreich geändert!",
            "password_confirm_not_match": "Passwortbestätigung stimmt nicht
             überein",
            "reset_code_no_match": "Code zum Zurücksetzen passt zu keinem Benutzer",
            "reset_code_expired": "Code zum Zurücksetzen abgelaufen",
            "signin_header": "Bitte melden Sie sich an.",
            "signin_submit": "Anmelden",
            "signin_forgotYourPassword": "Sie haben Ihr Passwort vergessen?",
            "signin_dontHaveAccountYet": "Sie haben noch keinen Zugang?",
            "signin_register_link": "Bitte registrieren Sie sich hier.",
            "signin_username_label": "Benutzername",
            "signin_username_placeholder": "Benutzername",
            "signin_password_label": "Passwort",
            "signin_password_placeholder": "Passwort",
            "signin_rememberme_label": "Anmeldedaten merken",
            "signin_error_signin_header": "Anmeldung fehlgeschlagen",
            "signin_error_signin_text": " Bitte überprüfen Sie Benutzername und
             Passwort und versuchen Sie es erneut.",
            "signin_error_signout_header": "Abmeldung fehlgeschlagen",
            "signin_error_signout_text": "Ihr Abmeldeversuch hat beim Server eine
             Fehlermeldung hervorgerufen. Bitte versuchen Sie es zu einem späteren
             Zeitpunkt erneut.",
            "signin_error_forbidden_header": "Die angeforderte Operation ist nicht
             erlaubt",
            "signin_error_forbidden_text": "Sie sind nicht berechtigt, die
             angeforderte Operation durchzuführen",
            "signin_error_serverfailed_header": "Unerwarteter Serverfehler",
            "signin_error_serverfailed_text": "Es ist ein interner Serverfehler
             aufgetreten. Bitte versuchen Sie es zu einem späteren Zeitpunkt
             erneut",
            "signin_error_unauthorized_header": "Unberechtigter Zugriff",
            "signin_error_unauthorized_text": "",
            "signin_success_signout_header": "Sie haben sich erfolgreich
             abgemeldet",
            "signin_success_signout_text": ""
        }
    }
```

```
    }
}
```

Code example 25: *Example of a public.json file for CM REST client configuration (this example is also provided in the Admin Tool)*

In order to change words and/or phrases which are displayed on public pages, just edit the `public.json` file in the Admin Tool and save it. The change will be visible on the GUI just in time. Please see also the following example.

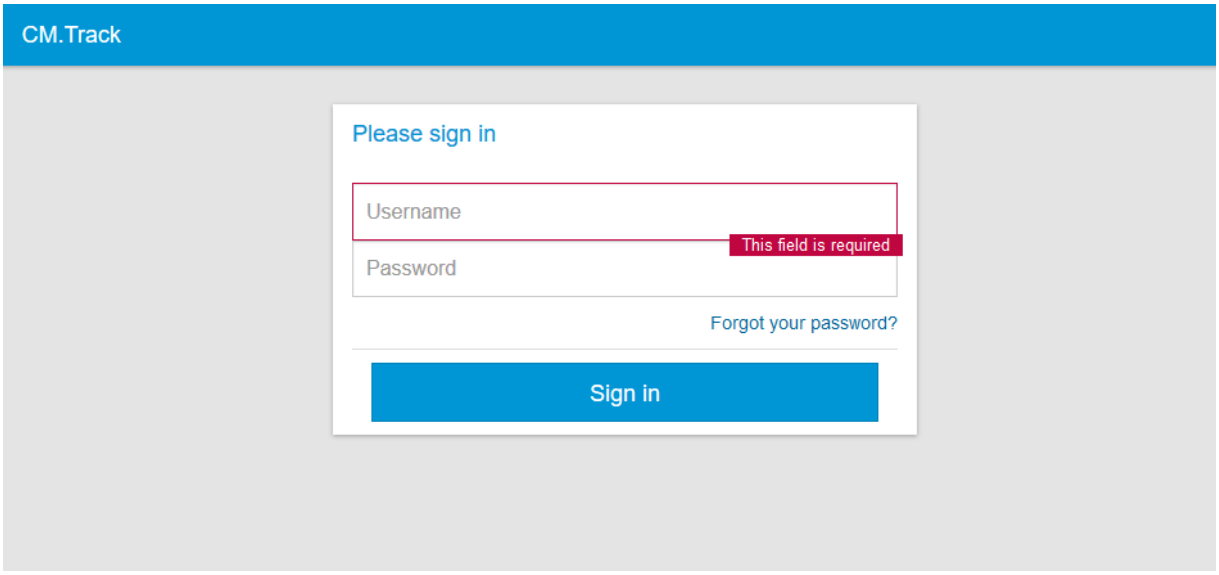


Figure 246: *The CM/Track sign-in page in standard format*

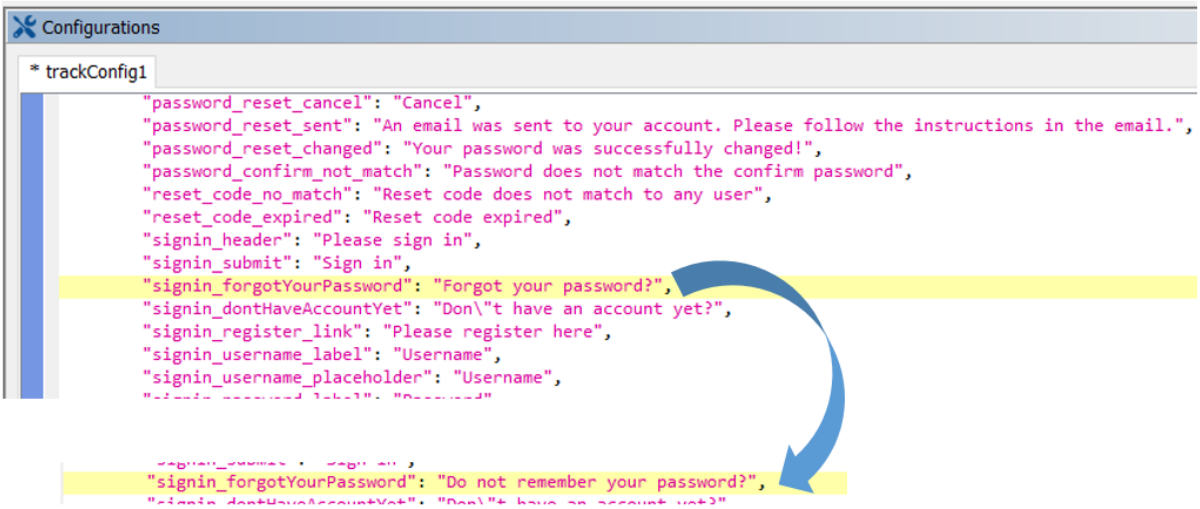


Figure 247: *Simple editing of the public.json file*

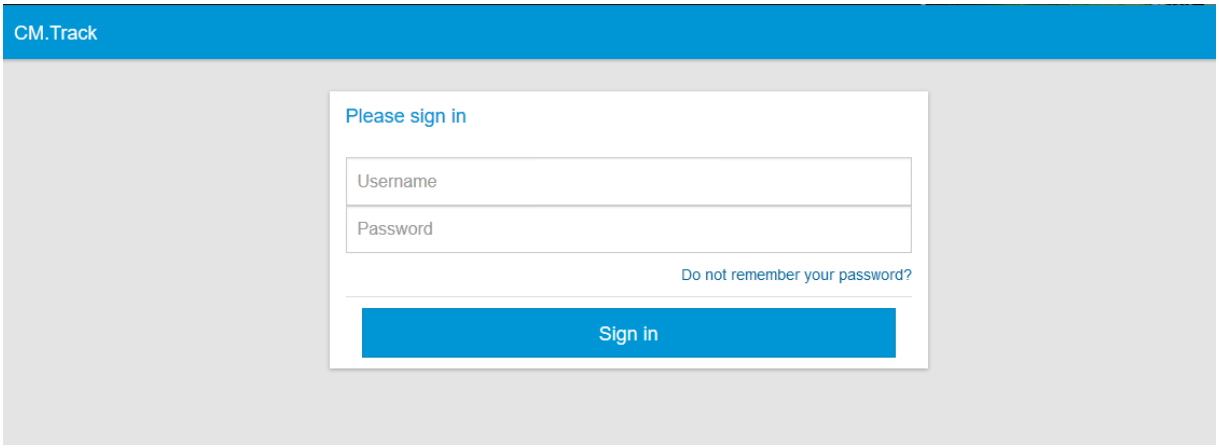


Figure 248: *The CM/Track sign-in page, slightly customized (one phrase changed)*

C.9.3.4 Configuration of the Password Reset and Change Functionalities

The functionality for password reset on the login page and password change after logging in can be enabled and disabled in the respective configuration files.

config.json

The `config.json` file allows you to determine whether the *Change password* button should be displayed after logging in to CM/Track. If `disablePasswordChange` is set to “true”, the user cannot change the password. If it is set to “false”, changing the password is possible.

```
"disablePasswordChange": true,
```

Code example 26: *Excerpt from config.json to disable the functionality to change the password*

Per default, i.e. when the `disablePasswordChange` parameter has not been defined explicitly, the menu item *Change password* is available.

public.json

The `public.json` file allows you to determine whether the *Forgot your password?* link should be displayed on the login page. If `disablePasswordReset` is set to “true”, the user cannot reset his password. If it is set to “false”, resetting the password is possible.

```
"disablePasswordReset": true,
```

Code example 22: Excerpt from public.json to disable the functionality to reset the password

Per default, i.e. when the `disablePasswordReset` parameter has not been defined explicitly, the link *Forgot your password?* is displayed.

C.9.4 Creating New Configuration Pages

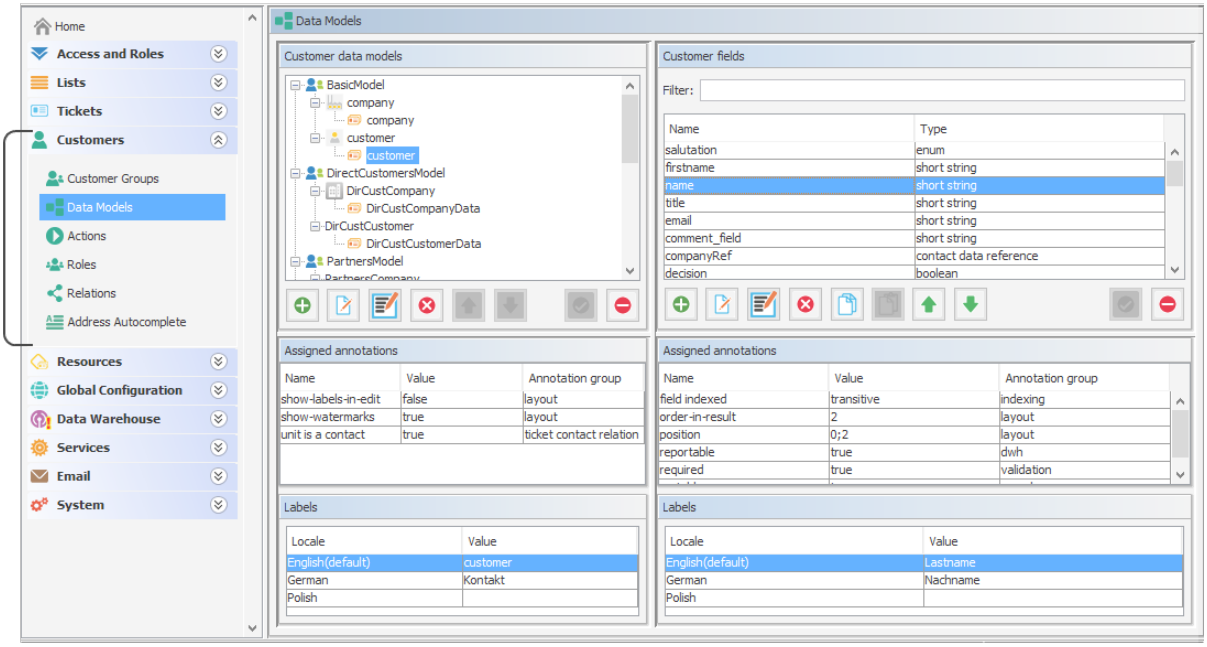
You can add a new JSON file to the configuration using the button *Create new file to the current configuration*. There are two use cases for this functionality.

- You want to add a new localization file, because you have added another language to your system configuration. Use `localization_<browser locale>` as name for such a file. You could, for example, add a file named `localization_fr` for French translations. Build the new file based on the existing example localization files.
- You want to add a new file which is required in a customer-specific project. This is a case for experienced CM consultants and developers.

C.9.5 Behavior Concerning System Installations and Updates

In a new installation of ConSol CM version 6.11.0.5 or during an update to this version, the templates/examples for the REST client GUI configuration are installed. All configurations, even the default configuration (*TrackV2*), have to be added manually if required.

D - Customer Data Model Section



Starting with version 6.9, ConSol CM offers a very flexible and powerful customer administration based on FlexCDM, the Flexible Customer Data Model.

In this section, you learn how to set-up and manage the Flexible ConSol CM Customer Data Model, FlexCDM, and all related topics. The following subjects are explained:

- Introduction to FlexCDM
- A Short Introduction to FlexCDM-Specific Web Client Functionalities
- Setting Up the Customer Data Model
- Customer Field Management and GUI Design for Customer Data
- Templates for Customer Data
- Managing Customer Groups
- Customer Roles
- Customer Relations
- Action Framework - Customer Actions
- Address Autocomplete

D.1 Introduction to FlexCDM

This chapter discusses the following:

Because the customer data model ***FlexCDM***, which has been introduced to ConSol CM with version 6.9, is rather complex and very powerful, a separate introduction chapter will help you to understand all the details.

D.1.1 FlexCDM at a Glance

D.1.1.1 Flexible Customer Data Model

As the name FlexCDM suggests, the ConSol CM customer data model offers a very high degree of flexibility. Various **customer groups** can be defined, each with its particular data model.

In ConSol CM, we talk about **customers** to describe the general CM object. This can be either a company or a contact. A **company** represents an object on company level which will, in most cases, be a real company, a subsidiary, a division or some other organizational unit on a higher level. It can also be a collection of products, a machine pool, or any other object which comprises sub-objects. A **contact** represents an object on contact level, i.e. on the lower level of the customer model. A contact will often be a real person but can also be a product, a machine or some other object.

Contacts as well as companies can be set as customer for a ticket.

There are different ways to configure customer data models for customer groups. Within a customer group, there might be ...

- a *contact* and a *company* level:
  then we talk about a **two-level customer data model** (where a company can contain several contacts)
- only a *contact* or only a *company* level:
  then we talk about a **one-level customer data model**

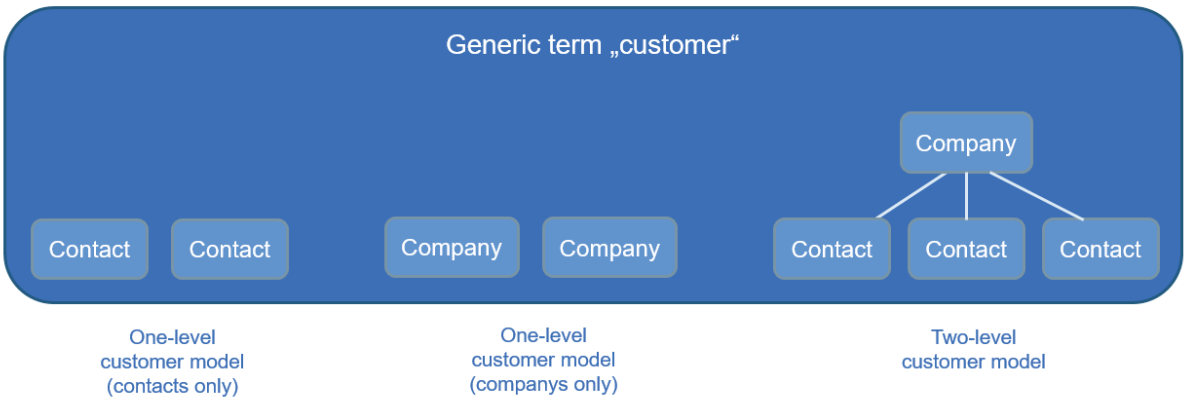


Figure 249: *Types of customer data models in ConSol CM*

For example, you could classify your customers in two customer groups:

1. **Resellers**
   With contact and company level.
2. **End customers**
   With contact level only.

You can configure as many customer data models as required. Every customer data model can be used for one or more customer groups.

A customer data model comprises the general model, i.e. the levels (contact and company or contact/company only) and all data fields for all components (e.g. name, address, and phone for a company or name, email, and room number for a contact).

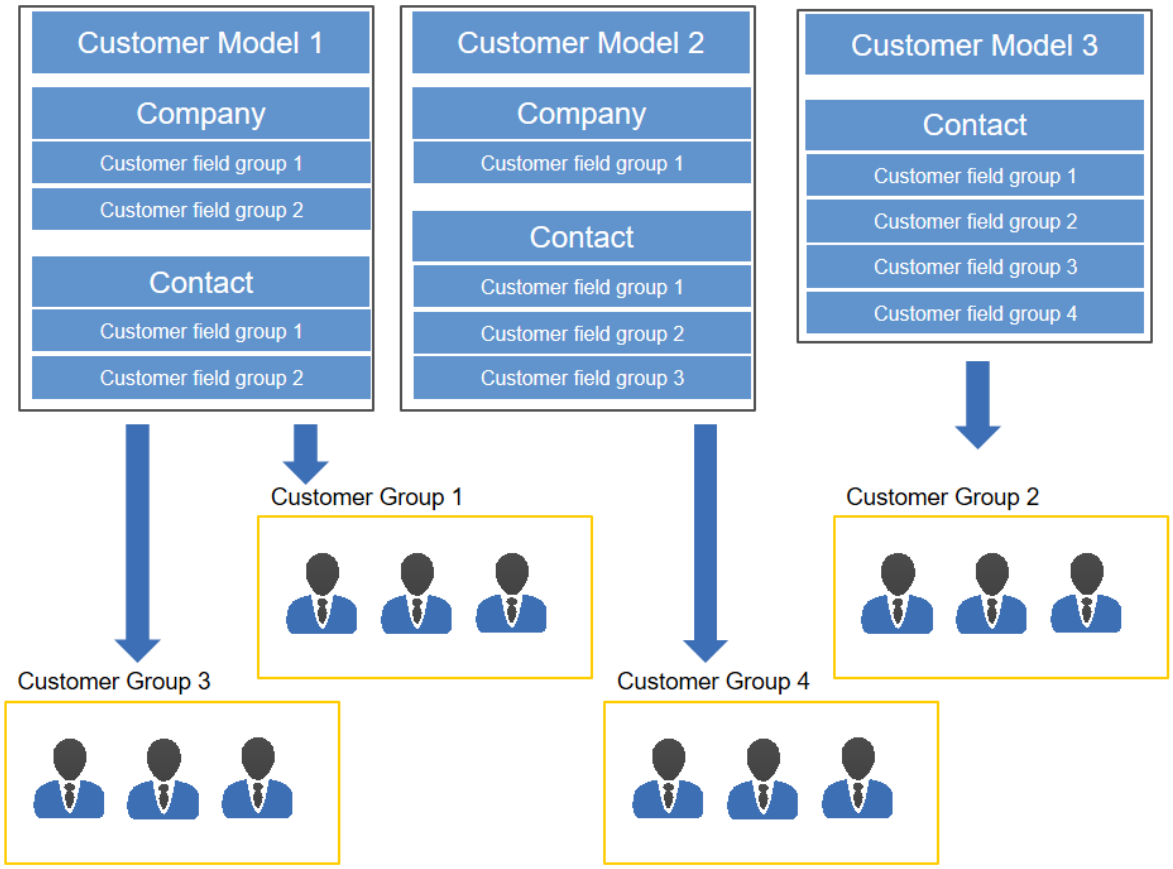


Figure 250: *ConSol CM FlexCDM - General principle*

For a two-level customer data model:

The terms *company* and *contact* are used to indicate the hierarchical level of an object within FlexCDM. An object of type *company* does not necessarily have to be a real company, it can also be a town with several machines (contacts) located in this town, an organization with several subsidiaries (contacts), or even a technical unit (e.g. a ship) with several contacts in the unit. Similarly, an object of type *contact* does not necessarily have to be a person, it can also be a location, a machine, or anything else which should represent the contact level.

For a one-level customer data model:

The customer objects in a one-level customer model are either of type *contact* or of type *company*.

For the customers which are managed by your ConSol CM system, the levels and names of all components entirely depend on the configuration of FlexCDM.

Using FlexCDM you can build different realms where each includes a specific customer group and the respective data and processes.

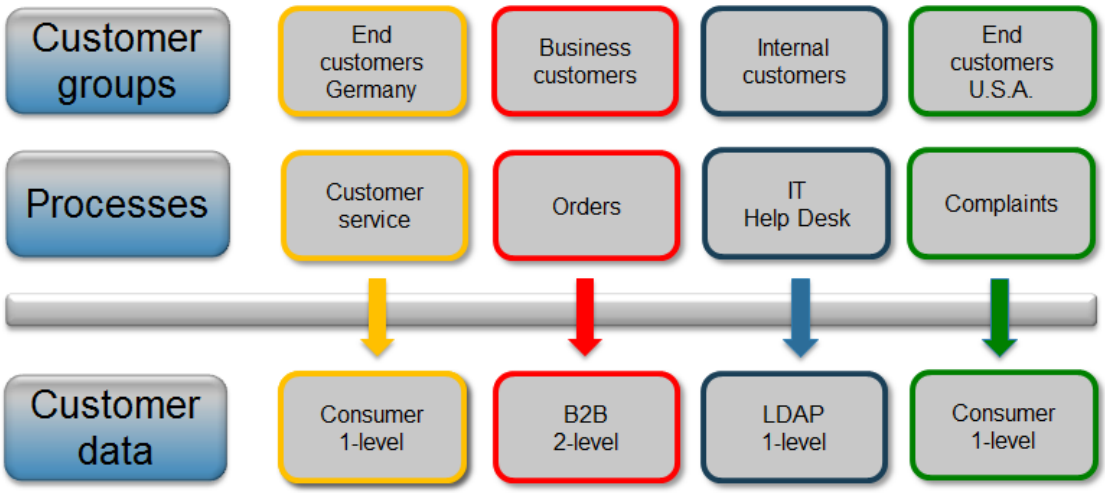


Figure 251: *ConSol CM FlexCDM - Customer data model*

Please see section Setting Up the Customer Data Model for a detailed description of the customer management.

D.1.2 Introduction to FlexCDM Objects

In this section, we will give you an overview of all objects which are relevant for the FlexCDM.

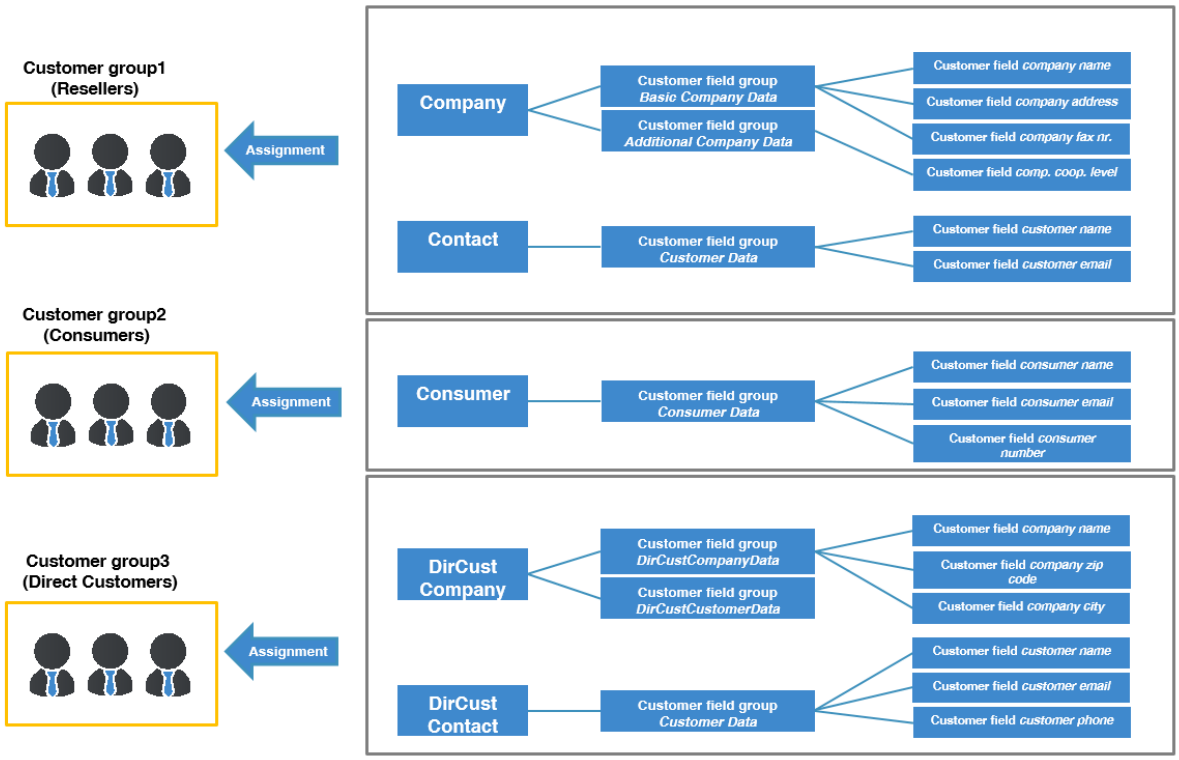


Figure 252: *ConSol CM FlexCDM - Example configuration*

D.1.2.1 Important Terms

Here are some important terms for the FlexCDM:

- **Customer**
  General term for customer objects, can be of type *contact* or of type *company*.
- **Company**
  Customer object of type *company*, company level.
- **Contact**
  Customer object of type *contact*, contact level.
- **Customer group**
  A group of customers with a specific customer data model. The engineer permissions (via roles) for customer management are assigned based on customer groups.
- **Customer object**
  An object within the customer data model. The object type can be *contact* (often times a person) or *company*. The technical (Groovy) equivalent is an object of class `Unit`.

- **Customer object definition**
  All definitions pertaining to the unit. For a company these are e.g. all customer field groups, all group annotations, and the assignment of all templates (for the display of customer data in the Web Client, not to be confused with other templates in ConSol CM!)
- **Customer field group**
  A group comprising one or more data field(s) (customer fields), analog to a ticket field group for ticket data. A customer field group can be shown or hidden or it can be displayed as a tab in the Details section.
- **Customer field**
  A single data field (types like ticket field types) that can contain customer data, analog to ticket fields when defining data fields for ticket data.
- **Customer data model**
  The whole data model that can be assigned to a customer group.
  The data model can have:
  - one level (only contact or only company)
  - two levels (company and contact)

  The customer data model also contains the definitions of customer field groups and customer fields.
- **Customer relations**
  Relations between a company and contacts or between companies or between contacts. All relations in their entirety represent the customer relations network.
- **Action Framework**
  A set of modules in CM which allows event-triggered, workflow-independent programming. The *customer actions* (also called *unit actions)* which are relevant in the FlexCDM context are part of the Action Framework.

D.1.3 Management of FlexCDM Objects Using the Admin Tool

In the Admin Tool, you reach the screen with the FlexCDM configuration options by opening the navigation group *Customers*.

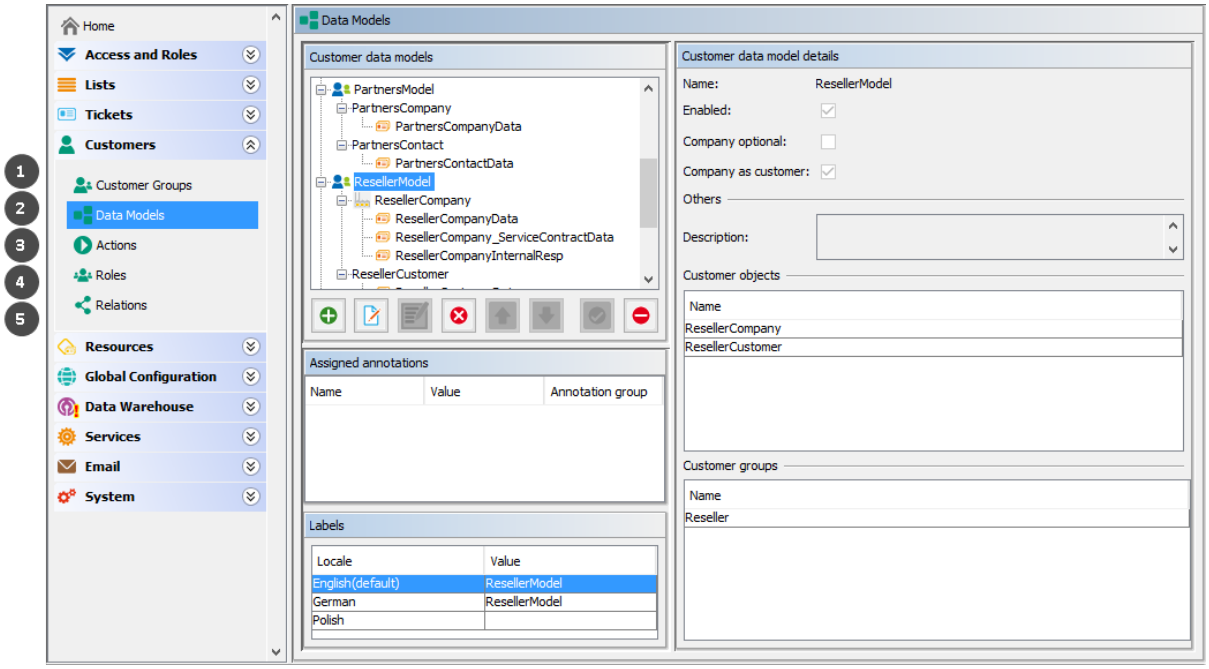


Figure 253: *ConSol CM Admin Tool - Customer navigation items relevant in FlexCDM*

The following navigation items are relevant for the definition and management of the customer data model:

- **Customer Groups (1)**
  Assignment of the following components (which have to be defined beforehand) to a customer group (see section Managing Customer Groups for details).
  - Customer data model
  - Customer actions
  - Search actions
  - CM/Phone configuration, if CM/Phone is active
- **Data Models (2)**
  Definition of the data models, i.e., definition of the data fields for customers (i.e., contacts and companies) and the GUI design (i.e., placing the data fields on the Web Client GUI). Please see sections Setting Up the Customer Data Model and Customer Field Management and GUI Design for Customer Data for details.
- **Actions (3)**
  Definition of customer actions, i.e., contact actions and company actions, please see section Action Framework - Customer Actions for details.

- **Roles (4)**
  Definition of customer roles, see section Customer Roles for details.
- **Relations (5)**
  Definition of customer relations, which represent references between customer (i.e., contact and company) objects, please see section Customer Relations for details.
  (Relations from resources to customers have to be defined in the Resource Pool data model, they cannot be configured here.)

D.2 A Short Introduction to FlexCDM-Specific Web Client Functionalities

This chapter discusses the following:

D.2.1 Introduction

You, as an administrator, might be wondering why a Web Client GUI introduction is provided in an administrator's manual. However, when you want to work with the CM customer data model, called ***FlexCDM***, you have to know the effects of all administration actions. And of course, those actions are visible in the Web Client. So in this section, we will take the role of an engineer and show several examples of working with the customer data model.

All configuration details which are required to understand the system's behavior are explained in the corresponding sections of the manual.

D.2.2 Working with the ConSol CM Web Client with FlexCDM

D.2.2.1 Example 1: Selecting the Customer Group

Provided that the engineers have access permissions for more than one customer group, they can **select the customer group** which should be used for certain operations using the *Customer Groups Filter*, a drop-down list in the main menu. The name which is displayed is the localized name of the customer group.

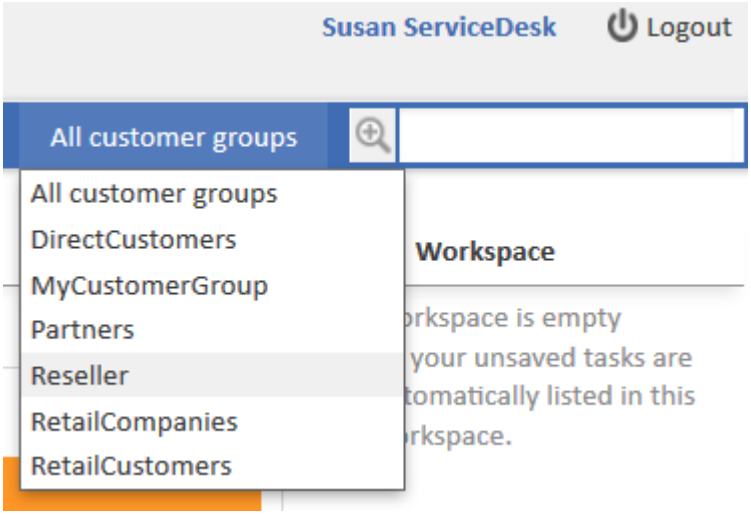


Figure 254: *ConSol CM Web Client - Selecting a customer group*

The selection influences the following actions:

- The Quick Search is performed only within the selected customer group.
- In the Detailed Search, the criterion *customer group* is only offered when *All customer groups* has been selected in the drop-down menu, otherwise the search implicitly uses only data from the selected customer group.
- In the Detailed Search, only the search fields from the selected customer group are offered.
- When a ticket is created, only the selected customer group is offered (implicitly) when a company and/or contact should be created in-line.
- A ticket can be created only in queues to which the selected customer group has been assigned.
- In the ticket list, views are only available if they contain tickets from queues to which the selected customer group has been assigned.

Please note that the Page Customization attribute *hiddenCustomerGroups* can influence the list of customer groups in the Customer Groups Filter. See section customerGroupSelector of the Page Customization for details.

D.2.2.2 Example 2: Creating a New Company and Contact

If engineers have access to several customer groups (and have selected *All customer groups* in the customer groups filter in the main menu, see example 1), they can **select the customer group when a new ticket is created** and a contact/company should be created in-line. This also depends on the selected queue. Only the customer groups which are assigned to the selected queue are available. If the option *All customer groups* has been selected in the drop-down menu, one tab is visible for the customer data of each customer group and the engineer can select the desired group.

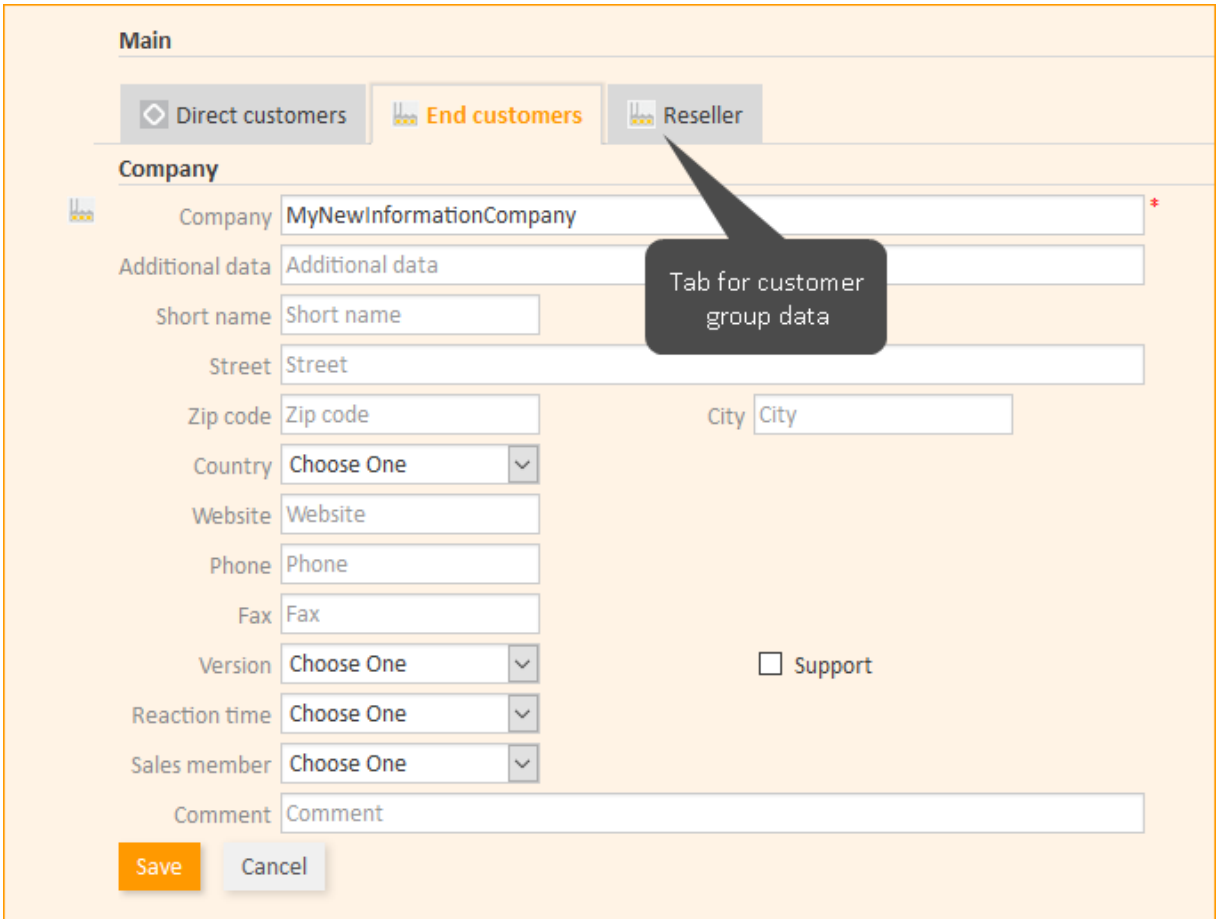


Figure 255: *ConSol CM Web Client - Creating a new company within a customer group*

If an engineer works primarily in a certain customer group but does not want to set the customer groups filter to this group, he can set his default customer group in his engineer profile. Then, the tab of this customer group will be opened / displayed when the ticket create page is opened.

D.2.2.3 Example 3: Using Company and Contact Page

Provided there is a two-level customer data model (company and contact) there are **separate company and contact pages**. On both pages, you can add comments and attach files. Those operations are then also visible in the history of the company (resp. contact) page.

For the company and for the contact object, icons can be defined for each customer data model, improving usability. The header of the contact and company page is defined by the localized value of the name of the respective customer object, see section Setting Up the Customer Data Mode, Name of the Customer Object.

Company Page

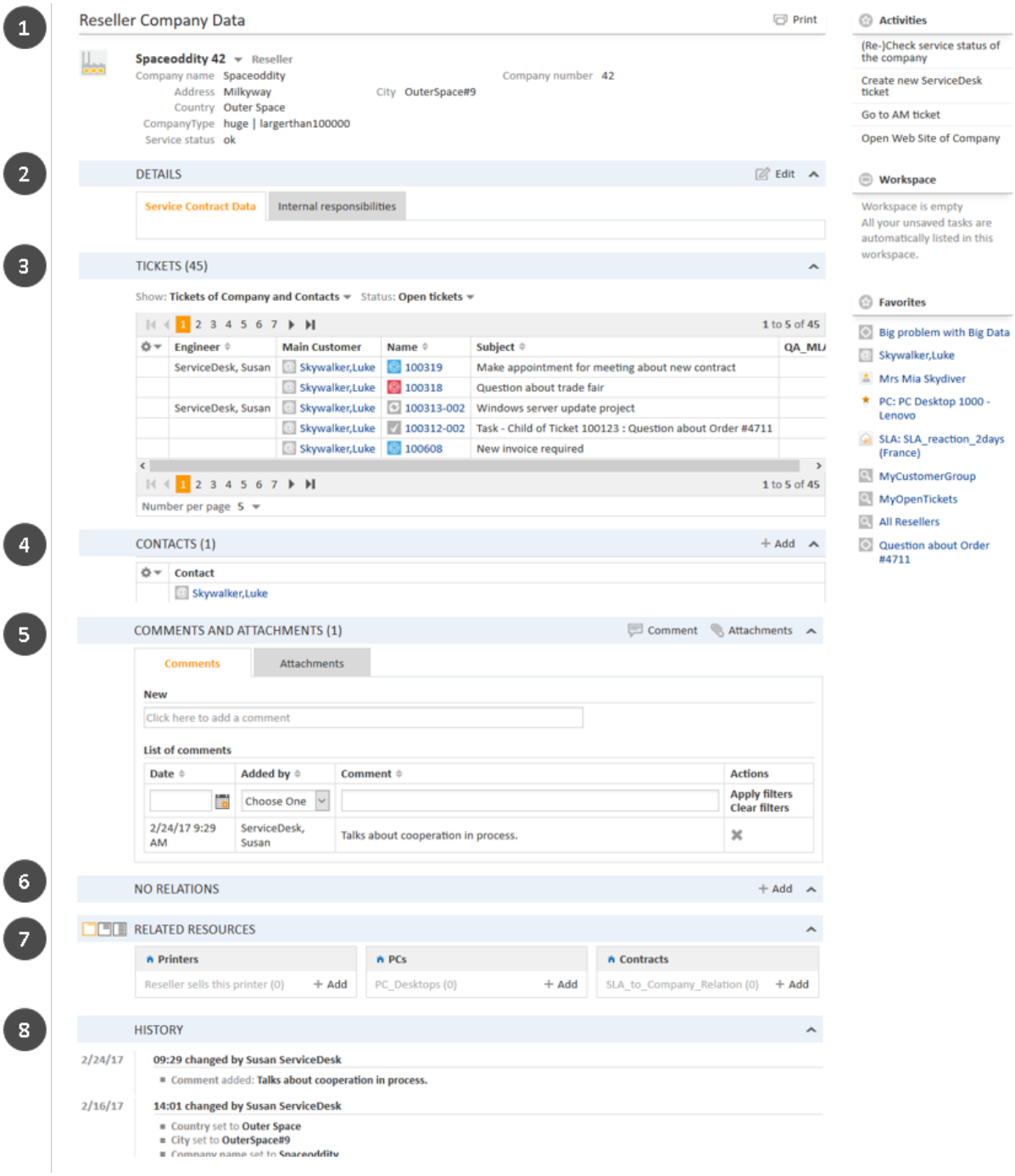


Figure 256: *ConSol CM Web Client - Company page*

The *Company* page contains the following sections:

- **Company data (1, 2)**
  Company data like address, phone number, service data (i.e., the customer fields you have

defined in the customer data model). All company data might be placed in the header section (1) or there might be one or more tabs in the *Details* section (2).
The name of the company is rendered using the Ticket Page template. If this is not defined, the Default template will be used. For details on customer templates, please refer to section Templates for Customer Data, especially subsection Ticket Page.

- **Tickets (3)**
  All tickets of the company and/or its contacts are listed in this section (also see Example 9: Using the Ticket Filters on Company or Contact Pages). Starting with CM version 6.9, it is possible to assign a ticket to a company directly. In the customer data model the option *Company as customer* has to be set to activate this functionality.
- **Contacts (4)**
  This section shows a list of all contacts belonging to this company. Click on a contact name to open the contact page.
- **Comments and Attachments (5)**
  There are two tabs:
  - **Comments**
    All comments concerning this company are listed here.
  - **Attachments**
    All attachments of the company are listed here. The list of attachments can be filtered or sorted based on file type, name, description, date, or engineer.
- **Relations (6)**
  All relations established between this company and other customers (companies or contacts) are listed here.
- **Related Resources (7)**
  This section is only displayed if CM/Resource Pool is activated in the CM system and if at least one resource relation to companies of the respective customer group is configured in the Admin Tool. It shows all resources linked to this company.
- **History (8)**
  All actions which have been performed with this company object are listed here, e.g., the change of a name or any other value of one of the customer fields.

Please refer to the *ConSol CM User Manual* for a detailed introduction of how to work with companies.

Contact Page

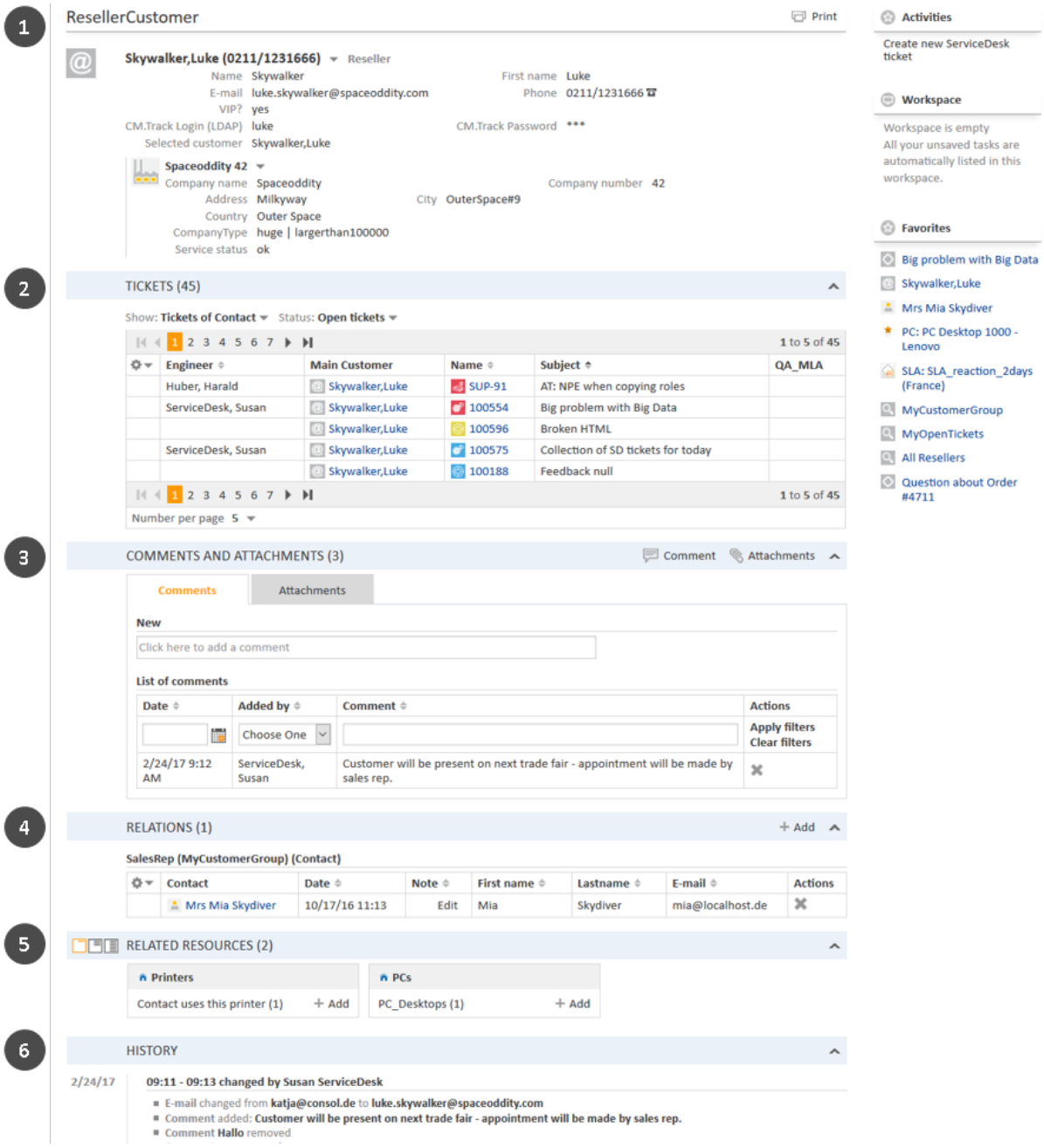


Figure 257: *ConSol CM Web Client - Contact page*

These are the sections on the *Contact* page:

- **Contact data (1)**
  Contact data like address, phone number, service data (i.e., the customer fields you have defined in the customer data model). All contact data might be placed in the header section or there might be one or more tabs in the *Details* section (not shown in figure above).
  The name of the contact is rendered using the Ticket Page template. If this is not defined, the

Default template will be used. For details on customer templates, please refer to section Templates for Customer Data, especially subsection Ticket Page.

- **Tickets (2)**
  All tickets for the contact and/or its company are listed in this section (also see Example 9: Using the Ticket Filters on Company or Contact Pages).
- **Comments and Attachments (3)**
  There are two tabs:
  - **Comments**
    All comments concerning this contact are listed here.
  - **Attachments**
    All attachments of the contact are listed here. The list of attachments can be filtered or sorted based on file type, name, description, date, or engineer.
- **Relations (4)**
  All relations established between this contact and other customers (companies or contacts) are listed here.
- **Related Resources (5)**
  This section is only displayed if CM/Resource Pool is activated in the CM system and if at least one resource relation to contacts of the respective customer group is configured in the Admin Tool. It shows all resources linked to this contact.
- **History (6)**
  All actions which have been performed with this contact object are listed here, e.g., the change of a name or any other value of one of the customer fields, or adding/removing relations, comments or attachments.

Please refer to the *ConSol CM User Manual* for a detailed introduction of how to work with contacts.

Please keep in mind that only engineers who have at least one role with the following access permissions for the respective customer group are allowed to access the Comments and Attachments section of the customer page:

- Details read
- Details write
- Details delete

Also note that only engineers who have at least one role with write permissions for the respective customer group can use the *Change* link in the context menu of the company (on the contact page) to assign the contact to another company.

D.2.2.4 Example 4: Setting a Company as Main Customer of a Ticket

If the configuration option *Company as customer* has been set for a customer data model, a **company can be used as main customer** for a ticket.

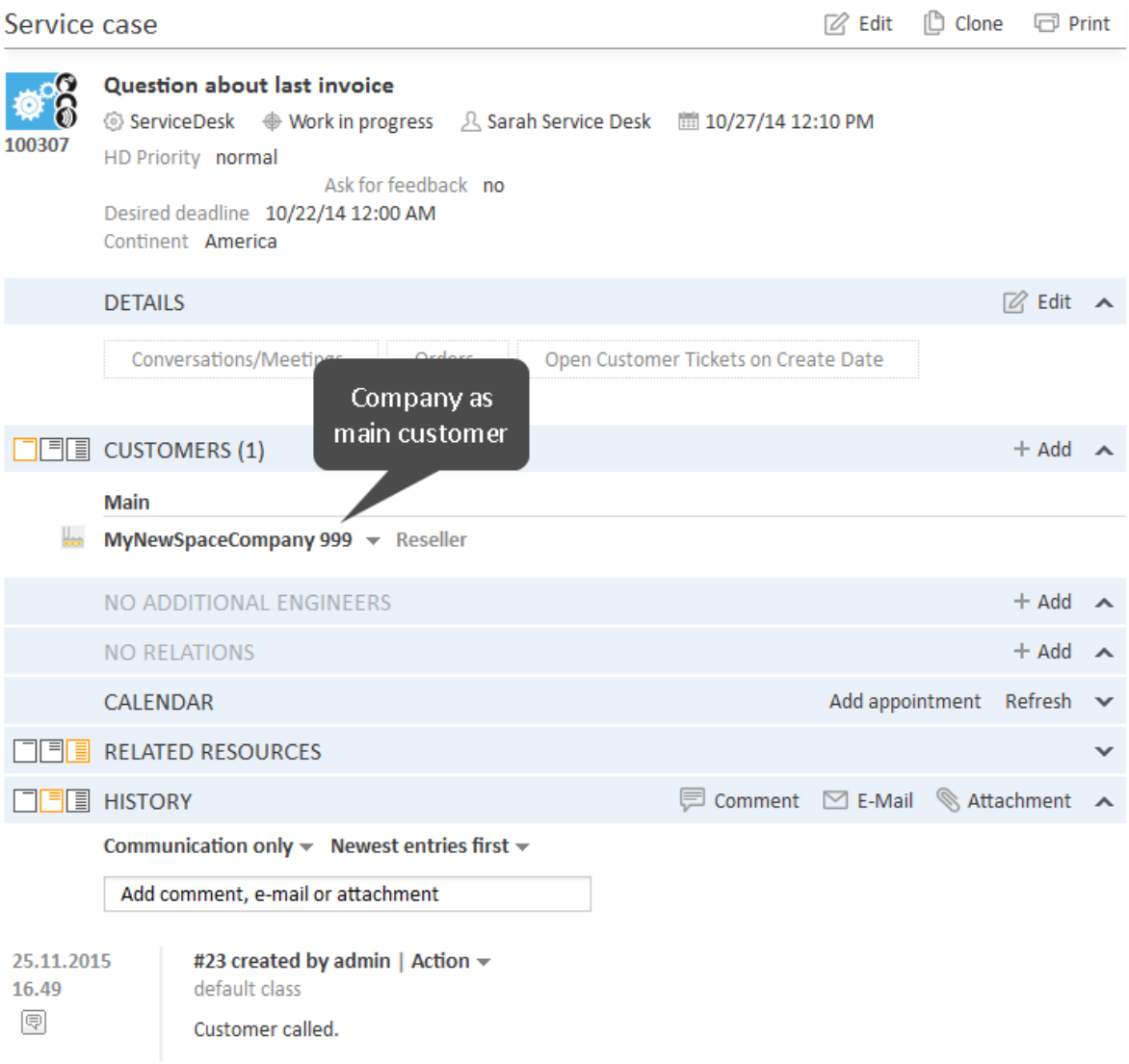


Figure 258: *ConSol CM Web Client - Using the company as main customer for a ticket*

D.2.2.5 Example 5: Using Company and Contact Actions

For companies and contacts, manual and automatic actions can be defined.

Manual actions are triggered using links in the Web Client, very similar to workflow actions (activities) for tickets. In this way, actions concerning the company or the contact data can be performed which are independent of ticket data. For example, an engineer can load the KPIs of the last month for the company, create a new contact within the company (see the following figure), update the contact data from another database, or create a ticket for the contact (see the figure after next).

Automatic actions can be performed when a system action takes place (create/update/delete a customer). The company and contact actions are part of the *Action Framework*.

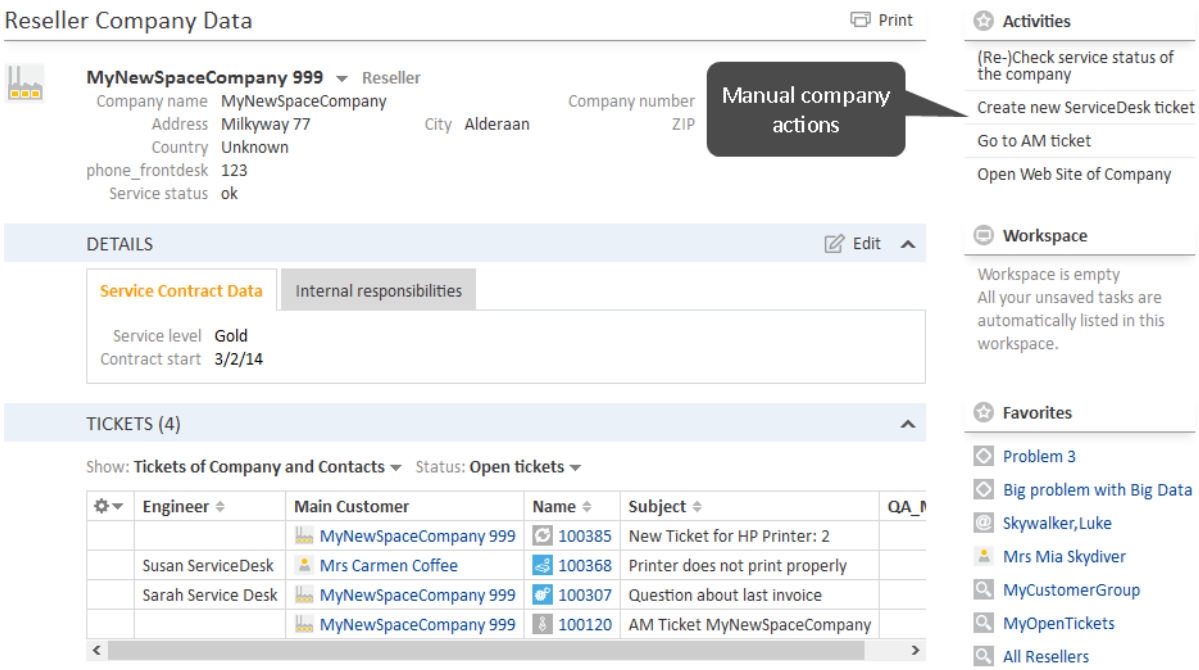


Figure 259: *ConSol CM Web Client - Manual company actions*

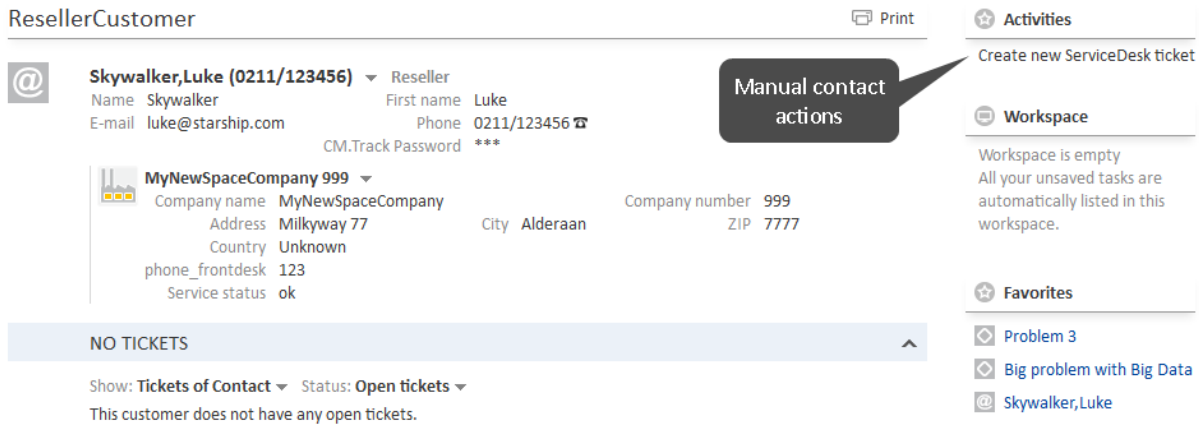


Figure 260: *ConSol CM Web Client - Manual contact actions*

⚠ Please keep in mind that only engineers who have at least one role with the following access permissions for the respective customer group are allowed to use the customer actions, i.e., only then the activities will be displayed in the Web Client:

- Act

D.2.2.6 Example 6: Setting Relations between Contacts and Companies

Particularly when you work with several customer groups it can be important to establish **relations between contacts and/or companies**. For example, your ConSol CM system can then represent a reference *sells products to ...* between a company and a contact. Or a relation *is supervisor of ...* between two contacts. In this way, you can create a network of your companies and contacts and use CM for *Customer Relationship Management* (CRM).

In the Web Client, relations between companies and/or contacts are established and displayed similarly to ticket relations. In the example, *MyNewSpaceCompany* sells products to the end customer *Mr. Sample*.

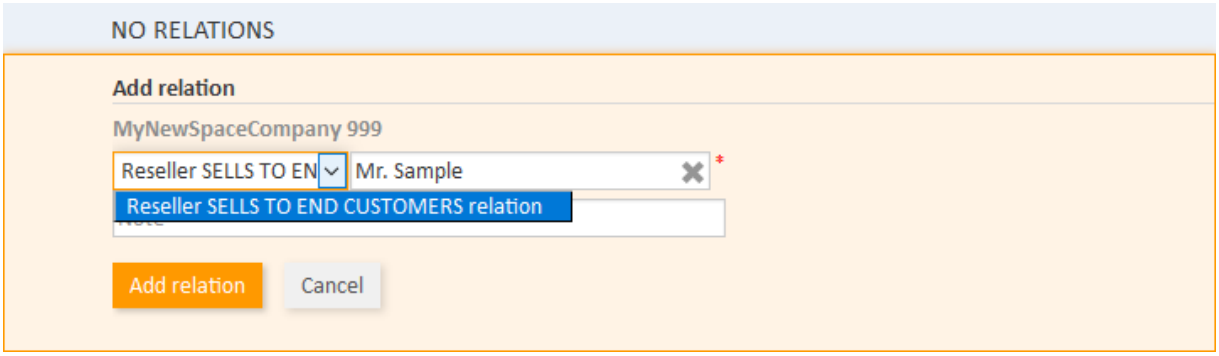


Figure 261: *ConSol CM Web Client - Establishing a company-contact relation*

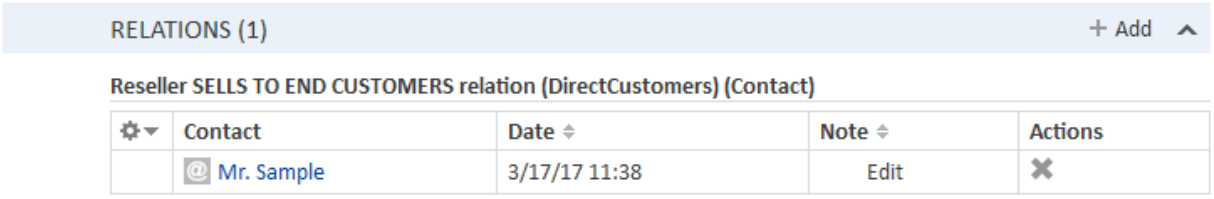


Figure 262: *ConSol CM Web Client - Display of company-contact relation*

D.2.2.7 Example 7: Deactivate a Customer (i.e., a Company or a Contact)

A customer, i.e., **a company or a contact**, **can be deactivated**. This feature might be useful when a contract with a company is no longer valid or when an employee (= contact) has left the company. In this way, the tickets can be kept and retrieved under the **old** contact/company name, but it is not possible to create new tickets for this customer. If the customer has to be deleted, all of their tickets (open and closed) have to be moved. In that case, the former contact-ticket or company-ticket relation is not as easy to find.

The contact or company can only be deactivated if no **open** tickets are assigned to this contact or company.

Please keep in mind that only engineers who have at least one role with the following access permissions for the respective customer group are allowed to deactivate (and reactivate) companies and contacts, i.e., only then will the *Deactivate/Activate* menu items be displayed in the Web Client:

- Deactivate/activate

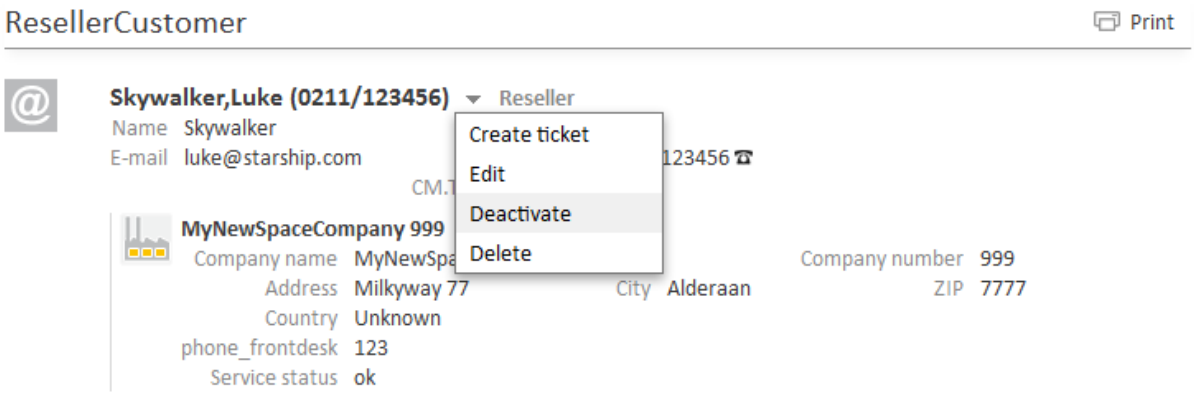


Figure 263: *ConSol CM Web Client - Deactivating a contact*

The following actions **can** be performed for a deactivated customer:

- Edit the customer data (e.g., name, address, phone).
- Delete the customer.
- Transfer the closed tickets to another customer.
- Only if the checkbox *include deactivated customers* is ticked: search the system for the customer using the Detailed Search.

The following actions **cannot** be performed for a deactivated customer:

- Create a new ticket.
- Assign a ticket to this customer.
- Assign a deactivated contact to another company.
- Assign contacts to a deactivated company.
- Search for the customer (deactivated contacts and companies are not shown in search results, except in the Detailed Search when the engineer specifically activates the checkbox *include deactivated customers*).

Deactivation in a two-level customer data model
When a company (or more generally spoken: an object at the company level) is deactivated, all assigned contacts are deactivated automatically.

There are two use cases:

1. **All** contacts of the company **can** be deactivated (no open ticket assigned).
   In this case the company **and** all assigned contacts will be deactivated. Afterwards the company page will be reloaded, company and contact data are marked as deactivated.
2. The company has still contacts which **cannot** be deactivated because of open tickets. Here, the deactivation of a company is **not** allowed. The deactivate option is not selectable.

Reactivation in a two-level customer data model
If a company is reactivated, the assigned contacts will not be reactivated automatically. They have to be reactivated manually.

D.2.2.8 Example 8: Removing Customer Data

There are three options to delete a customer and/or remove customer data from the CM system:

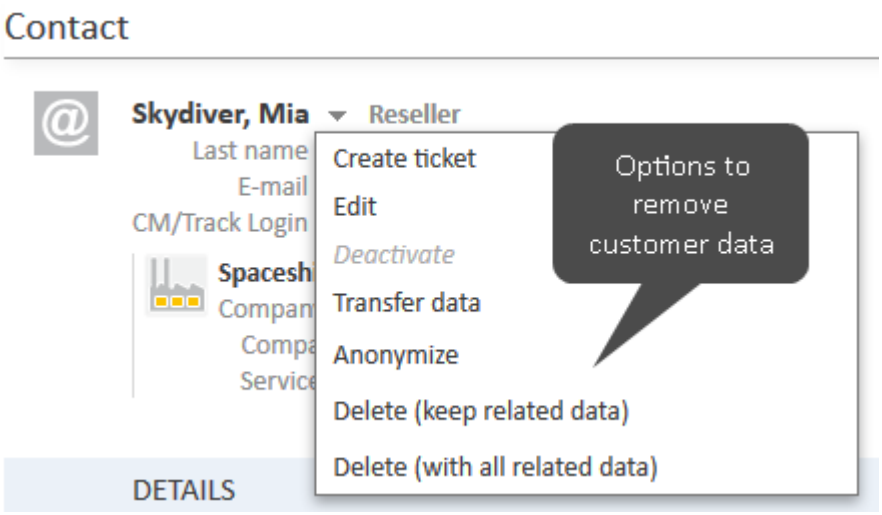


Figure 264: *Contact page with context menu showing delete options*

- **Delete (keep related data) / Delete**

  The customer is deleted, but some references to the customer might remain in the system in history entries of other objects. Before deleting a customer, all his tickets have to be transferred to another customer.

  The option is called *Delete (keep related data)* for contacts and *Delete* for companies.

- **Delete (with all related data)**

  This option is only available for contacts. The contact and all the tickets where the contact is the main customer are deleted. All references to the contact in history entries of other objects are anonymized.

  > This option can be used for GDPR-compliant removal of contact data if the contact and his tickets are not needed for reports.

- **Anonymize**

  The personal data within the contact and within the tickets where the contact is the main customer are removed. Please see ... define a field containing personal data for how personal data is defined. All existing history entries, comments, and attachments of the contact are removed, and an entry stating that the contact has been anonymized is added to the history. The complete history of the contact's tickets is removed and an entry stating that the data has been anonymized is added to the history. Relations to other objects are removed and the contact is anonymized in the history of these objects.

  > This option can be used for GDPR-compliant removal of contact data if the contact and his tickets are still needed for reports.

D.2.2.9 Example 9: Using the Ticket Filters on Company or Contact Pages

On the company and contact pages, ticket filters are available, i.e., **filter options** can be used to display selected tickets for the company or contact.

Ticket Filters on Company Page

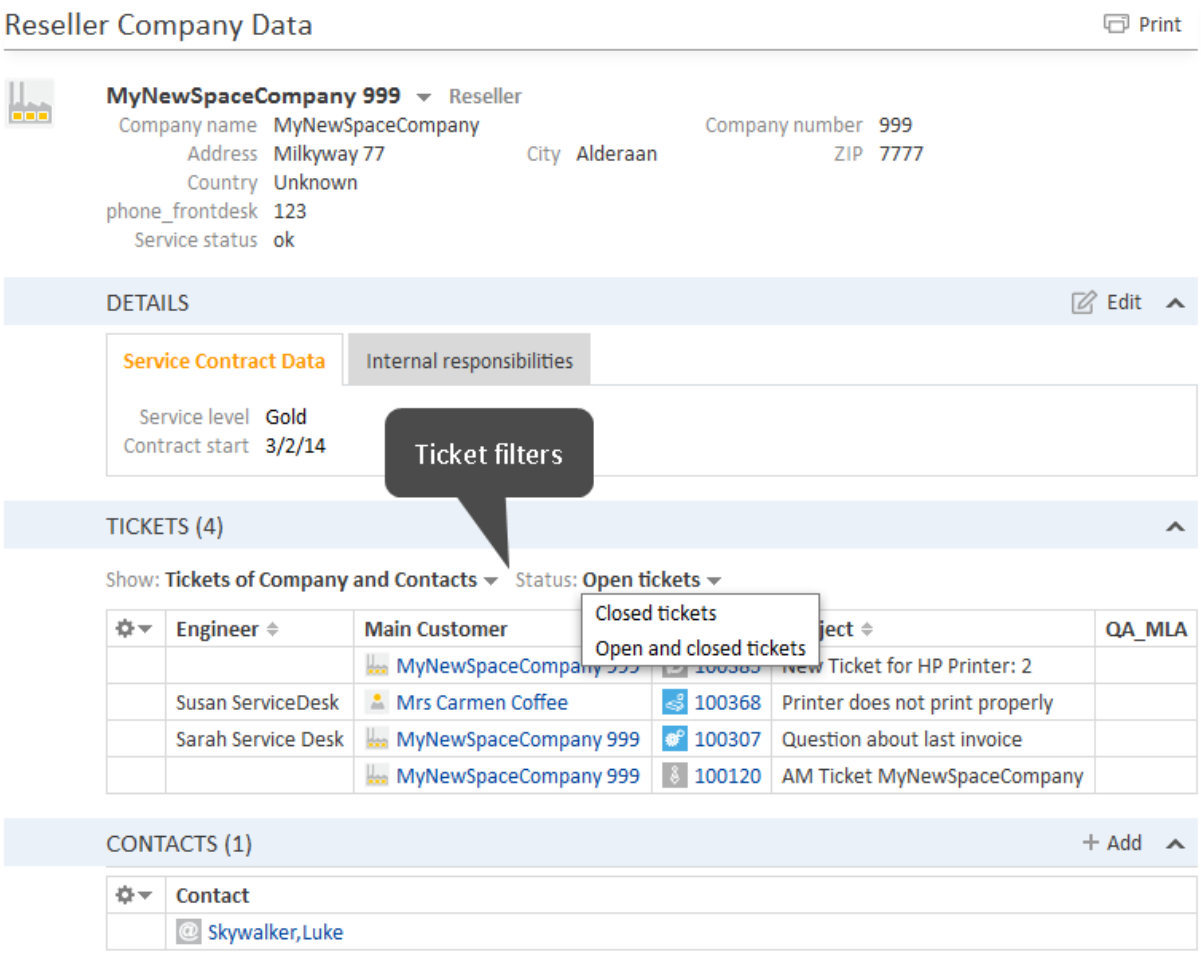


Figure 265: *ConSol CM Web Client - Ticket filters on company page*

The displayed tickets depend on your selection in the two ticket filters:

- **Show**
  - **Tickets of Company:** Tickets where the company is either the main or an additional customer.
  - **Tickets of Company (main customer only):** Tickets where the company is the main customer.
  - **Tickets of Contacts:** Tickets where a contact of the company is the main customer or an additional customer.
  - **Tickets of Company and Contacts:** Tickets where the company or a contact of the company is the main customer or an additional customer.

- **Status**
  - Open tickets
  - Closed tickets
  - Open and closed tickets

Ticket Filter on Contact Page

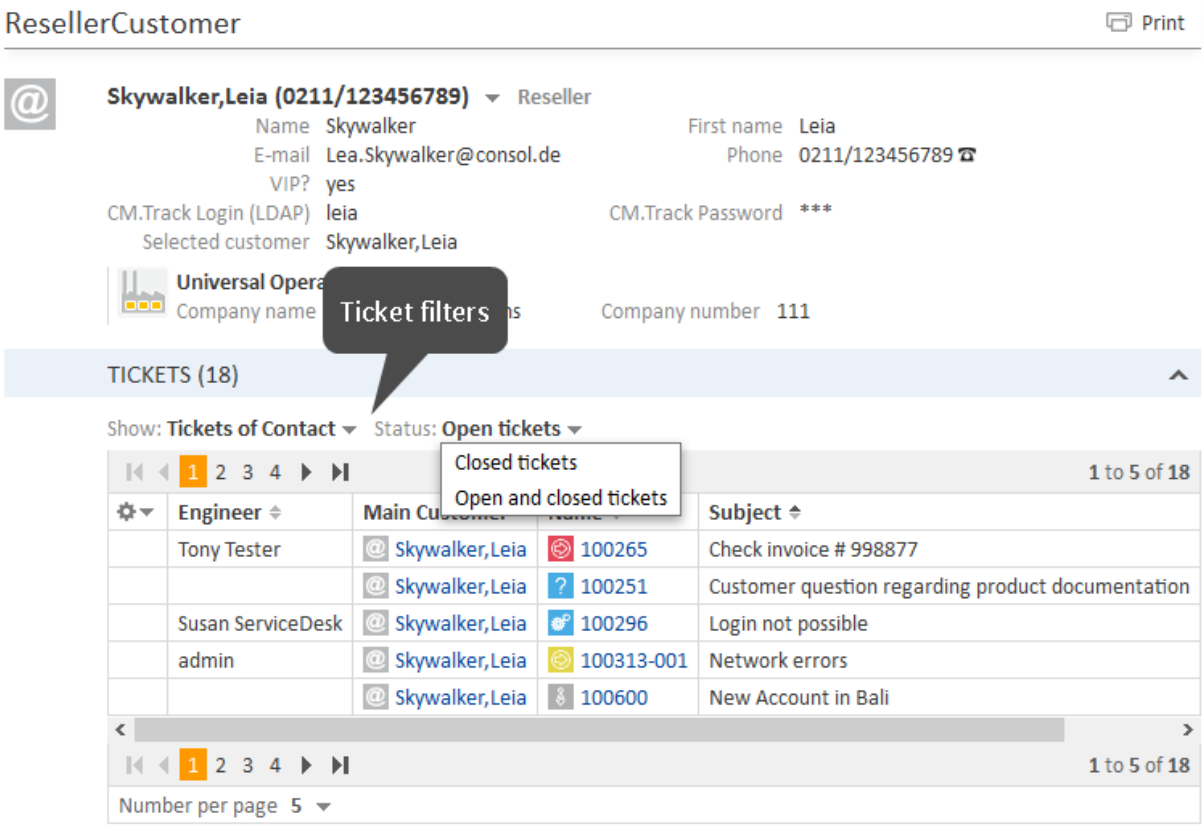


Figure 266: *ConSol CM Web Client - Ticket filters on contact page*

The displayed tickets depend on your selection in the two ticket filters:

- **Show**
  - **Tickets of Contact:** Tickets where the contact is either the main or an additional customer.
  - **Tickets of Contact (main customer only):** Tickets where the contact is the main customer.
  - **All Tickets of Company:** Tickets where the company of the contact, the contact itself or another contact of the company is the main or additional customer.

- **Status**
  - Open tickets
  - Closed tickets
  - Open and closed tickets

D.2.2.10 Example 10: Customer Group Displayed in Quick Search

The **customer group is displayed for all search results** in the list. The following notation is used:

```
<Localized name of customer field group> (<localized name of customer group>)
```

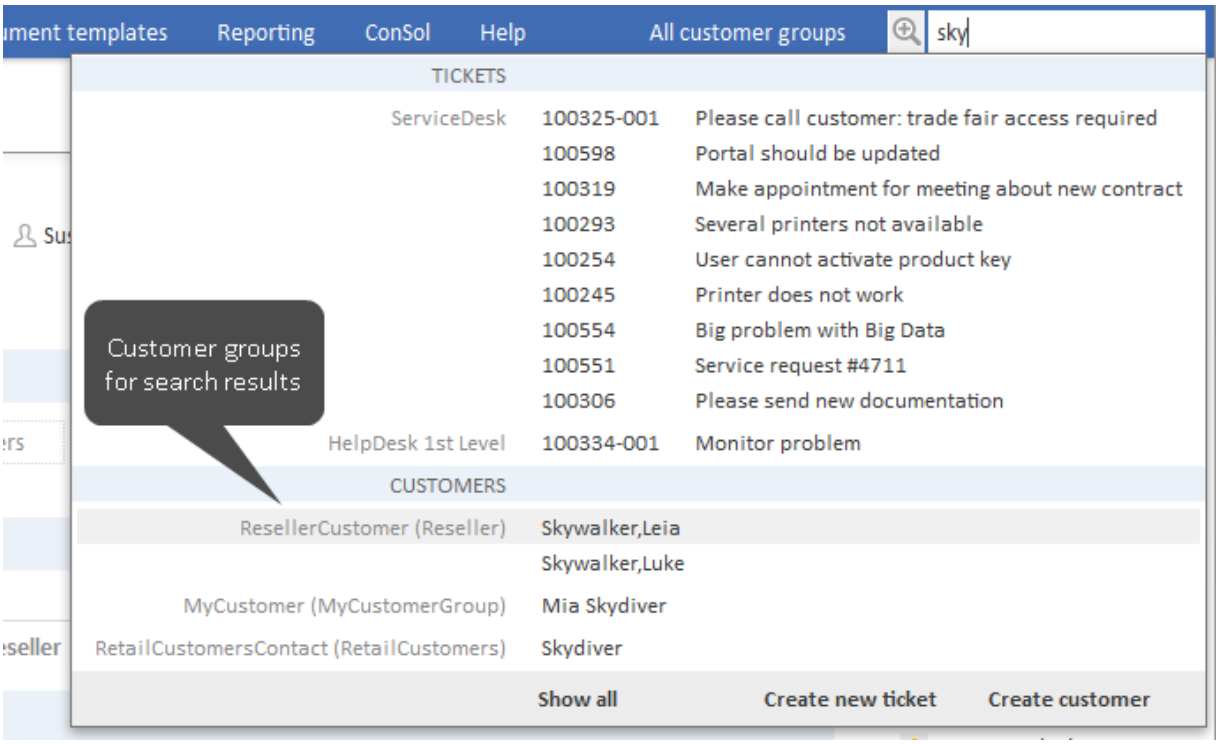


Figure 267: *ConSol CM Web Client - Search results for Quick Search*

D.2.2.11 Example 11: Using Search Actions for Company or Contact Data

See section Action Framework - Search Actions.

D.3 Setting Up the Customer Data Model

This chapter discusses the following:

D.3.1 Introduction to Setting Up the Customer Data Model Based on FlexCDM

With ***FlexCDM*** various customer data models can be implemented. Please refer to section Introduction to FlexCDM for a detailed introduction. To work with a new customer data model within a certain customer group (or in several customer groups), the following steps have to be performed:

1. Create a customer data model.
   (This implies you have already decided whether this should be a one- or a two-level data model. In this example, we will create a two-level model.)
2. Create a new customer group.
3. Assign the customer data model to the group.

A customer data model comprises objects on three model levels:

1. **The customer data model definition**
2. **The customer objects within this model**
   A customer object can be of one of two types:
   a. **Company**
      E.g., an institution, but can also be a machine, a ship, or anything else which represents the company level.
   b. **Contact**
      E.g., a person, but can also be a machine, a hardware device, a product, or anything else which represents the contact level.
      If a company level is present, the contact is a sub-level of the company. For a simple customer data model, use only the contact object or only the company object.
3. **The customer fields**
   These are the data fields for the customer objects, i.e., either the customer fields for company data (e.g., ZIP, address, phone) or the customer fields for contact data (e.g., surname, name, email address).

D.3.2 Managing Contacts and Companies Using the Admin Tool

To manage components of the customer data model, use the items in the navigation group *Customers* in the Admin Tool. Open the navigation item *Data Models* to set up a new data model or to edit existing models.

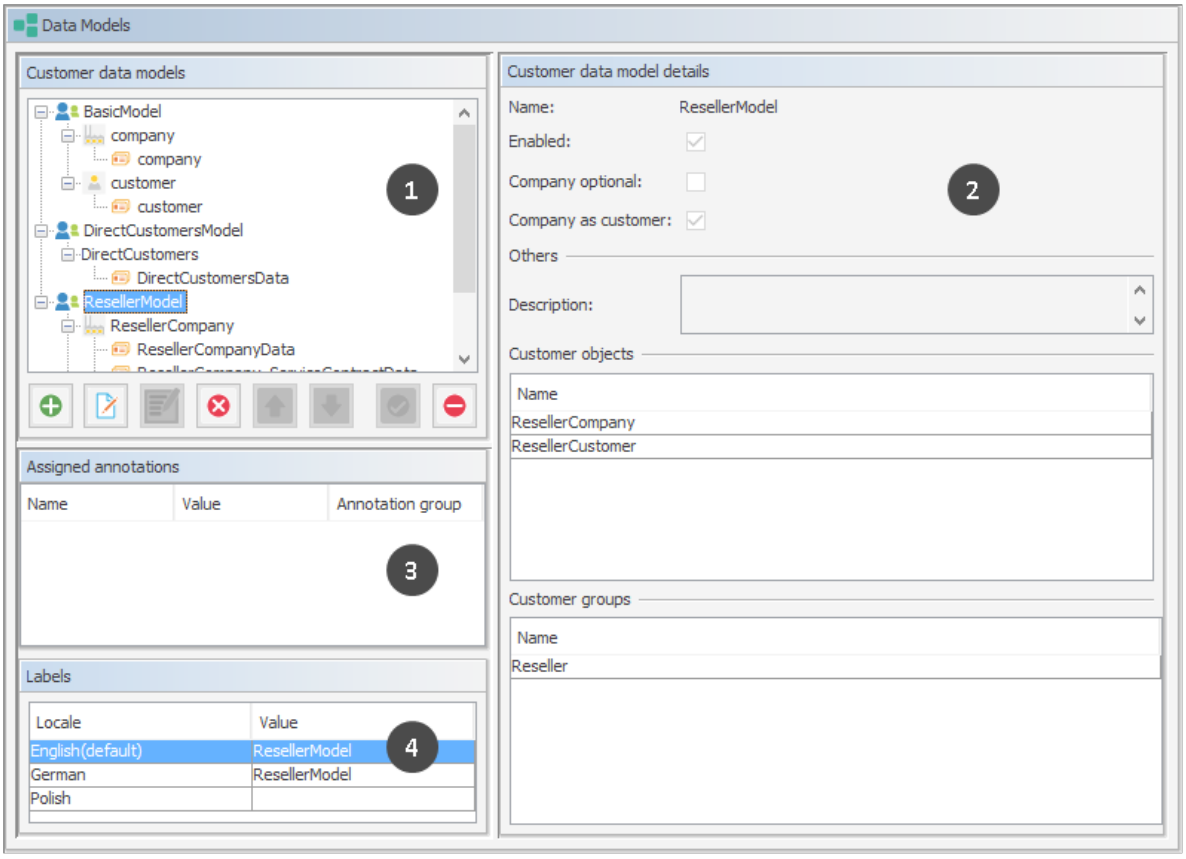


Figure 268: *ConSol CM Admin Tool - Customers, Data Models: Customer data model definition*

The figure shows the following sections:

- tree of all components of all customer data models (1)
- details of the selected component (2)
- annotations for customer field groups of a data model, empty for data models or if no group is selected (3)
- labels for the selected component (4)

To explain how to work with the customer data model, we will walk you through an example in the next sections.

D.3.2.1 Creating a New Two-Level Customer Data Model

To create a new customer data model you have to create the objects on all levels of the data model. In the following example, we will build a customer data model for partner data. We will create a customer data model with a company and a contact object, i.e., we will have to create the following objects:

- the customer data model itself
- the company object (1st level)
- the customer fields for the company
- the contact object (2nd level)
- the customer fields for the contact

After having defined an object, the parameters for this object can be (or rather should be) configured.

Step 1: Create the Customer Data Model with the First Customer Object

When you create a new customer data model, you have to add a customer object and the respective customer fields in one step.

To create a new customer data model, mark another customer data model (that way you select the level on which you want to work) and use the *Add* button to open the pop-up window.

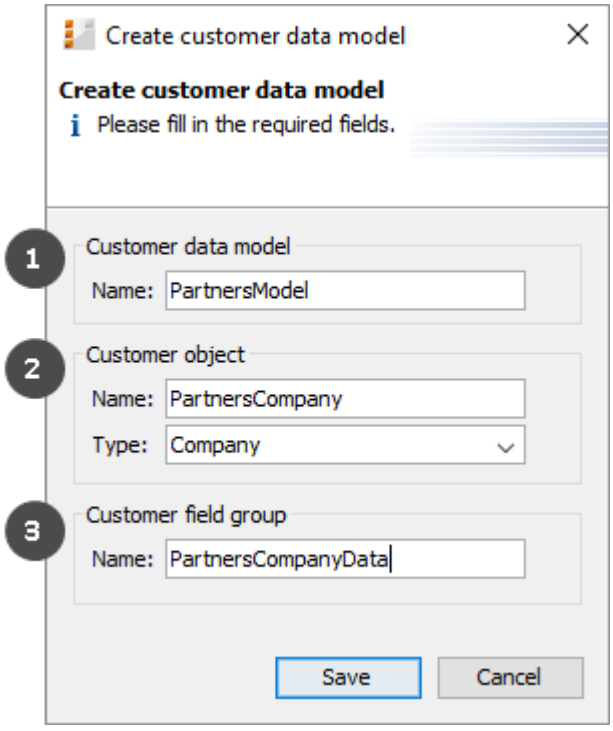


Figure 269: *ConSol CM Admin Tool - Customers, Data Models: Creating a new customer data model*

You have to fill in the following fields:

- **Customer data model (1)**
  - **Name**
    The name of the new customer data model. The localized names for the data model are set in the *Labels* section of the main *Data Models* panel. For details, please refer to section Localization of Data Fields.
- **Customer object (2)**
  - **Name**
    The unique technical name of the company/contact object. The localized names for the customer object are set in the *Labels* section of the main *Data Models* panel. For details, please refer to section Localization of Data Fields
  - **Type**
    Select *Contact* or *Company*. There can be only one company object and one contact object within one customer data model.
- **Customer field group (3)**
  - **Name**
    The unique technical name of the first customer field group for company data within the defined customer object. More customer field groups can be added later on. The localized names for the customer field group are set in the *Labels* section of the main *Data Models* panel. For details, please refer to section Localization of Data Fields

Step 2: Create Another Customer Object

In the next step, you have to add the contact object. Select the object *PartnersCompany* (to set the correct level for the following *Add* operation) and click the *Add* button to open the pop-up window.

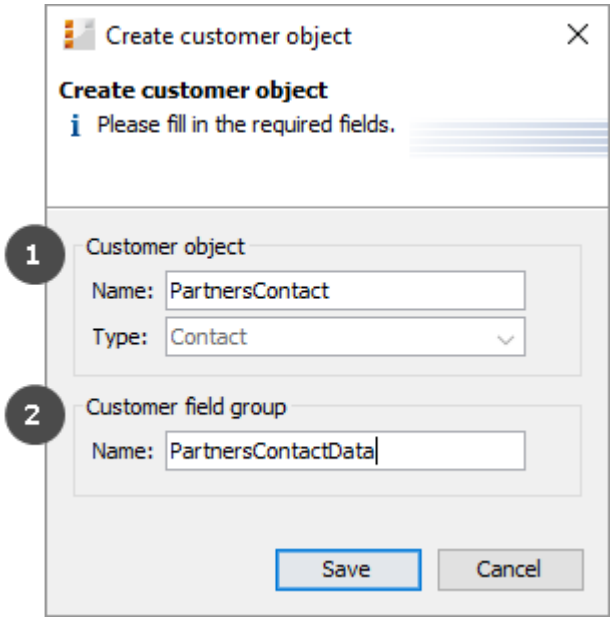


Figure 270: *ConSol CM Admin Tool - Customers, Data Models: Adding a new customer object*

You have to fill in the following fields:

- **Customer object (1)**
  - **Name**
    The unique technical name of the contact object. The localized names for the customer object are set in the *Labels* section of the main *Data Models* panel. For details, please refer to section Localization of Data Fields. The localized name will also be used as header of the customer page (i.e., contact or company page, see section Example 3: Using Company and Contact Page).
  - **Type**
    Here, *Contact* is pre-selected and cannot be modified, because a company object is already present in the customer data model.
- **Customer field group (2)**
  - **Name**
    The unique technical name of the first customer field group for contact data within the defined customer object. More customer field groups can be added later on. The localized names for the customer field group are set in the *Labels* section of the main *Data Models* panel. For details, please refer to section Localization of Data Fields

Step 3: Configuring the Parameters for the Defined Objects

Parameters for the Customer Data Model

Double-click on the name of the customer data model (*PartnersModel* in our example) or mark the customer data model in the list and click the *Edit* button to open the pop-up window where you can define the parameters for the model.

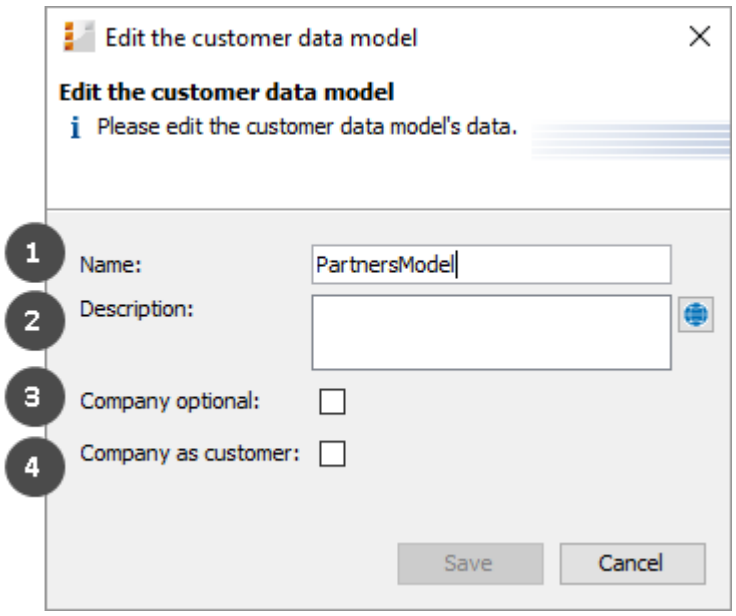


Figure 271: *ConSol CM Admin Tool - Customers, Data Models: Parameters for a customer data model*

You can fill in the following fields:

- **Name (1)**
  Check or modify the existing name of the model.
- **Description (2)**
  Optional description. Can be localized. Please see section Localization of Objects in General, Type 1 for details.
- **Company optional (3)**
  If this checkbox is checked, it is possible to add a contact to a ticket without the contact being part of a company. The company might be set later but it is not required. So, here you can enable working with single contacts, even within a two-level customer data model.
- **Company as customer (4)**
  Check this checkbox if it should be allowed to create tickets not only for contacts, but also for companies within the model.

Parameters for the Customer Object

Double-click on the name of a customer object, e.g., the *PartnersCompany* or select the object and click the *Edit* button to open the pop-up menu where you can configure the parameters for this object.

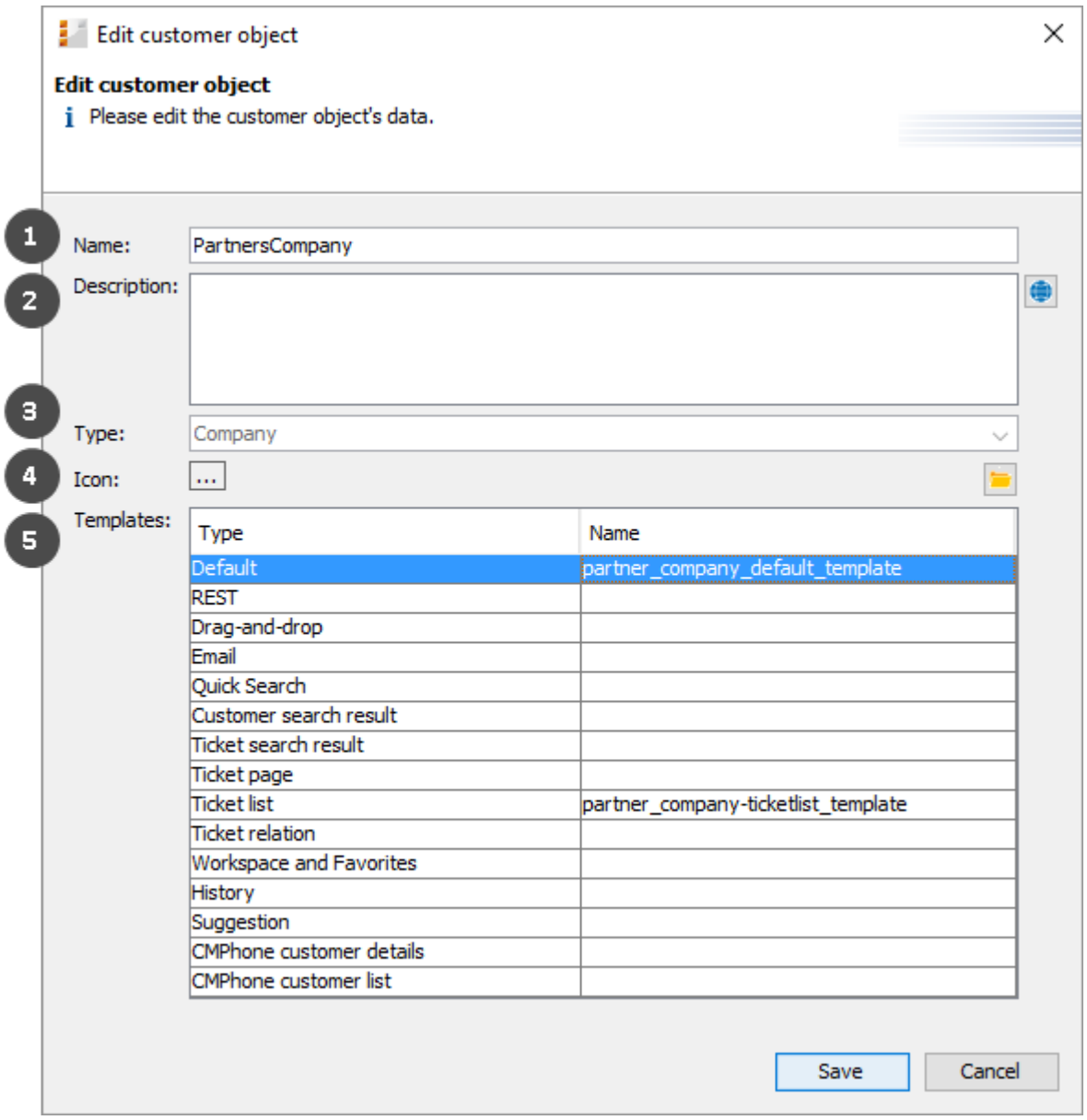


Figure 272: *ConSol CM Admin Tool - Customers, Data Models: Parameters for a customer object*

You can fill in the following fields:

- **Name (1)**
  The unique technical name of the customer object with technical name and localized name(s). The localized name will also be used as header of the customer page (i.e., contact or company page, see section Example 3: Using Company and Contact Page).
- **Description (2)**
  The description of the customer object. Will be used in future ConSol CM versions. Can be localized. Please see section Localization of Objects in General, Type 1 for details.
- **Type (3)**
  A read-only field which displays the type (*contact* or *company*) of the customer object.

- **Icon (4)**
  The icon for all companies within this model. It will be displayed in the Web Client. You can either use one of the standard CM icons using the button (...) or upload an icon using the *File explorer* button.
- **Templates (5)**
  The templates which are used to render the data of the customer object, i.e., the templates which define the data fields that are displayed in the Web Client for objects of this type. There are various positions in the GUI for which the layout of the company or contact data can be defined. The templates are stored in the *Script and Template* section of the Admin Tool. Please see section Templates for Customer Data for a detailed explanation.

Parameters for the Customer Field Group

Double-click on the name of the customer field group to change the technical name (only possible when respective fields in the customer data sets are empty) or to assign/unassign Dependent Enum Scripts for the group.

To define the customer fields, use the GUI elements on the right-hand side. In the following figure, an example for the definition of customer fields for the *PartnersCompany* is shown. Here, only one customer field group (*PartnersCompanyData*) is used. You can use as many customer field groups as you need in one customer object.

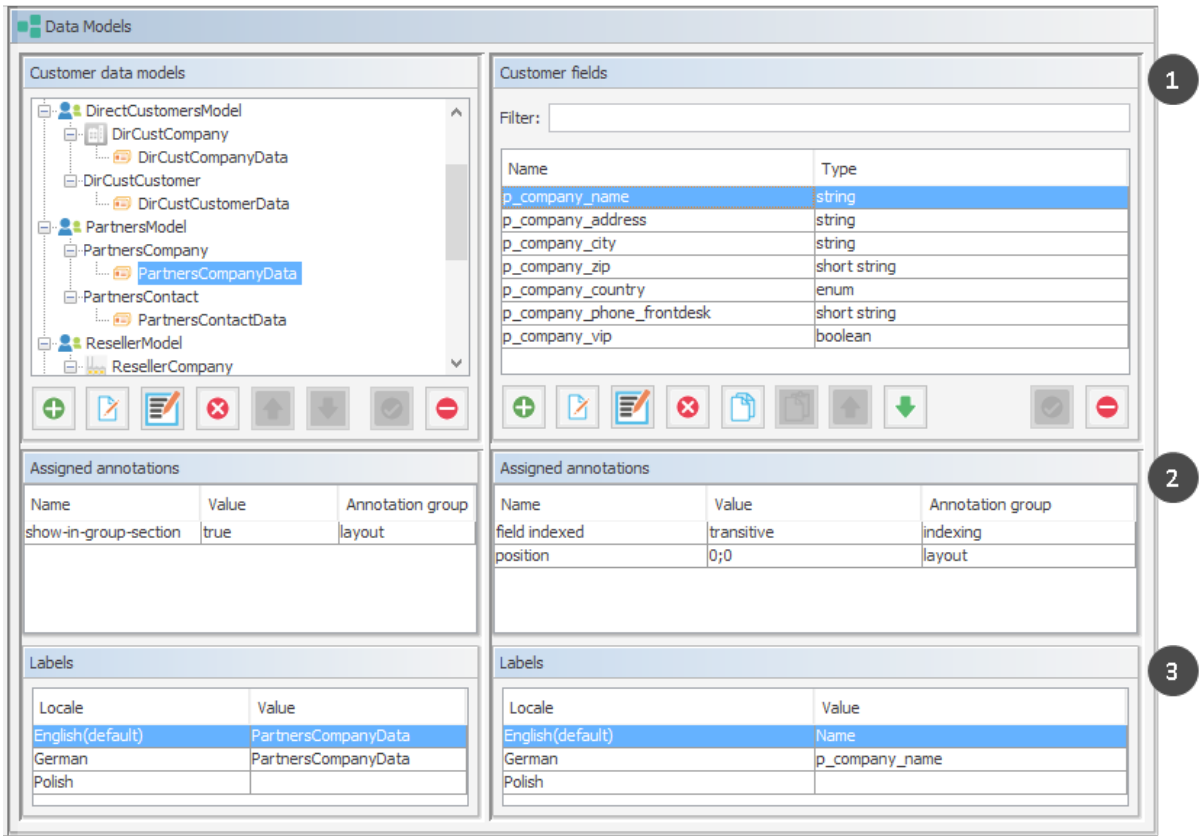


Figure 273: *ConSol CM Admin Tool - Customers, Data Models: Parameters for the customer field group*

The figure shows the following sections:

- list of customer fields in the selected customer field group (1)
- field annotations for the selected customer field (2)
- labels for the selected customer field (3), i.e., the localized values for the name of the customer field

The definition of customer fields within customer data models is based on the same principles as the definition of ticket fields for ticket data. For a detailed introduction to the definition and management of ticket fields, please refer to sections Ticket Field Administration (Setting Up the Ticket Data Model) and Customer Field Management and GUI Design for Customer Data.

The available customer field annotations are listed in section Annotations. The annotation `unit is a contact` is no longer in use because the level of a unit (i.e., the company or contact) is defined by its unit type (*company* or *contact*).

⚠ Please make sure that the annotation `field indexed` is set for all fields which should be searchable. This affects the Quick Search, Detailed Search, and all auto-complete operations! See also section Search Configuration.

Congratulations! When you have completed all the steps in the previous sections, you have created a new ConSol CM customer data model and can now go on to assign the model to one or more customer group(s).

D.3.2.2 Creating a New Customer Group Using the New Customer Data Model

When the customer data model has been defined, it can now be assigned to one or more customer groups. In the example, we will create the new customer group *OurPartnerCompanies* which will use the new *PartnersModel*.

Use the navigation item *Customer Groups* in the navigation group *Customers* of the Admin Tool to create a new customer group and to assign the desired customer data model.

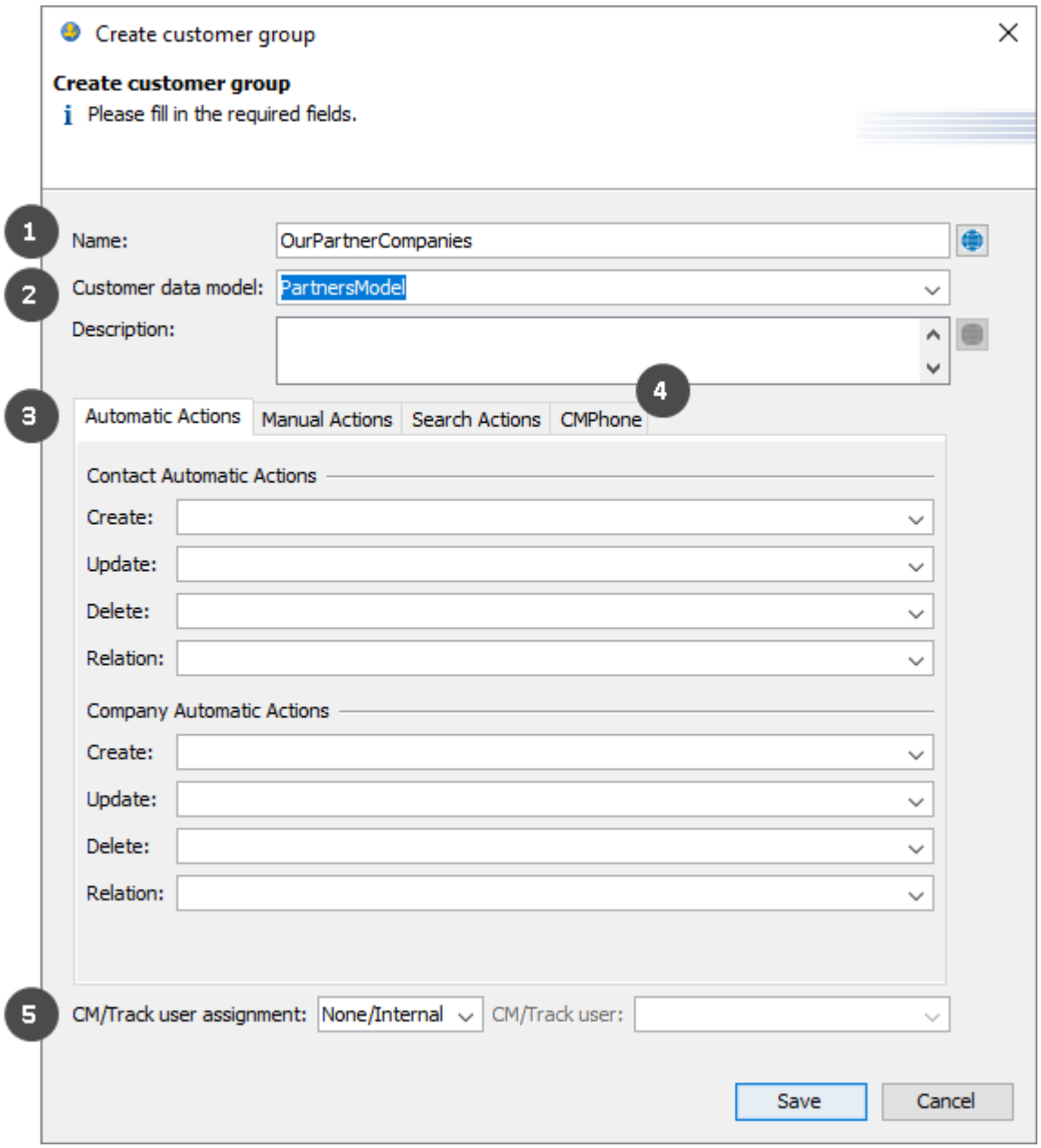


Figure 274: *ConSol CM Admin Tool - Customers, Customer Groups: Definition of a new customer group*

You can fill in the following fields:

- **Name (1)**
  The unique technical name (and localized name) of the new customer group. Can be localized. Please see section Localization of Objects in General, Type 1 for details.
- **Customer data model (2)**
  Select the desired data model from the drop-down menu.

- **Automatic/Manual/Search Actions (3)**
  On those tabs you can assign customer actions. This is explained in detail in section Action Framework - Customer Actions.
- **CMPhone (4)**
  This tab will only be displayed if CM/Phone is installed. Please see section CM/Phone: CTI with ConSol CM for details.
- **CM/Track user assignment (5)**
  Select the assignment mode of the CM/Track user profile. Please see section Defining the User Assignment Mode for further information.

An engineer who has access permissions for six customer groups will see the following in the Web Client.

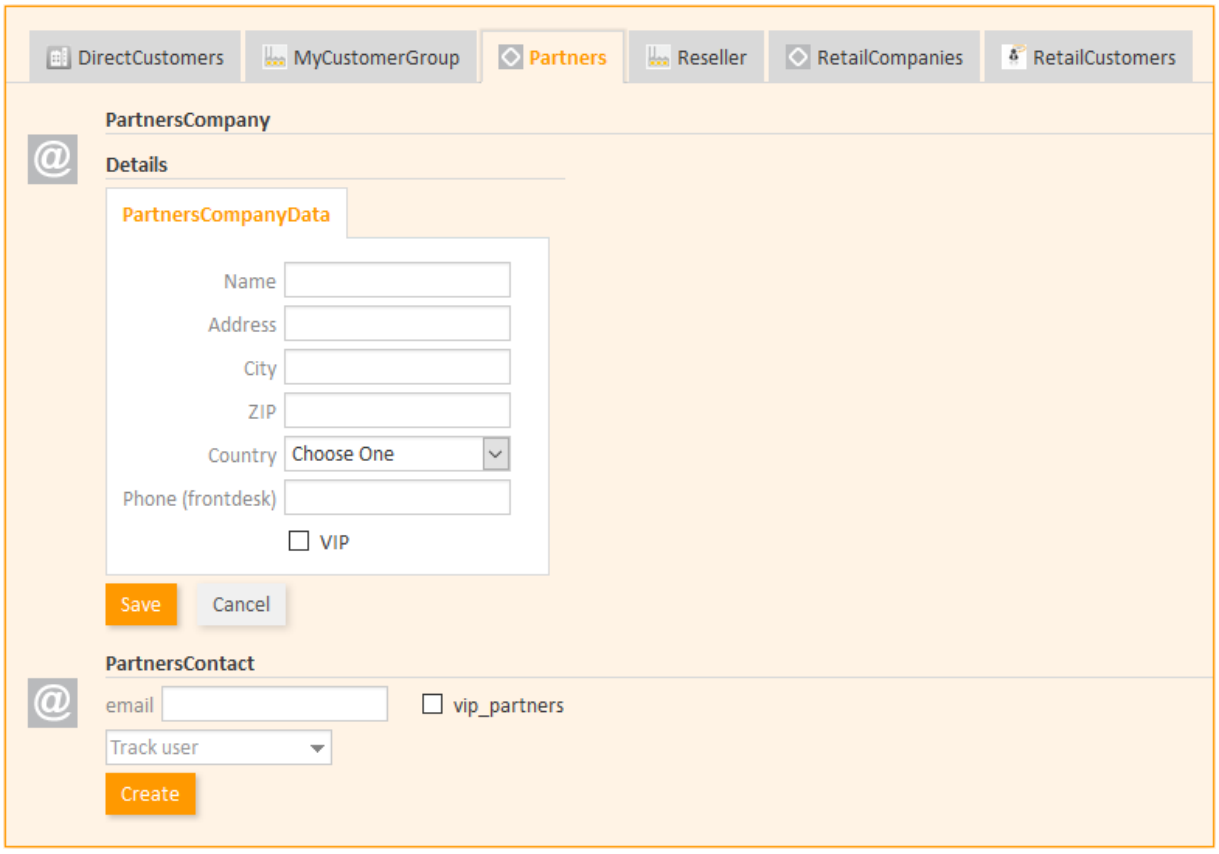


Figure 275: *ConSol CM Web Client - Creating a new company and contact*

D.3.2.3 Assigning Access Rights for Customer Groups with the New Model to Roles

In order to let engineers work with customer data from the new customer group, i.e., to create new partner data sets or to modify existing sets, you have to grant access permissions for the customer groups to one or more roles.

See section Tab Customer Group Permissions.

D.3.2.4 Assign the New Customer Groups to Queues

Please keep in mind that you have to assign the new customer group to all queues where tickets should be created for customers of this group. See section Queue Administration for details.

D.3.2.5 Deactivate Objects in the Customer Data Model

The following objects within a customer data model can be deactivated (this functionality is implemented only very rudimentary in current CM versions):

- an entire customer data model
- an object on company level - currently (as of CM version 6.11) no effect
- an object on contact level - currently (as of CM version 6.11) no effect
- a customer field group - currently (as of CM version 6.11) no effect

In order to deactivate a certain object, select the object on the desired level (be careful to distinguish between the entire data model, the company or contact level, and the customer field groups).

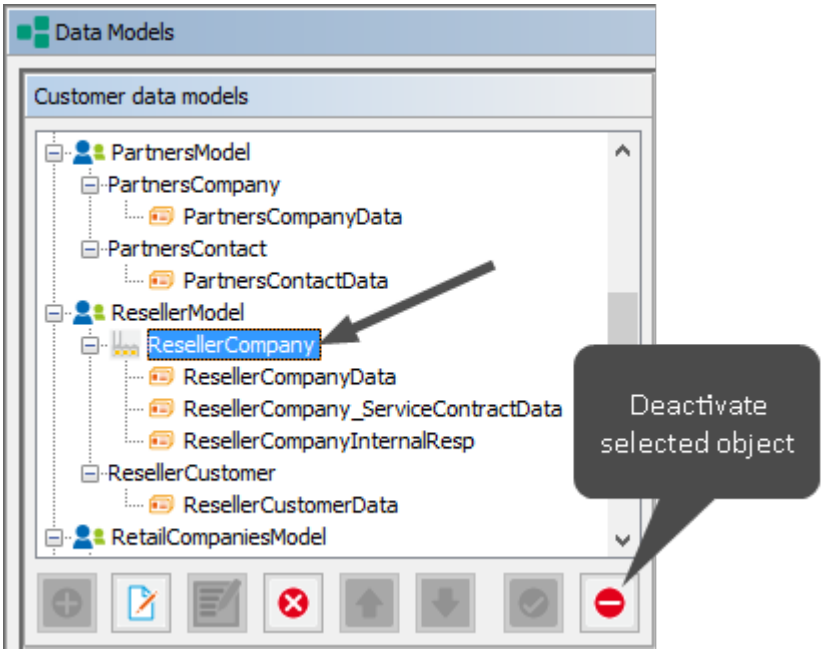


Figure 276: *ConSol CM Admin Tool - Customers, Data Models: Deactivate a FlexCDM object (here: company object in ResellerModel)*

D.4 Customer Field Management and GUI Design for Customer Data

This chapter discusses the following:

D.4.1 Introduction

One feature of ConSol CM versions 6.9 and later is great flexibility as far as customer data model (FlexCDM) and GUI design are concerned. You, as an administrator, can define any data field which is required and place it in the user interface wherever it is suitable. The basic principle is now the same as the one you know for ticket fields: full flexibility.

The management of the ticket data model and GUI design is explained in section Ticket Field Administration (Setting Up the Ticket Data Model). The management of objects within the customer data model is explained in section Setting Up the Customer Data Model. Please refer to those sections for a detailed explanation. In this chapter, we assume that you have a working knowledge of those topics.

D.4.2 Defining Customer Fields for Customer Data Using the Admin Tool

D.4.2.1 Admin Tool GUI

The data field definition for customer data is part of the definition of the entire customer data model, see section Setting Up the Customer Data Model. The data model is defined on the navigation item *Data Models* in the navigation group *Customers*, in the Admin Tool.

Data fields for customer objects within the customer data model are called *customer fields*. Customer fields are based on the same principles as ticket data fields (ticket fields): data fields are managed in groups and the groups, as well as the single fields, can be annotated.

You reach this screen by opening the navigation item *Data Models* in the navigation group *Customers*.

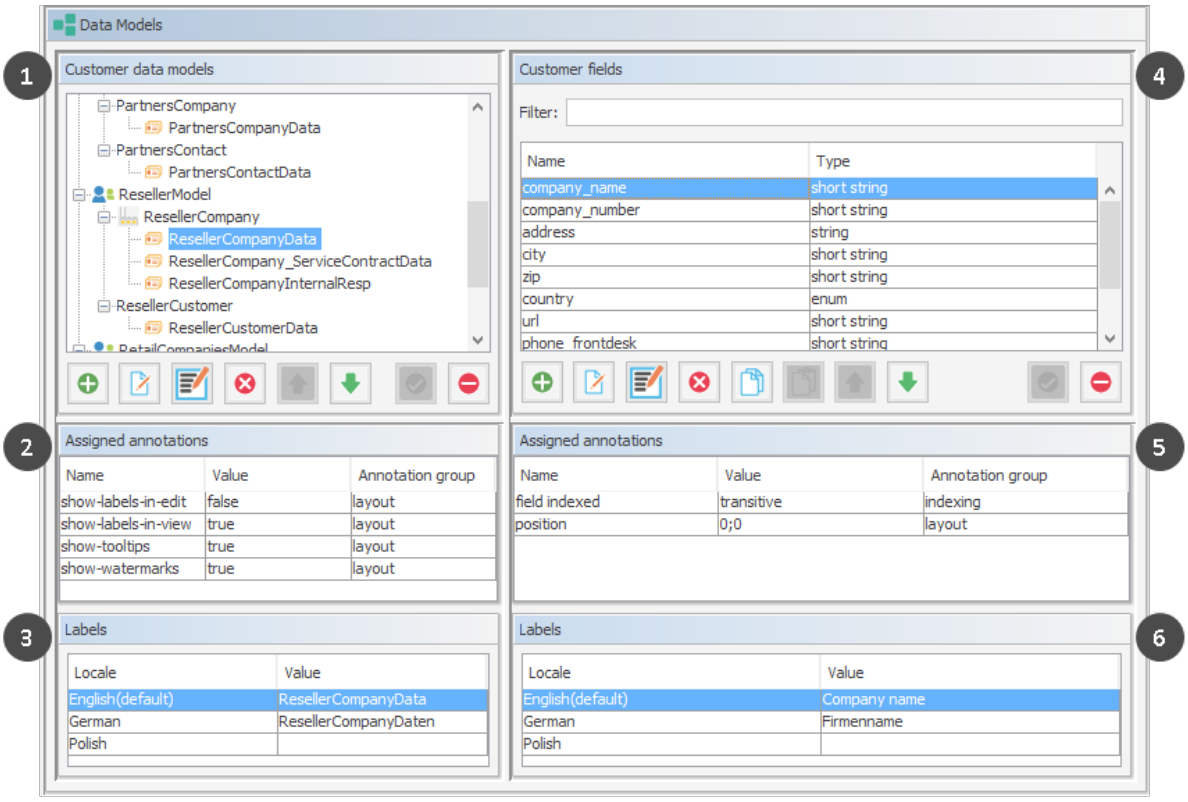


Figure 277: *ConSol CM Admin Tool - Customers, Data Models: Definition of customer fields*

The *Data Models* screen consists of six sections:

- Customer data models (1)
- Annotations for the selected customer field group (2)
- Labels for the selected customer field group (3), i.e., the localized values for the name of the customer field group
- Customer fields of the selected customer field group (4)
- Annotations for the selected customer field (5)
- Labels for the selected customer field (6), i.e., the localized values for the name of the customer field

D.4.2.2 Customer Field Groups

Like the ticket fields which you are already familiar with from previous CM versions, customer fields are placed in groups, the *customer field groups*. Each customer object within a customer data model can have as many customer field groups as required. For example, for a reseller company there can be a customer field group for the general data, one for the contract data, and one for the persons who are responsible for this reseller. For contacts within the reseller data model, one customer field group with general data is defined.

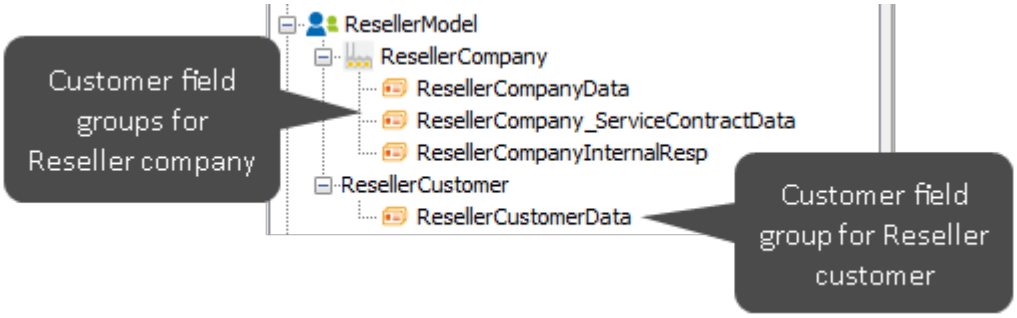


Figure 278: *ConSol CM Admin Tool - Customers, Data Models: Customer data model with several customer field groups*

The organization of data fields in groups has several implications. Please keep them in mind to make sure your data model design meets the users' needs.

A customer field group ...

- can be faded in and out in the GUI during the process, but only the whole group, not single fields (= customer fields).
- can be displayed as a tab or in the customer data section. The title (and mouse-over) of the tab is the (localized) name of the customer field group. For details about localization of customer field groups, please refer to section Localization of Data Fields.
- is configured by group annotations.
- is placed in the GUI based on its position in the customer field group list (defines, e.g., the order of tabs).

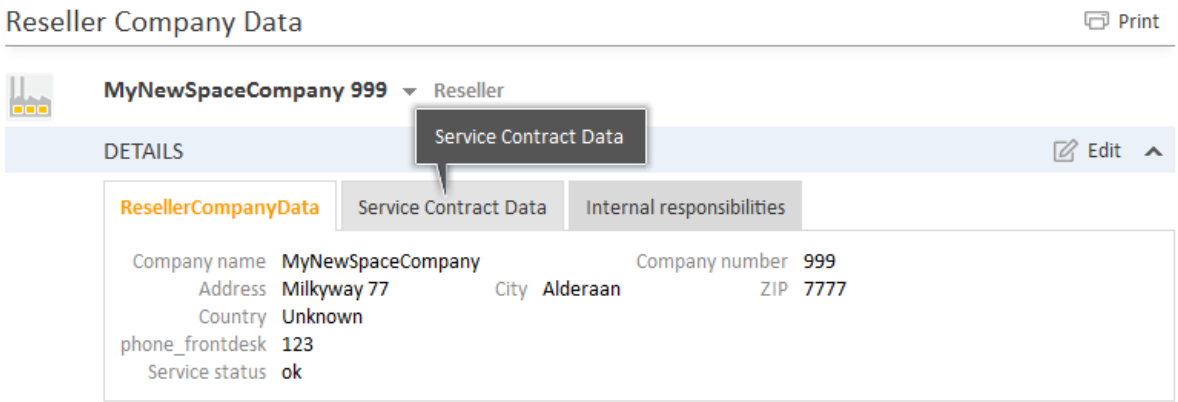


Figure 279: *ConSol CM Web Client - Company data organized in tabs (based on customer field groups)*

D.4.2.3 Customer Field Definition

The definition of customer fields (i.e., data fields like *name*, *address*, or *phone number*) is based on the proven concepts which have been used for ticket fields since the first CM6 version.

A customer field ...

- is defined by a data type.
- is configured using field annotations (e.g., `position` or `field-indexed`), as described in Annotations.

> In contrast to ticket fields and ticket layout, where you can use three columns for data fields, you can use as many data field columns as you like for the definition of customer fields.
>
> Please keep in mind that in CM versions 6.10.4 and below, the Business Card Feature will be active, i.e., in order to present all customer data aligned as on a business card, all fields of a line are presented left-aligned (i.e., starting with column #0), even if the first field in the line has the position column #2 (or other) according to the position annotation.

The following data types are available for ticket fields, customer fields and resource fields.

- **autocomplete**
  A data field which contains a scripted autocomplete list. This is a dynamic list which is based on a script of type *Text Autocomplete*. A detailed explanation of scripted autocomplete lists is given in section Scripted Autocomplete Lists.
- **boolean**
  Values: true/false. Depending on the annotation `boolean-type`, the value is displayed as checkbox, radio buttons, or drop-down list.

  > If you work with scripts, either in CM workflows or in the Admin Tool, please note that the behavior of boolean fields which are represented as checkboxes, i.e. with annotation `boolean-type = checkbox` (default) is different depending on the CM version!
  >
  > - **In CM versions prior to 6.9.4.0:**
  >   If a boolean field has not been touched, its value is “false”. If it is checked, its value is “true”, and if it is unchecked again, its value is “false ”again.
  > - **In version 6.9.4.0 and up:**
  >   If a boolean field has not been touched, its value is “NULL”. If it is checked, its value is “true”, and if it is unchecked again, its value is “false”.
  >
  > Fields which have already been filled with values in the database will not be changed during an update from a version prior to 6.9.4.0 to a version 6.9.4.0 and up.
  >
  > Boolean fields represented as radio buttons (annotation `boolean-type = radio`) or drop-down menu (annotation `boolean-type = select`) are always shown and behave as described for versions 6.9.4.0 and up, i.e., with “NULL ”value if untouched.

- **date**
  Format and accuracy can be set by annotations.

- **enum**
  For sorted lists. The engineer can choose one of the enum values in the Web Client. Enums and values have to be created previously within the Managing Sorted Lists: Enum Administration. Select the desired *Enum type* and *Enum group* in the fields below.
- **list**
  A data field of this data type is the first step to creating a list (one column) or a table (multiple columns) of input fields in the Web Client.
  - For a **table** the next step will be to create another field of type *struct* (see below) to contain the input of the individual list fields (which will become the columns of the table). So, if you want to create a table you have to define a field of the type *struct* first (see below) before you can add the fields for the table columns.
  - For a **simple list**, the next step will be to create fields which belong to the list. No *struct* is required.

  For all fields belonging to a list or table you have to set the dependencies in the field *Belongs to* (see below). For example, a table field (which is a regular data field) always belongs to a *struct*, a struct always belongs to a *list.*
- **struct**
  A data field of this type defines a data structure (line of a table) which groups one or multiple fields. It is the second step to building a table after you have created a field of the type *list*. Add the fields for the columns of the table in the next step. The dependencies have to be set for each field in the *Belongs to* field (see below), i.e., a *struct* always belongs to a *list*.

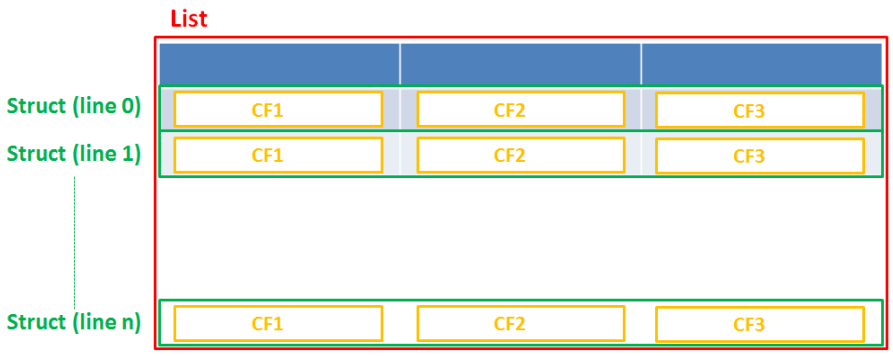


Figure 280: *Scheme: List of structs*

Technically spoken, the list is an array which contains a map (= key:value pairs) in each field.

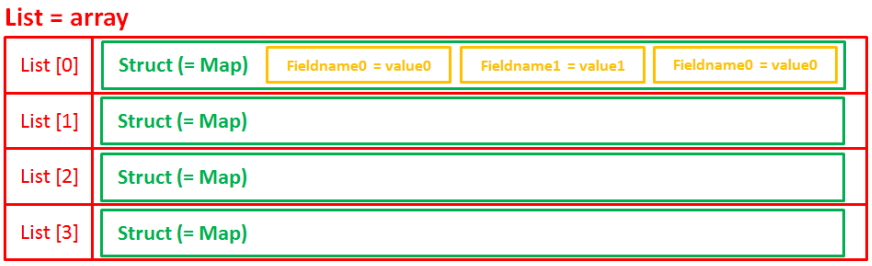


Figure 281: *List of structs, technical principle*

- **number**
  For integer values.
- **fixed point number**
  For numbers with a fractional part, e.g., currencies. You have to enter the total number of digits (*Precision*) and the number of digits that fall to the right of the decimal point (*Scale*) in the respective fields below.
- **string**
  For up to 4000 alphanumeric characters.

  ⚠ Restriction when using an Oracle database: at most 4000 bytes can be saved in UTF encoding. Starting with Oracle12c.

- **long string**
  For large objects.

  ⚠ For *long strings* the limit depends on the database system used for ConSol CM: MS SQL Server: 2 GByte; MySQL: 4 GByte; Oracle: 16 - 64 GByte (depending on page size of tablespace).

- **short string**
  For up to 255 alphanumeric characters.

  ⓘ For *string* fields, you can use specific annotations to fine-tune the field definition. For example, a *string* field can be defined to contain a URL which will automatically be displayed as hyperlink or can be the hook for an autocomplete list. Please read the following section.

- **contact data reference**
  Special data type used internally for referencing the contacts associated with a ticket. In FlexCDM (i.e., starting with CM version 6.9.0), this data type is no longer displayed but only used internally in the CM system.

- **MLA field**
  This data type is used for fields that contain hierarchical lists with a tree structure called *MLA* (Multi Level Attributes). The name of the field is the name of the new MLA that has to be defined within the MLA Administration. The group of the field has to be referenced when the MLA is created.

⚠ The data type you choose on creating a data field cannot be changed afterwards!

String fields are widely used for customer, ticket, and resource data and strings can be used to contain various content, for example, a text box with a comment, a simple input field with only 20 characters, a URL or a password. The fine-tuning of string fields is implemented using specific annotations which are all listed on the Annotations page. However, since work with these annotations is an every-day task of CM administrators, the most important and most commonly used annotations will be explained here as well.

How can I ...

... insert a **text box** instead of a single line?

Value for annotation `text-type`: “textarea”

The size of the text box can be adjusted, displayed as standard text box depending on web browser. Use the `field-size` annotation in case a specific size of the text box is required.

... hide the input of the fields for **passwords**?

Value for annotation `text-type`: “password”

Only dots will be displayed. This annotation does **not** define the field to contain a password! It only defines the display mode! Use the `password` annotation to define a string field to contain the CM/Track password.

... display a **hyperlink**, display the name instead of the link?

Value for annotation `text-type`: “url”

Input will be displayed as a hyperlink in view mode. String has to match a specific URL pattern:

- "^((?:mailto\:|(?:(?:ht|f)tps?)\://)1\S+)(?: (?:\|)?(.*))?$"

First part of the string is the link (url), second part is the name which should be displayed.

Example: "http://consol.de ConSol"

... display a **file link**?

Value for annotation `text-type`: “file-url”

Input will be displayed as a link to a file on the file system. The web browser has to allow/support those links!

Example: Enabling file:// URLs in a Firefox browser

Add the following lines to either the configuration file `prefs.js` or to `user.js` in the user profile. On a Windows system usually in a folder like `C:\Users\<USERNAME>\AppData\Roaming\Mozilla\Firefox\Profiles\uvubg4fj.default`

- user_pref("capability.policy.localfilelinks.checkloaduri.enabled", "allAccess");
- user_pref("capability.policy.localfilelinks.sites", "http://cm-server.domain.com:8080");
- user_pref("capability.policy.policynames", "localfilelinks");

Alternatively a Firefox browser add-on like ***Local Filesystem Links*** can be installed for better access to the referenced files and folders.

The link will also be displayed as tooltip.

The URL is correctly formed if the following conditions are met:

- It starts with `file:` followed by regular slashes:
  - three slashes "///" for files on the same computer as the browser (alternatively "//localhost/") or
  - two slashes followed by the server name followed by another slash for files on file servers accessible from the computer running the browser.
- These are followed by the full path to the file ending with the file name.
- The path on Microsoft Windows systems is also written with forward slashes instead of backslashes.
- The drive letter of a local path on Microsoft Windows systems is noted as usual, for example `C:`.
- Paths with spaces and special characters like "{, }, ^, #, ?" need to be percent encoded ("%20" for a space for example) for Microsoft Windows systems.

Example URLs:

- file://file-server/path/to/my/file.ext
- file:///linux/local/file.pdf
- file:///C:/Users/myuser/localfile.doc

See also the explanation about `file-url` in the section text-type

... define a **label**?

Value for annotation `text-type`: “label”

This will be a read-only field which is displayed in gray, use the ***label-group*** annotation to link label and input fields which belong together. Please take a look at the annotations for labels (`show-label-in-edit`, `show-label-in-view`) before implementing special label fields!

... define a field for the **valid email addresses**?

Value for annotation `email`: “true”

The field may only contain valid email addresses. Input will be validated according to standard email format <name>@<domain>.

... define a scripted autocomplete list?

Value for the annotation `text-type` = “autocomplete”

Optional: value for the annotation `autocomplete-script` = <name of the respective script>

A scripted autocomplete list is used to provide a drop-down menu which is filled dynamically using the input the engineer has provided so far. For example, when the user types "Mil", the possible values "Miller", "Milberg", and "Milhouse" are displayed as list and the engineer can select the one required for the field. You know this behavior from other autocomplete fields, e.g., the search for engineers for a ticket or the search for customers while creating a ticket. However, in these cases, CM generates the list automatically. The behavior cannot be influenced or customized. Scripted autocomplete lists, on the contrary, can be implemented by the CM administrator. The values are based on a result set which is dynamically created. The result set can contain strings, engineers, customers (Units), and resources.

A detailed description of scripted autocomplete lists is provided in section Scripted Autocomplete Lists.

... define a field for the **CM/Track login**?

Value for annotation `username`: “true”

Will be used for authentication against CM/Track server. Only for customer fields in a contact object.

... define a field for the **CM/Track password**?

Value for annotation `password`: “true”

Will be used for authentication against CM/Track server (in DATABASE mode). Only for customer fields in a contact object.

... define a field containing personal data?

Value for annotation `personal-data`: “true”

This annotation can be assigned to ticket and contact fields. Contact fields with this annotation will be deleted when a contact is anonymized. Ticket fields with this annotation will be deleted when the main customer of the ticket is anonymized. See Example 8: Removing Customer Data for information about how to anonymize a contact.

When defining a field to contain personal data, please take into account that the deletion of the field during the anonymization process is treated as a regular update. Therefore, business event triggers reacting on changes to ticket fields fire, and the contact update action script is executed.

This might lead to unwanted side-effects.

Labels for Customer Fields

In CM versions prior to 6.9.4, the labels for customer fields (e.g., name, address) are displayed in the Web Client as watermarks per default. If a distinct label is required, a separate customer field of type STRING with the annotation `text-type` = “label” has to be used, as shown in the following figures.

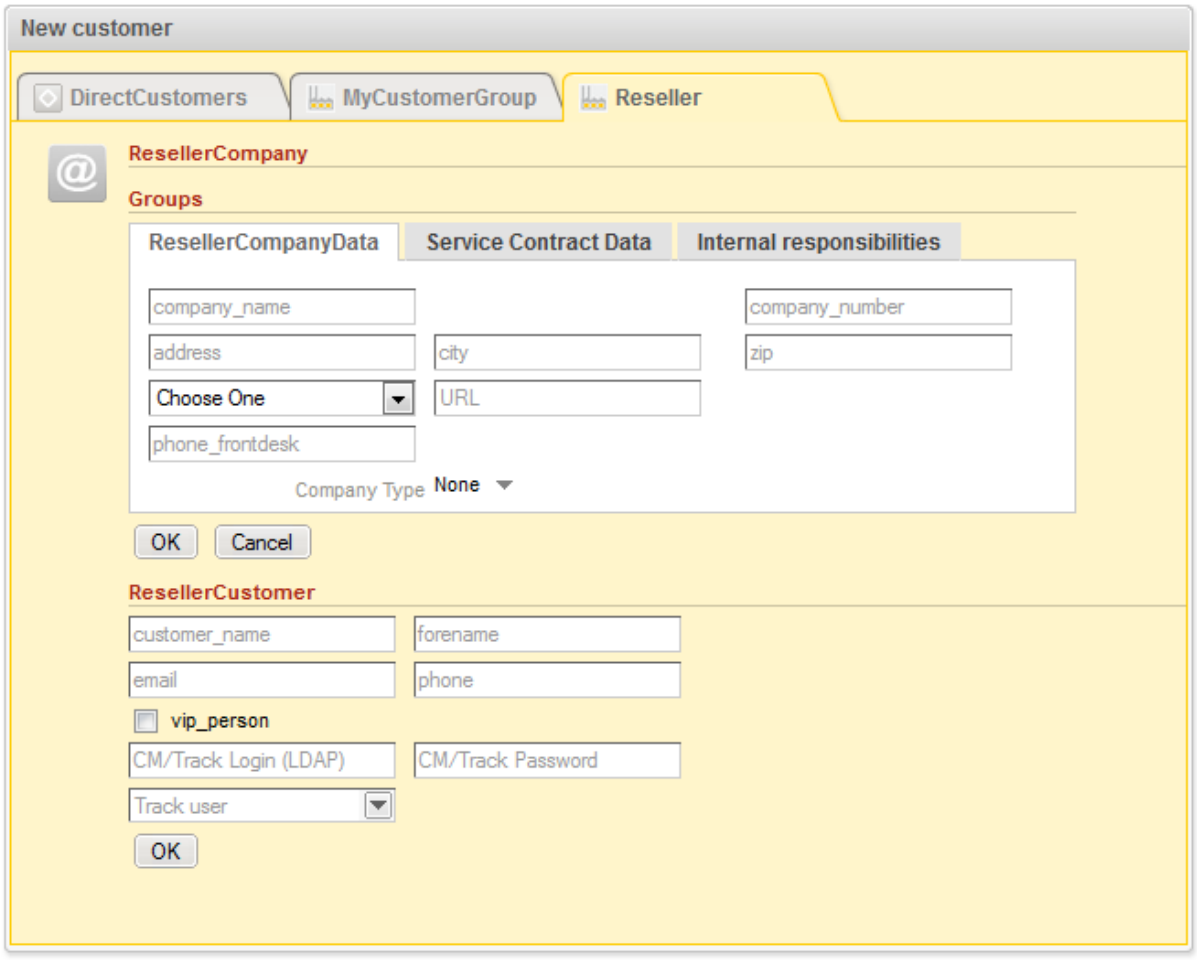


Figure 282: *ConSol CM Web Client - Default display of customer field labels in versions prior to 6.9.4*

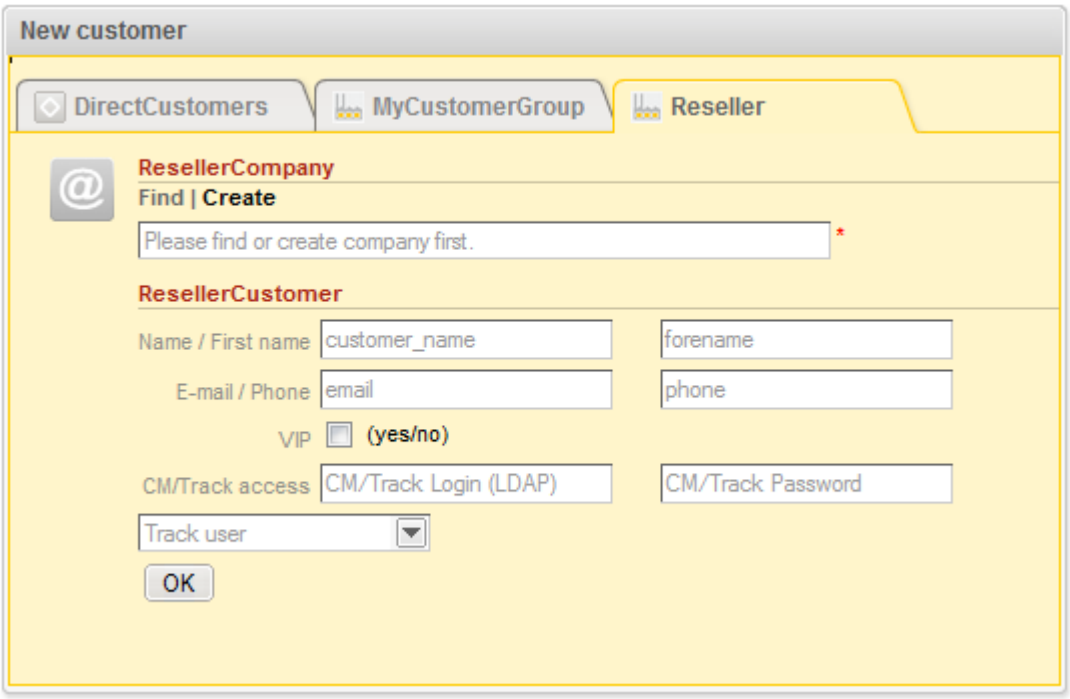


Figure 283: *ConSol CM Web Client - Labels as distinct customer fields in versions prior to 6.9.4*

As of CM version 6.9.4, you, as an administrator, can decide whether watermarks or labels (or both) should be used. Labels are now implemented similar to the labels for ticket fields (i.e., the data fields for ticket data). Furthermore, tool tips can be added. The display modes of labels for customer fields is controlled by annotations. Annotations can be set on customer field group level and on customer field level. The field annotations overwrite the group annotations. In this way, you can define a group display behavior but can modify one or more single fields, as demonstrated in the example below.

Annotations for customer field groups:

- layout:show-labels-in-view (“true” if not set)
- layout:show-labels-in-edit (“true” if not set)
- layout:show-watermarks (“false” if not set)
- layout:show-tooltips (“true” if not set)

Annotations for customer fields:

- layout:show-label-in-view
- layout:show-label-in-edit
- layout:show-watermark
- layout:show-tooltip

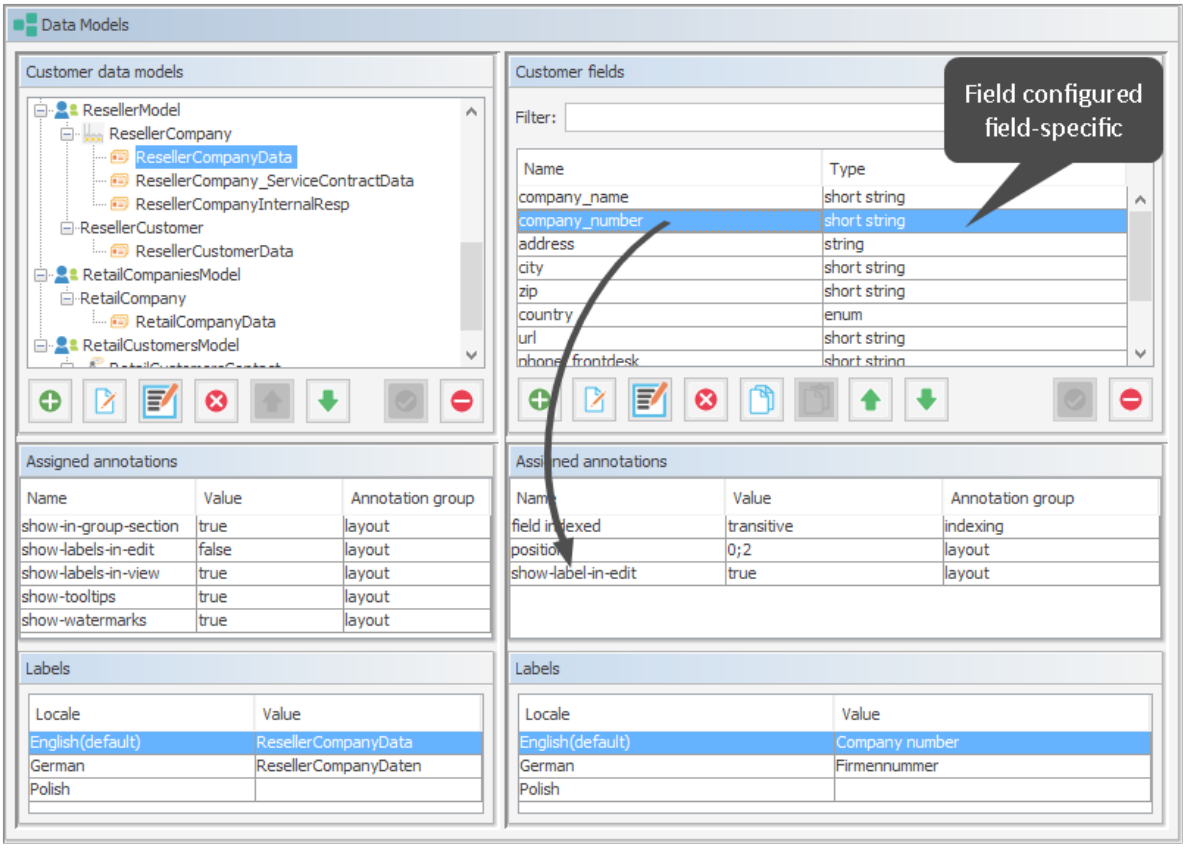


Figure 284: *ConSol CM Admin Tool - Customers, Data Models: Configuration for Reseller customer field group, one field configured field-specific*

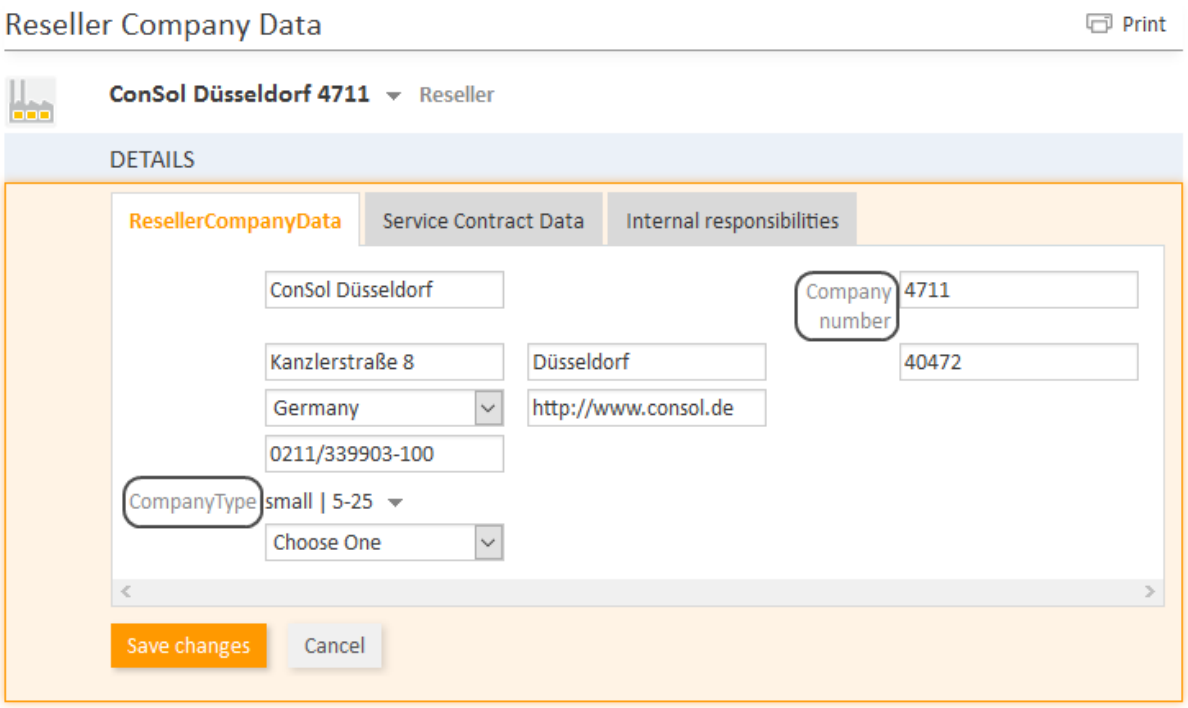


Figure 285: *ConSol CM Web Client - View for Reseller customer field group, one field (company_number) annotated field-specific (edit mode)*

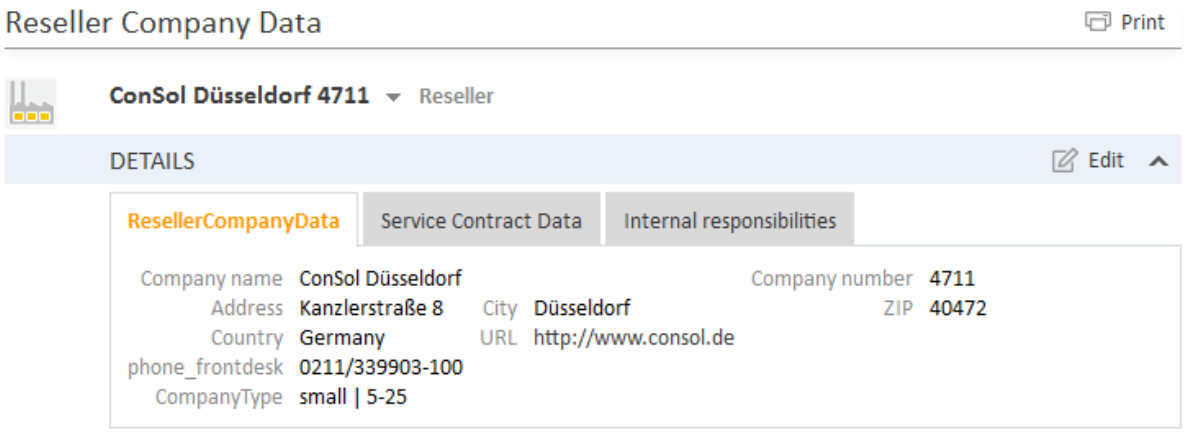


Figure 286: *ConSol CM Web Client - View for Reseller customer field group, one field (company_number) annotated field-specific (Example: No influence on view mode)*

The labels are inserted automatically, so in the end there might be, at most, six columns. In the following example, two columns are used.

- Field_1: position 0;0
- Field_2: position 0;1
- Field_3: position 1;0
- Field_4: position 1;1
- Field_5: position 2;0
- Field_6: position 2;1

Label_1	Field_1	Label_2	Field_2
Label_3	Field_3	Label_4	Field_4
Label_5	Field_5	Label_6	Field_6

Figure 287: *Example for customer field positions in the grid*

Label Mode after an Update from CM Version Prior to 6.9.4 to a Version 6.9.4 and Up

The layout of the customer data model will be preserved during the update! All annotations will be set accordingly and can be modified later.

For all existing customer field groups, the annotation setting layout:`show-labels-in-edit` = "false" (which will hide standard labels in edit mode) is applied during the update.

Default Mode for New Customer Field Groups and Customer Fields

All new customer field groups will be rendered with the new default configuration. This means:

- standard labels
- watermarks disabled
- tool tip enabled

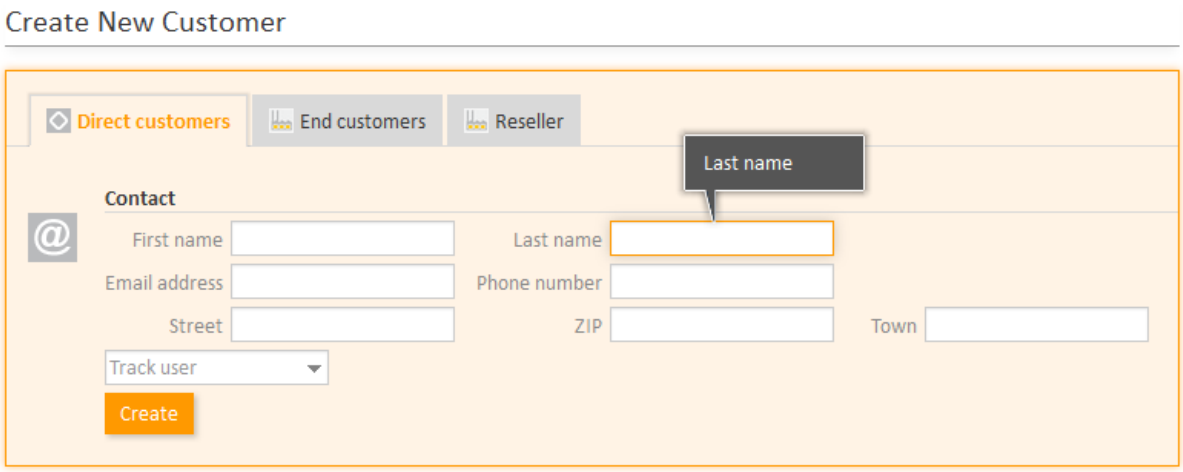


Figure 288: *ConSol CM Web Client - Default display mode for customer fields*

D.4.3 Scripting Using Flex CDM Objects

In this book we will use the terms *customer* and *customer definition*. In previous versions of ConSol CM, the term *data object* was used to refer to customers. However, the names of the corresponding Java classes are *Unit* and *UnitDefinition*. All other Java classes which deal with customer objects are also still named *Unit*... Please keep that in mind if you work as both a ConSol CM administrator and programmer. Please refer to the *ConSol CM Java API Doc* for details.

D.4.3.1 New Scripting Features Since ConSol CM Version 6.9

Up to ConSol CM version 6.8, a *Unit* could have only one ticket field group. The expression`unit.get("name")` was always valid because a ticket field with a specified name could exist only once, for example *"group1.name"*.

Starting with ConSol CM version 6.9, a customer (*Unit*) may have one or more customer fields with the same field name, e.g., *"name"* can be represented as *"group1.name"* and *"group2.name"*. In such cases, the expression `Unit.get("name")` is not semantically valid and throws an exception.

For backwards compatibility the code `unit.get("name")` will work as long as the customer field *name* is unique. If another customer field with the same name is added, the code `unit.get("name")` will no longer work!

Use the following notation to retrieve unit field (customer field) data:

- **For one field:**
  unit.get("group1:name")
- **For numerous fields:**
  unit.get("group1:field1.group2:field2")

```
unit.get("contactFields:companyReference.companyFields:name")
```

Code example 27: *Example to get the name of the company's contact*

D.4.3.2 Extension of the *Custom Field Expression Language* (CFEL)

Starting with ConSol CM version 6.9, the Custom Field Expression Language (CFEL) has been improved to provide simple access to many objects. This applies to scripting for tickets and other objects as well as objects from the customer data model, i.e., units (customer objects: objects at the contact or company level). The improvements concerning units (customers) are explained here. For a detailed explanation on how to write Java or Groovy code which uses customer data model objects, please refer to the *ConSol CM Process Designer Manual*.

You can access the customer data model objects from different scripts:

- **Workflow:**
  - Scripts in workflow activities
  - Scripts in workflow conditions
- **Admin Tool (AT) scripts of type:**
  - Dependent enum
  - Email
  - Clone
  - Default values
  - Customer action
  - Customer condition
  - Workflow

D.4.3.3 Convenience Methods

Examples of convenience methods:

Object.Method	Explanation
`def contacts = unit.get ("contacts()")` `List contacts = company.getContacts()`	Using CFEL ("contacts()") a list of all contacts is retrieved for the company (unit).
`Unit company = mainContact.getCompany()`	For a contact, the company can be retrieved easily.
`newContact.set("company()", newCompany)`	For a (new) contact, the company is assigned the CFEL expression "company()", provides easy access to the company object.
`List tickets = company.get ("tickets()")`	For a company, all tickets are retrieved.

`Ticket ticket = getTicket();`
`Unit mainContact = ticket.getMainContact()`
`List tickets = mainContact.get ("tickets()")`
	For a contact, all tickets are retrieved.

`Integer count = contact.get ("company().contacts() [0].tickets()[count]");`	A chain of expressions is used to get the number of tickets for a specific contact.

```
TicketCriteria ticketCriteria = new TicketCriteria();
Unit patternContact = new Unit("contact", customerGroup);
mdcmCriteriaBuilder.setReferencedContactCriteria(ticketCriteria, patternContact);
```

Code example 28: *Example 1: Search for the tickets of a contact or of a company*

```
TicketCriteria ticketCriteria = new TicketCriteria();
Unit contactPattern = new Unit("contact", customerGroup);
mdcmCriteriaBuilder.setReferencedContactCriteria(ticketCriteria, contactPattern);
Unit companyPattern = new Unit("company", customerGroup);
companyPattern.setFieldValue("name", „ConSol");
mdcmCriteriaBuilder.setReferencedCompanyCriteria(contactPattern, companyPattern);
```

Code example 29: *Search for the tickets of the contact who is member of a certain company*

```
UnitCriteria unitCriteria = new UnitCriteria();
Unit companyPattern = new Unit("company", customerGroup);
mdcmCriteriaBuilder.setReferencedCompanyCriteria(unitCriteria, companyPattern);
```

Code example 30: *Search for contacts of a certain company*

For detailed information about the methods, including input parameters and output data type (method signatures), please refer to the *ConSol CM Java API Doc*.

D.4.3.4 Important Objects

The objects which are available in the script obviously depend on the script's context. The following examples demonstrate some of the possible use cases:

Startig point	Script	Objects	Example
Company page	customer action script	unit represents the company	`def contacts = unit.get("contacts()")`
Contact page	customer action script	unit represents the contact	`List tickets = unit.get("tickets()")`

Startig point	Script	Objects	Example
Workflow activity	workflow action or condition script	ticket	`def id = ticket.getId()`
Workflow activity with script in AT	workflow action or condition script	ticket not present implicitly!	`import com.consol.cmas.common.model.ticket.Ticket` `def id = ticket.getId()`

D.4.4 Using Scripted Field Visualization for Customer Fields

Using scripted field visualization, you can enhance the display of data in customer fields. Please see section Scripts of Type Field Visualization for details.

D.5 Templates for Customer Data

This chapter discusses the following:

D.5.1 Introduction to Using Templates for the Display of Customer Data

In the ***ConSol CM*** Web Client, customer data sets are displayed in short form at various locations, based on templates. For example, in the ticket list the contact name and company name might be required whereas in the customer data section of the ticket the family name, first name, and phone number of a contact might be needed. This section will show you where short forms are used and how the respective templates are configured using the Admin Tool.

The configuration is based on the following principle:

- Templates are assigned to a customer, i.e., to a contact definition or company definition, in the navigation item *Data Model* in the navigation group *Customers* in the Admin Tool. These templates control the display in certain areas of the Web Client. As shown in the following figure, several template types can be defined. A template of type *Default* is always required and serves as a fallback for all other template types except for REST. If the REST API is used (either directly or via CM/Track), the definition of a REST template is also mandatory.
- The referenced template must be stored in the Admin Tool, navigation group *System*, navigation item *Scripts and Templates* , tab *Templates*. The name of a template is user-defined.

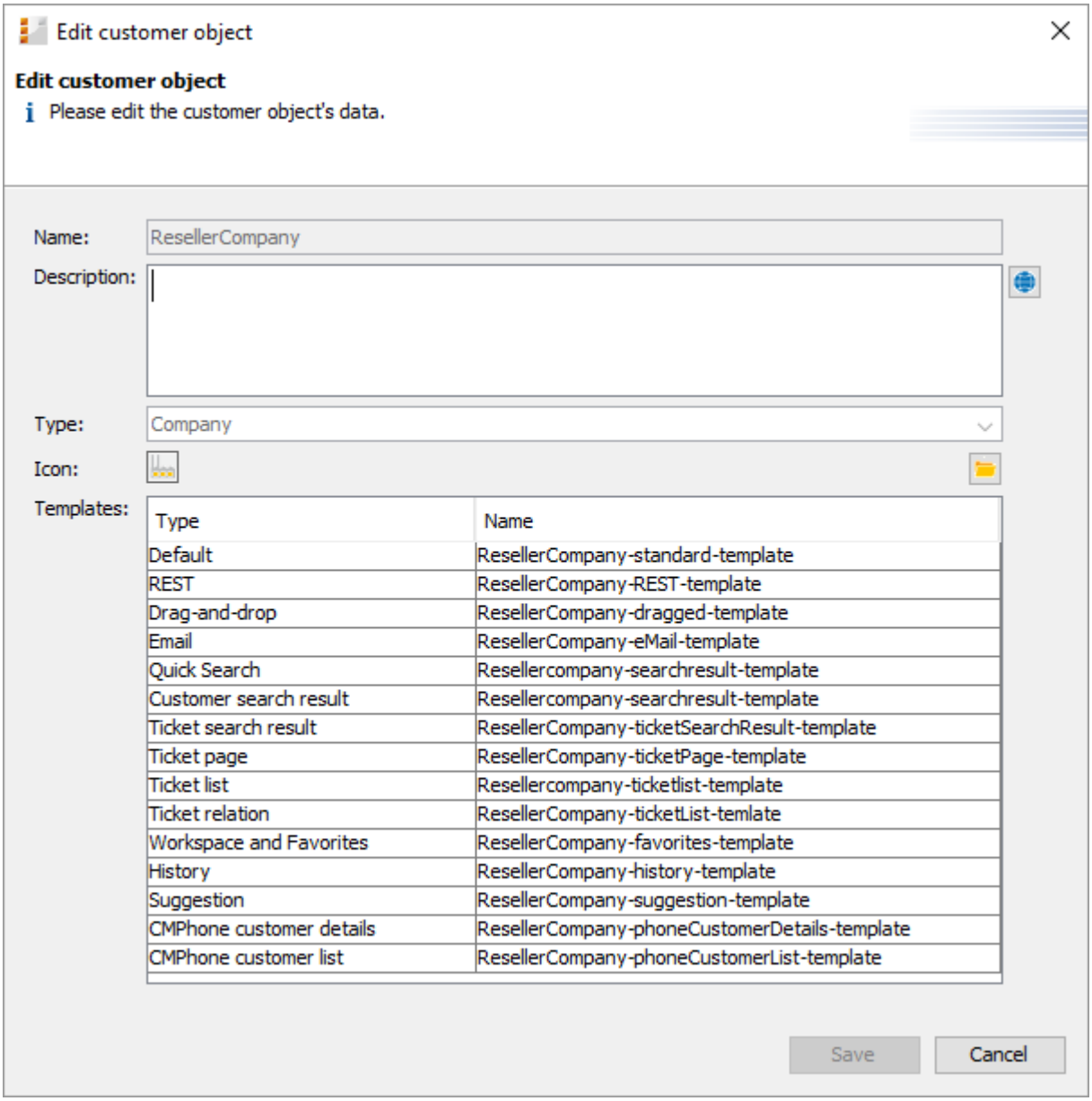


Figure 289: *ConSol CM Admin Tool - Customers, Data Models: Template annotations for customer object (here: company)*

In the following paragraphs, the syntax and coding for templates and all possible template types are explained.

D.5.2 Coding Templates

The templates are written using ***FreeMarker*** notation. For detailed information, please refer to the FreeMarker web site.

Within the templates, you work with three object types:

1. **Name of the customer object**
   i.e., the name of a company or contact object
2. **Name of the customer field group**
   within the customer object (a customer field group is similar to a ticket field group for ticket data)
3. **Name of the customer fields**
   within the customer field group

See the following figure for an example:

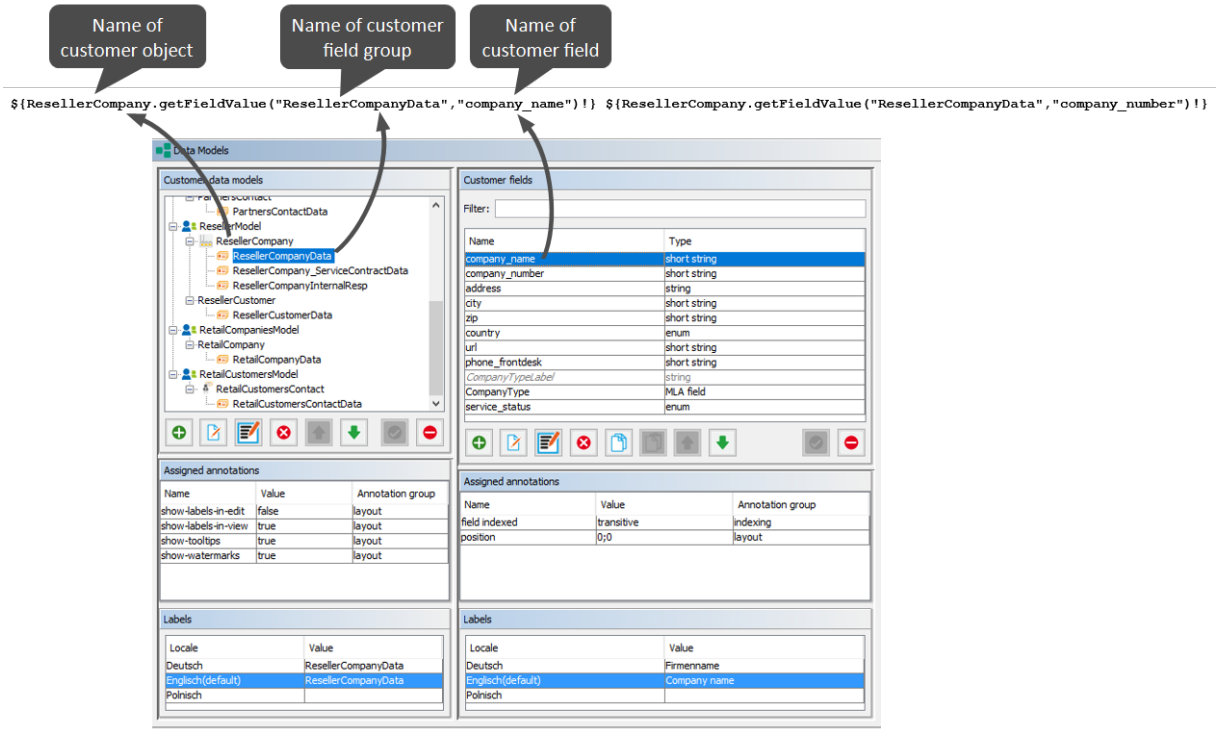


Figure 290: *ConSol CM Admin Tool - Customers, Data Models: Writing customer templates*

⚠ The customer templates must be single-row! They must not contain line-breaks!

D.5.2.1 Examples of Templates

Here are some examples of templates:

```
<#if ResellerCustomer.getFieldValue("ResellerCustomerData","customer_name")?has_
 content &&
 ResellerCustomer.getFieldValue("ResellerCustomerData","firstname")?has_content>
 ${ResellerCustomer.getFieldValue("ResellerCustomerData","customer_name")!},
 ${ResellerCustomer.getFieldValue("ResellerCustomerData","firstname")!}
 <#else> ${ResellerCustomer.getFieldValue("ResellerCustomerData","customer_
 name")!}</#if>
```

Code example 31: *Example of a customer template with customer name (has to be written in one line!)*

```
${company.getFieldValue("company", "name1")!}${company.getFieldValue("company",
 "name2")!}
 ${company.getFieldValue("company", "mainaddr_city")!},${customer.getFieldValue
 ("customer", "firstname")!}
 ${customer.getFieldValue("customer", "name")!}
```

Code example 32: *Example of a company-contact template (has to be written in one line!)*

```
<#if company??>${company.getFieldValue("company", "name1")!}${company.getFieldValue
 ("company", "name2")!}
 ${company.getFieldValue("company", "mainaddr_city")!},
 </#if>${customer.getFieldValue("customer", "firstname")!}${customer.getFieldValue
 ("customer", "name")!}
```

Code example 33: *Example of a search-customer template (has to be written in one line!)*

```
<#setting number_format="#"/>${customerModelCompany.getFieldValue("groupName",
 "numberValueField")!}
```

Code example 34: *Setting number format: removing "." in number display*

D.5.3 Localizing Enum Values in Customer Templates

Starting with ConSol CM version 6.10.5.4, enum values can be localized, i.e., you can have the localized value displayed in the Web Client. For example, an ENUM *salutation* contains “mr” and “mrs” as technical values, but in the Web Client, the correct salutation in the respective browser locale should be displayed. If the locale of the browser is not configured explicitly, the standard CM locale will be used.

The following example works with the *salutation* ENUM.

```
${localize(customer.getFieldValue("customer", "salutation"))!}
 ${customer.getFieldValue("customer", "firstname")!} ${customer.getFieldValue
 ("customer", "name")!}
```

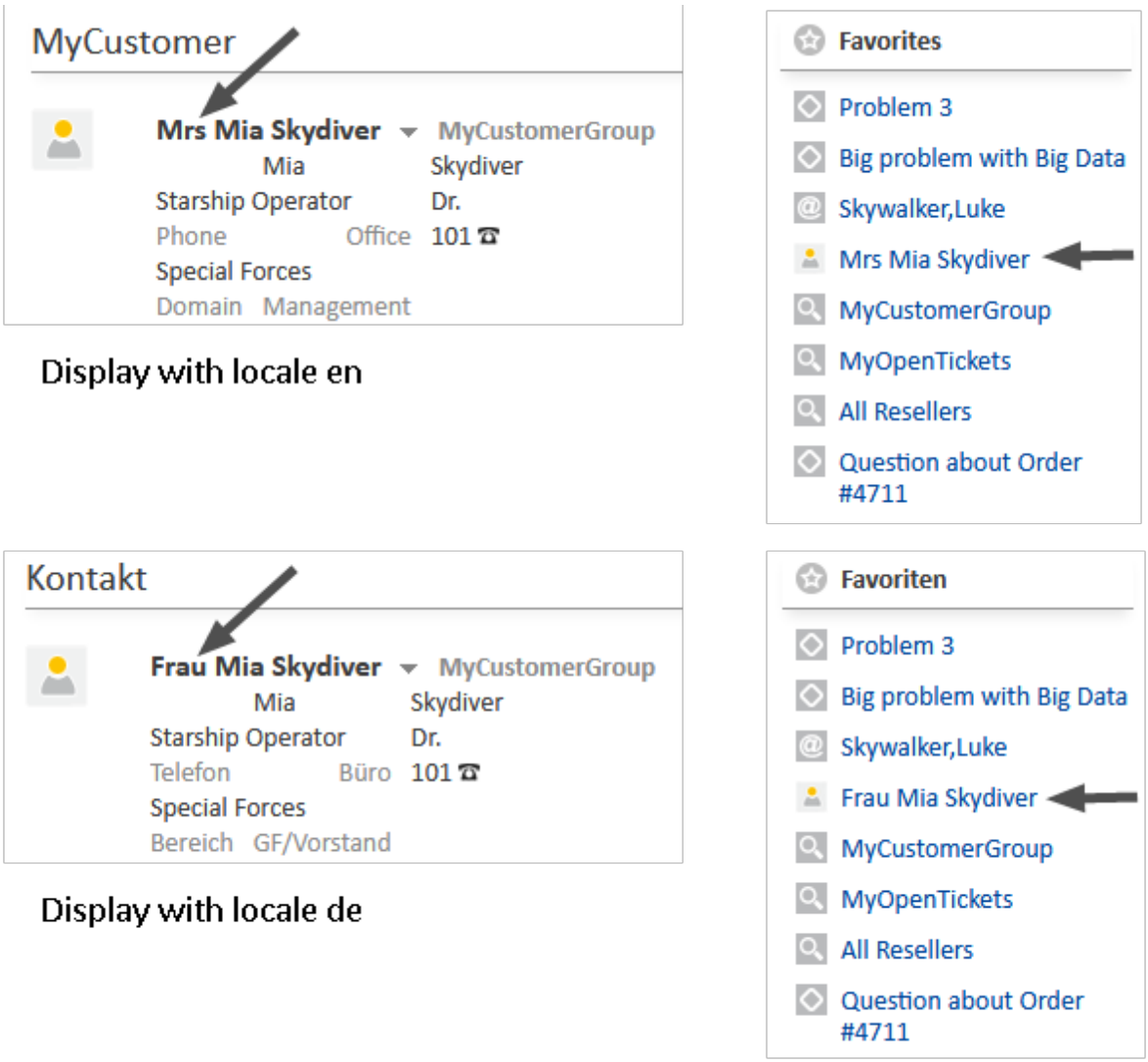
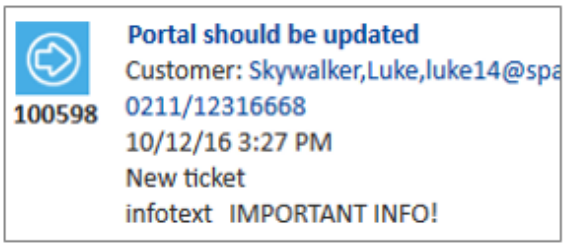


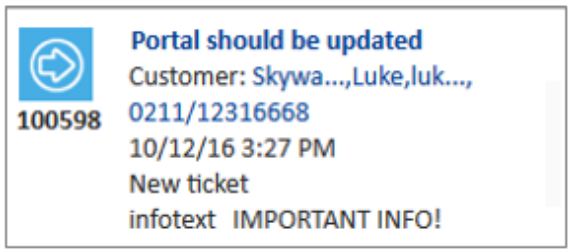
Figure 291: *ConSol CM Web Client - Display of an enum value based on a customer template*

D.5.4 Abbreviating Values in Customer Templates

In order to avoid that values of templates are not displayed, because the space in the GUI is not sufficient, each value can be abbreviated, using the `abbreviate()` method. In this way, all fields of the template can be displayed, even if the first fields of the template contain very long values. Please see also the following example for an explanation.



Not abbreviated



Abbreviated using template shown below

```
${abbreviate(ResellerCustomer.getFieldValue("ResellerCustomerData
","customer_name"),8)!},${ResellerCustomer.getFieldValue("Reselle
rCustomerData","forename")!},${abbreviate(ResellerCustomer.getFie
ldValue("ResellerCustomerData","email"),6)!},
${ResellerCustomer.getFieldValue("ResellerCustomerData","phone")!
}
```

The `abbreviate()` method needs two parameters:

- the value which should be abbreviated
- the number of characters to be displayed. Please note that this is the number of all characters which are displayed, i.e. including the three dots. Thus, for example, to display one letter and three dots, use “4” as parameter. 4 is also the minimum value!

D.5.5 Template Types

D.5.5.1 Standard (Default)

This template is used in all of the following locations if no special templates have been defined, i.e., all other special templates could be omitted if a standard template is defined.

> ⚠ If no template is defined for a certain Web Client location and no standard template has been defined either, there will be an error in the log file and -- *unknown* -- will be displayed in the Web Client.
>
> So make sure that at least a standard template is defined. In a two-level customer data model, this has to be done for the company and for the contact level!

D.5.5.2 REST

This template configures the appearance of customer data when accessed using the ***REST API***. In the standard configuration, no customer data are displayed in the CM portal, CM/Track, which is based on the REST API. This template is only used when you address the REST API directly, e.g., for programming CM interfaces. The following example shows a REST client request for customer data and the resulting answer of the CM server via REST API. The value within the <mark> tag in the XML output is the customer information, formatted using the REST template. In this example, the company name (“Spaceoddity”) and the company number (“42”) are part of the template.

```
${ResellerCompany.getFieldValue("ResellerCompanyData","company_name")!}
 ${ResellerCompany.getFieldValue("ResellerCompanyData","company_number")!}
```

Code example 35: *Example REST template*

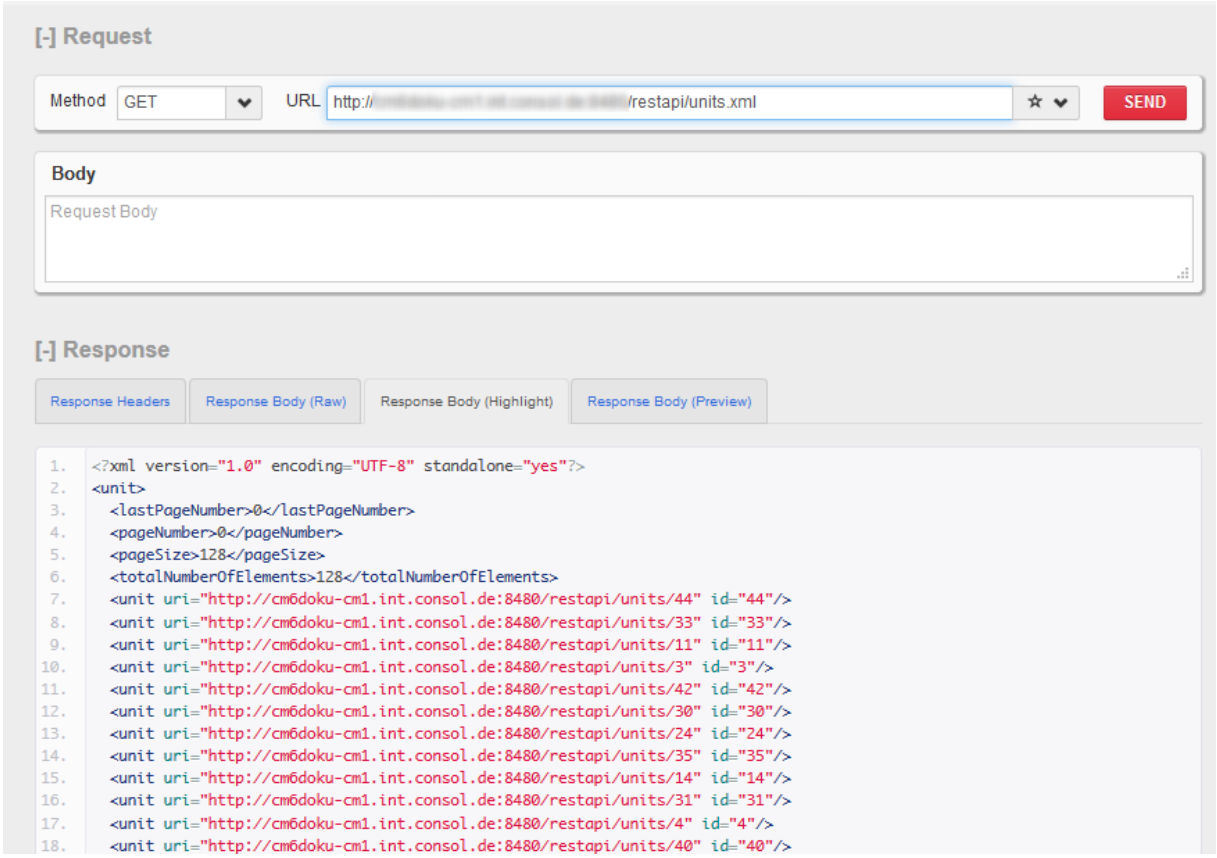


Figure 292: *REST API - Request for all units (only part of the list is shown in the figure)*

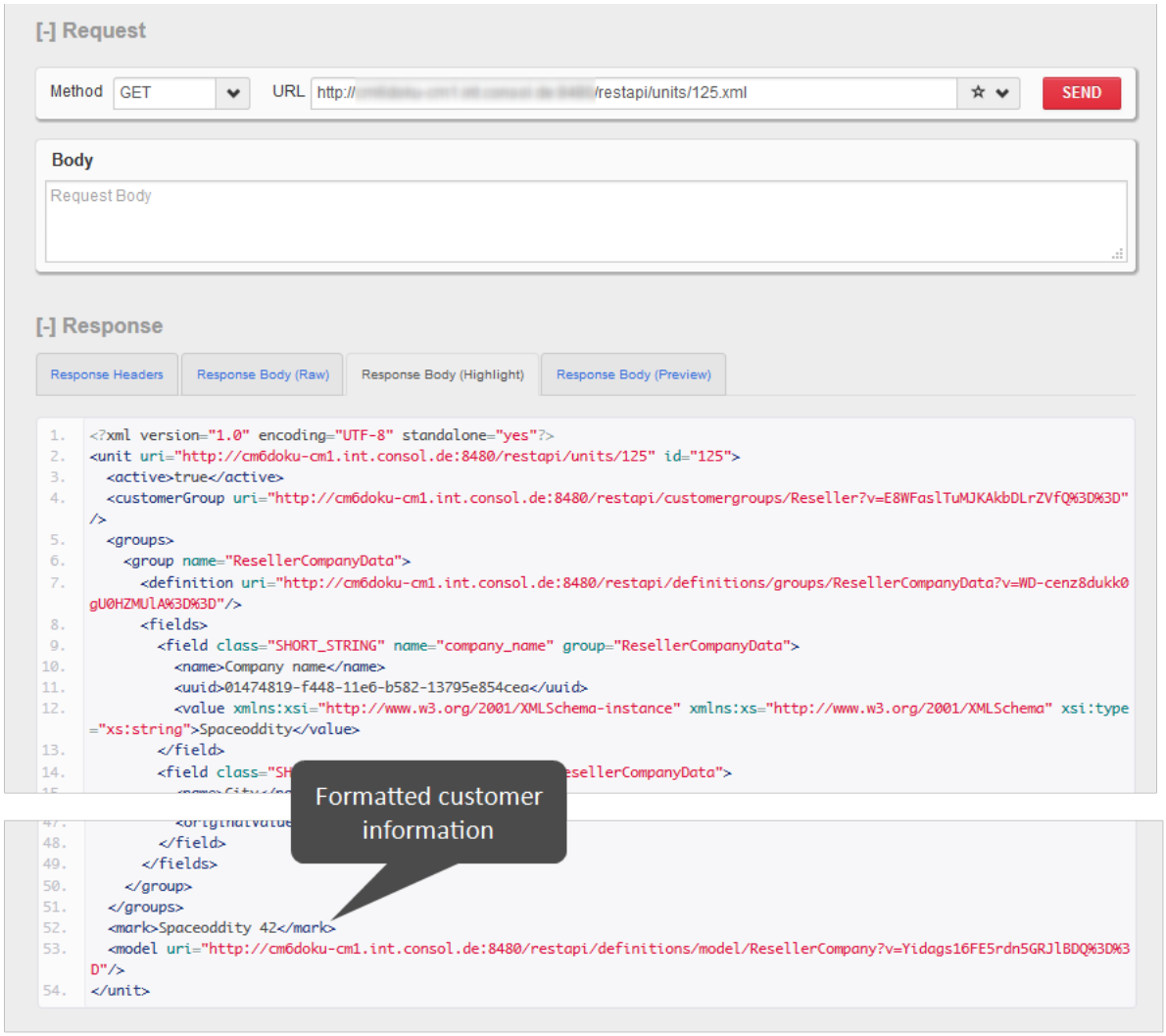


Figure 293: *REST API - Request for one unit (ID 125) as XML output*

> Starting with CM version 6.10.3, the ConSol CM REST API can handle various request parameters concerning objects from CM/Resource Pool. If you want to work with those objects via REST, make sure the REST template or at least a Default template is defined for each resource type!
>
> Remember: CM/Track also uses the REST API and thus needs the REST templates!

D.5.5.3 Drag-and-drop

This defines the format of a customer data set while the data set is dragged, e.g., from the *Customers* section to the *Favorites* section.

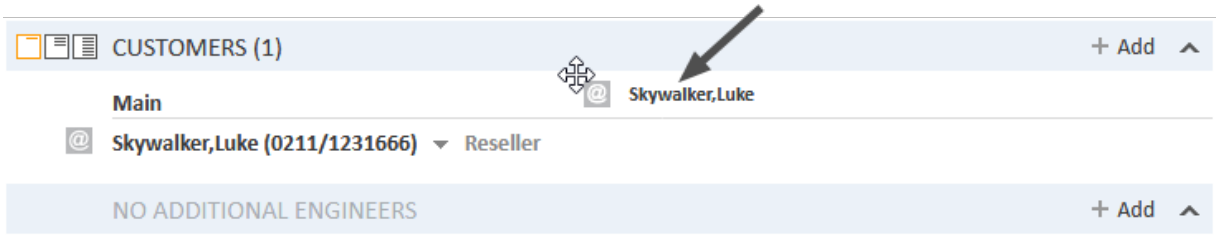


Figure 294: *ConSol CM Web Client - Customer drag-and-drop template*

D.5.5.4 Email

In the Ticket Email Editor an automatic search provides search results in-line in the form of a drop-down list. The format of those search results can be configured using this template.

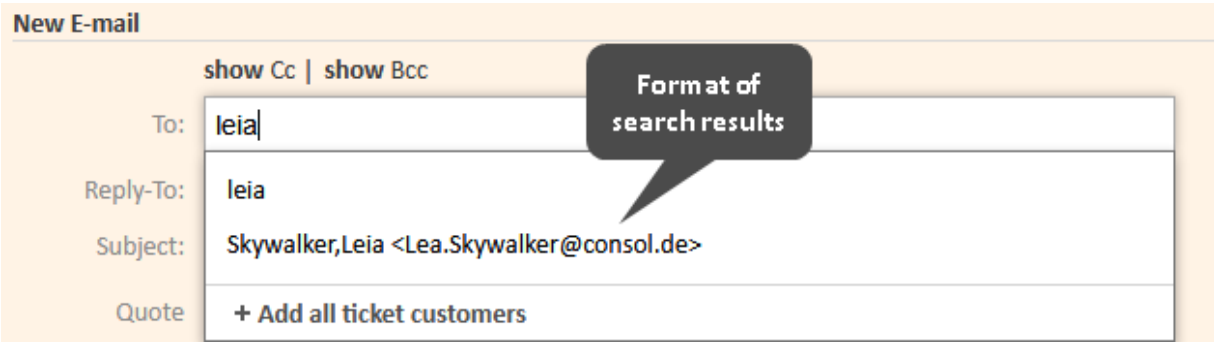


Figure 295: *ConSol CM Web Client - Customer email template*

```
<#if customer.getFieldValue("customer","name")?has_content
 && customer.getFieldValue("customer","firstname")?has_content
 && customer.getFieldValue("customer","division")?has_content>
 ${customer.getFieldValue("customer","name")!},
 ${customer.getFieldValue("customer","firstname")!},
 ${customer.getFieldValue("customer","division")!}
 <#else> ${customer.getFieldValue("customer","name")!},
 ${customer.getFieldValue("customer","division")!}</#if>
```

Code example 36: *Email template (has to be written in one line!)*

D.5.5.5 Quick Search

This template defines the format of the search result for contacts or companies in the Quick Search. The template is restricted to a single line of output.

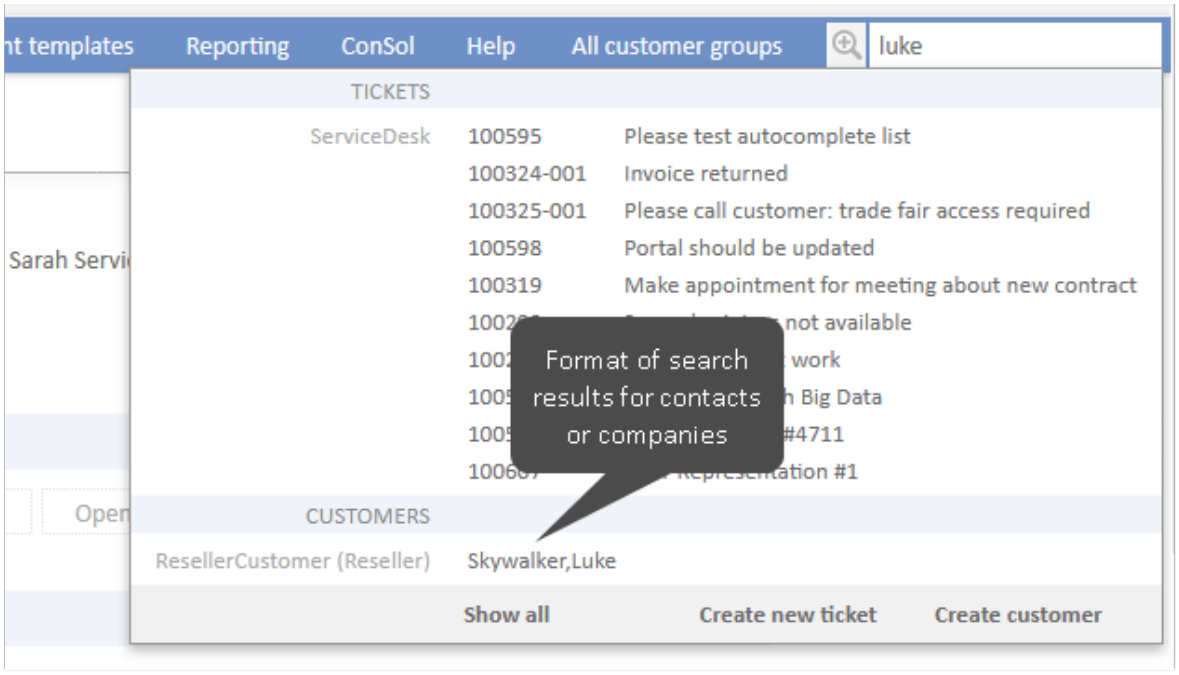


Figure 296: *ConSol CM Web Client - Customer Quick Search template*

D.5.5.6 Customer Search Result

This template defines the search results for automatic searches in auto-complete fields.

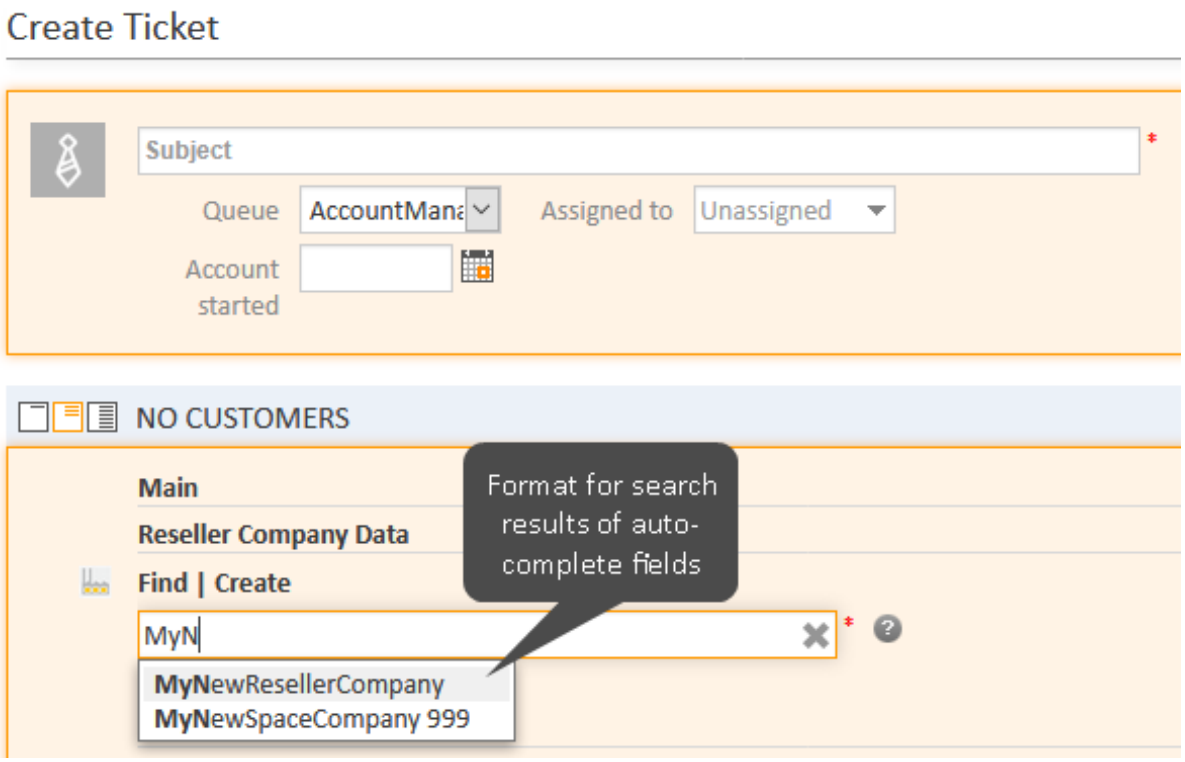


Figure 297: *ConSol CM Web Client - Customer search result template*

This is the applied template:

```
${ResellerCompany.getFieldValue("ResellerCompanyData","company_name")!}
 ${ResellerCompany.getFieldValue("ResellerCompanyData","company_number")!}
```

Code example 37: *ResellerCompany search result template (has to be written in one line!)*

D.5.5.7 Ticket Search Result

In the results page of the Detailed Search the tickets found by the search are displayed as a list. One column of this list contains the main customer of the ticket. The `ticket search result` template defines the layout of the customer data in this column.

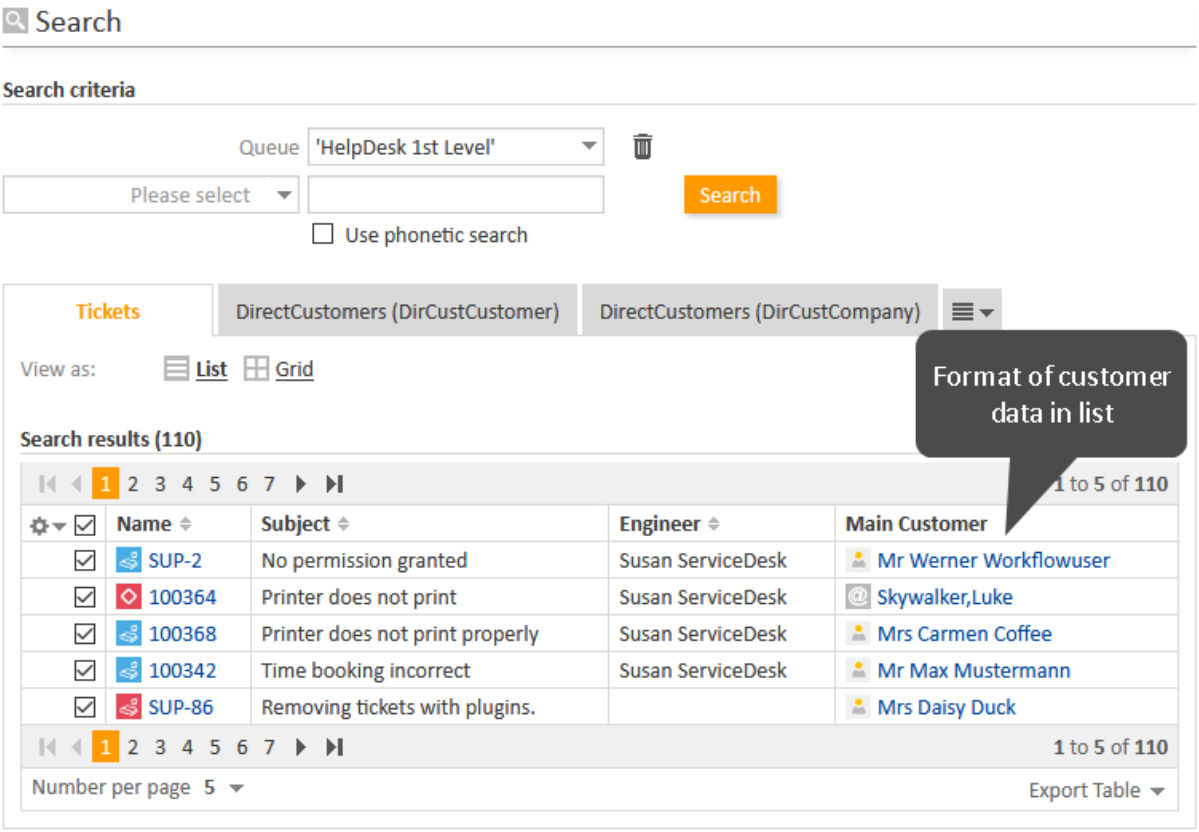


Figure 298: *ConSol CM Web Client - Customer ticket search result template*

D.5.5.8 Ticket Page

This template defines the presentation of customer data in the *Customers* section of a ticket. The influence of this template varies depending on the ConSol CM version:

- In CM versions 6.10 and below, the template defines the appearance of the customer data only for the minimum display level. In the medium and large levels, the first line (positions 0;x) of the customer fields definitions is displayed. This applies only if the `position` annotation has been set! If no `position` annotation is set and the Web Client renders the order of the data fields automatically, the mechanism will not work - the fields of the first line will not be used as template!
- In CM versions 6.11.0 and up, the template is used on all three levels.

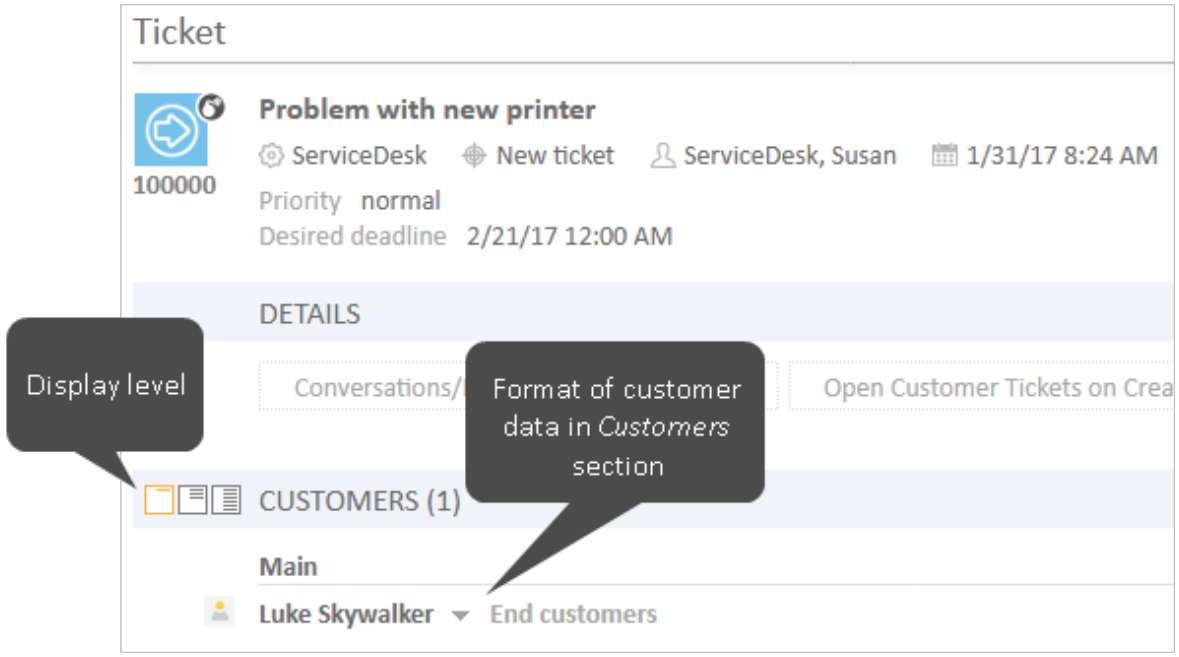


Figure 299: *ConSol CM Web Client - Customer ticket page template*

In ConSol CM versions 6.11 and up, the ticket page template is also used for rendering the customer data on the customer pages, i.e., on contact and company pages. The following figure shows an example of a customer page.

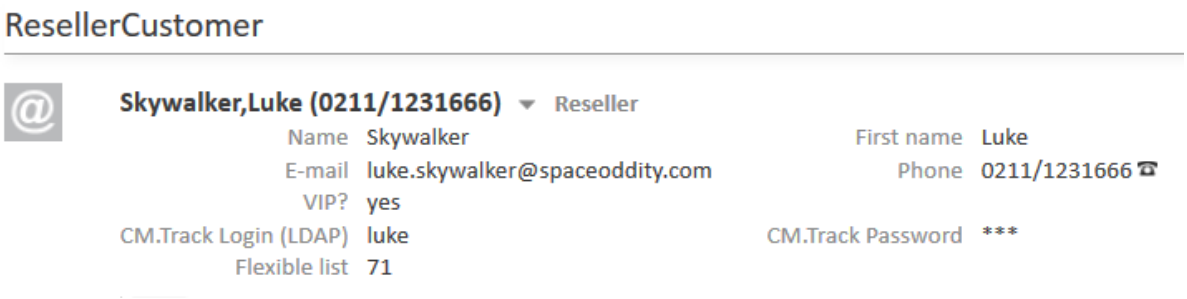


Figure 300: *ConSol CM Web Client - Header of a contact page, contact name rendered using ticket page template*

The following template is used:

```
<#if ResellerCustomer.getFieldValue("ResellerCustomerData","customer_name")?has_
 content>${ResellerCustomer.getFieldValue("ResellerCustomerData","customer_
 name")!},${ResellerCustomer.getFieldValue
 ("ResellerCustomerData","forename")!}<#else> ${ResellerCustomer.getFieldValue
 ("ResellerCustomerData","customer_name")!}</#if><#if
 ResellerCustomer.getFieldValue("ResellerCustomerData","phone")?has_content>
 (${ResellerCustomer.getFieldValue("ResellerCustomerData","phone")!})</#if>
```

Please note that you might have to adapt the template configuration if you perform an update from a CM version previous to 6.11 to version 6.11 or higher. In previous versions, the first line, i.e., fields with position (0,x), of customer fields (former Data Object Group Fields) was used to display the basic customer data in the header of the customer page. In case you had adapted the display by setting the visibility of some fields to “edit”, you might want to remove this annotation.

D.5.5.9 Ticket List

This template defines the presentation of the customer data in the ticket list.

If you would like to work with this template type, please make sure that the page customization parameter *accordionTicketList.mainCustomerDescriptionVisible* is set to “true”. Otherwise customer data cannot be displayed in the ticket list.

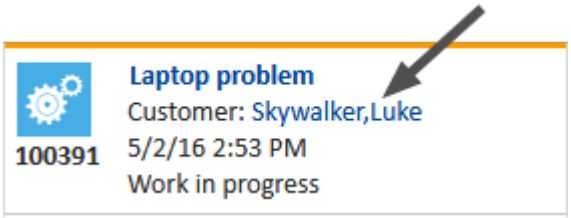


Figure 301: *ConSol CM Web Client - Customer ticket list template*

D.5.5.10 Ticket Relation

This template defines the presentation of the customer data in ticket references in the *Relations* section of a ticket. Please keep in mind that customer data of referenced tickets are only displayed in visibility level “extended”.

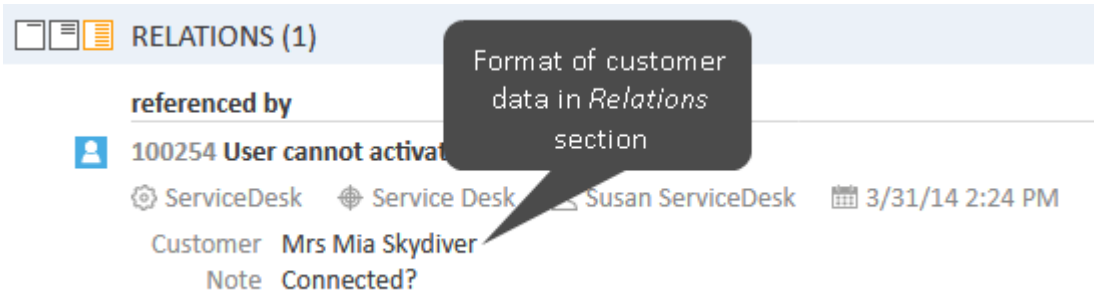


Figure 302: *ConSol CM Web Client - Customer ticket relation template*

D.5.5.11 Workspace and Favorites

This template defines the presentation of customer data in the *Favorites* section.

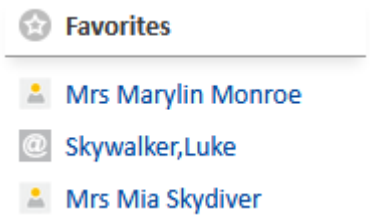


Figure 303: *ConSol CM Web Client - Customer favorites template*

D.5.5.12 History

This template defines the presentation of customer data in the ticket protocol, i.e., in the *History* section of a ticket.

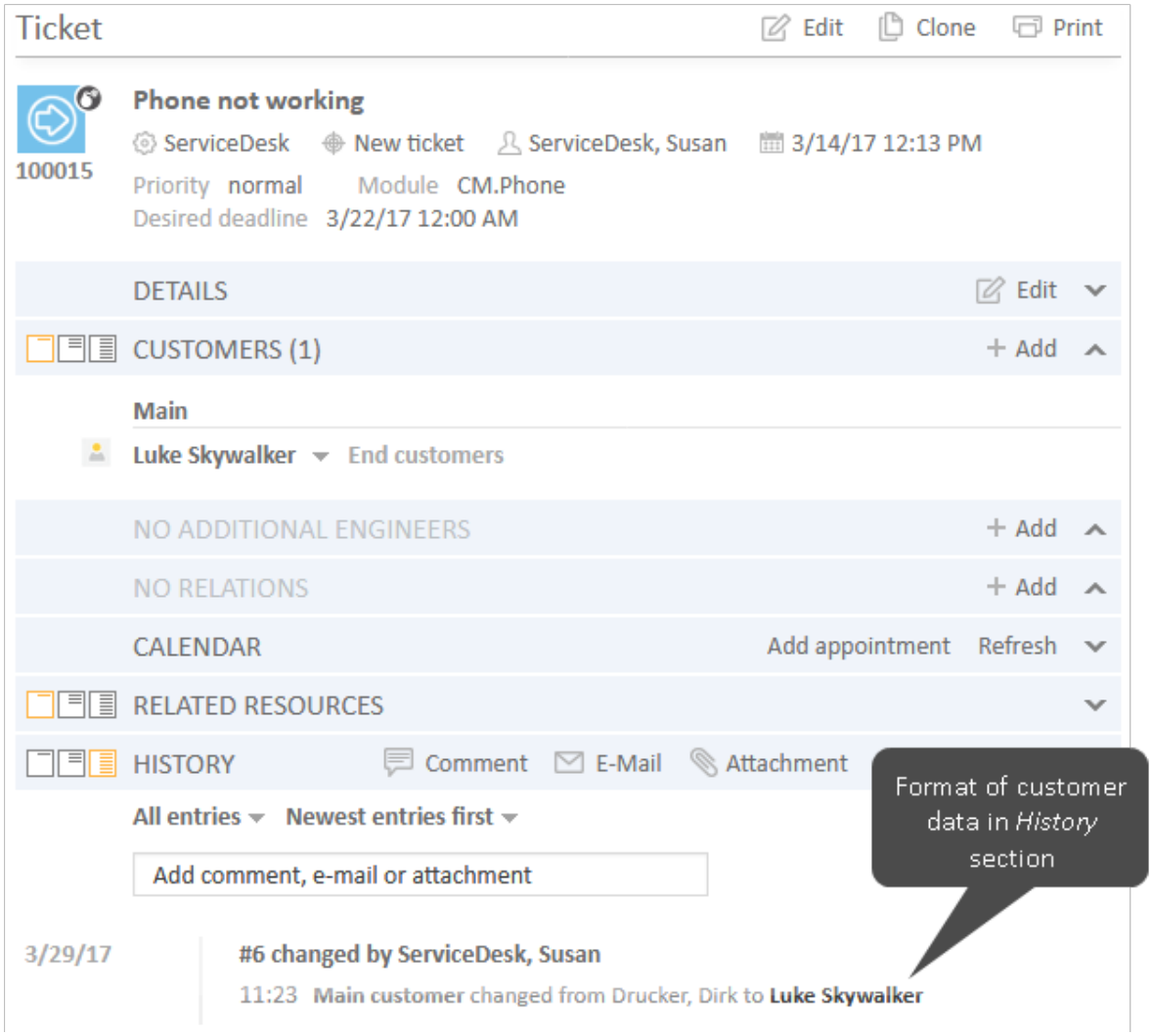


Figure 304: *ConSol CM Web Client - Customer history template*

D.5.5.13 Suggestion

This template defines the presentation of customer data for suggestions which are displayed when a ticket is created.

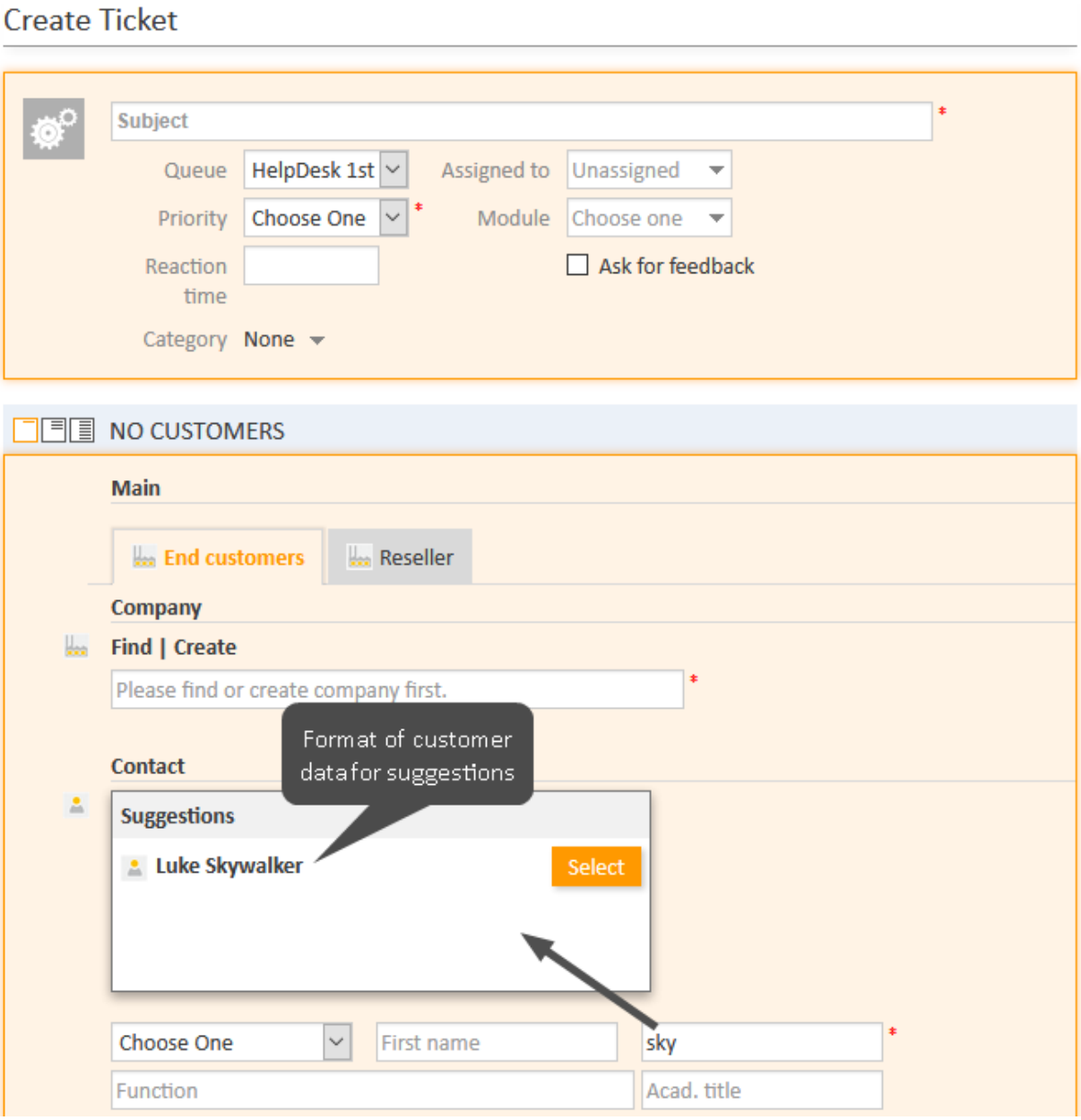


Figure 305: *ConSol CM Web Client - Customer suggestion template*

D.5.5.14 CM/Phone Customer Details

See section CM/Phone: CTI with ConSol CM.

D.5.5.15 CM/Phone Customer List

See section CM/Phone: CTI with ConSol CM.

D.6 Managing Customer Groups

This chapter discusses the following:

D.6.1 Basic Principles of Customer Data Models and Customer Groups

In a ConSol CM system multiple customer groups can be used.

> ⚠ **Principles:**
>
> There can be any number of customer groups and any number of customer data models.
>
> Each customer group has exactly one customer data model.
>
> Each customer data model can be assigned to any number of customer groups.

In the following example, the system contains four customer groups, each with its specific customer data model.

D.6.2 Managing Customer Groups Using the Admin Tool

In the Admin Tool, customer groups are managed using the navigation item *Customer Groups* in the navigation group *Customers*.

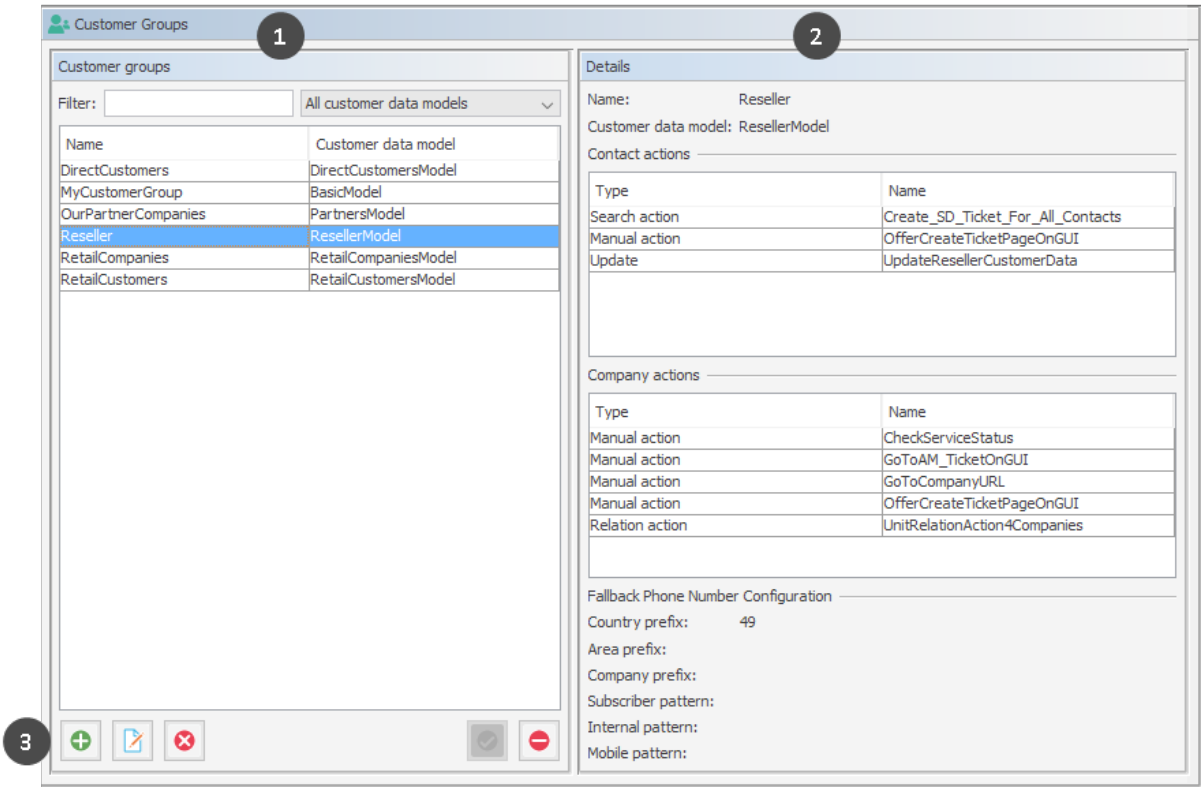


Figure 306: *ConSol CM Admin Tool - Customers, Customer Groups: Managing customer groups*

- List of all customer groups (1). Can be sorted by clicking the column header, and can be filtered.
- Details of the selected customer group (2)
- Buttons to add, edit, delete, activate or deactivate a customer group (3)

D.6.2.1 Customer Groups List

On the left side, all customer groups are listed:

- **Name**
  The technical name of the customer group.
- **Customer data model**
  The name of the customer data model which has been assigned to the customer group.

You can apply two sorts of filters:

- **Name filter**
  Enter a text or some characters in the field *Filter*. Only the customer groups where the name contains the text/characters will be displayed in the list.
- **Customer data model filter**
  Select a customer data model from the drop-down list. Only customer groups with the selected data model will be displayed in the list.

D.6.2.2 Customer Group Details

On the right side, the details of the selected customer group will be displayed. An explanation of all parameters is given in the following section.

D.6.2.3 Creating a New Customer Group

You create a new customer group by clicking the *Add* button below the group list. A pop-up window is opened where you have to enter the customer group parameters.

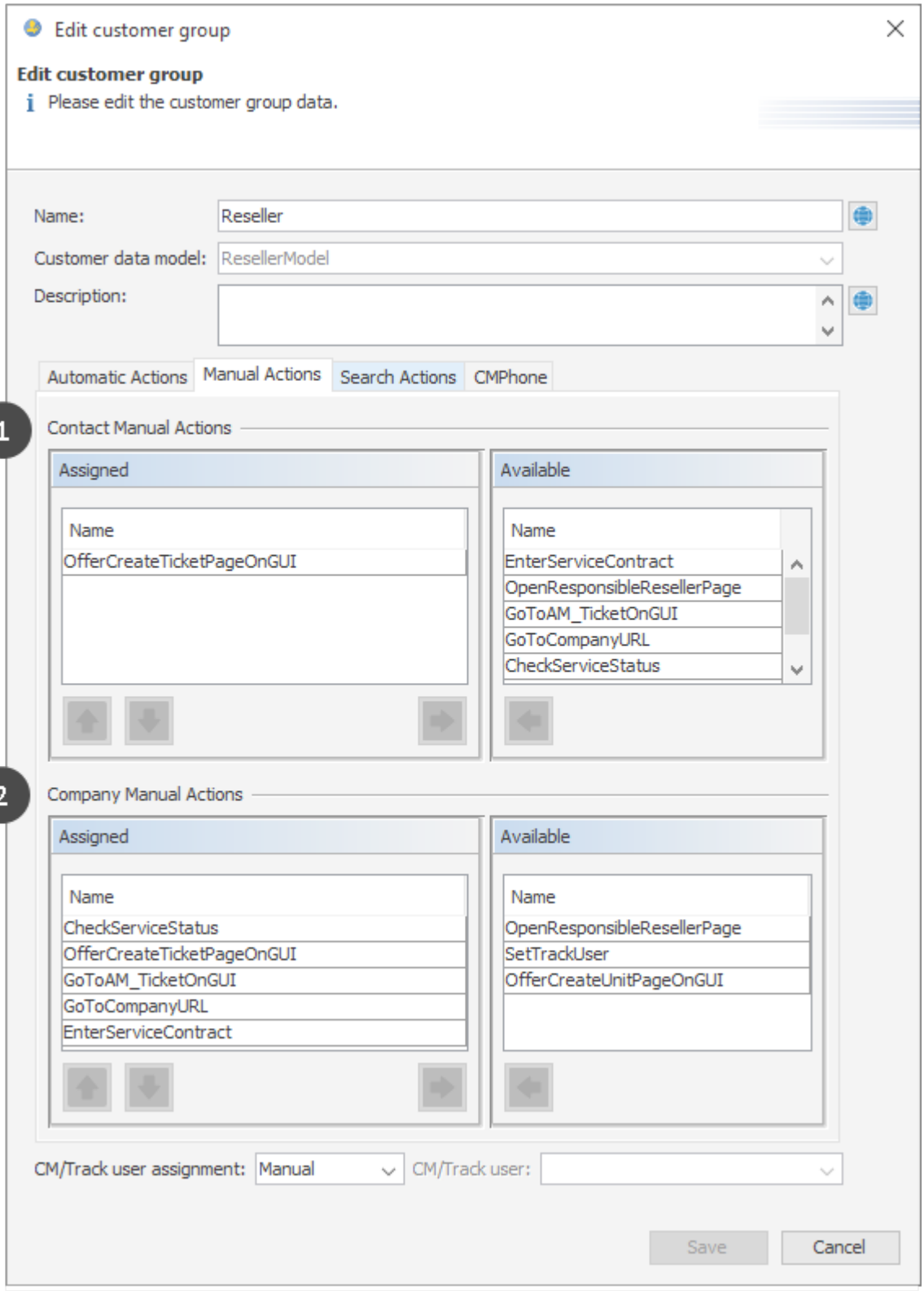


Figure 307: *ConSol CM Admin Tool - Customers, Customer Groups: Parameters for a customer group*

- **Name**
  The unique technical name of the customer group. Can be localized using the *Localize* button. For details, please refer to section Localization of Objects in General, Type 1.
- **Customer data model**
  Select the customer data model from the drop-down list. All customer data models which have been defined (see section Setting Up the Customer Data Model) are available.
- **Automatic Actions**
  Here, automatic customer actions can be assigned to a customer group. All customer actions which have already been defined will be in the lists. You can assign those actions to contacts and/or companies of the customer group. Automatic actions will be triggered when a customer is created, edited or deleted. The entire configuration is explained in detail in section Action Framework - Customer Actions.
- **Manual Actions**
  Manual actions can also be assigned to contacts and/or to companies. All customer actions which have already been defined will appear in the lists. The actions will be listed as activities on the contacts or company page and have to be triggered manually by an engineer, very similar to workflow activities for tickets. You can use the arrow buttons below the lists of assigned actions to define the order of the activities in the Web Client. Details about manual customer actions are provided in section Action Framework - Customer Actions.
- **Search Actions**
  Search actions are offered as activities for result lists of Detailed Searches and have to be assigned to contacts and/or to companies of a customer group. For example, a company search action for the customer group *Reseller* will always be offered in search result lists which contain lists of Reseller companies. All search actions for customers which have already been defined will be offered in the lists. Search actions are part of the Action Framework and are explained in detail in section Action Framework - Search Actions.
- **CMPhone**
  Tab for all CM/Phone parameters. Only available if CM/Phone is active, see section CM/Phone: CTI with ConSol CM.
- **CM/Track user assignment**
  Decide how the CM/Track user profile should be assigned. Possible values are:
  - **Fixed**
    Select a CM/Track user profile (i.e., an engineer which has been defined as CM/Track user profile). This CM/Track user profile is used for all contacts of this customer group. The *Track user* field is not displayed in the Web Client.

    ⚠ As soon as the customer group has contacts, the assignment mode *Fixed* cannot be changed anymore!

  - **Manual**
    Default. The assignment of a CM/Track user profile is done manually by an engineer in the Web Client as described in section Granting Access to CM/Track V2 for Customers. The user profile can also be set using the REST API.

- **None/internal**
  The assignment of a CM/Track user profile can only be done via script, e.g., in a contact action. The *Track user* field is not displayed in the Web Client and the assignment cannot be done via REST API either. You can use the following methods to set a CM/Track user in a contact action script:

```
Engineer trackUser = engineerService.getByName("mytrackuser");
 unitEngineerRelationServiceImpl.updateEngineer(unit, trackUser);
```

Actions can be assigned to the contacts (1) and/or to the companies (2) of the customer group.

D.6.2.4 Editing a Customer Group

If you want to edit a customer group, select it in the list and click the *Edit* button or just double-click the name of the customer group. Modify the customer group parameters and click *Save* to store your modifications.

D.6.2.5 Deleting a Customer Group

Select the customer group you want to delete in the list and click the *Delete* button. If you confirm the following dialog with *Yes*, the customer group will be deleted and will no longer be available in the system. A customer group can only be deleted if it is not assigned to a queue and if there are no tickets for customers of the group. In a system which has been in operation for a while, it will usually not be possible to delete a customer group.

D.6.2.6 Disabling and (Re-)Enabling a Customer Group

To disable a customer group, select the customer group in the list and click the *Deactivate* button. The entry in the list is now shown in italics. Just click the *Activate* button at the bottom of the page to enable the customer group again. If a customer group is disabled, it is not possible to create new tickets for companies or contacts of the group. Tickets of the group are still visible.

D.6.3 Assigning Access Rights for Customer Groups

In order to let engineers work with customer data of a customer group, e.g., to create new reseller data sets or to modify them, you have to grant access permissions for the user groups to one or more roles.

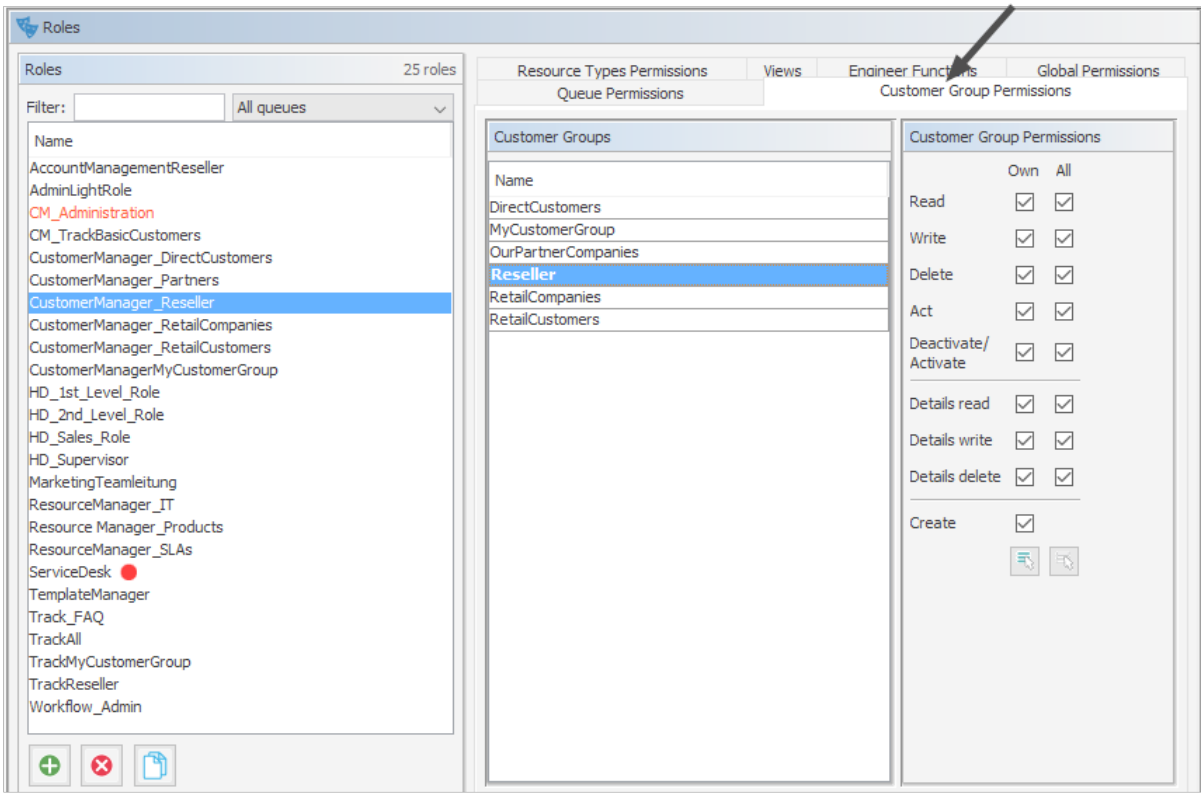


Figure 308: *ConSol CM Admin Tool - Access and Roles, Roles: Assigning permissions on customer groups to a role*

The access rights which can be granted for customer data also comprise the *Comments and Attachments* section of the customer page, i.e. of the company page and/or the contact page.

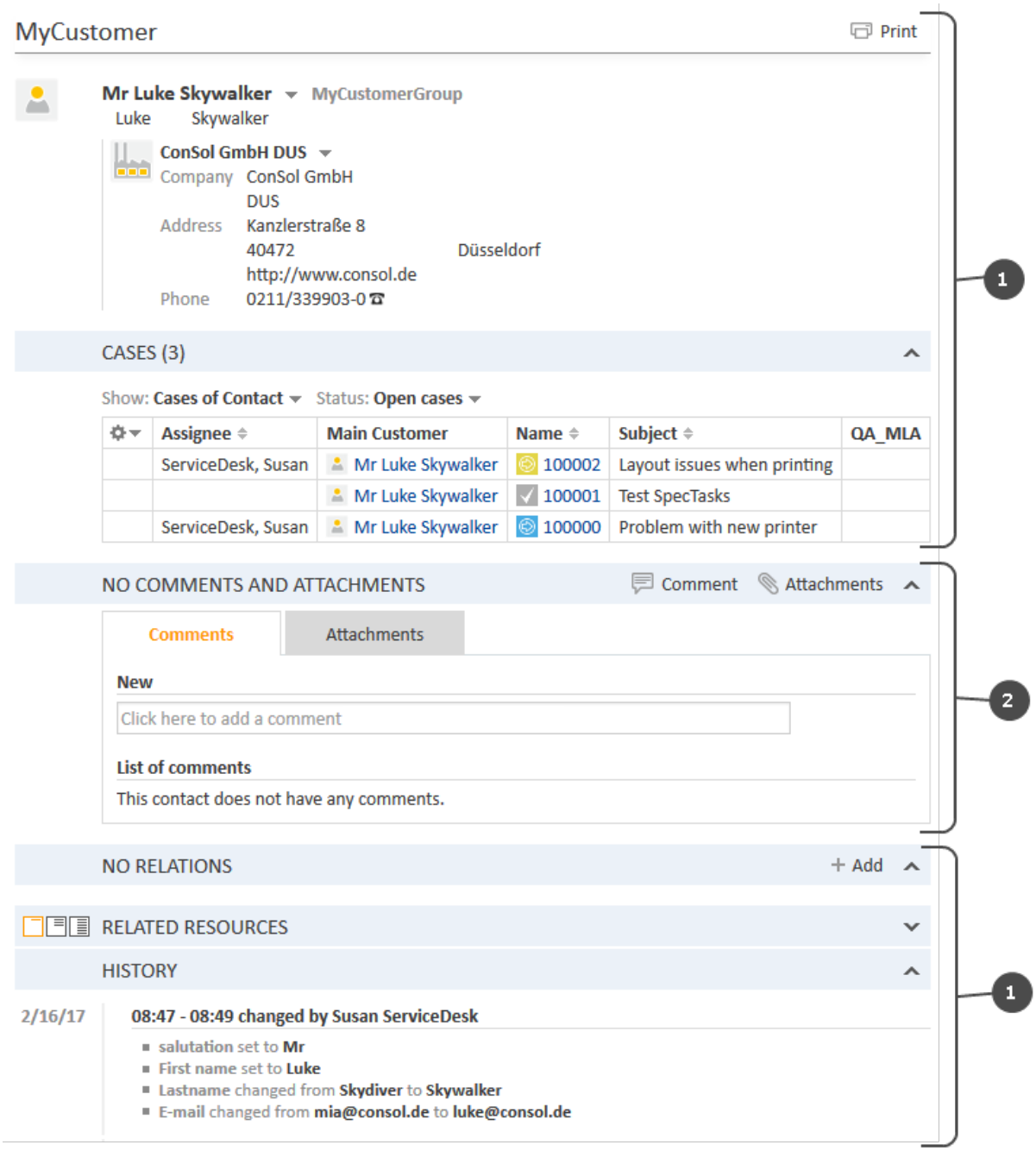


Figure 309: *ConSol CM Web Client - Contact page: Comments and Attachments section*

The following access permissions can be granted:

- **Customer type**
  Refers to the tickets of the customer.

    - **Own**
      All (main or additional) customers of tickets which are currently assigned to the engineer or where the engineer is set as an additional engineer.
    - **All**
      All customers.
- **General sections (1)**
    - **Read**
      Read the customer data.
    - **Write**
      Write/modify the customer data.
    - **Delete**
      Delete a customer data set.
    - **Act**
      Execute actions for this customer (see section Action Framework - Customer Actions for details about customer actions).
    - **Deactivate/activate**
      Deactivate and (re-)activate the customer. It is not possible to create tickets for a deactivated customer.
- **Comments and Attachments section (2)**
    - **Details read**
      Read customer data in the *Comments and Attachments* section.
    - **Details write**
      Write/modify customer data in the *Comments and Attachments* section.
    - **Details delete**
      Delete customer data in the *Comments and Attachments* section.
- **General**
    - **Create**
      Create a customer data set. In a two-level customer data model this refers to contact as well as to company data sets.

⚠ Please keep in mind that an engineer must have at least read permissions for a customer group to open and/or create tickets for customers in this group!

D.7 Customer Roles

This chapter discusses the following:

D.7.1 Introduction

Customer roles help you distinguish different additional customers who are linked to a ticket. For example, it will help a great deal if the engineer finds an incident ticket and instantly knows which of the customers is the manager, who is the technical contact person and who he might have to contact for questions concerning billing.

You can define any number of customer roles. Technically, it is a simple list, and the engineer, working with the Web Client, can (but does not have to) assign a customer role to any additional customer of the ticket. An additional customer can only have zero or one roles, not more than one.

Besides helping engineers in their everyday work, customer roles can also be helpful for designing processes and writing scripts, e.g., to write an email to all managers of all open Service Desk tickets of a company.

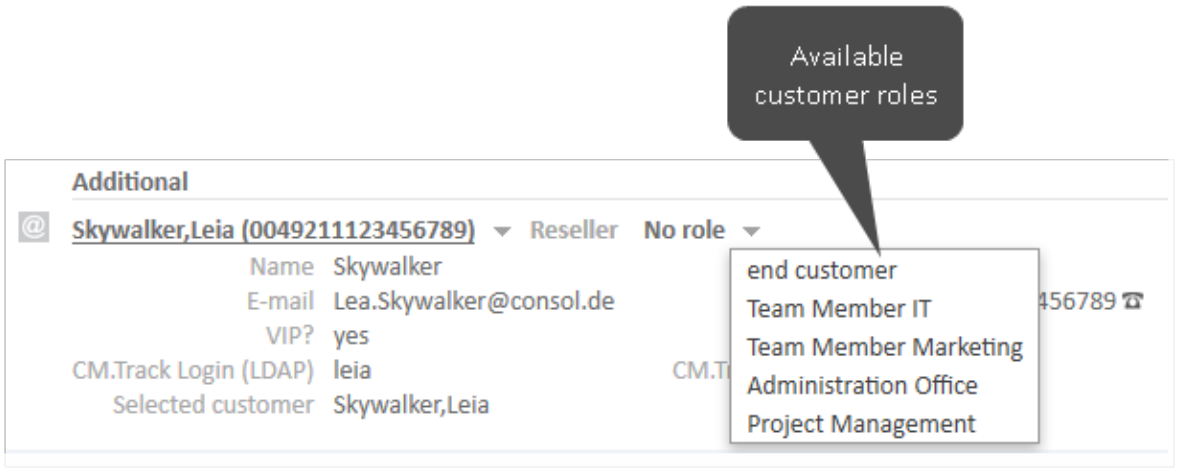


Figure 310: *Customer roles for an additional customer*

D.7.2 Defining Customer Roles Using the Admin Tool

Open the navigation item *Roles* in the navigation group *Customers* to add (create), edit, or delete customer roles.

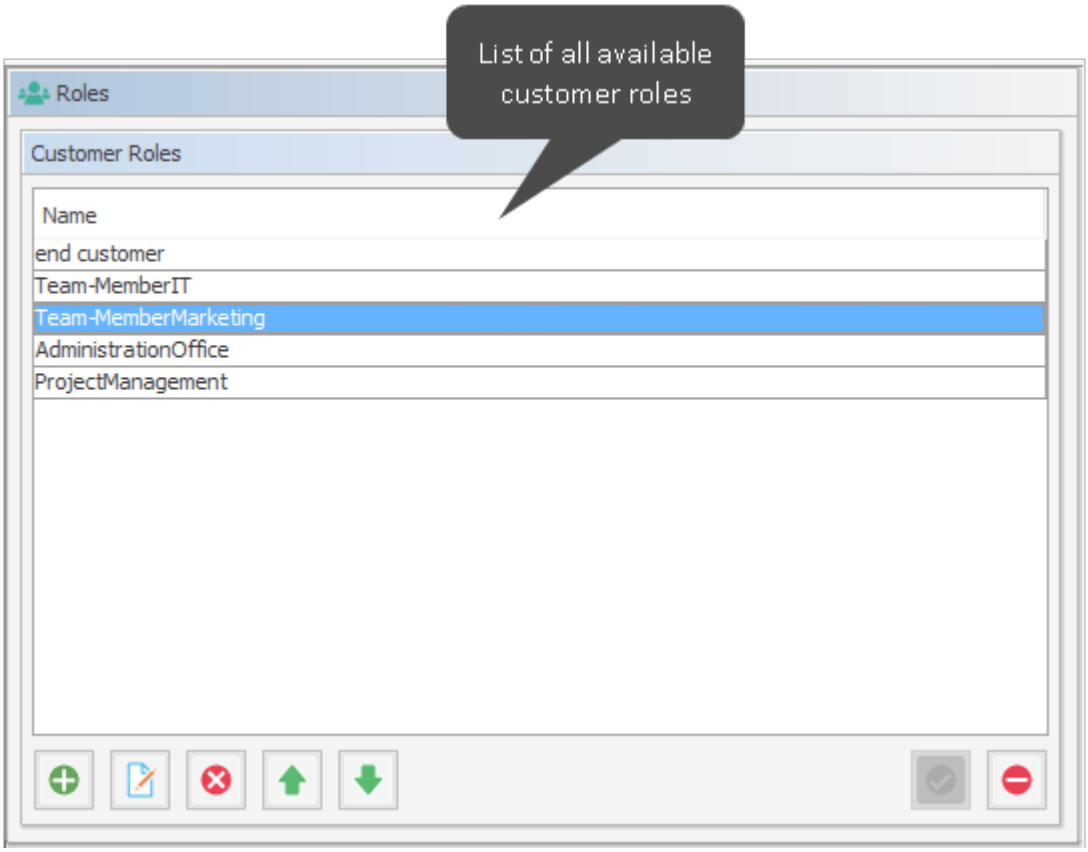


Figure 311: *ConSol CM Admin Tool - Customers, Roles: Customer Roles*

In the Web Client these roles can be assigned to additional customers of a ticket to show the function of these customers, e.g., project manager or end customer.

There are two implications of the assignment of a customer role to an additional customer:

1. It provides information in the Web Client (e.g., you would not want to send a log file to a *manager* but to the *Team Member IT*).
2. The customer role can be used in workflow programming to control the process flow (e.g., send an email to all *Team Members IT* but not to contacts with other roles).

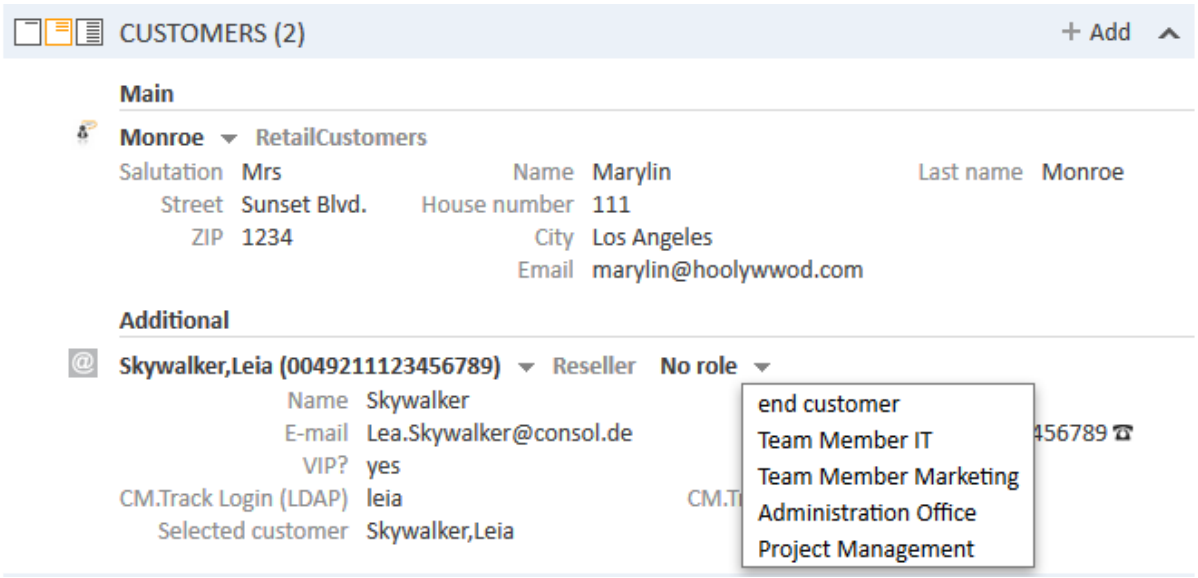


Figure 312: *ConSol CM Web Client - Setting a customer role for an additional customer*

D.7.2.1 Create or Edit a Customer Role

A customer role is defined by its name. By clicking the *Add* button a pop-up window appears where you can enter the name. Using the *Localize* button next to the name field you can localize the name (see below). For details about localization, please refer to section Localization of Objects in General, Type 1. The checkbox *Enabled* is already selected to set the customer role active in the system (see also Disable or Enable a Customer Role). You will see the same window when you click the *Edit* button in order to edit a customer role.

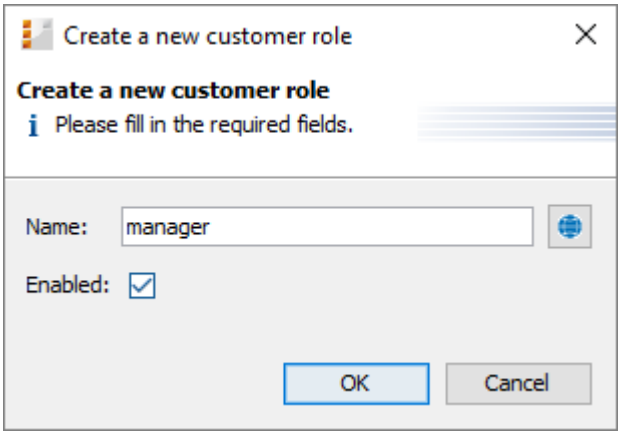


Figure 313: *ConSol CM Admin Tool - Customers, Roles: Create or edit a customer role*

D.7.2.2 Delete a Customer Role

A customer role can only be deleted if it is not assigned to any customers, otherwise you get a warning and can only disable this customer role (see below).

In order to delete a customer role, select it in the list and click the *Delete* button. After choosing *Yes* in the confirmation dialog the customer role will be removed from the list and the system.

D.7.2.3 Disable or Enable a Customer Role

If a customer role is still assigned to a customer but is not needed anymore you can disable it. To do this select the customer role and click the *Deactivate* button. The entry in the list is shown in italics afterwards. The customer role cannot be assigned anymore. Just click the *Activate* button at the bottom of the page if you want to enable the role again.

You can also enable or disable a customer role in the window used for editing customer roles by selecting or de-selecting the *Enabled* checkbox. When you create a customer role, this check box is automatically selected.

D.7.2.4 Localize a Customer Role

Click the *Localize* button in the create or edit window to enter the localized name(s) of a customer role. For details about localization, please refer to section Localization of Objects in General, Type 1.

D.8 Customer Relations

This chapter discusses the following:

D.8.1 Introduction

Customer relations represent relations between customers, i.e., companies and contacts. They can be one of two types:

- **directional** (different levels in a hierarchy)
- **reference** (same level, no hierarchy)

A relation is of one of the following types:

- **company - company**
  e.g., ... *has a cooperation with* ... (company X cooperates with company Y)
  - The companies can belong to the same or to different customer groups.
  - The involved customer groups can have the same or different customer data models.
- **company - contact**
  e.g., ... *is customer of* ... (contact X is customer of company Y)
  - The company and the contact can belong to the same or to different customer groups.
  - The involved customer groups can have the same or different customer data models.
- **contact - contact**
  e.g., ... *is serviced by* ... (contact X from company X is serviced by contact Y from company Y)
  - The companies and contacts can belong to the same or to different customer groups.
  - The involved customer groups can have the same or different customer data models.

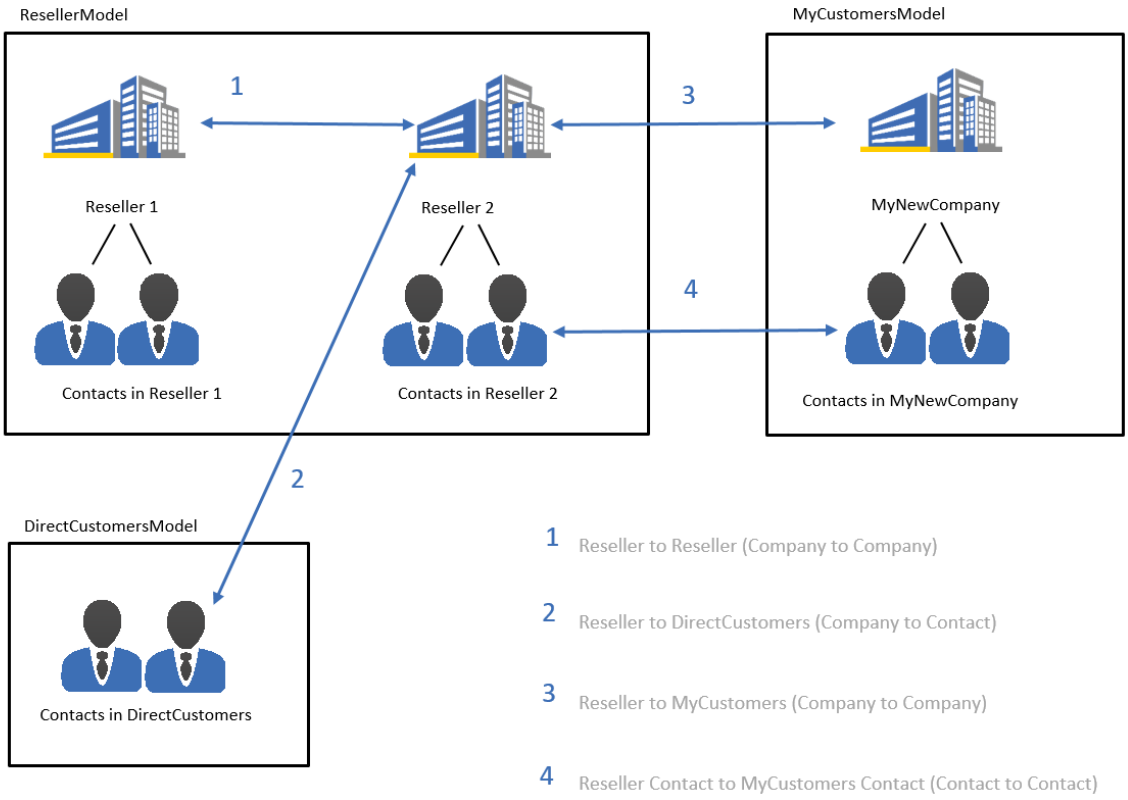


Figure 314: *ConSol CM FlexCDM - Examples of customer relations*

D.8.2 Management of Customer Relations Using the Admin Tool

To make customer relations available to the engineers, the relations have to be defined in the Admin Tool. Open the navigation item *Relations* in the navigation group *Customers*. All relations are listed, new relations can be added, and old ones can be deleted.

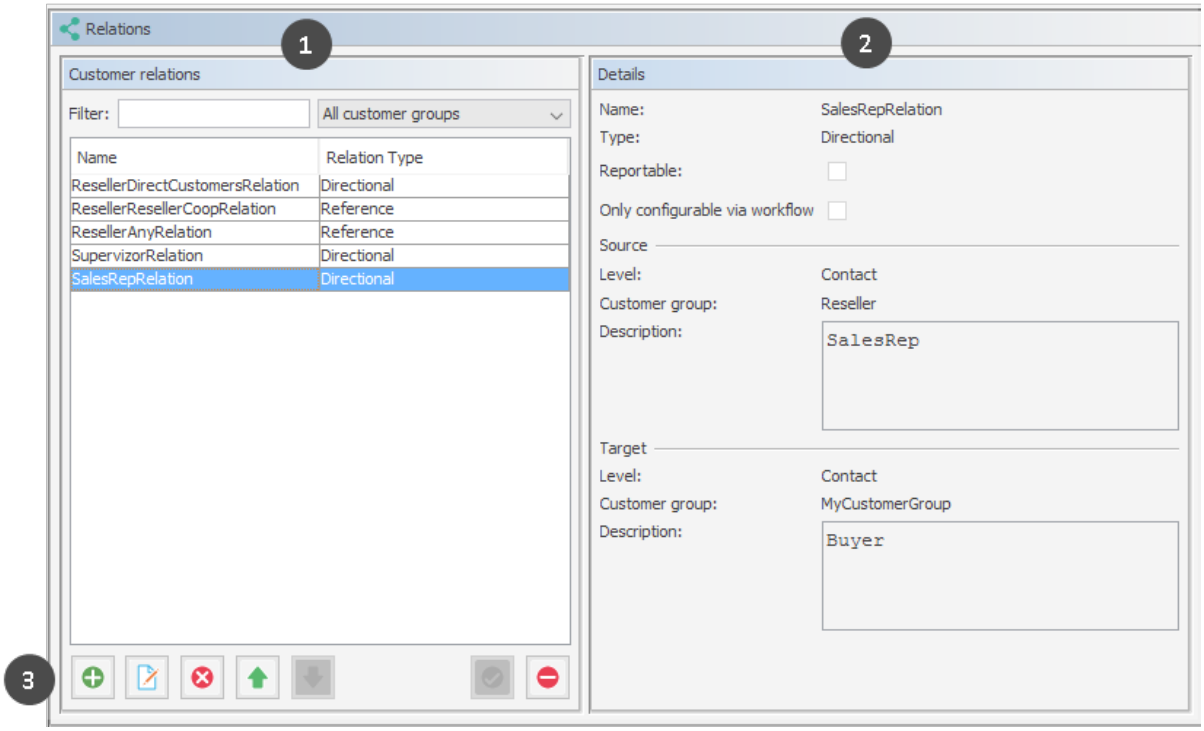


Figure 315: *ConSol CM Admin Tool - Customers, Relations: Managing customer relations*

The following elements are available:

- **List of relations (1)**
- **Filter**
  - Filter for an expression which has to be entered into the field *Filter*. Use the asterisk as a placeholder for any character.
  - Filter for customer groups using the drop-down menu.
- **Relation details (2)**
- **Buttons (3)**
  - **Add button**
    Add a new relation. The pop-up window *Create customer relation* (see next section) is opened.
  - **Edit button**
    Modify the parameters of a relation. The pop-up window *Edit customer relation* (see next section) is opened.

- **Delete button**
  Delete an existing relation. This is only possible when no relations of this type have been set (using the Web Client).
- **Change order (arrows up and down)**
  Place a relation at a specific position in the list. This defines the order of the manual relations as they are displayed in the Web Client.
- **Activate / deactivate relations**
  A deactivated relation is not available in the Web Client, i.e., a relation of this type can no longer be created. Existing relations of this type are not modified and are displayed in the GUI.

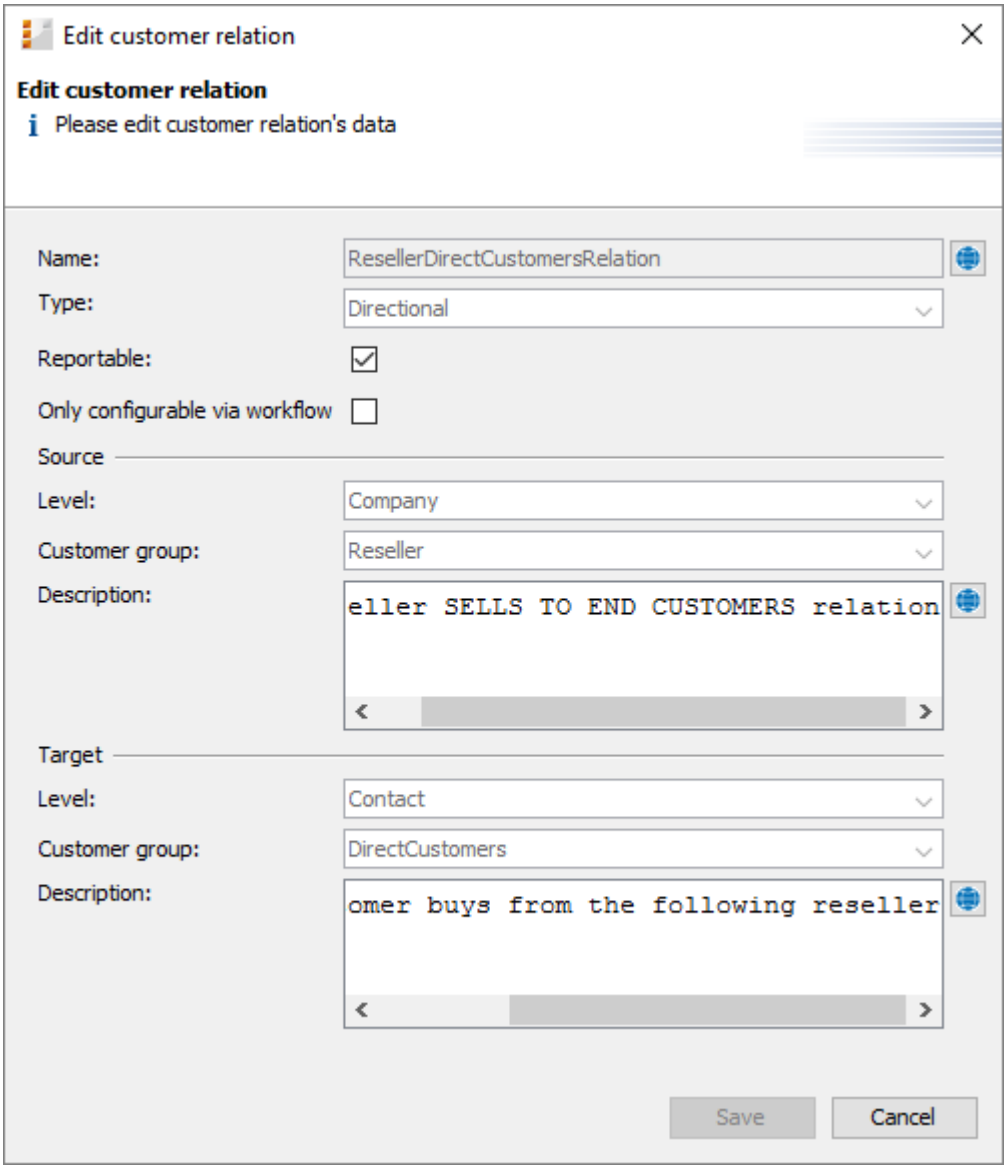


Figure 316: *ConSol CM Admin Tool - Customers, Relations: Details of a customer relation*

To create a new relation, use the *Add* button, to edit a relation use the *Edit* button. In both cases, the detail information pop-up window for a relation is opened where you can edit the following fields:

- **Name**
  Name of the relation. The technical name is used for internal use cases (scripts) whereas the localized name will be displayed in the Web Client (as for most fields in ConSol CM). For details about localization, please refer to section Localization of Objects in General, Type 1.
- **Type**
  Select one of the two types:
  - **Directional**
    A directional relation has a defined source and a defined target. A customer can be source and target for different relation types at the same time. An example for a directional relation is a reseller (company) to end customer (contact) relation: *sells products to*. A company (reseller) sells products to a contact (end customer). A relation between two contacts of a company: *is boss of*. The other direction, *works for*, can also be used, but designing a consistent structure for the entire system, to avoid misunderstandings, is highly recommended.
  - **Reference**
    A reference is an undirected relation with no hierarchical implications, e.g., *has a cooperation with*.
- **Reportable**
  Defines if the relations of this type should be transferred to the data warehouse.
- **Only configurable via workflow**
  If this checkbox is marked then the relation is not available in the Web Client but can only be created via workflow scripts. Such relations cannot be manipulated manually.
- For a directional relation select:
  - **Source**
    - **Level**
      Level of the relation source, i.e., *company* or *contact* or *any* (choose the latter if the source can be either a company or a contact).
    - **Customer group**
      The customer group of the source customer.
    - **Description**
      Will be displayed in the Web Client as the description of the relation on the source side.
  - **Target**
    - **Level**
      Level of the relation target, i.e., *company* or *contact* or *any* (choose the latter if the target can be either a company or a contact).
    - **Customer group**
      The customer group of the target customer.

- **Description**
  Will be displayed in the Web Client as the description of the relation on the target side.

D.8.3 Creating Customer Relations Using the Web Client

In spite of this being an administrator's manual, we will show you how relations are set using the Web Client, because, as an administrator, you should always know what the consequences of administration modifications are.

As an engineer who has the access permissions to the source and to the target customer group, you can add a relation of one customer to another in the *Relations* section of the source customer. You have to have at least one role with the access permission *Write* for the source customer group and the target customer group to perform this operation. You can set the relations on the customer-specific page, i.e., open the company page or the contact page of the potential source object.

For example, you can create a relation *Sells to end customers* from a company in the *Reseller* customer group to a contact in the *End customers* customer group. Of course, this relation has to be defined in the Admin Tool first. Use the *Add* link in the *Relations* section and then select the relation from the drop-down menu. Enter the target name (e.g., contact name) in the auto-complete text field. You can also add a note. Then click *OK*. The relation can be edited or deleted afterwards using the respective links (*Edit*, *Remove*).

Please refer to the *ConSol CM User Manual* for a detailed explanation of working with customer relations.

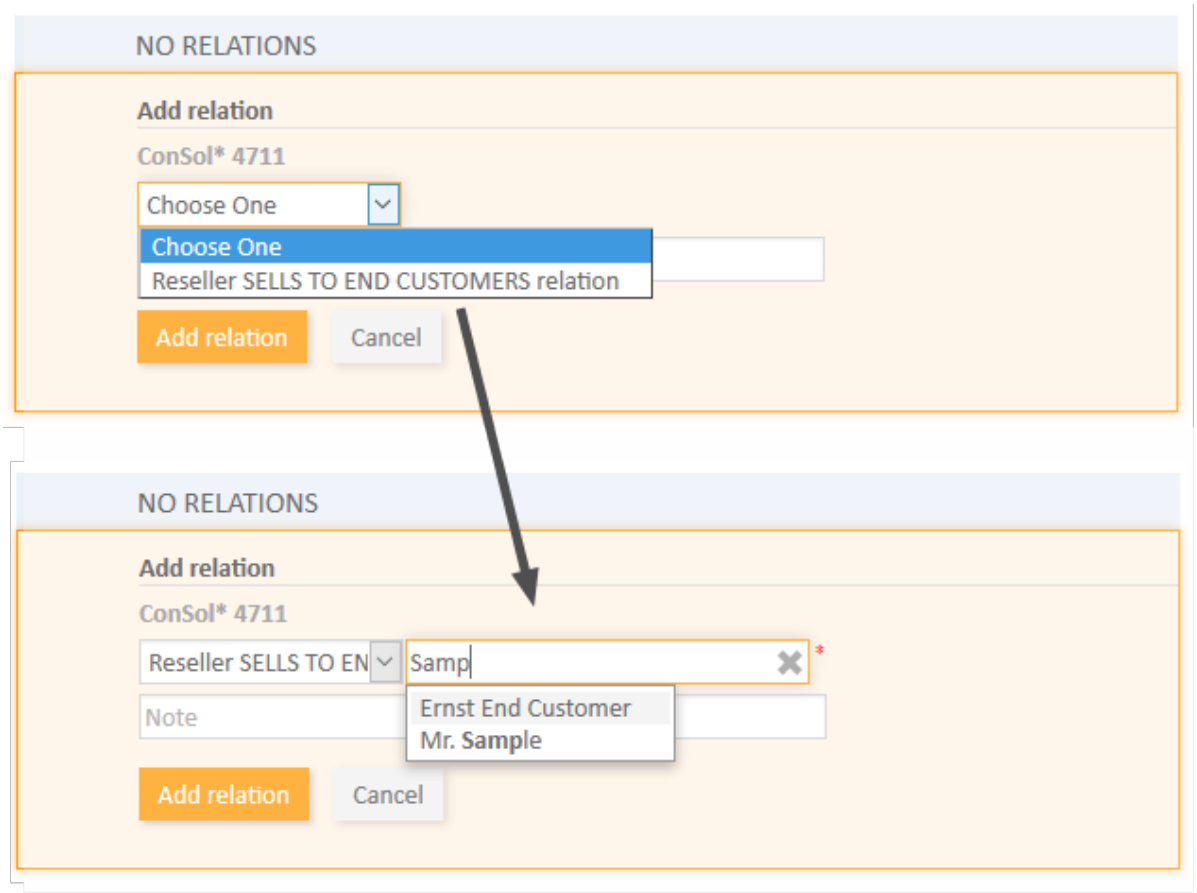


Figure 317: *ConSol CM Web Client - Setting a relation*

D.8.4 Scripting Using Relations

A new class, the `UnitRelationService`, is available. For details please refer to the *ConSol CM API Java Doc*.

In this book we will use the terms *customer* and *customer definition*. In previous versions of ConSol CM, the term *data object* was used to refer to customers. However, the names of the corresponding Java classes are *Unit* and *UnitDefinition*. All other Java classes which deal with customer objects are also still named *Unit*... Please keep that in mind if you work as both a ConSol CM administrator and programmer. Please refer to the *ConSol CM Java API Doc* for details.

```
// Creates the unit relation
UnitRelation create(UnitRelation pUnitRelation)
// Deletes the unit relation
void delete(UnitRelation pUnitRelation)
// Get a set of relations by criteria
PageResult<UnitRelation> getByCriteria(UnitRelationCriteria pCritieria)
// Gets unit relations by source and target units
Set<UnitRelation> getBySourceAndTarget(Unit pSourceUnit, Unit pTargetUnit)
// Gets unit relations by source or target units
Set<UnitRelation> getByUnits(Collection<Unit> pUnits)
// Updates the unit relation
void update(UnitRelation pUnitRelation)
```

Please refer to the *ConSol CM Process Designer Manual* for a detailed explanation on how to write scripts which use customer relations.

D.8.5 Customer Relations to Resources

The relations of resources to contacts or to companies are always defined as resource relations, i.e., defined for resources, not for customers. The company or contact objects are assigned as potential target objects when the resource relation is defined. resource relations are explained in section CM/Resource Pool - Resource Relations.

D.9 Action Framework - Customer Actions

This chapter discusses the following:

D.9.1 Introduction

Customer actions (also called *Unit actions*, previously called *Data Object Actions)* are system actions which are either triggered manually or by a certain incident concerning a customer, i.e. a customer is deleted or a customer is modified. Customer actions are components of the ConSol CM Action Framework. They can be performed for a customer, i.e., a contact or a company. The actions can be performed automatically by the system or manually, triggered by an engineer (via Web Client) who has the required permissions. You might want to apply customer actions for use cases like the following:

- Load additional data into a company's data set.
- Build an automatic report about company-specific KPIs.
- Transfer ConSol CM data to another system (e.g., an ERP system).
- Create/update a Google Maps (see Trademarks) link within the address data.

You can use the following types of customer actions:

- **Automatic** actions which are performed by the system after one of the following customer operations:
  - CREATE
  - UPDATE
  - DELETE
  - RELATION
  - SEARCH
- **Manual** actions which are performed by an engineer using *Activities* links in the customer page (company or contact page) of the Web Client (similar to *Workflow activities* for tickets). Which actions are available for manual execution depends on the customer which is currently being displayed, i.e., when the company page is open, company actions will be offered, when the contact page is open, contact actions will be offered.

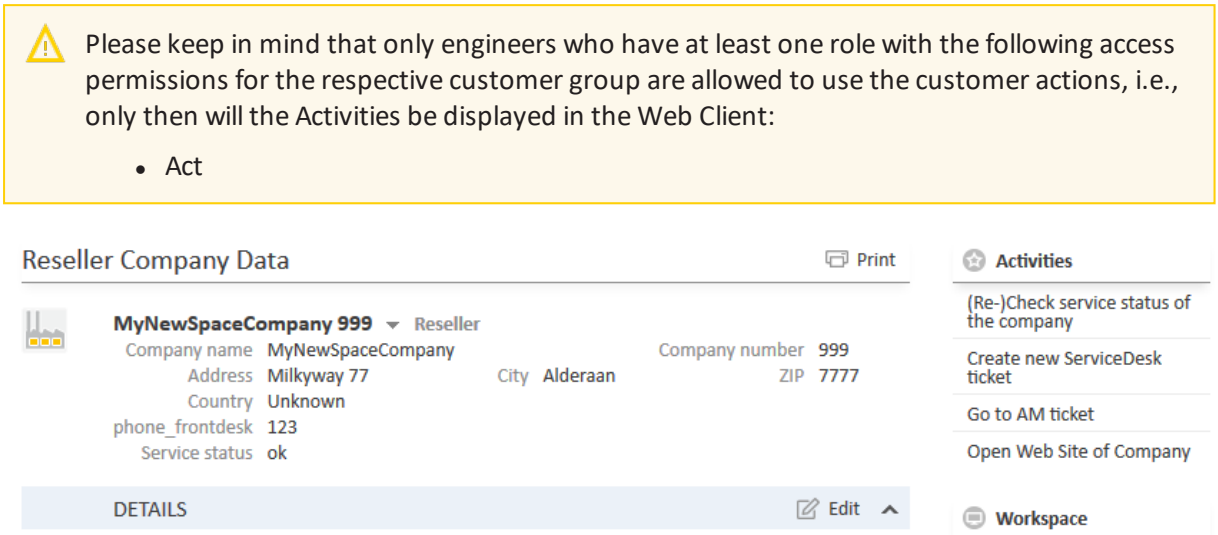


Figure 318: *ConSol CM Web Client - Example for manual customer activities*

Customer actions are based on Groovy scripts which are stored in the *Scripts and Templates* section of the Admin Tool.

The execution of customer actions can be controlled using condition scripts, i.e., you can implement a customer condition script which is run before the action script of the customer action. The action script is only run if this condition script returns “true”.

So there are two types of scripts you have to deal with when you use the ConSol CM Action Framework:

- **Customer action scripts**
  Defines the action which should be performed.
- **Customer condition scripts**
  Defines one or more conditions for the display of the action in the Web Client. Has to return “true” or “false”. If “false” is returned the action is not displayed on the GUI and therefore cannot be performed.

When you want to implement a customer action you have to proceed in three steps:

1. Create a script for the customer action (either an action script only or an action script and a condition script).
2. Create the customer action(s) which use(s) the script(s).
3. Assign the customer action(s) to the customer group(s) where they should be available. You can assign the actions to contacts and/or companies of a customer group.

In the following sections, all three steps are explained in detail.

D.9.2 Managing Customer Actions Using the Admin Tool

In this book we will use the terms *customer* and *customer definition*. In previous versions of ConSol CM, the term *data object* was used to refer to customers. However, the names of the corresponding Java classes are *Unit* and *UnitDefinition*. All other Java classes which deal with customer objects are also still named *Unit*... Please keep that in mind if you work as both a ConSol CM administrator and programmer. Please refer to the *ConSol CM Java API Doc* for details.

D.9.2.1 Step 1: Write the Customer Action Script

Create a new Admin Tool script of type *Customer action*. If required, create another script of type *Customer condition*.

For a detailed explanation of Admin Tool scripts, please refer to section Admin Tool Scripts. For an introduction to Admin Tool scripts used for customer actions, please read section Scripts for the Action Framework: Programming Customer Actions in this chapter.

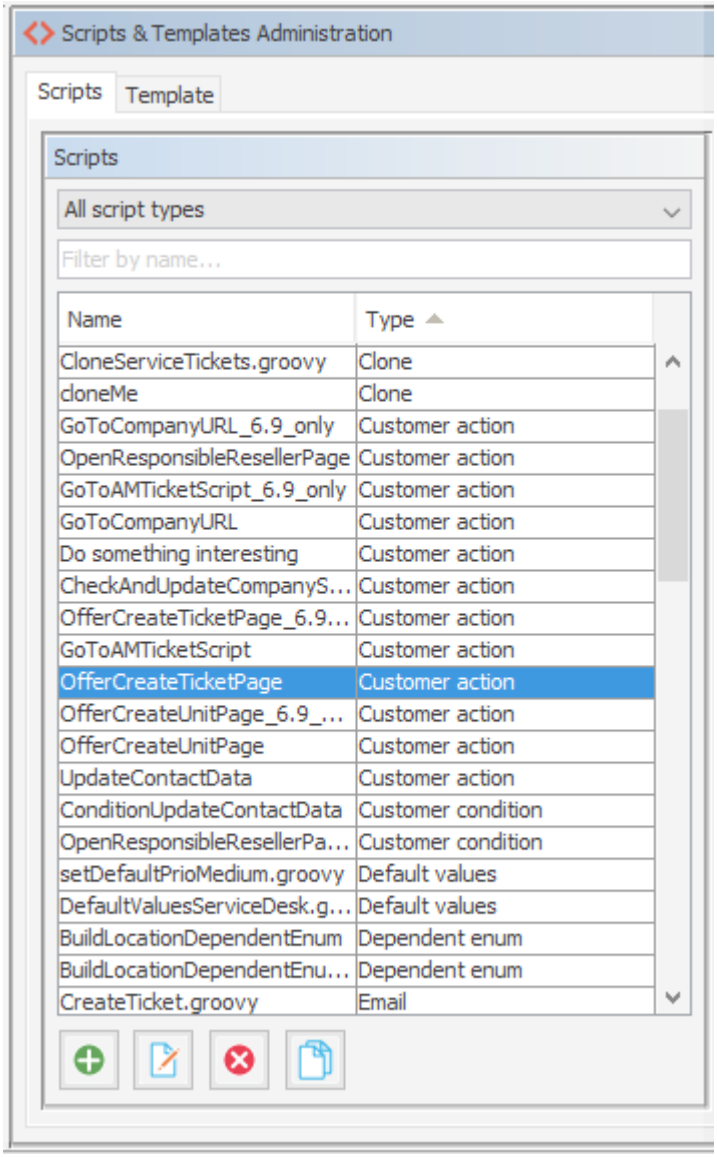


Figure 319: *ConSol CM Admin Tool - System, Scripts and Templates: Scripts for customer actions*

```
//create and return action result that will tell the web to create a new ticket
 with unit as
//a main customer
def queueId = queueService.getByName("Helpdesk").getId();
Map<String, Object> valuesMap = new HashMap<String, Object>
valuesMap.put(PostActionParameter.UNIT_ID, unit.getId())
valuesMap.put(PostActionParameter.QUEUE_ID, queueId)
return unitActionScriptResultFactory.getPostAction(PostActionType.CREATE_TICKET,
 valuesMap)
```

Code example 38: *Customer action script for CM version 6.9.4*

```
// offer Create Ticket page for a new Service Desk ticket
import com.consol.cmas.core.server.service.action.PostActionType
import com.consol.cmas.common.model.ticket.Ticket
def newtic = new Ticket()
def qu = queueService.getByName("ServiceDesk")
newtic.setQueue(qu)
return actionScriptResultFactory.getPostAction(PostActionType.CREATE_TICKET,
 newtic, unit)
```

Code example 39: *Customer action script for CM version 6.10*

D.9.2.2 Step 2: Create Customer Action(s) Which Use the Script

Open the navigation item *Actions* in the navigation group *Customers* in the Admin Tool and add (= create) a new action object using the *Add* button . The same pop-up window appears for both adding and editing a customer action.

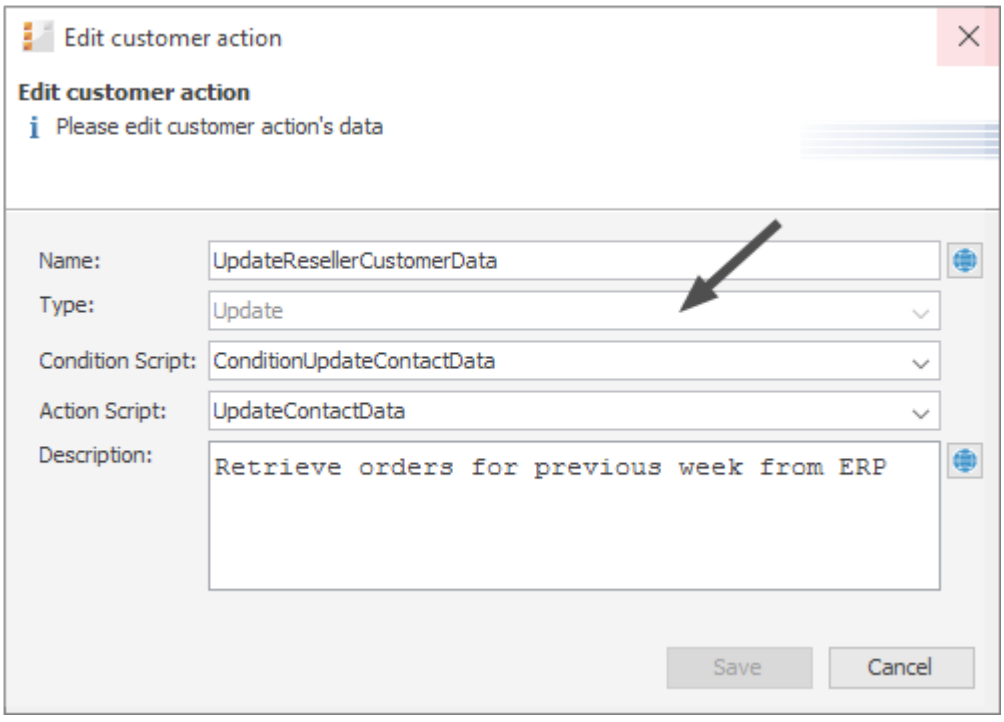


Figure 320: *ConSol CM Admin Tool - Customers, Actions: Editing a customer action*

In the pop-up window, the parameters for the new action have to be defined:

- **Name**
  The unique technical name of the action. Can be localized using the *Localize* button. Localization is required for manual actions because the localized name is displayed under *Activities* in the Web Client. For details about localization, please refer to section Localization of Objects in General, Type 1.

- **Type**
  The action type which defines when it should be executed. Select one of the following types:
  - **Create**
    This script will be executed automatically when the contact/company is created.
  - **Update**
    This script will be executed automatically when the contact/company is updated, i.e., when the data has been modified (either manually or automatically) and is saved again. Starting with CM version 6.10.5.4, the changes which have been performed during the *Update* action can be monitored using the `changes` object. This is explained in detail in section Working with the Changes Object in Customer Update Actions.
  - **Delete**
    This script will be executed automatically when the contact/company is deleted.
  - **Relation**
    This script will be executed automatically when a relation to or from a customer of this customer group is
    - created
    - deleted

    (The script will not be executed when the comment of a relation is changed.)
  - **Manual**
    This script will be available on the contact/company page as a manual activity, provided the engineer has the required access permissions (*act* permission for the respective customer group).
- **Condition Script**
  If a condition script should be run before the action script, the name of the condition script has to be entered here. The action script is only run if the condition script has returned “true”. If there is no condition, just leave this field empty.
- **Action Script**
  The name of the action script which should be run. This has to be the exact name under which the script is stored in the *Scripts and Templates* section of the Admin Tool.
- **Description**
  Enter the description which should be displayed as mouse-over in the Web Client (for manual actions only).

Save the action. You can then assign it to customer groups, as described in the following step.

D.9.2.3 Step 3: Assign Customer Actions to Customer Groups

For the customer action to become effective, you have to assign it to a customer group, at which point it will be available for all customers of this customer group. Depending on the initial definition (contact or company action), the action will be available for contacts or for companies in the customer group. To assign a customer action to a customer group, open the navigation item *Customer Groups* in the navigation group *Customers* of the Admin Tool. Select the customer group you would like to edit and click the *Edit* button to open the pop-up window to assign the customer actions. All cus-

tomer actions which have been stored under *Actions* (see step 2) will be available here, each for the corresponding action type. For example, an action which has been defined for the type *Update* during definition (see step 2) will only be available as an *Update* action.

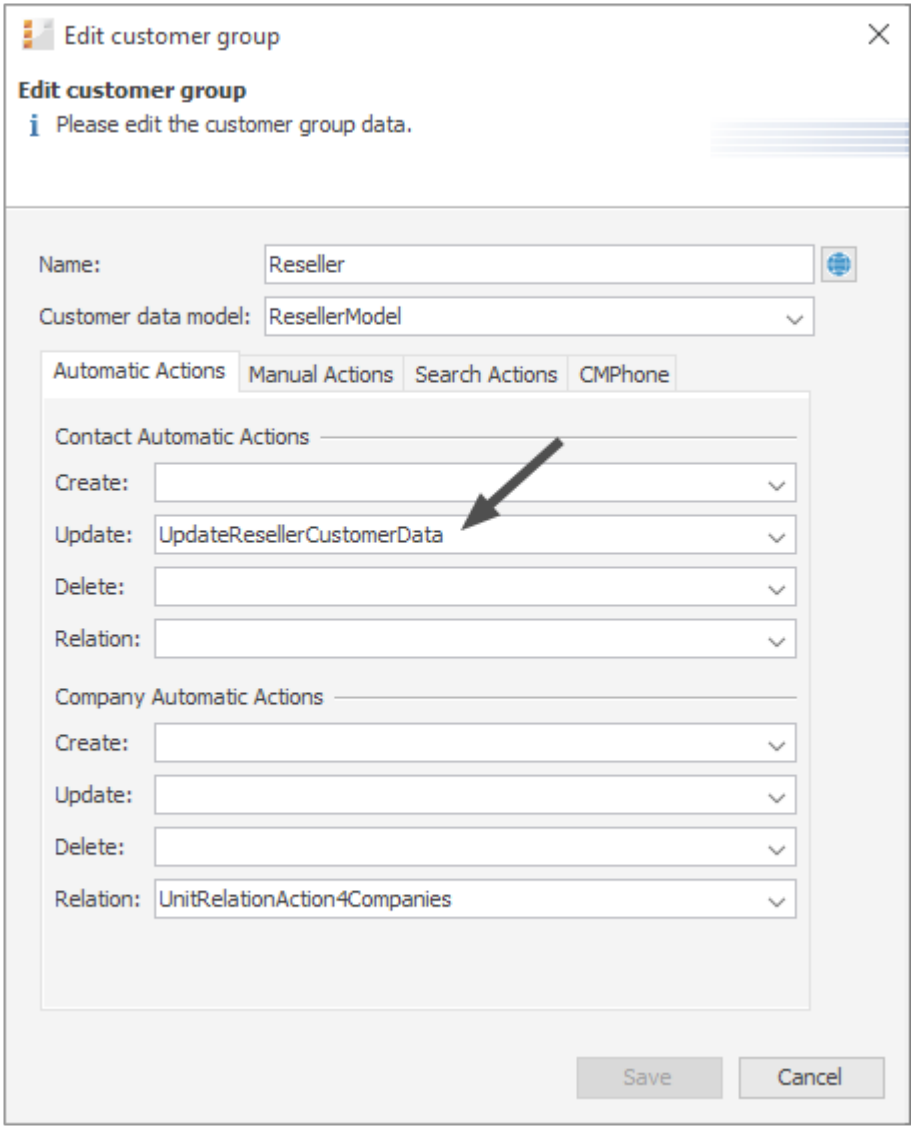


Figure 321: *ConSol CM Admin Tool - Customers, Customer Groups: Assigning customer actions to a customer group*

You can assign the following action types to a customer group:

- **Automatic Actions**
  - **Contact Automatic Actions**
    The script will be executed for the contact.
  - **Company Automatic Actions**
    The script will be executed for the company.

  For each type you can determine the system behavior for the following actions:

  - **Create**
    This script will be executed automatically when the contact/company is created.
  - **Update**
    This script will be executed automatically when the contact/company is updated, i.e., when the data has been modified (either manually or automatically) and is saved again. Starting with CM version 6.10.5.4, the changes which have been performed during the *Update* action can be monitored using the `changes` object. This is explained in detail in section Working with the Changes Object in Customer Update Actions.
  - **Delete**
    This script will be executed automatically when the contact/company is deleted.
  - **Relation**
    This script will be executed automatically when a relation to or from a customer of this customer group is
    - created
    - deleted

    (The script will not be executed when the comment of a relation is changed.)
- **Manual Actions**
  This script will be available on the contact/company page as a manual activity, provided the engineer has the required access permissions (*act* permission for the respective customer group).
- **Search Actions**
  These are explained in section Action Framework - Search Actions.

D.9.3 Using Customer Actions as an Engineer (User)

As an engineer (user), only the customer action type *manual* is relevant for you. The *CREATE, UPDATE* and *DELETE* scripts run in the background.

Manual actions are offered in the Web Client, similar to workflow activities for a ticket. Please see *Example 1* in the next section.

D.9.4 Examples for Customer Action Scripts

D.9.4.1 Example 1: Simple Manual Action

A manual action is coded and stored as an Admin Tool script, then a company action is defined using the script, and the action is assigned to a customer group.

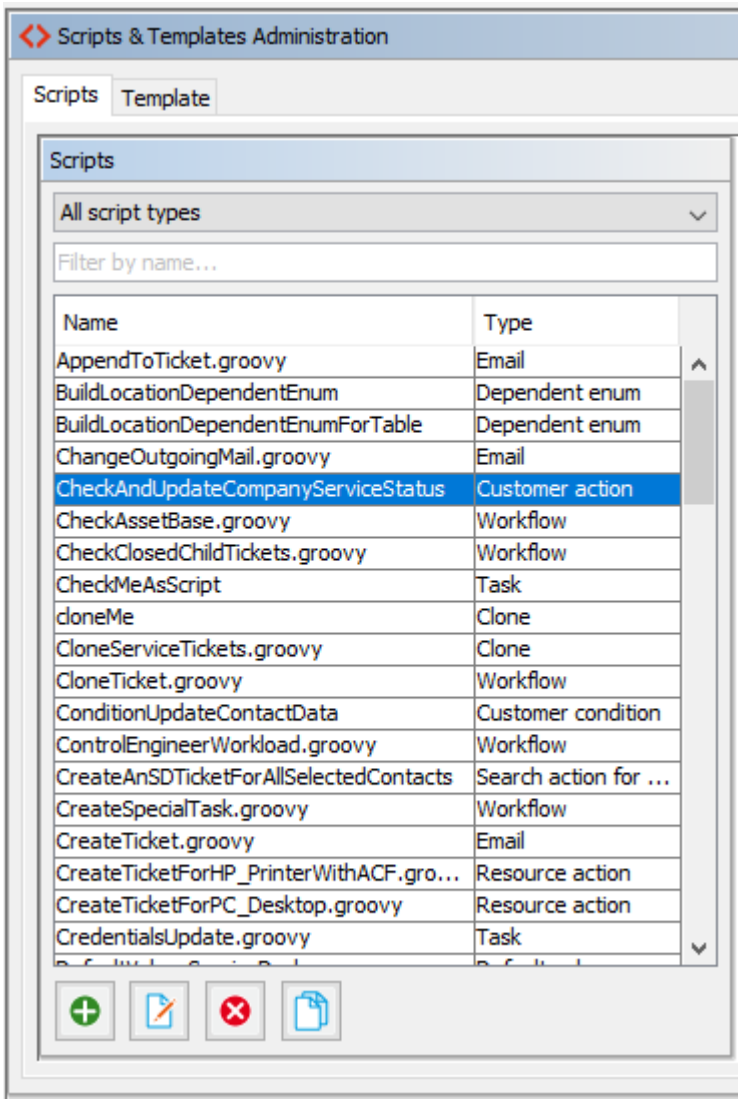


Figure 322: *ConSol CM Admin Tool - System, Scripts and Templates: Customer action script for a company action*

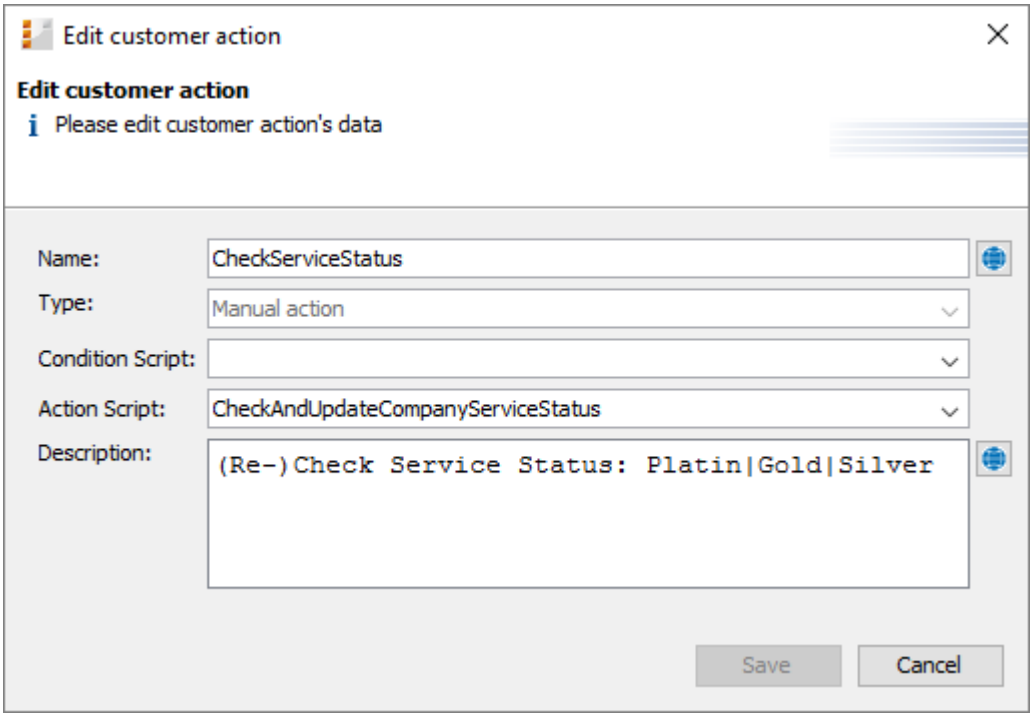


Figure 323: *ConSol CM Admin Tool - Customers, Actions: Defining the company action*

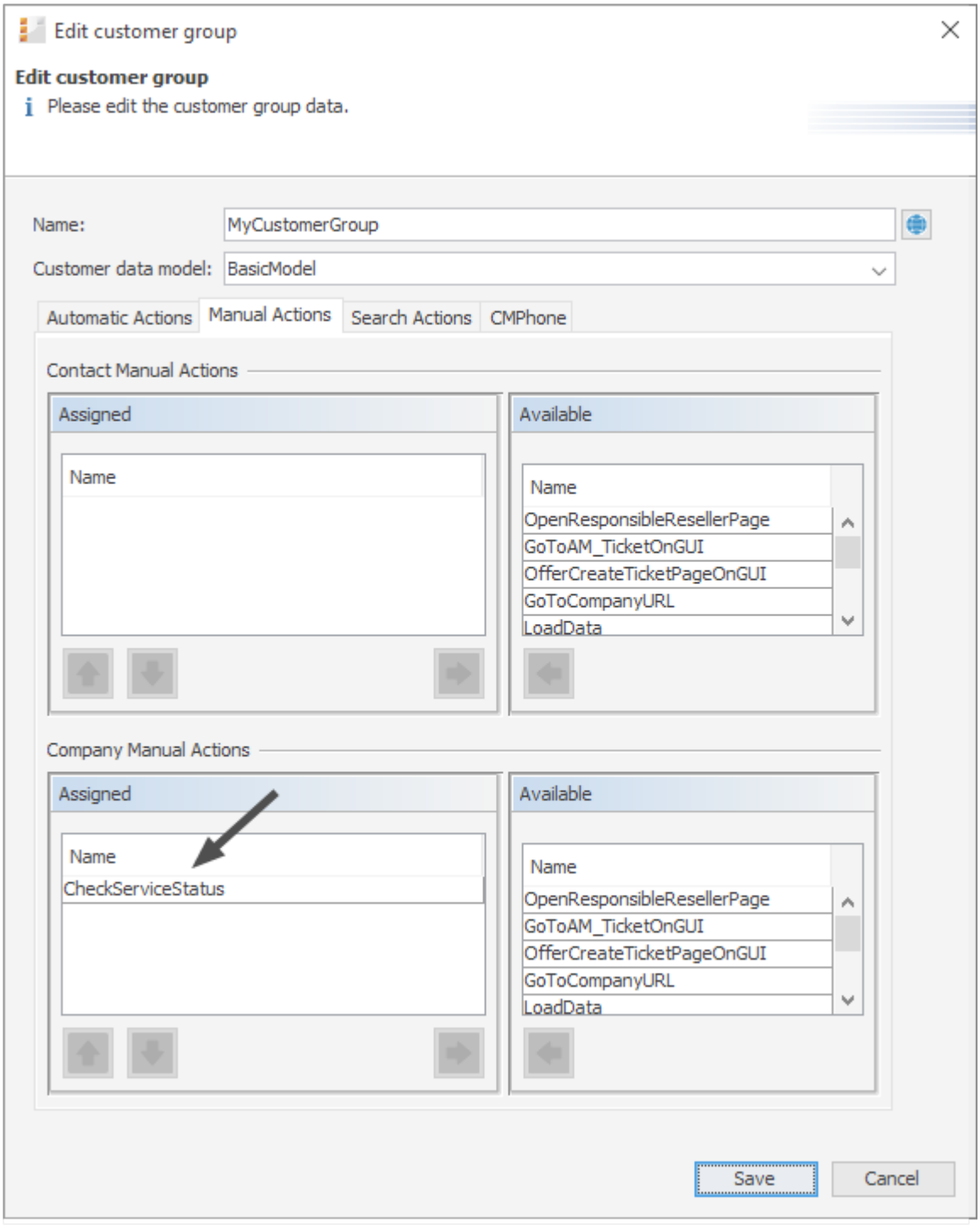


Figure 324: *ConSol CM Admin Tool - Customers, Customer Groups: Assigning a company action to a customer group*

The engineer can use the action manually in the Web Client.

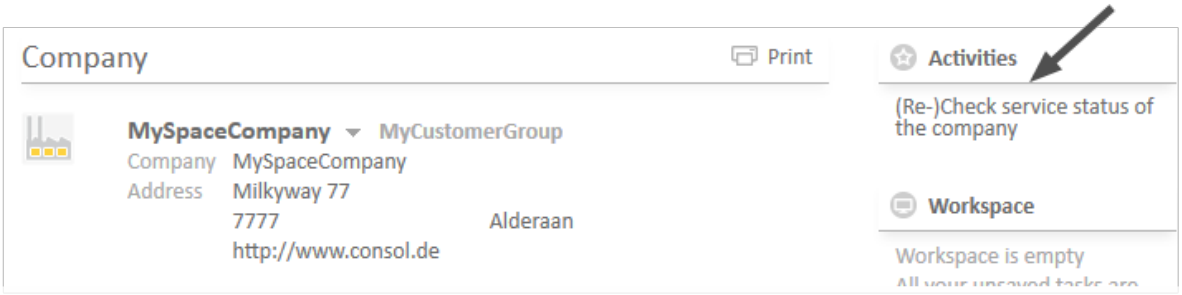


Figure 325: *ConSol CM Web Client - Using a manual company action*

A script similar to the following could be used:

```
// check service status for a Reseller company and set new status
// this is an example for documentation purposes

import com.consol.cmas.common.model.customfield.enums.EnumValue
import com.consol.cmas.core.server.service.action.PostActionType

// ... do something, e.g., reference an external system to find the current service
 status of the company) ...
def ser_stat = enumService.getValueByName("service_status","ok")

// set the new service status for the company
unit.set("ResellerCompanyData:service_status","ok")
unitService.update(unit)

return actionScriptResultFactory.getPostAction(PostActionType.SUCCESS,
 "info.dataobject.action.success").withRefreshContent();
```

Code example 40: *Customer action script, CM version 6.10*

D.9.4.2 Example 2: New Ticket for a Customer

The following script opens the *Create ticket* page for the contact or company the action was performed for. The target queue is *ServiceDesk*, enabling a new Service Desk ticket to be created in no time for the open contact or company. For an introduction to Admin Tool scripts for the Action Framework, please read the following section.

```
import com.consol.cmas.common.model.scripting.unit.PostActionParameterimport
import com.consol.cmas.core.server.service.UnitActionScriptResultFactoryimport
import com.consol.cmas.common.model.scripting.unit.PostActionType

def queueId = queueService.getByName("ServiceDesk").getId();
Map<String, Object> valuesMap = new HashMap<String, Object>()
valuesMap.put(PostActionParameter.UNIT_ID, unit.getId())
valuesMap.put(PostActionParameter.QUEUE_ID, queueId)
return unitActionScriptResultFactory.getPostAction("createTicket", valuesMap)
```

Code example 41: *Customer script (CM version 6.9.4)*

```
import com.consol.cmas.core.server.service.action.PostActionType
import com.consol.cmas.common.model.ticket.Ticket
def newtic = new Ticket()
def qu = queueService.getByName("ServiceDesk")
newtic.setQueue(qu)
return actionScriptResultFactory.getPostAction(PostActionType.CREATE_TICKET,
 newtic, unit)
```

Code example 42: *Customer script (CM version 6.10)*

Please note that in case a ticket should be created with a company as main customer, the checkbox *Company as customer* must be checked for the customer data model of the respective customer group.

D.9.5 Scripts for the Action Framework: Programming Customer Actions

Customer actions are defined by Admin Tool scripts, i.e., by Groovy scripts which are stored in the *Scripts and Templates* section of the Admin Tool. The predefined object `unit` (i.e., an object of class `Unit`) is available for those scripts. Objects of the class `Unit` can represent a company or a contact, depending on the context.

There are two types of scripts for the Action Framework:

- Customer action scripts
- Customer condition scripts

D.9.5.1 Customer Action Scripts

The actions in this script are either triggered automatically by the system operations *CREATE*, *UPDATE*, or *DELETE* or by a manual action of the engineer (using *Activities* in the Web Client).

Automatic Customer Action Scripts

```
unit.set("personalData.name", "Skywalker")
unitService.update(unit)
```

Code example 43: *Set a value in customer data and update the unit*

> ⚠ When you use `unitService.update(unit)`, as in the example above, you should use a customer condition script to avoid infinite loops. See the notes in section Customer Condition Scripts.

Manual Customer Action Scripts

For manual customer action scripts you can make use of some specific methods and objects.

- **Methods** (**fields of the Interface PostActionType**, com.consol.cmas.core.server.service.action.PostActionType)
  - **CREATE_RESOURCE**
    Open the create resource page
  - **CREATE_UNIT**
    Open the create unit page.
  - **CREATE_TICKET**
    Open the create ticket page.
  - **GOTO_RESOURCE**
    Open the resource page
  - **GOTO_UNIT**
    Open the unit detail page.
  - **GOTO_TICKET**
    Open the ticket page.
  - **GOTO_PAGE**
    Open a web page (URL).
- **Objects**
  - actionScriptResult

For a detailed explanation of the PostActionTypes, please refer to section Scripts for the Action Framework.

Create a Unit

(`PostActionType.CREATE_UNIT`) redirects the user to the create unit page. It uses the optional parameter `PostActionParameter.CUSTOMER_GROUP_ID` to define for which customer group a new unit has to be created and optionally the map of customer fields (`PostActionParameter.FIELDS_MAP`) to fill the unit's customer fields with the values which were passed on.

```
import com.consol.cmas.common.model.customfield.meta.FieldKey
import com.consol.cmas.common.model.customfield.AbstractField
import com.consol.cmas.common.model.customfield.StringField
import com.consol.cmas.common.model.scripting.unit.PostActionParameter
import com.consol.cmas.common.model.scripting.unit.PostActionType

Map<FieldKey, AbstractField<?>> fieldsMap = new HashMap<FieldKey, AbstractField<?>>
 ()
FieldKey firstName = new FieldKey("customer", "firstname")
FieldKey name = new FieldKey("customer", "name")
fieldsMap.put(firstName, new StringField(firstName, "Han"))
fieldsMap.put(name, new StringField(name, "Solo"))

Map<String, Object> valuesMap = new HashMap<String, Object>()
valuesMap.put(PostActionParameter.CUSTOMER_GROUP_ID, unit.getCustomerGroup().getId
 ())
valuesMap.put(PostActionParameter.FIELDS_MAP, fieldsMap)

return unitActionScriptResultFactory.getPostAction(PostActionType.CREATE_UNIT,
 valuesMap)
```

Code example 44: *Company script which fills some unit data, CM version 6.9.4*

```
// used for companies in MyCustomerGroup to create new contacts easily

import com.consol.cmas.common.model.customfield.meta.*
import com.consol.cmas.common.model.customfield.*
import com.consol.cmas.core.server.service.action.PostActionType

def myunit = new Unit()

def mycustomergroup = customerGroupService.getByName("MyCustomerGroup")
myunit.setCustomerGroup(mycustomergroup)

def mycustomerdefinition = unitDefinitionService.getByName("customer")
myunit.setDefinition(mycustomerdefinition)
myunit.set("company()", unit)

myunit.set("customer.firstname", "Han")
myunit.set("customer.name", "Solo")

return actionScriptResultFactory.getPostAction(PostActionType.CREATE_UNIT, myunit)
```

Code example 45: *Company script which fills some unit data, CM version 6.10*

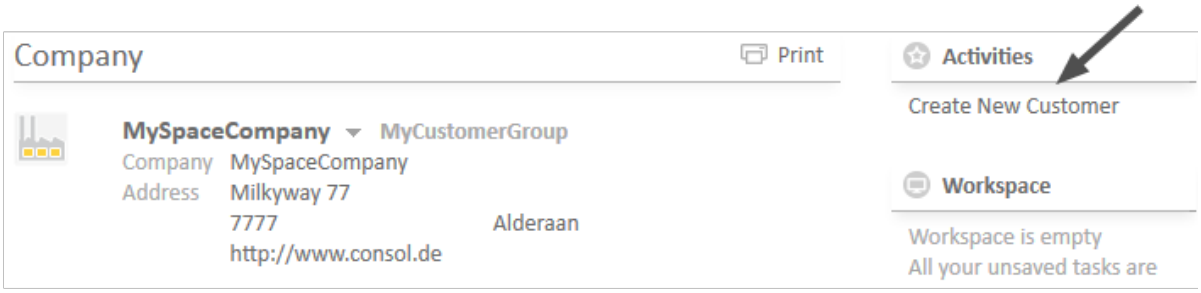


Figure 326: *ConSol CM Web Client - Manual company action*

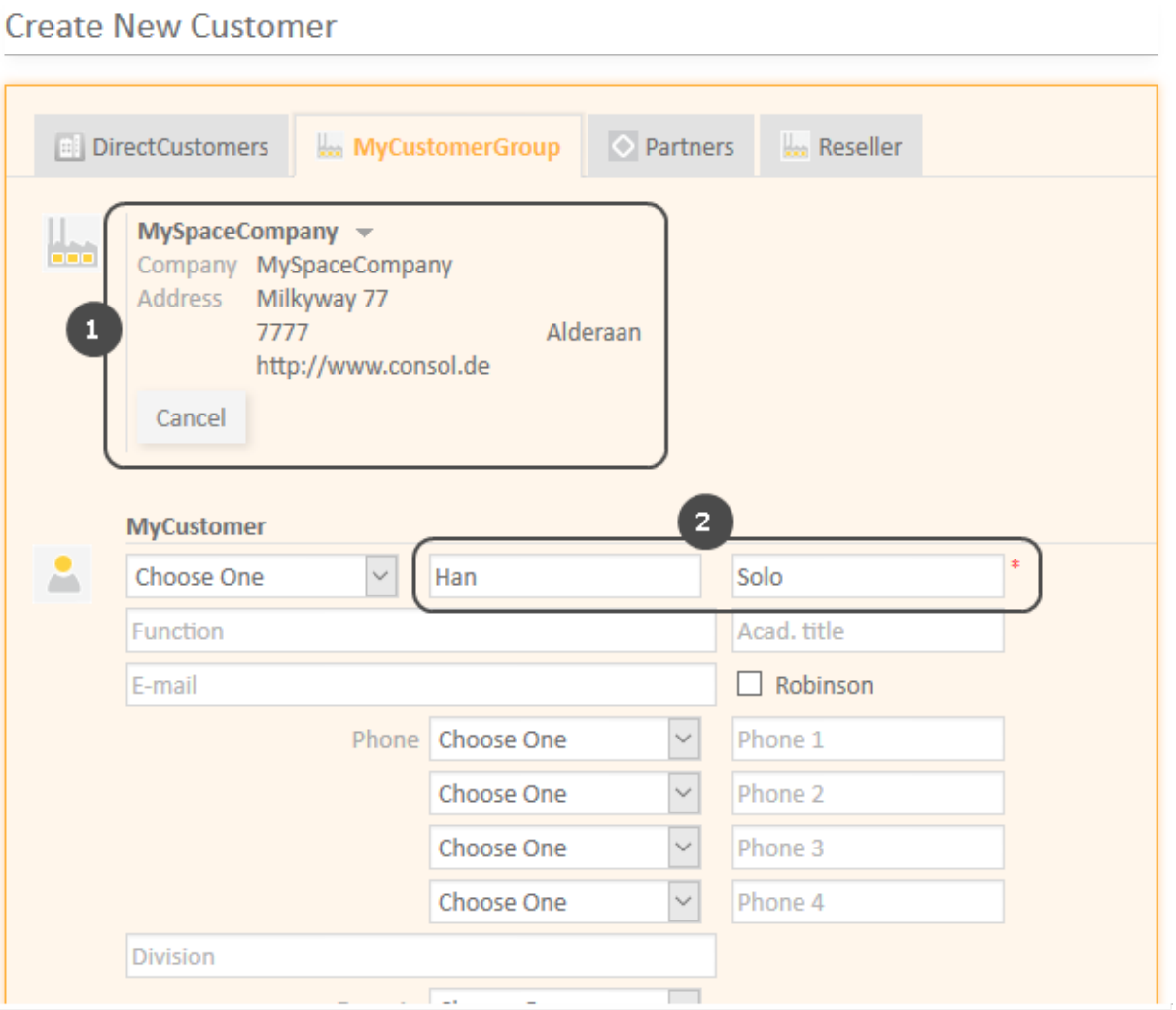


Figure 327: *ConSol CM Web Client - Create Unit page opened and pre-filled by company action*

- (1) The company for which the *Create New Customer* activity has been performed is preselected.
- (2) Some data fields of the contact data are pre-filled

Of course, the names of the customer fields which are used in the script have to be the ones from the customer data model which has been assigned to the customer group for which a new contact should be created.

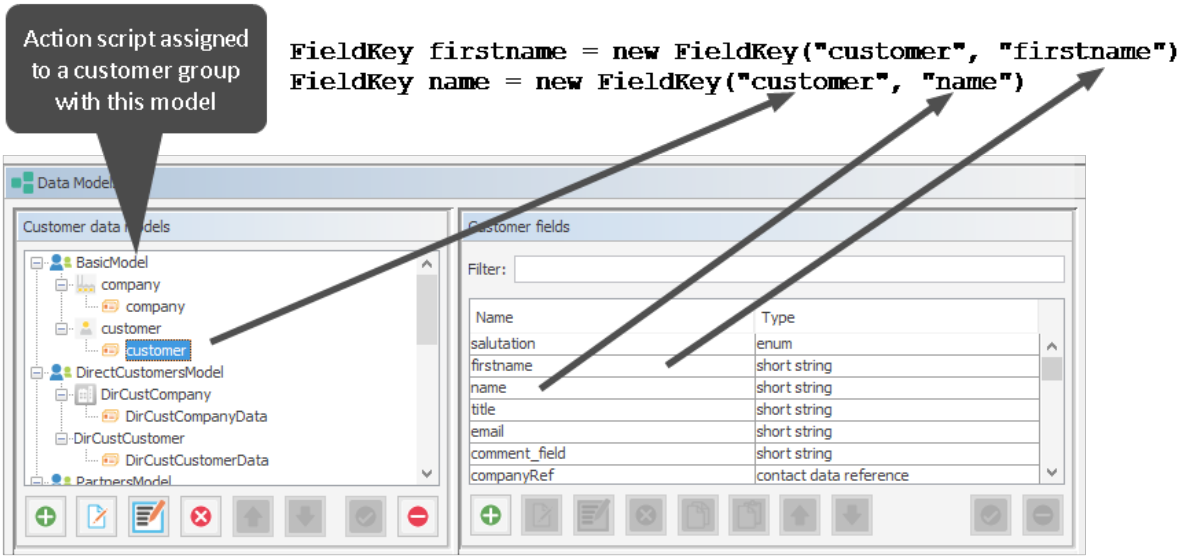


Figure 328: *ConSol CM Admin Tool - Customers, Data Models: Script for creating Unit page as company action*

Create a Ticket

(`PostActionType.CREATE_TICKET`) redirects the user to a ticket creation page. It uses the optional `PostActionParameter.UNIT_ID` with the ID of the main customer, `PostActionParameter.QUEUE_ID` with the ID of the queue, and the ticket fields map `PostActionParameter.FIELDS_MAP`.

```
// offer Create Ticket page for a new HelpDesk 1st level ticket

import com.consol.cmas.common.model.scripting.unit.PostActionType
import com.consol.cmas.common.model.scripting.unit.PostActionParameter

def queueId = queueService.getByName("HelpDesk_1st_Level").getId()
Map<String, Object> valuesMap = new HashMap<String, Object>()
valuesMap.put(PostActionParameter.UNIT_ID, unit.getId())
valuesMap.put(PostActionParameter.QUEUE_ID, queueId)
return unitActionScriptResultFactory.getPostAction(PostActionType.CREATE_TICKET,
 valuesMap)
```

Code example 46: *Script creates and returns action result that will tell the Web Client to create a new ticket with unit as the main contact, CM version 6.9.4*

```
// offer Create Ticket page for a new Service Desk ticket

import com.consol.cmas.core.server.service.action.PostActionType
import com.consol.cmas.common.model.ticket.Ticket
def newtic = new Ticket()
def qu = queueService.getByName("ServiceDesk")
newtic.setQueue(qu)
return actionScriptResultFactory.getPostAction(PostActionType.CREATE_TICKET,
 newtic, unit)
```

Code example 47: *Script creates and returns action result that will tell the Web Client to create a new ticket with unit as the main contact, CM version 6.10*

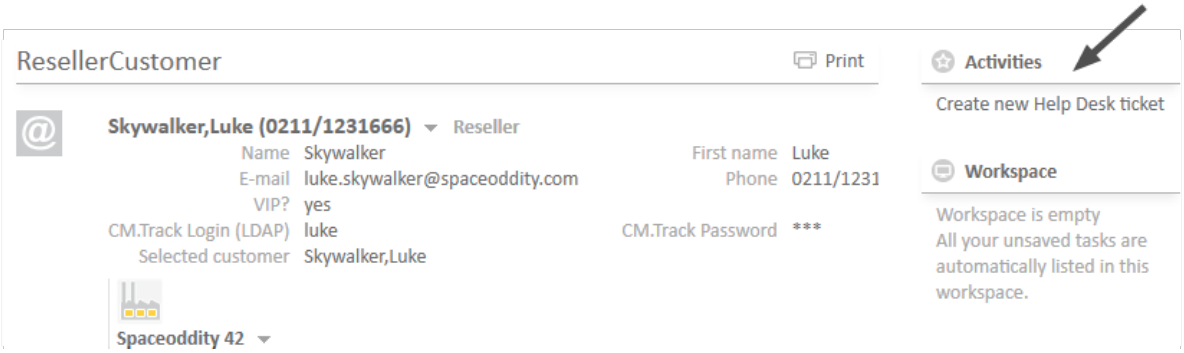


Figure 329: *ConSol CM Web Client - Manual contact action*

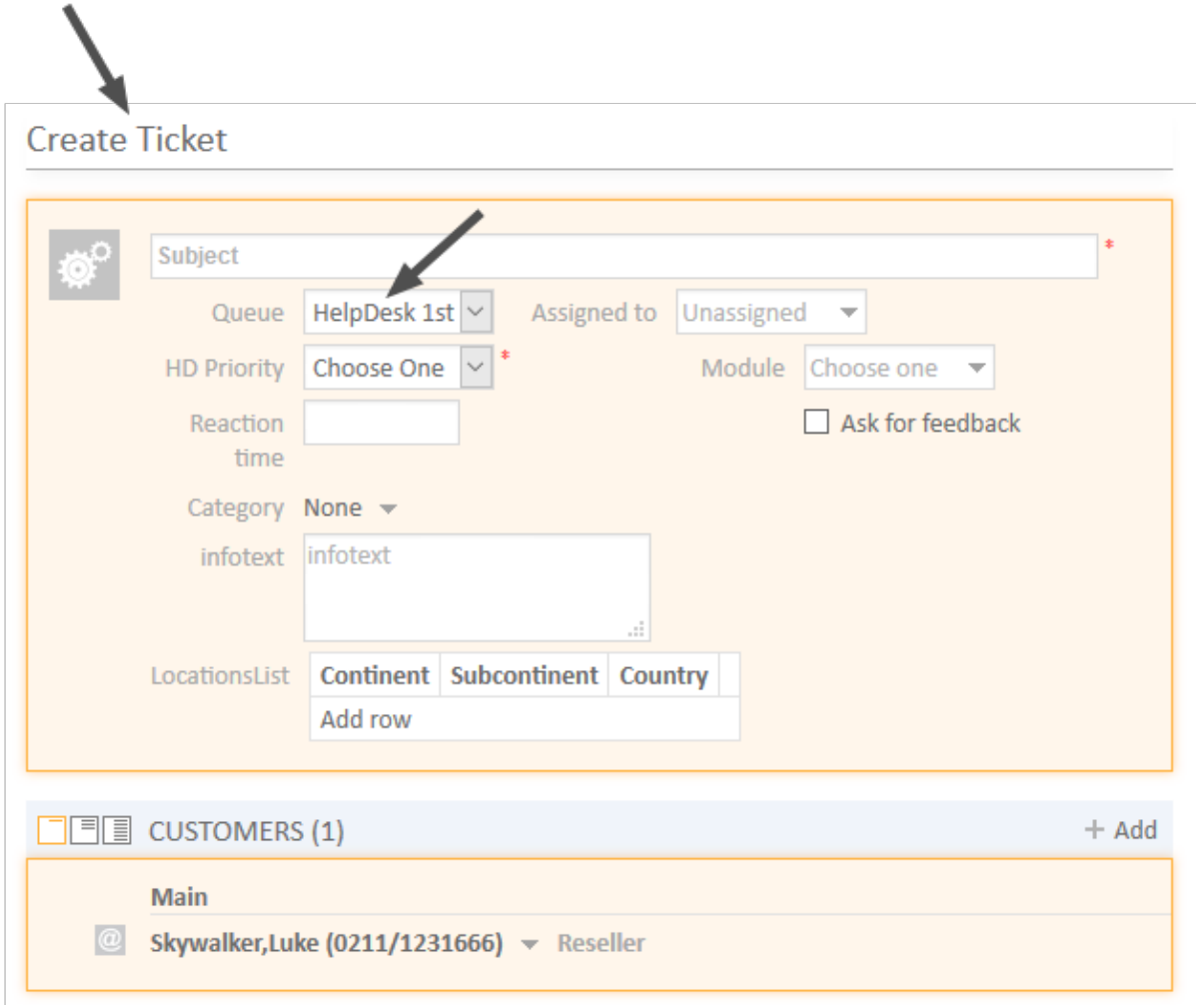


Figure 330: *ConSol CM Web Client - Create ticket page opened and pre-filled by contact action*

Open a Unit Page

(`PostActionType.GOTO_UNIT`) redirects to a unit page. It uses the obligatory parameter `PostActionParameter.UNIT_ID` with the ID of the unit.

```
import com.consol.cmas.common.model.scripting.unit.PostActionType
import com.consol.cmas.common.model.scripting.unit.PostActionParameter

Map<String, Object> valuesMap = new HashMap<String, Object>()
valuesMap.put(PostActionParameter.UNIT_ID, unit.get("company()").getId())
return unitActionScriptResultFactory.getPostAction(PostActionType.GOTO_UNIT,
 valuesMap)
```

Code example 48: *Script which opens the company page, CM version 6.9.4*

Example: Go to the contact detail page of an end customer and open the company detail page of the Reseller company which is responsible for this end customer. A company-contact relation has been established before. If more than one reseller relation has been defined, the first relation in the list is used.

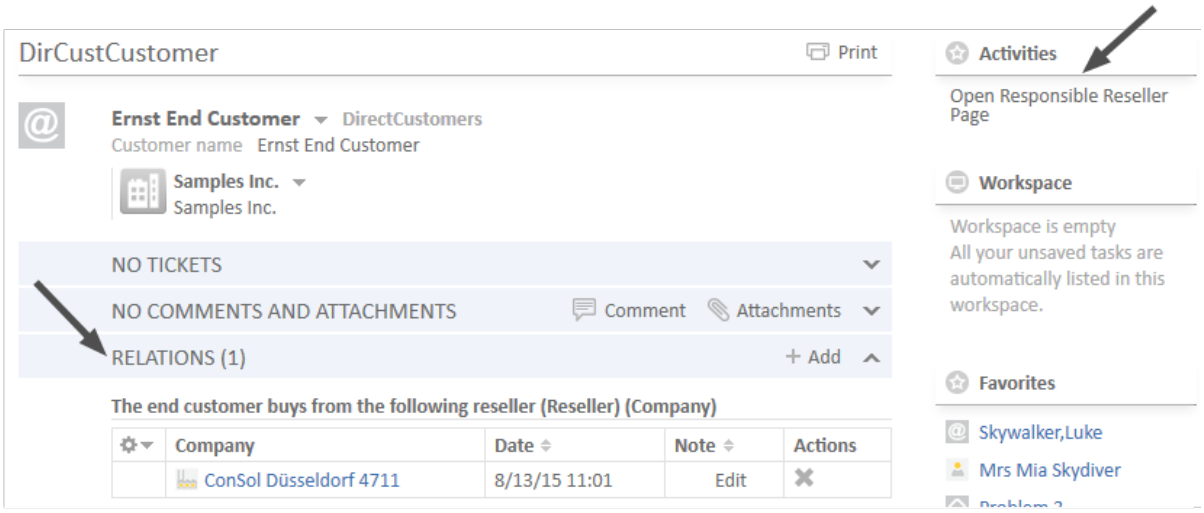


Figure 331: *ConSol CM Web Client - Customer action on contact detail page (activity opens company detail page of responsible reseller, CM version 6.10)*

```
// Open company detail page of responsible reseller company, uses first reseller in
 list
// should only be executed if this relation exists. A condition script is used to
 check

import com.consol.cmas.common.service.UnitRelationDefinitionService
import com.consol.cmas.core.server.service.action.PostActionType
import com.consol.cmas.common.model.customfield.UnitRelation

// find responsible reseller
def unit_rel_def = unitRelationDefinitionService.getByName
 ("ResellerDirectCustomersRelation")

Set<UnitRelation> res_relations = unitRelationService.getByDefinitionAndTarget
 (unit_rel_def, unit)

if (res_relations.size() > 0) {
   def source_unit = res_relations.toArray()[0].getSourceUnit()
   // log.info("SOURCE UNIT IS NOW " + source_unit.get
    ("ResellerCompanyData:company_name"))
   return actionScriptResultFactory.getPostAction(PostActionType.GOTO_UNIT, source_
    unit)

} else {
   log.info("ERROR -- no responsible reseller unit found")
   return actionScriptResultFactory.getPostAction(PostActionType.FAILURE,
    "action.result.failure" )
}
```

Code example 49: *Script which opens the company page, CM version 6.10*

If the action *OpenResponsibleResellerPage* should only be offered in the Web Client when a relation to the responsible reseller has been set, you can work with a customer condition script. It has to be created as an Admin Tool script of type *Customer condition* and has to be assigned to the customer action in the *Actions* section.

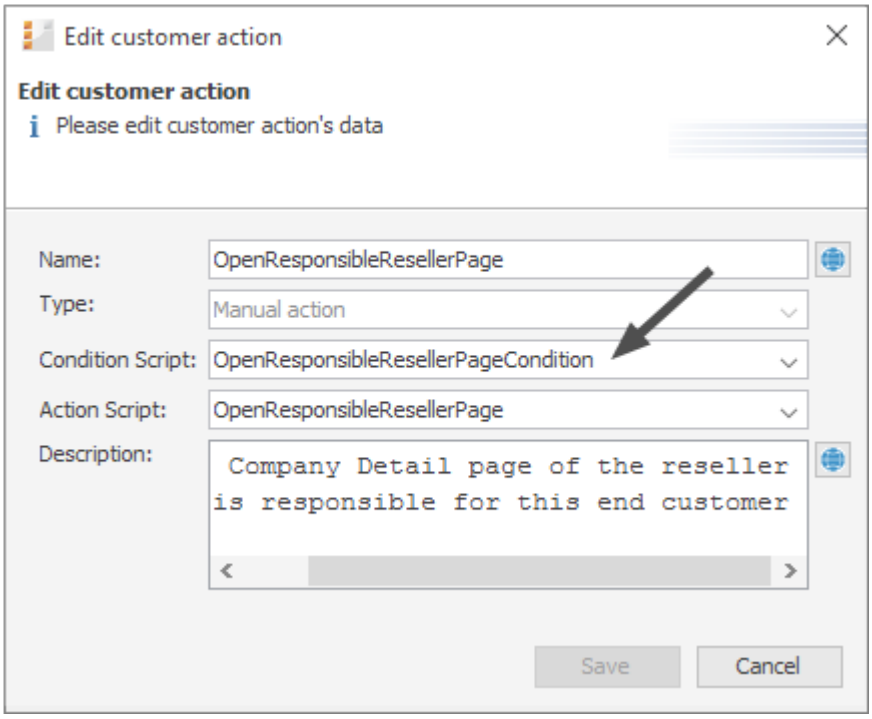


Figure 332: *ConSol CM Admin Tool - Customers, Actions: Assignment of customer condition script*

```
// Checks if reseller relation is set

import com.consol.cmas.common.service.UnitRelationDefinitionService
import com.consol.cmas.core.server.service.action.PostActionType
import com.consol.cmas.common.model.customfield.UnitRelation

// find responsible reseller
def unit_rel_def = unitRelationDefinitionService.getByName
 ("ResellerDirectCustomersRelation")
Set<UnitRelation> res_relations = unitRelationService.getByDefinitionAndTarget
 (unit_rel_def, unit)
if (res_relations.size() > 0) {
   return true
} else {
   return false
}
```

Code example 50: *Customer condition script which checks if reseller relation is present*

Open a Ticket Page

(`PostActionType.GOTO_TICKET`) redirects to a ticket page. It uses the obligatory parameter `PostActionParameter.TICKET_ID` with the ID of the ticket.

```
import com.consol.cmas.common.model.scripting.unit.PostActionType
import com.consol.cmas.common.model.scripting.unit.PostActionParameter
import com.consol.cmas.common.model.customfield.Unit
import com.consol.cmas.common.model.ticket.TicketCriteria
import com.consol.cmas.common.model.customfield.ListField
import com.consol.cmas.common.model.customfield.ContactReferenceField
import com.consol.cmas.common.model.customfield.UnitReferenceSearchField
import com.consol.cmas.common.model.customfield.ContactReferenceSearchField
import com.consol.cmas.common.model.customfield.meta.FieldKey
import com.consol.cmas.common.model.ticket.Ticket
import com.consol.cmas.common.model.ContactTicketRole
import com.consol.cmas.common.model.customfield.StringField
import com.consol.cmas.common.model.scripting.unit.UnitActionScriptResult


//get AM queue for search
def q_id = (workflowApi.getQueueByName("AccountManagement")).id
def q_ids = new HashSet()
q_ids.add(q_id)

//find AM ticket for the company
def crit = new TicketCriteria()
crit.setQueueIds(q_ids)

// create list field key
def contactSearchListFieldKey = new FieldKey("queue_fields","contacts")

// prepare list field
def contactsListField = new ListField(contactSearchListFieldKey )

// create member field key
def contactSearchFieldKey = new FieldKey("queue_fields","contacts_member")

// create unit member field with Unit and ticket main role

def contactsMember = new
ContactReferenceSearchField(contactSearchFieldKey, unit,
ContactTicketRole.MAIN_ROLE)

// put member field in Unit list field
contactsListField.addChild(contactsMember)

// put field(s) into the criteria
crit.setFields([contactsListField] as Set)

// seek and find
def foundTickets = ticketService.getByCriteria(crit)

if ( foundTickets ) {
   def AM_tic = foundTickets.first()
   def AM_tic_id = AM_tic.id

   // go to AM ticket
```

```
   Map<String, Object> valuesMap = new HashMap<String, Object>()
   valuesMap.put(PostActionParameter.TICKET_ID, AM_tic_id)
   return unitActionScriptResultFactory.getPostAction(PostActionType.GOTO_TICKET,
    valuesMap)
}

// Default: found nothing
return null
```

Code example 51: *Open a ticket page in view mode, CM version 6.9*

```
import com.consol.cmas.common.model.customfield.Unit
import com.consol.cmas.common.model.ticket.TicketCriteria
import com.consol.cmas.common.model.customfield.ListField
import com.consol.cmas.common.model.customfield.ContactReferenceField
import com.consol.cmas.common.model.customfield.UnitReferenceSearchField
import com.consol.cmas.common.model.customfield.ContactReferenceSearchField
import com.consol.cmas.common.model.customfield.meta.FieldKey
import com.consol.cmas.common.model.ticket.Ticket
import com.consol.cmas.common.model.ContactTicketRole
import com.consol.cmas.common.model.customfield.StringField
import com.consol.cmas.core.server.service.action.PostActionType

//get AM queue for search
def q_id = (workflowApi.getQueueByName("AccountManagement")).id
def q_ids = new HashSet()
q_ids.add(q_id)
//find AM ticket for the company
def crit = new TicketCriteria()
crit.setQueueIds(q_ids)
// Listenfeld-Key erzeugen
def contactSearchListFieldKey = new FieldKey("queue_fields","contacts")
// Listenfeld vorbereiten
def contactsListField = new ListField(contactSearchListFieldKey )
// Memberfeld-Key erzeugen
def contactSearchFieldKey = new FieldKey("queue_fields","contacts_member")
// Unit-Memberfeld mit Unit und Ticket-Hauptrolle erzeugen
// COmpany is MAIN CONTACT at the AM ticket!
def contactsMember = new ContactReferenceSearchField(contactSearchFieldKey, unit,
 ContactTicketRole.MAIN_ROLE)
// Member-Feld in Unit-Listenfeld stopfen
contactsListField.addChild(contactsMember)
// Feld(er) in die Kriterien stopfen
crit.setFields([contactsListField] as Set)
// Suchen und finden
def foundTickets = ticketService.getByCriteria(crit)
println "Found tickets: ${foundTickets}"

if ( foundTickets ) {
   def AM_tic = foundTickets.first()
   return actionScriptResultFactory.getPostAction(PostActionType.GOTO_TICKET, AM_
    tic)
}
// Default: found nothing
return null
```

Code example 52: *Open a ticket page in view mode, CM version 6.10*

⚠ When you use one of the scripts above, please keep in mind that the ticket search requires that the customer fields of the company be indexed (annotation `field-indexed` = “true”).

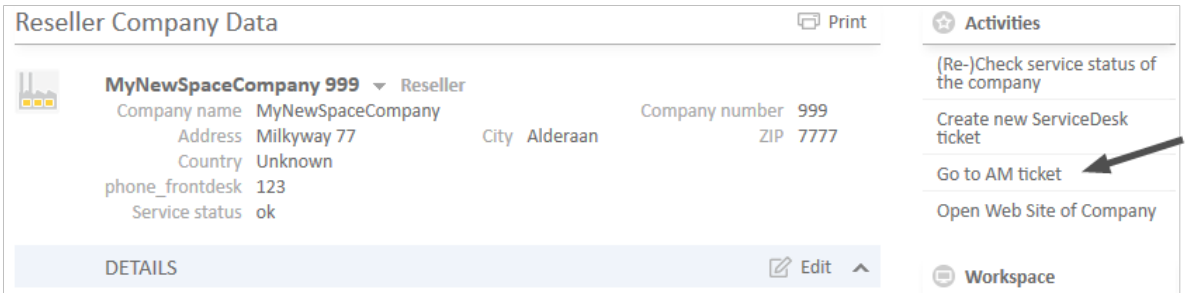


Figure 333: *ConSol CM Web Client - Company action available on company page (1)*

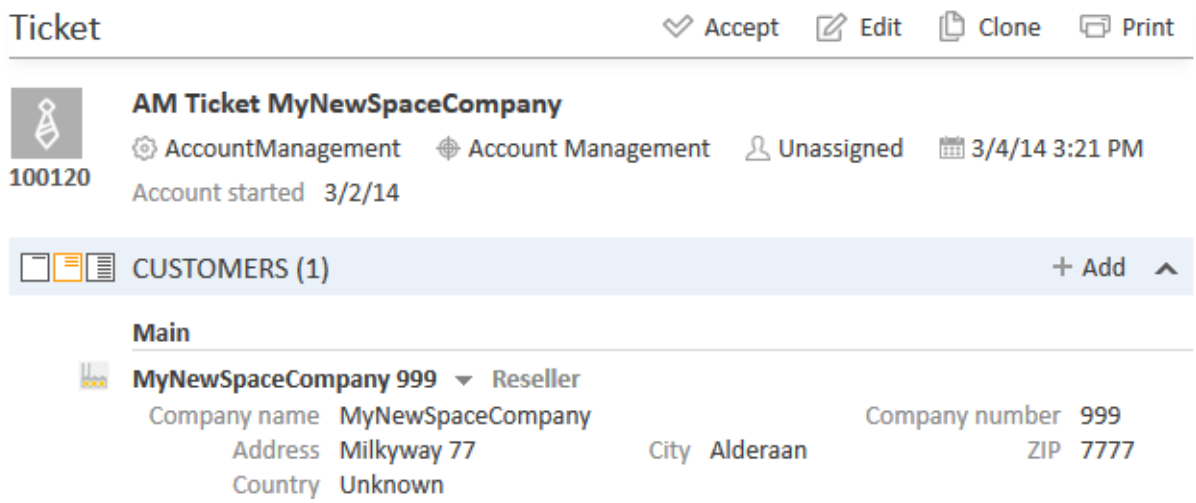


Figure 334: *ConSol CM Web Client - AM page opened after company action Go to AM ticket*

Open a Web Page

(`PostActionType.GOTO_PAGE`) redirects to a URL. It uses the obligatory `PostActionParameter.URL` with the URL.

The following code shows a simple example with a fixed URL for each company.

```
import com.consol.cmas.common.model.scripting.unit.PostActionType
import com.consol.cmas.common.model.scripting.unit.PostActionParameter

Map<String, Object> valuesMap = new HashMap<String, Object>()
valuesMap.put(PostActionParameter.URL, unit.get("company:www"))
return unitActionScriptResultFactory.getPostAction(PostActionType.GOTO_PAGE,
 valuesMap)
```

Code example 53: *Script which opens a certain web site (URL), CM version 6.9.4*

```
// opens company's web site
import com.consol.cmas.core.server.service.action.PostActionType

def url = unit.get("url")

if (!url) {
   return actionScriptResultFactory.getPostAction(PostActionType.FAILURE,
    "error.script.no.url")
} else {
   return actionScriptResultFactory.getPostAction(PostActionType.GOTO_PAGE,url)
}
```

Code example 54: *Script which opens a certain web site (URL), CM version 6.10*

The string *error.script.no.url* is a label which has been defined in the navigation group *Labels*, as described in section Labels.

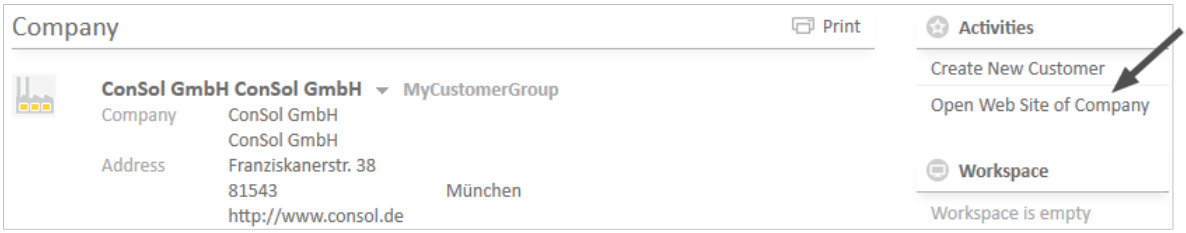


Figure 335: *ConSol CM Web Client - Company action available on company page (2)*

> To open a fixed URL, you can use a customer field of type STRING with the annotation `text-type` = "url". This will automatically create a hyperlink. Thus, the use of the `GOTO_URL` parameter in customer action scripts is only recommended when an URL is built dynamically within a script.

D.9.5.2 Customer Condition Scripts

A customer condition script defines whether an action should be shown in the Web Client or not. It is executed before the customer action script. If it returns "false", the customer action script will not be executed.

```
if(unit.getFieldValue("customer.personalData") == null) {
   return true
} else {
   return false
}
```

Code example 55: *Customer condition script*

To put a customer condition script into operation, you have to perform the following steps:

1. Write an Admin Tool script of type *Customer condition*. The script has to define the conditions for the return values “true” and “false”, as demonstrated in the example above. At this point you must already know for which customer groups and data models the script will be used, to make sure you reference the correct data fields within the script.
2. Assign the customer condition script to one or more customer action(s). This is done in the *Actions* section, navigation group *Customers*. Once in place, this specific customer condition script will always be executed before the customer action script of the same customer action.

That's it. You do not have to assign the condition script to a customer group. This happens implicitly when the respective customer action is assigned to contacts or companies of certain customer groups.

D.9.5.3 Important Groovy Objects

Object `UnitActionScriptResult` (CM Version 6.9)

The object `UnitActionScriptResult` is only taken into account for manual actions. For actions like *CREATE*, *UPDATE*, or *DELETE* it is not available. The `UnitActionScriptResult` object is created by the `unitActionScriptResultFactory.getPostAction(String, Map<String, Object>)` method. This class (resp. the object) is used to store information that will influence the process flow of the Web Client after the manual action has been executed. The `UnitActionScriptResult` object contains the manual action type, the IDs of the ticket, the unit, the queue, and the customer group. After executing the manual action, the user can be redirected to a different page.

Object `actionScriptResult` (CM Version 6.10 and Up)

Starting with CM version 6.10, the Action Framework offers the classes `ActionScriptResult` and `ActionScriptResultFactory`. An object (singleton) of the class `ActionScriptResultFactory` is available as `actionScriptResultFactory` in every action script, no matter which type of action script.

See section Scripts for the Action Framework for details.

The customer actions CREATE, UPDATE, and DELETE are executed in the core methods *create*, *update*, and *delete* of the object `unitService`.

Thus, if the update action script updates the customer using the `unitService.update(Unit)` method then a `java.lang.StackOverflowError` error may be thrown because the update action will infinitely recurse. In that case a customer condition script should be used to avoid such infinite loops.

D.9.5.4 Working with the Changes Object in Customer Update Actions

Starting with CM version 6.10.5.4, it is possible to monitor the changes which have been performed during a customer update action. (The same applies to resource *Update* actions, explained in section Working with the Changes Object in Resource Update Actions).

To find out which changes have been performed use the object of class `UnitChanges` in Unit actions.

Please remember, the *Update* script will be executed:

- in an explicit *Update* action
- when comments or attachments are added
- when comments or attachments are removed

There are two methods of the `unitChanges` object which provide information about the changed data:

- `getCustomFieldChangeInfo()`
  provides information about changes of unit data (in customer fields)
- `getContentChangeInfo()`
  provides information about changes in the unit history (comments, attachments)

Since the method return parameters contain rather complex components, we recommend to read the API doc of the `UnitChanges` class. The following code provides an example for a script where a `unitChanges` object is used.

```
// Update Action Script UpdateContactData.groovy for contacts in Reseller group
import com.consol.cmas.common.model.content.unit.UnitCommentEntry
import com.consol.cmas.common.model.content.unit.UnitAttachmentEntry
log.info 'Contact data have been UPDATEd!'
// Are there any changes?
if (changes) {
   log.info 'Yes, changes have been made to unit'
   log.info 'Changes object is a ' + changes.class
}
// Have Custom Fields (Data object Group Fields) been changed? If yes - which?
if (changes.customFieldChangeInfo) {
   log.info 'Yes, changes have been made to Custom Fields'
   log.info changes.customFieldChangeInfo
   log.info changes.customFieldChangeInfo.each { k, v ->
   log.info "Changed field: ${k.groupName}/ ${k.fieldName}"
   log.info "New value: ${v.value.value}"
   log.info "Old value: ${v.previousValue.value}"
}
} else {
   log.info 'No changes to Custom Fields'
}

// Have comments or attachmenst been changed? If yes - which?
log.info changes.contentChangeInfo
if (changes.contentChangeInfo){
   log.info 'Yes, changes have been made in detail section'
   if (changes.contentChangeInfo.value){
      log.info changes?.contentChangeInfo.each { ctEntry ->

         if (ctEntry?.value[0] instanceof UnitCommentEntry){
            log.info 'A comment has been added.'
            log.info 'Old value: ' + ctEntry?.previousValue
            log.info 'New value: ' + ctEntry.value[0]?.text
            log.info 'Made by the engineer ' + ctEntry.value[0]?.engineer?.name
            log.info 'Creation date of the comment: ' + ctEntry.value
             [0]?.creationDate
         } else if (ctEntry?.value[0] instanceof UnitAttachmentEntry){
            log.info 'An attachment has been added.'
            log.info 'Old value: ' + ctEntry?.previousValue
            log.info 'New value text: ' + ctEntry.value[0]?.text
            log.info 'New value file name: ' + ctEntry.value[0]?.filename
         }
      }
   } else {
      log.info 'Entry has been deleted.'
   }
}
```

Code example 56: *Unit Update script where changes are monitored and printed out to server.log*

When for a contact in the customer group *Resellers* changes have been made to the two customer fields *phone* and *vip_person*, the following text is displayed in the `server.log` file.

```
database_UpdateContactData] [Susan-] Contact data have been UPDATEd!
database_UpdateContactData] [Susan-] Yes, changes have been made to unit
database_UpdateContactData] [Susan-] Changes object is a class
 com.consol.cmas.common.model.history.unit.UnitChanges
database_UpdateContactData] [Susan-] Yes, changes have been made to Custom Fields
database_UpdateContactData] [Susan-] {(phone,ResellerCustomerData)=Modification
 {value=AbstractField{key=(phone,ResellerCustomerData), value=0211/1231777},
 previousValue=AbstractField{key=(phone,ResellerCustomerData),
 value=0211/1231666}}, (vip_person,ResellerCustomerData)=Modification
 {value=AbstractField{key=(vip_person,ResellerCustomerData), value=false},
 previousValue=AbstractField{key=(vip_person,ResellerCustomerData), value=true}}}
database_UpdateContactData] [Susan-] Changed field: ResellerCustomerData/ phone
database_UpdateContactData] [Susan-] New value: 0211/1231777
database_UpdateContactData] [Susan-] Old value: 0211/1231666
database_UpdateContactData] [Susan-] Changed field: ResellerCustomerData/ vip_
 person
database_UpdateContactData] [Susan-] New value: false
database_UpdateContactData] [Susan-] Old value: true
```

Code example 57: *Log output from the script above*

D.10 Address Autocomplete

This chapter discusses the following:

D.10.1 Introduction

In some ConSol CM systems it is required that engineers enter or edit a great number of customer data manually. In such cases it can be helpful to have system support to

- provide suggestions for the input into some data fields (like e.g., zip code or address) to ensure that the entered address really exists
- avoid entering duplicates

To support such cases, ConSol CM offers the feature *Address Autocomplete*. This feature can be switched on/off using a system property. In standard ConSol CM installations, it is switched off. When it has been switched on, the engineer can receive suggestions for the input in one or more of the following customer fields:

- one or more fields which contain the zip code
- one or more fields which contain the city
- one or more fields which contain the address (street and number)

This applies to the following operations in the CM Web Client:

- Create a customer on the *Create customer* page
- Create a customer in the *Customers* section of a ticket
- Edit a customer on the customer page
- Edit a customer in the *Customers* section of a ticket
- Entering customer data on the *Detailed Search* page
- Entering customer data in an ACF (Activity Control Form)

Usually, a data set which is publicly available, e.g., a data collection on CD ROM, serves as source for the import of the address data. In this way, the engineers can implicitly use a vast collection of zip code/city/address mappings.

> ⚠ Please note that in order to use this feature, your company will have to purchase an address collection, i.e., the feature Address Autocomplete is based on the import of external data which is not part of a ConSol CM distribution!

In the following example, a German address collection has been imported into a ConSol CM demo system. Please see the three example figures from the respective ConSol CM Web Client to learn how ConSol CM can help engineers improve their data input quality.

Example 1: The engineer starts typing into the *ZIP* field. Only the existing ZIP codes will be offered. The number of suggestions which are displayed is part of an Address Autocomplete Configuration, please see section Define the Address Autocomplete Configuration Using the Admin Tool for details.

Figure 336: *ConSol CM Web Client - Suggestions of the Address Autocomplete feature, example 1*

Example 2: The *ZIP* field has already been filled. For the *City* field, only the correct possible values are suggested.

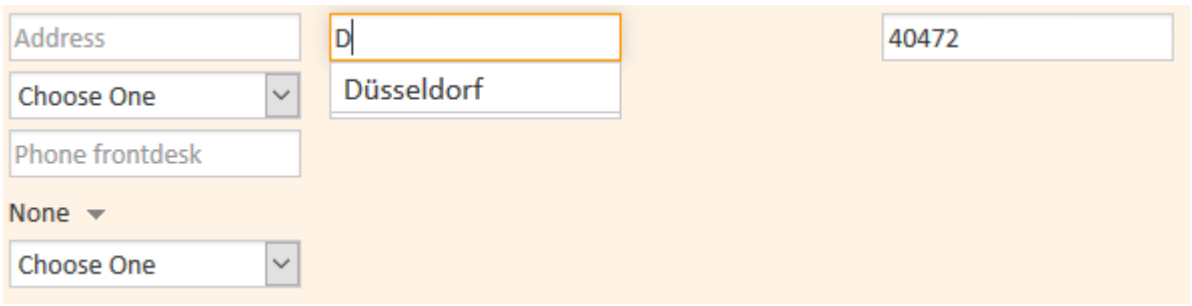


Figure 337: *ConSol CM Web Client - Suggestions of the Address Autocomplete feature, example 2*

Example 3: The *ZIP* field and the *City* field have already been filled. For the *Address* field, only correct possible values are suggested.

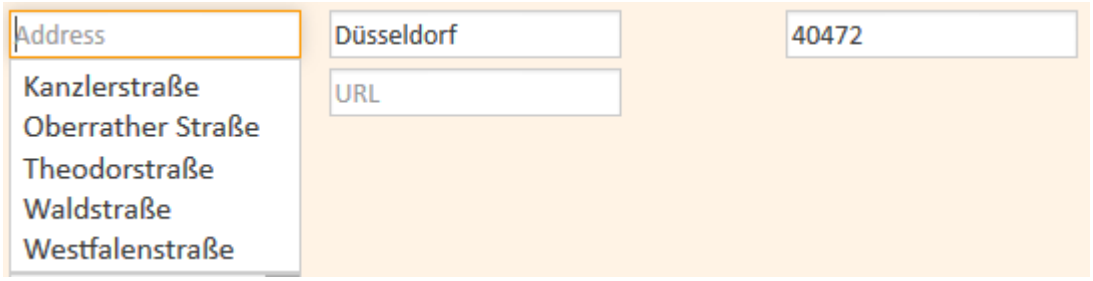


Figure 338: *ConSol CM Web Client - Suggestions of the Address Autocomplete feature, example 3*

Please note that the values which are displayed are only suggestions. The engineer can always overwrite the suggested values by typing into the field manually.

Of course, parallel to this feature, the ConSol CM standard feature *Autocomplete Search* works and will display suggestions for the customers which are already part of the ConSol CM database.

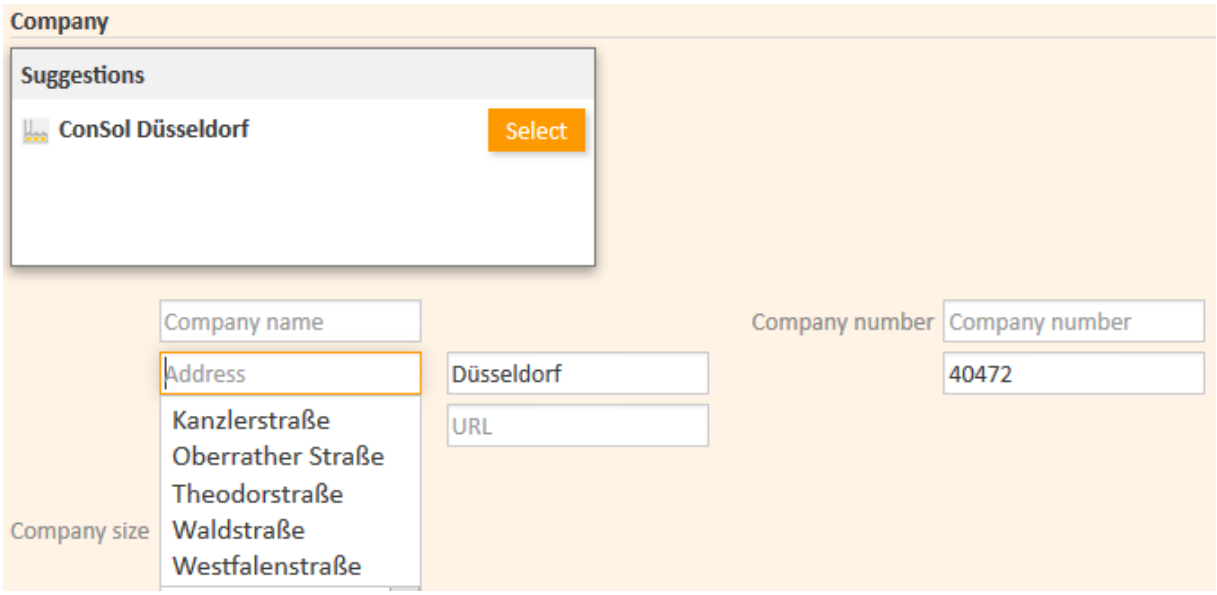


Figure 339: *ConSol CM Web Client - Suggestions of the standard Autocomplete Search feature in combination with Address Autocomplete*

To configure your ConSol CM system for this feature, the following steps have to be performed:

1. Switch on the *Address Autocomplete* feature using the Admin Tool, see section Switch on the Address Autocomplete Feature Using the Admin Tool.
2. Import zip code/city/address data into the ConSol CM database. See section Import Zip/City/Address Data into the ConSol CM Database.
3. Define the autocomplete strategy by creating one or several address autocomplete configurations using the ConSol CM Admin Tool, see section Define the Address Autocomplete Configuration Using the Admin Tool.
4. Refresh the index, as described in section Refresh the Index.

D.10.2 Switch on the *Address Autocomplete* Feature Using the Admin Tool

Enter the system property cmas-app-admin-tool, autocomplete.enabled and set its value to “true”. The system property is not present in a standard ConSol CM installation and has to be added manually.

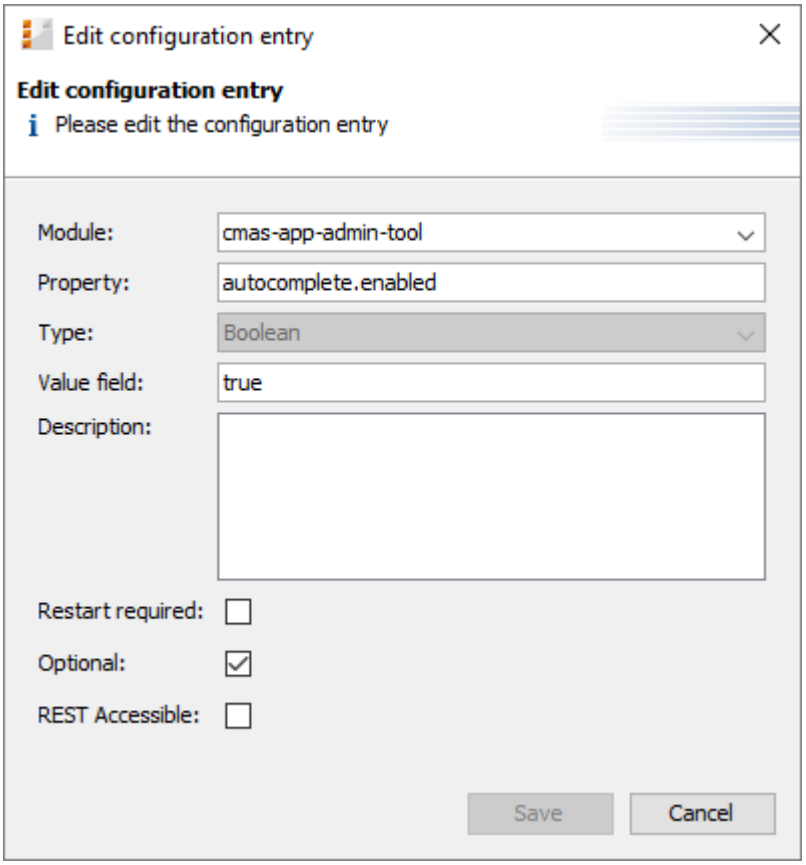


Figure 340: *ConSol CM Admin Tool - System, System Properties: System property for the Address Autocomplete feature*

You will then see the new navigation item *Address Autocomplete* in the navigation group *Customers*. You will learn how to configure the autocomplete strategy in section Define the Address Autocomplete Configuration Using the Admin Tool. But before it makes sense to define the strategy, the source data have to be imported. Please proceed to the next section to learn how to do this.

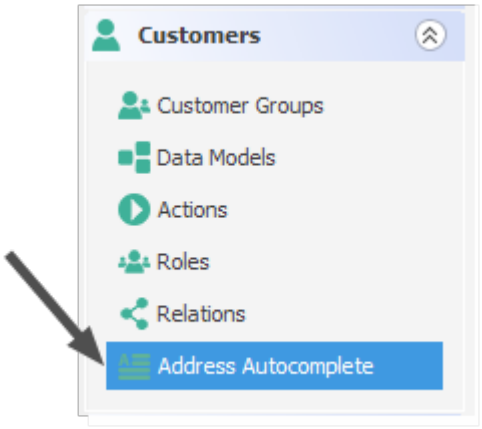


Figure 341: *ConSol CM Admin Tool - Navigation item Address Autocomplete in navigation group Customers*

D.10.3 Import Zip/City/Address Data into the ConSol CM Database

Import the data into the table `cmas_autocomplete_address` in your ConSol CM database.

This import has to be implemented by a person who knows how to insert data correctly into a relational database! An import or import script are neither part of the ConSol CM distribution nor part of standard maintenance. You can implement the import script using a tool of your choice. In case you need any support, please ask your ConSol CM Consultant for help and advice.

The import has to comprise three fields for each data set (see the figure below):

- city
- street
- zip

```
mysql> show fields from cmas_autocomplete_address;
+--------+--------------+------+-----+---------+----------------+
| Field  | Type         | Null | Key | Default | Extra          |
+--------+--------------+------+-----+---------+----------------+
| id     | bigint(20)   | NO   | PRI | NULL    | auto_increment |
| city   | varchar(255) | YES  | MUL | NULL    |                |
| street | varchar(255) | YES  | MUL | NULL    |                |
| zip    | varchar(32)  | YES  | MUL | NULL    |                |
+--------+--------------+------+-----+---------+----------------+
```

Figure 342: *ConSol CM Database - Table cmas_autocomplete_address*

D.10.4 Define the Address Autocomplete Configuration Using the Admin Tool

To define the autocomplete strategy, you have to perform the following steps:

1. Create one or more Address Autocomplete Configuration
2. Configure the behavior for each Address Autocomplete Configuration

D.10.4.1 Create One or More Address Autocomplete Configurations

An Address Autocomplete Configuration represents a mapping of a customer field to one of the key fields in the `cmas_autocomplete_address` table, e.g., you want to define "When the engineer starts inputting data in the field *ResellerCompanyData*:*zip*, the system should search in *zip*, and when the engineer starts typing in the customer field *ResellerCompanyData*:*city*, the system should search in the *city* field".

You have to define each of the Address Autocomplete Configurations in the Admin Tool in the navigation item *Address Autocomplete* in the navigation group *Customers*. Add a new Address Autocomplete Configuration by clicking the *Add* button, entering the name of the new configuration, and clicking *Save*.

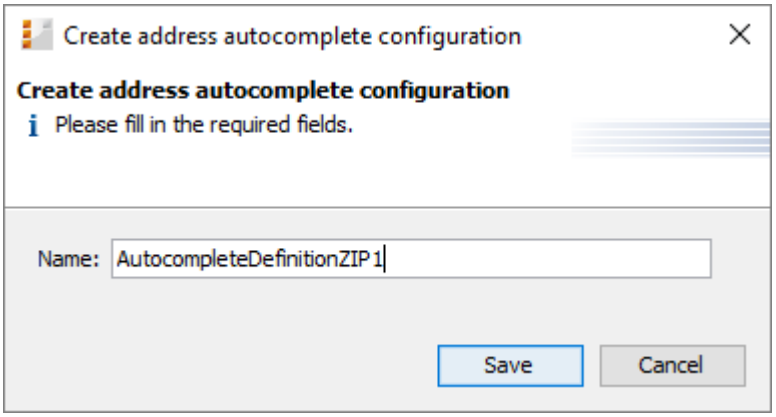


Figure 343: *ConSol CM Admin Tool - Customers, Address Autocomplete: Adding a new Address Autocomplete configuration*

D.10.4.2 Configure the Behavior for Each Address Autocomplete Configuration

To configure an Address Autocomplete Configuration, mark the definition in the list and add one or more Address Autocomplete Fields. These are the mapping rules from the customer fields to the key fields in the `cmas_autocomplete_address` table.

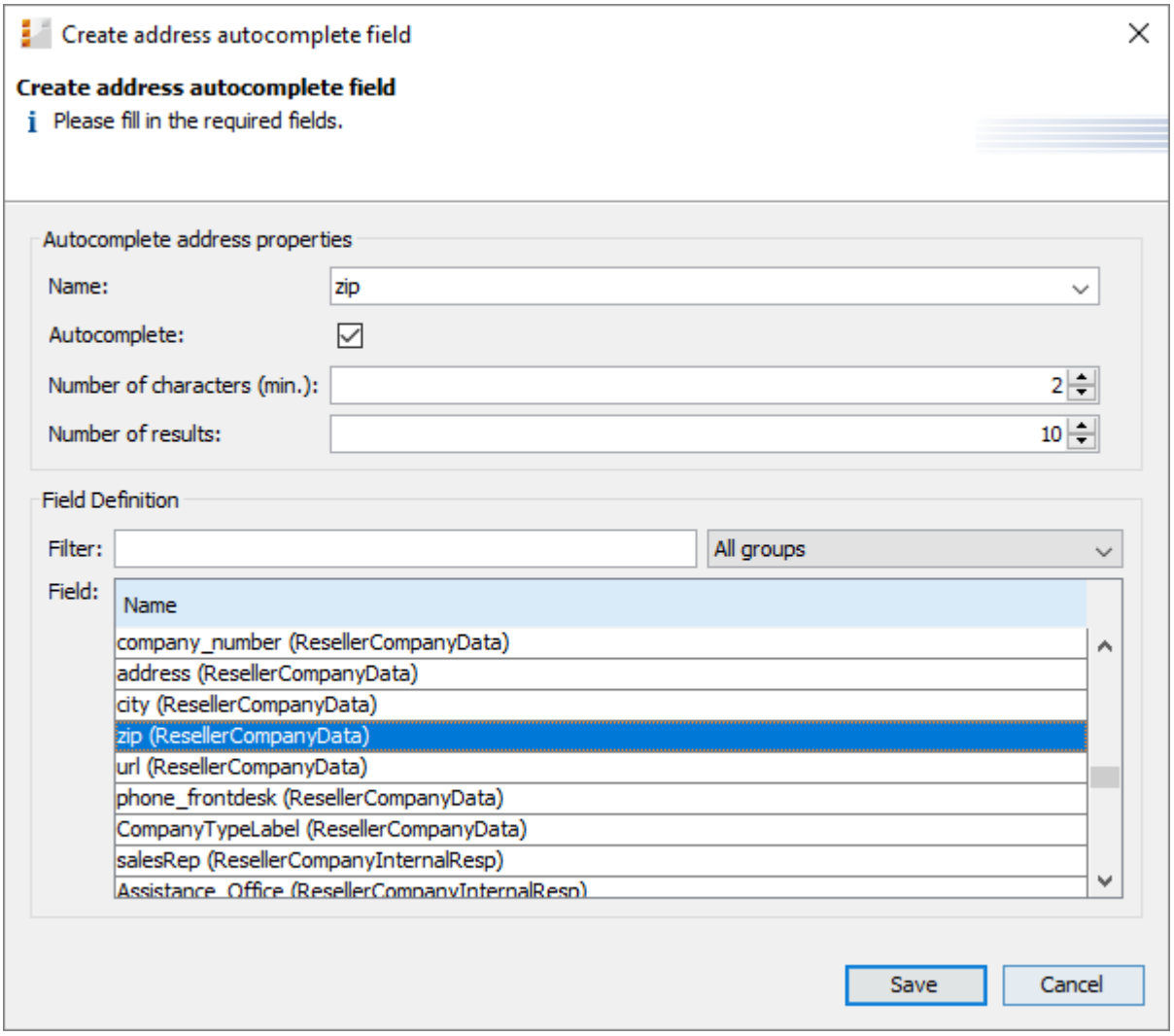


Figure 344: *ConSol CM Admin Tool - Customers, Address Autocomplete: Definition of an Address Autocomplete field*

The following fields have to be defined for each Address Autocomplete Field:

- **Name:**
  Select *city*, *street*, or *zip*. Here you define which key field from the `cmas_autocomplete_address` table is used.
- **Autocomplete:**
  - **If switched on:**
    The field itself will be an autocomplete field without any dependencies on other fields.
  - **If switched off:**
    The field will not be an autocomplete field itself but will be filled depending on other fields. E.g., a *city* field with *Autocomplete* switched off will be automatically filled when the *zip* field or the *address* field is filled but will not directly react to input from the engineer.

- **Number of characters (min.):**
  Here you define the number of characters the engineer has to type into the customer field before the value recognition and autocompletion start. If you want the system to display the list as soon as the cursor has been placed in the customer field in the Web Client, leave the configuration field here empty.
- **Number of results:**
  Here you define the maximum number of suggestions in the (drop-down) list.
- **Field Definition:**
  Here you select one customer field which should react to the autocomplete input. Only a single selection is possible.

In the following example, two Address Autocomplete Fields have been defined for the Address Autocomplete configuration *AutocompleteDefinitionZIP1*, shown in the following figure.

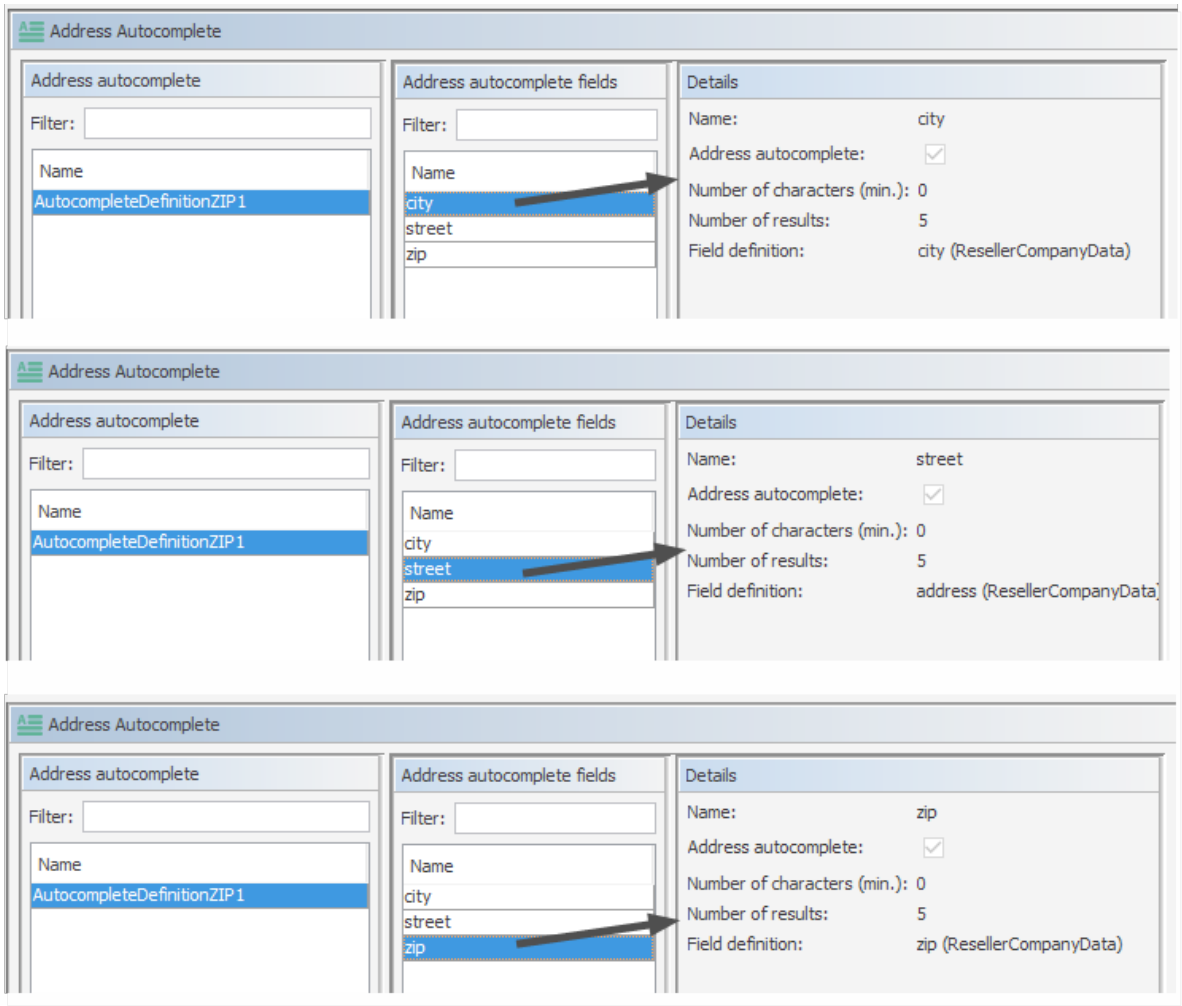


Figure 345: *ConSol CM Admin Tool - Customers, Address Autocomplete: Complete Address Autocomplete configuration*

D.10.4.3 Refresh the Index

When you have entered and configured all required Address Autocomplete Configurations, you have to refresh the index. For details about the index, please refer to section Indexer Management Using the Admin Tool.

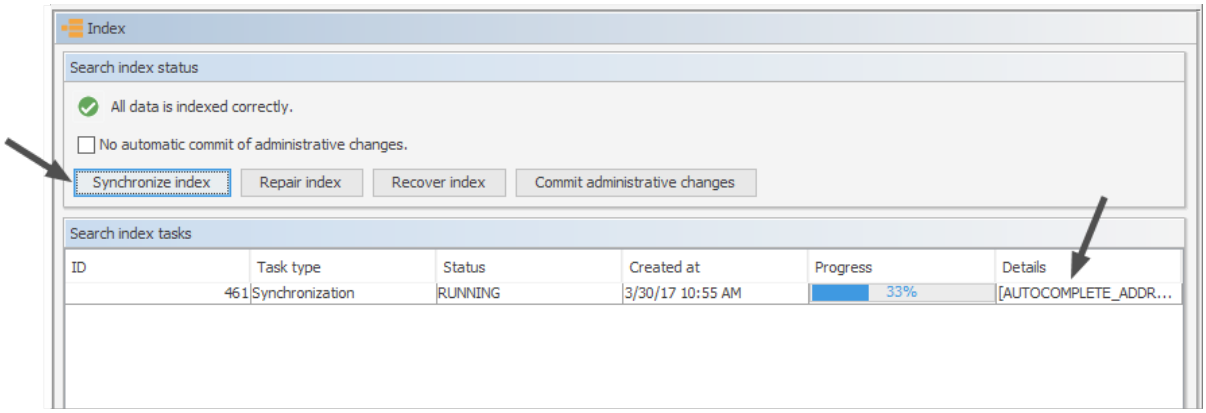


Figure 346: *ConSol CM Admin Tool - Services, Index: Index refresh*

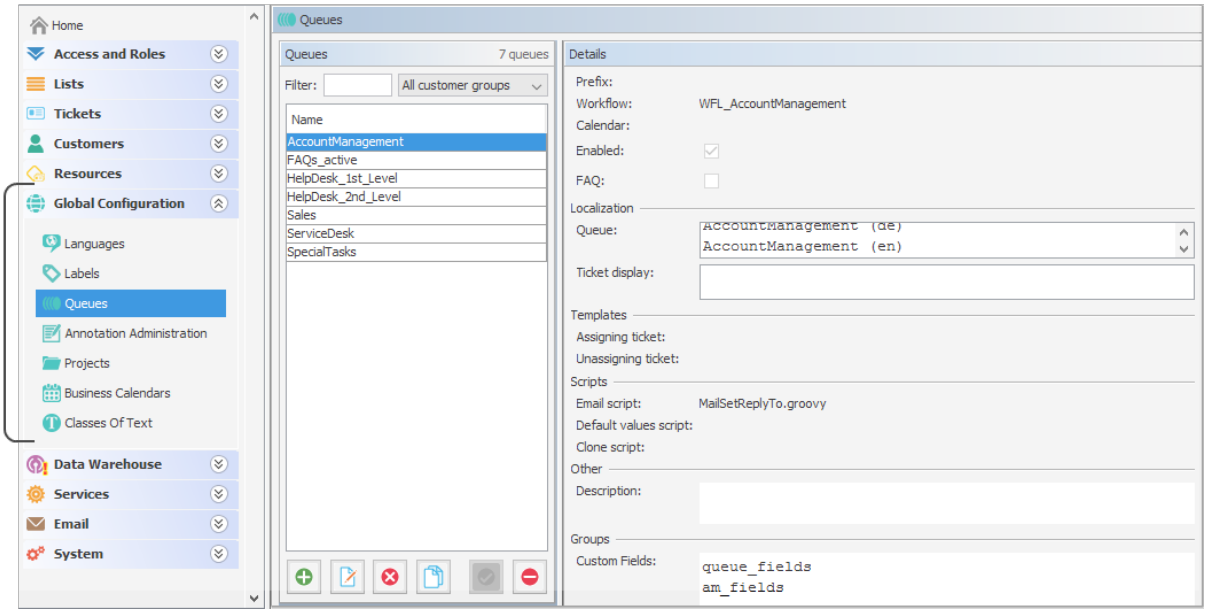
D.10.5 Edit an Address Autocomplete Configuration

In order to edit an existing Address Autocomplete Configuration, use the *Edit* button . You can edit the Address Autocomplete Configuration and/or each of the Address Autocomplete Fields.

D.10.6 Delete an Address Autocomplete Configuration or Address Autocomplete Fields

In order to delete an Address Autocomplete Configuration or Address Autocomplete Field, mark the configuration or the field in the respective list and press the *Delete* button.

E - Global Configuration Section



In this section you will learn how to configure some global settings in the ConSol CM system:

- Languages
- Labels
- Queue Administration
- Projects
- Working with Calendars
- Business Calendars
- Microsoft Exchange Calendar Integration
- Classes of Text

E.1 Languages

This chapter discusses the following:

E.1.1 Languages

ConSol CM can be configured to offer localization functionalities for the Web Client in one or more of any number of languages. In the Admin Tool, this is configured on the navigation item *Languages* in the navigation group *Global Configuration*. This will also influence the languages which are offered in the Process Designer.

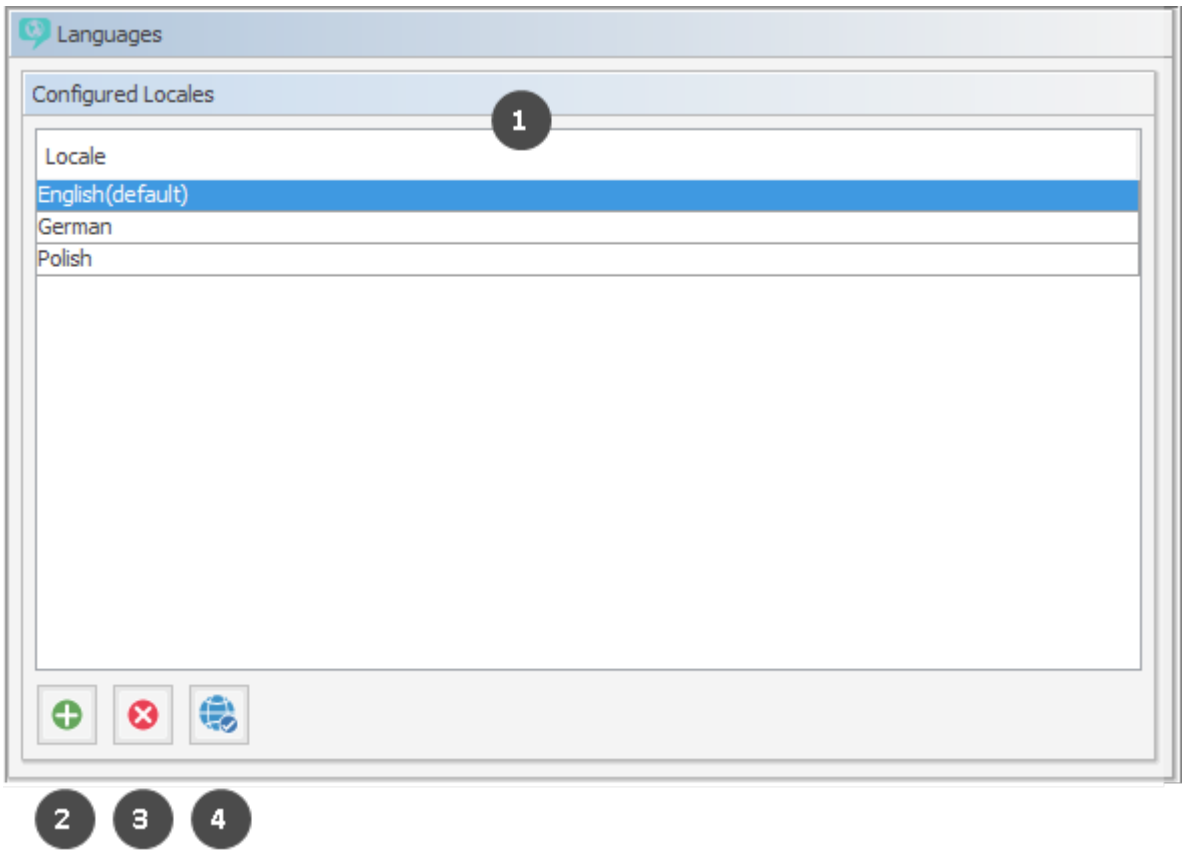


Figure 347: *ConSol CM Admin Tool - Global Configuration, Languages*

Configured Locales:
In this list, the locales (1) which will be available in the entire system are configured. This influences the lists for localized values in the Process Designer (e.g., for activities) and in the Admin Tool (e.g., for ticket field values). The displayed values for those activities or fields then depend on the locale of the web browser which is running the ConSol CM Web Client.

- Click the *Add* button to add more locales (2).
- Click the *Delete* button to remove the selected locale from the list (3).
- Click the *Locale* button to set the selected locale as the default locale. The default locale is used when the browser's preferred locale is not present in ConSol CM. E.g., if the engineer has set the browser locale to FR and in the ConSol CM administration only English (default), German, and Polish are available, English will be used as the default (4).

⚠ Make sure that the configured languages are installed on each machine where ConSol CM is running or is used. This will not be checked automatically.

E.1.2 The Use of Locales

For an engineer who works with the ConSol CM Web Client, the interface is displayed in the language that is configured in the web browser if it is a locale that is configured in ConSol CM. If no matching CM locale can be found, the default locale which has been set in the Admin Tool is used.

Depending on the location in the Admin Tool, the mechanism to enter the localized terms is slightly different. Please read the following section for a details explanation: The Different Modes of Localizing Terms and Labels Using the Admin Tool

In the Process Designer, the locales which have been configured in the Admin Tool are available. However, you can also delete locales in the Process Designer. Please refer to the *ConSol CM Process Designer Manual* for details.

E.2 Labels

This chapter discusses the following:

E.2.1 Introduction

It is possible to manipulate labels which are displayed on the GUI, i.e. in the Web Client. In this way, you, as an administrator, can customize the graphical user interface according to your company's requirements concerning terminology and/or Corporate Design. For example, in your company , it might be required to talk about cases instead of tickets, and a queue should be referred to as process. All this can be manipulated using the label settings in the Admin Tool.

E.2.2 Configuring Labels Using the Admin Tool

In the Admin Tool, labels are configured on the navigation item *Labels* in the navigation group *Global Configuration*.

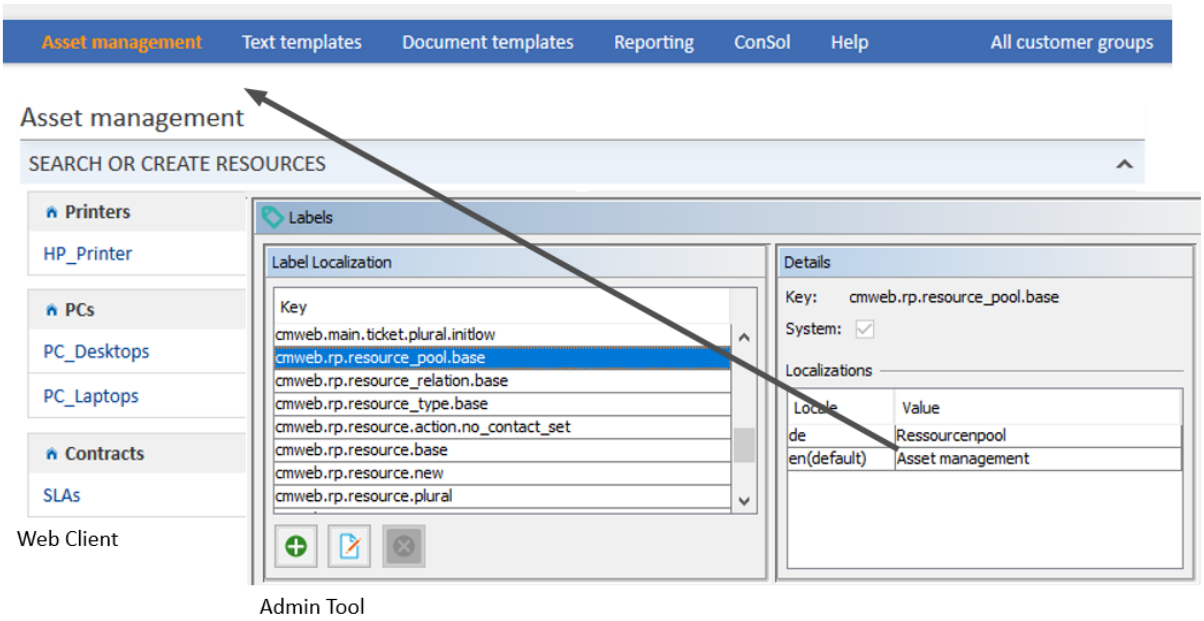


Figure 348: *ConSol CM Admin Tool - Global Configuration, Labels*

The following labels are currently available:

Label	GUI location	Default value
cmweb.main.engineer.base.initcap	Term for one engineer, capitalized. Used, e.g., in ticket header section.	Engineer
cmweb.main.engineer.base.initlow	Term for one engineer.	engineer
cmweb.main.engineer.plural.initcap	Term for more than one engineers, capitalized.	Engineers
cmweb.main.engineer.plural.initlow	Term for more than one engineers.	engineers
cmweb.main.queue.base.initcap	Term for one queue, capitalized. Used, e.g., in ticket list filter.	Queue
cmweb.main.queue.base.initlow	Term for one queue.	queue
cmweb.main.queue.plural.initcap	Term for more than one queue, capitalized.	Queues
cmweb.main.queue.plural.initlow	Term for more than one queue.	queues
cmweb.main.ticket.article.dative.initcap	Demonstrative pronoun (dative) for one ticket, capitalized	This
cmweb.main.ticket.article.dative.initlow	Demonstrative pronoun (dative) for one ticket	this
cmweb.main.ticket.article.demonstrative.initcap	Demonstrative pronoun (nominative) for one ticket, capitalized	This
cmweb.main.ticket.article.demonstrative.initlow	Demonstrative pronoun (nominative) for one ticket	this
cmweb.main.ticket.article.initcap	Article for one ticket, capitalized	The
cmweb.main.ticket.article.initlow	Article for one ticket	the
cmweb.main.ticket.base.initcap	Term for one ticket, capitalized	Ticket
cmweb.main.ticket.base.initlow	Term for one ticket	ticket

Label	GUI location	Default value
cmweb.main.ticket.new	Entry in Main menu for "Create new ticket"	Create ticket
cmweb.main.ticket.plural.initcap	Term for more than one ticket, capitalized	Tickets
cmweb.main.ticket.plural.initlow	Term for more than one ticket	tickets
cmweb.rp.resource.base	Title for the resource page and for the resource list on the resource type page	Resource
cmweb.rp.resource.new	Entry for "Create a new resource"	New resource
cmweb.rp.resource.plural	Term for more than one resources	Resources
cmweb.rp.resource.to	Used for the context menu Jump to ... (resource)	Jump to resource
cmweb.rp.resource_pool.base	Entry in main menu.	Resource pool
cmweb.rp.resource_relation.base	Expression for resource relation.	Relation
cmweb.rp.resource_type.base	Expression for resource type, e.g. on Detail Search page for resources .	Resource type

E.2.2.1 For CM Programmers: Defining Labels for Messages

You can also use labels to define error or information messages which should be displayed in the Web Client in certain situations. In this way, you do not have to use only standard CM properties but you can define very specific messages. The following three figures provide an example.

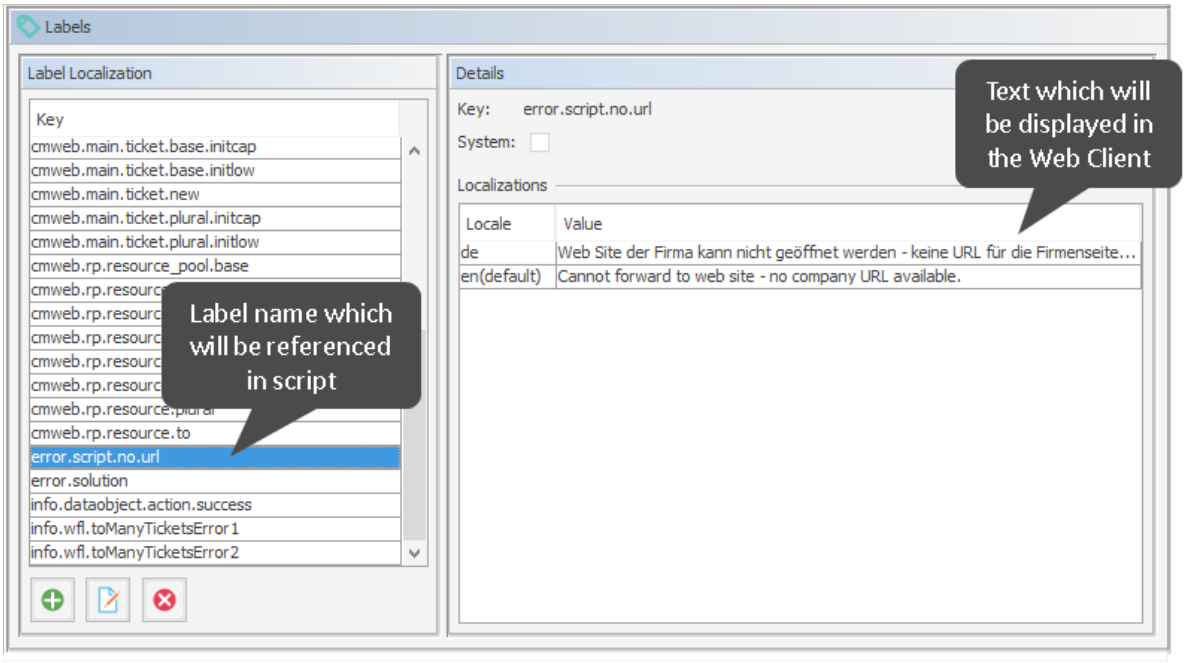


Figure 349: *ConSol CM Admin Tool, Global Configuration, Labels - System-specific label used for an error message (1): Label definition*

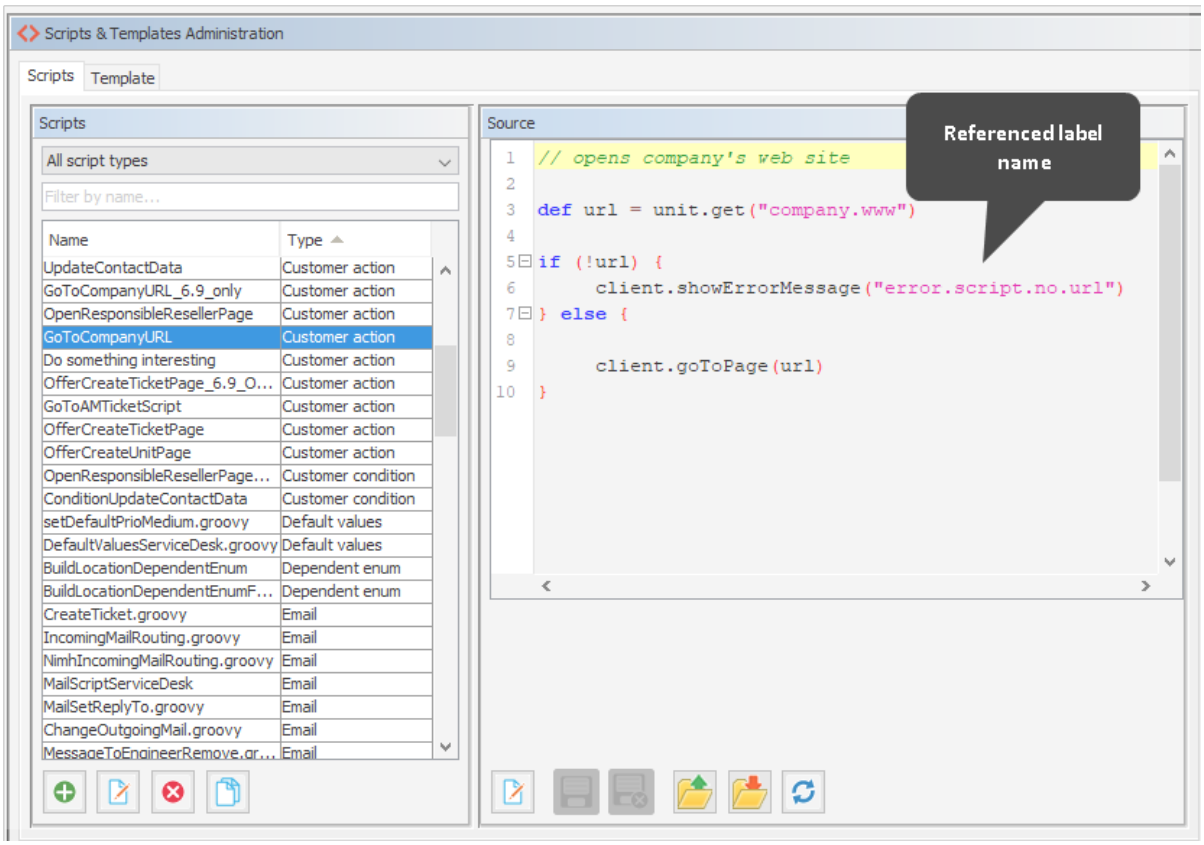


Figure 350: *ConSol CM Admin Tool, System, Scripts and Templates - System-specific label used for an error message (2): Script where the label is referenced*

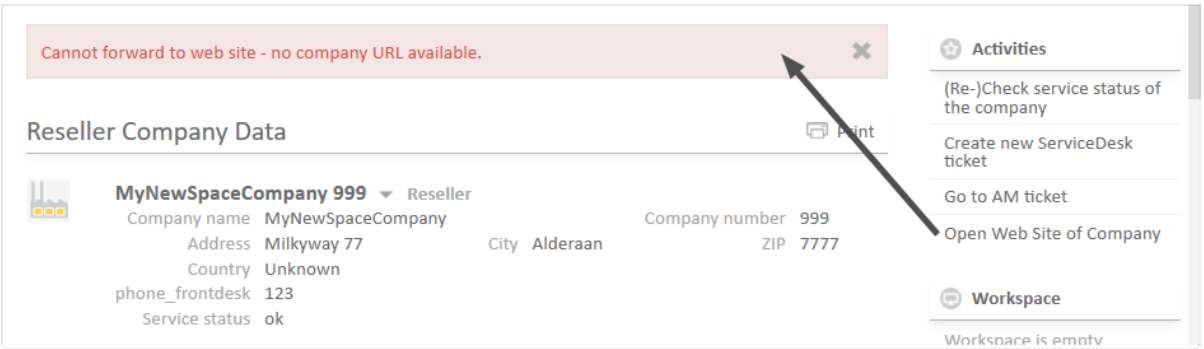


Figure 351: *ConSol CM Web Client - System-specific label used for an error message (3): Display in the Web Client*

You can also work with the `MessageProviderService`. This class provides a very generic way of using labels, not only for messages, but also for other GUI elements like names in charts or other objects. As you know from other services, a singleton of the class `MessageproviderService` is potentially available as `messageProviderService` object in all scripts.

You might need the method `getMessage(String pKey)`. This takes the key of a label as parameter. As a result, the localized value of the label is displayed, according to the default system locale.

Examples, taken from a widget script:

```
 //Some messages used by this widget
title = messageProviderService.getMessage
 ("web.chart.company.ticket.creators.title")
titleNoTicketsYet = messageProviderService.getMessage
 ("web.chart.company.ticket.creators.noResults")
seriesName = messageProviderService.getMessage
 ("web.chart.company.ticket.creators.series")
```

In case you want to display the message in a specific locale, you can work with the other signature of the method `getMessage`:

`getMessage(String pKey, Locale pLocale)`. This can also be used for another locale than the standard one.

Example, taken from a workflow script which checks the engineer workload. Here, the standard locale of the engineer who is logged in is used.

```
// Engineer can only accept ticket if he does not have too many tickets already
def curr_eng = workflowApi.currentEngineer
def max_tics = configurationService.getValue("custom-
 servicedesk","engineer.max.open.tickets")

// look for open tickets of current engineer
def engs = []
engs.add(curr_eng.id)

TicketCriteria tic_crit = new TicketCriteria()
tic_crit.engineerCriteria = TicketCriteria.EngineerCriteria.assigned(engs as Set)
tic_crit.status = TicketCriteria.Status.OPEN

List<Ticket> open_eng_tics = ticketService.getByCriteria(tic_crit)

def tic_number = open_eng_tics.size
def loc = engineerService.currentLocale

if (tic_number > max_tics) {
   log.info 'Too many tickets for engineer ' + engineerService.current + '. Current
    number is ' + tic_number
   // get text from labels defined in AT:

   def infoText1 = messageProviderService.getMessage
    ("info.wfl.toManyTicketsError1", loc)
   def infoText2 = messageProviderService.getMessage
    ("info.wfl.toManyTicketsError2", loc) + max_tics
   // alternative solution: workflowApi.addValidationError("INFO","You have too
    many tickets (" + tic_number + ") already, so you cannot accept another ticket.
    Maximum allowed number is " + max_tics)
   workflowApi.addValidationError("INFO",infoText1 + " " + infoText2)

} else {
   ticket.setEngineer(curr_eng)
}
workflowApi.reinitializeTrigger("defaultScope/Service_
 Desk/TimeTriggerDesiredDeadline")
```

Code example 58: *Script of activity "New IT ticket (Accept ticket)", engineer workload is checked*

A message like this might be displayed in this case:

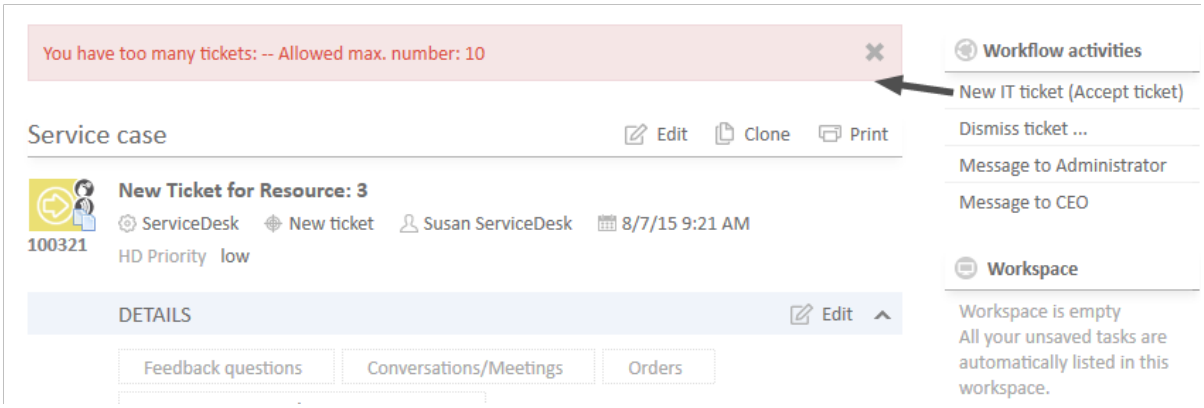


Figure 352: *ConSol CM Web Client - Message for English browser locale if the engineer has too many tickets to accept another one*

E.3 Queue Administration

This chapter discusses the following:

E.3.1 Introduction

Queues are a central element of ConSol CM. Tickets are grouped in queues, e.g., for certain tasks or work groups. Each queue is assigned a single workflow which controls the processing steps of all tickets in this queue. For example, there might be one queue *Service Desk* one queue *Marketing*, and one queue *Sales*.

The following parameters and objects are assigned to a queue (the parameters and objects are defined on other Admin Tool tabs, and are assigned to a new or existing queue here):

- The workflow of the queue (mandatory), i.e., the process which should be used for all tickets in the queue (e.g., all tickets of a department). A queue can only have one workflow but a workflow can be used by multiple queues.
- The template for the emails which are sent to engineers when a ticket is assigned or removed (optional).
- Several scripts that define the behavior of tickets in this queue (optional).
- One or more customer group(s) which are associated with the queue. Tickets can only be added to the queue for customers of this/these group(s) (one customer group is mandatory, more are optional).
- The business calendar (i.e., the working hours) which should be applied for tickets in this queue (optional).
- The data fields (ticket fields) which should be available in tickets in this queue. They are defined by assigning ticket field groups to the queue (some mandatory, some optional).
- The classes of text which should be available for tickets in this queue (optional).
- The project(s) which should be available for time booking in tickets of the queue (optional).

As a central element, the queue uses various objects and elements that are defined in other places, i.e., on another page of the Admin Tool, so usually all elements which are required for the definition of a queue are defined first. However, except for the workflow, all parameters can be modified after a queue definition has been saved for the first time (i.e. after the queue has been created), so you can configure the queue using an iterative approach if you like.

Furthermore, a queue is the basis for the assignment of access permissions, please see section Role Administration for details.

E.3.2 Queue Administration Using the Admin Tool

In the Admin Tool, queues are managed on the navigation item *Queues* in the navigation group *Global Configuration*.

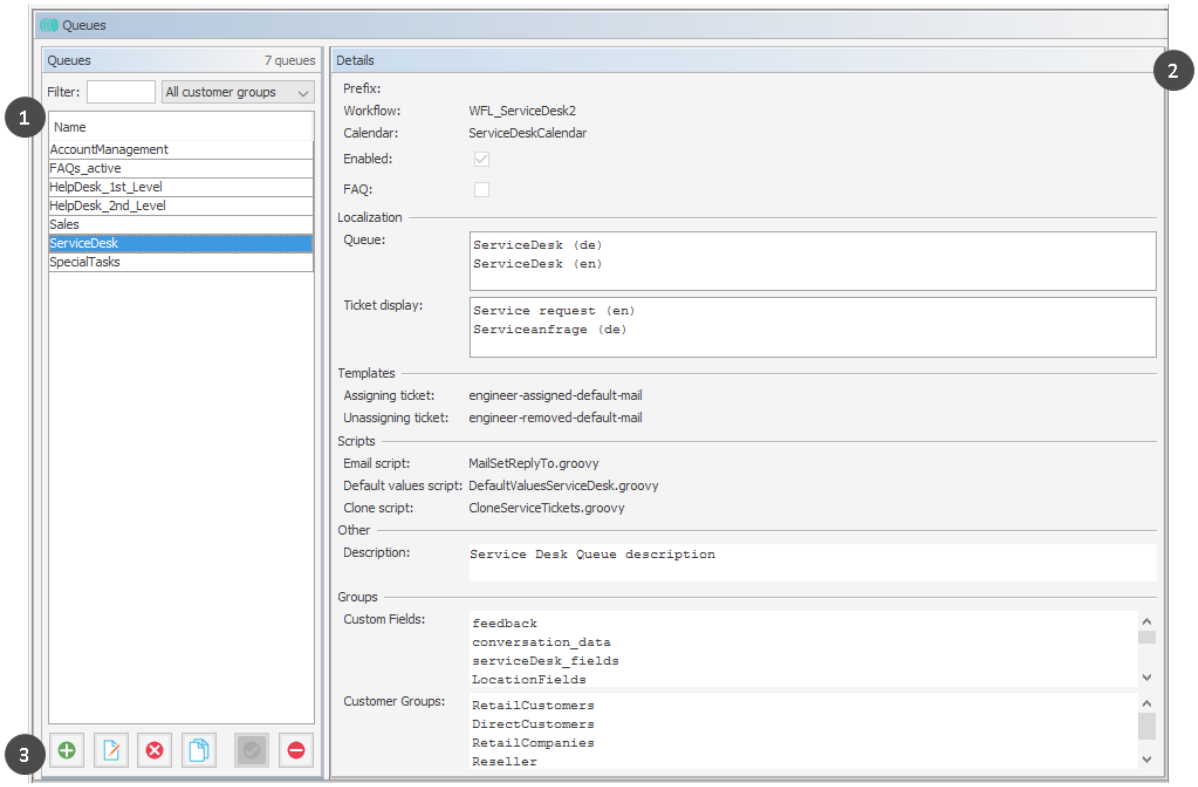


Figure 353: *ConSol CM Admin Tool - Global Configuration, Queues: Queue administration*

The navigation item *Queues* contains a list of all queues (1) which can be filtered using a string filter and/or a pull-down menu where the customer group can be selected. In the *Details* section (2) on the left hand side, all queue details are displayed (in read-only mode).

To add, edit, delete, copy or (de-) activate a queue, use the respective button in the button bar (3) below the queue list.

E.3.2.1 Filter the Queue List

Queues you want to edit or copy can be found most quickly if you enter filter information in the fields above the queue list.

You can filter for queues which

- contain a certain text string (blanks are interpreted, too) and/or
- are assigned to customer groups.

E.3.2.2 Create a Queue

You create a new queue by clicking the *Add* button below the queue list. The pop-up window which appears is the same for adding and for editing a queue. In both cases, you have to edit the queue details.

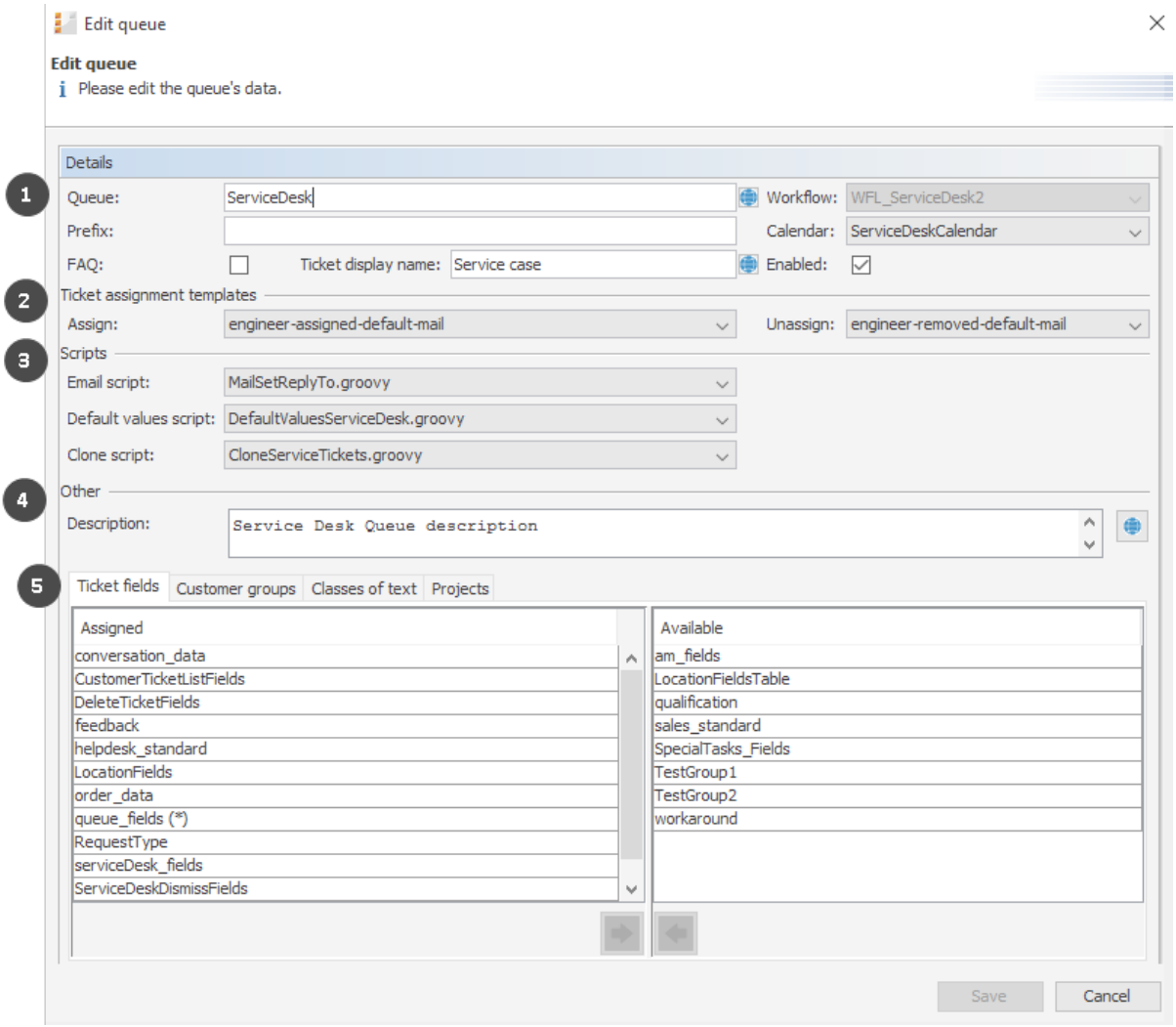


Figure 354: *ConSol CM Admin Tool - Global Configuration, Queues: Create a queue*

The following fields are available:

- **General configuration (1):**
  - **Queue:**
    Enter the technical queue name in this field. Click the *Localize* button to enter the localized queue name for all languages that are available in the system. The localized queue name (depending on the web browser locale) will be displayed in the Web Client in the ticket header. If no localized values are provided, the technical name will be displayed.

- **Workflow:**
  Choose the workflow for the queue from this list.

  > When you have developed and deployed a new workflow, it will only be available in the Admin Tool after a reload of Admin Tool data!

  > Once you have assigned a workflow to a queue it cannot be changed anymore!

- **Prefix:**
  You can enter a prefix for the ticket IDs of a queue, e.g., when the ticket ID should indicate to which queue or organizational structure it belongs.

  > The prefix remains with the ticket name if the ticket is moved to another queue.

- **Calendar:**
  Choose the business calendar for the queue from the list. Business Calendars in CM define working hours, holidays and the valid time zone (see section Business Calendars). They are used, e.g., for time triggers in the workflow and have to be activated explicitly for each trigger, i.e., in order to work with time calculations based on a business calendar, it has to be configured in three places:
  - In the navigation group *Global Configuration*, navigation item Business Calendars the calendar is created and the active and vacation times are configured.
  - In the queue administration a calendar is assigned to the queue.
  - For each time trigger in the workflow the use of the queue-specific calendar can be activated or not. Refer to the *ConSol CM Process Designer Manual* for a detailed explanation of working with time triggers.
- **FAQ:**
  Ticking this checkbox marks the queue as a knowledge base for CM/Track users. They can search for tickets in this queue in CM/Track, the ConSol CM Web Portal. Please see also section CM/Track V1: FAQs in CM/Track or CM/Track V2: FAQs in CM/Track for more information about this topic.
- **Ticket display name:**
  Optional. The string entered here will be displayed as ticket header. If a localized value is provided, this will be used, otherwise, the technical name will be used. In this way, you can adapt the CM system to display terms like *Service case*, *Request* or *Case* and you are not limited to the term *Ticket*. If nothing has been defined manually, the default value *Ticket* is used.
  Please note that two manually defined parameters can potentially define the value for the ticket display name.

- The ticket display name set here, in the queue details definition. This is queue-specific, i.e. for each queue, a different display name can be defined. For example, in the Service Desk queue, this might be *Service case*, whereas in the Sales queue, this might be *Opportunity*.
- The value which is defined using the label `ticket.base.initcap`. This is used only if the queue-specific ticket display value is not set. For details about labels, please read the section about Labels.

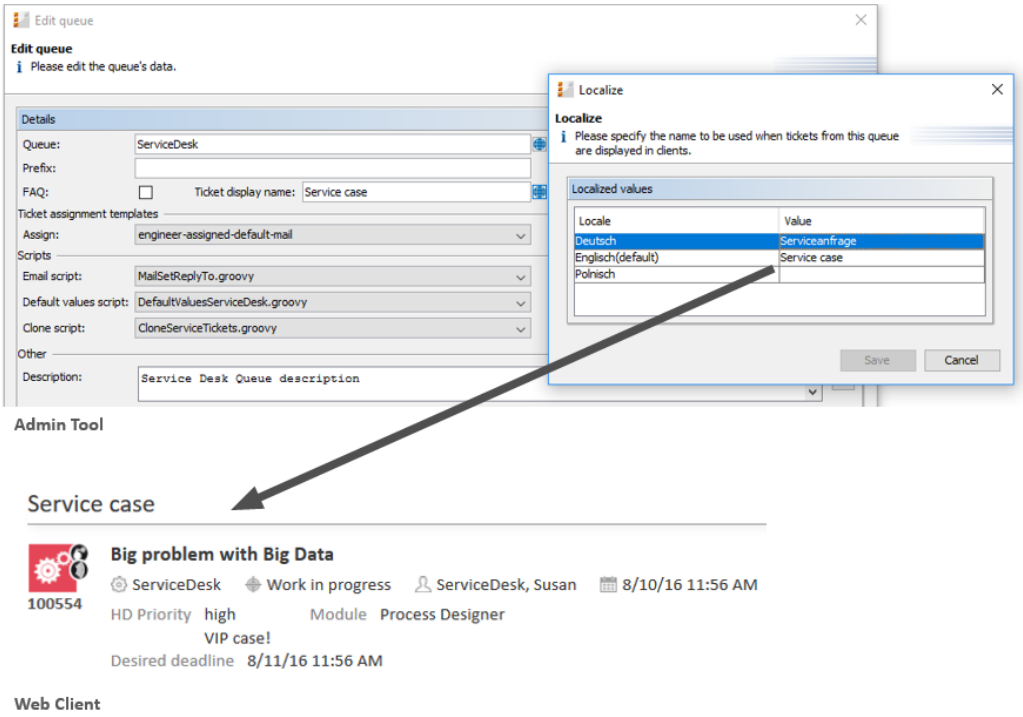


Figure 355: *ConSol CM Admin Tool and Web Client: Setting and viewing a queue-specific ticket header*

- **Enabled:**
  If this checkbox is checked, the queue is immediately available in the system after saving, otherwise the queue is disabled. In enabled queues, you can create tickets. In disabled queues, this is not possible.

- **Ticket assignment templates (2):**
  Here you can choose email templates which are used for automatic emails which are sent to the (new) engineer when a ticket is assigned to an engineer (*Assign*), or to the (old) engineer when a ticket is taken away (*Unassign*). When you have defined the templates in the *Scripts & Templates Administration* of the Admin Tool (see section Admin Tool Templates) they will be available in the drop-down menu. If you do not want the ConSol CM system to send an automatic email in case of the engineer operation, just leave the field empty. Please keep in mind

that the system properties cmas-core-server, mail.notification.engineerChange (= “true”) and cmas-core-server, mail.notification.sender have to be set, see sections System Properties and Administrator and Notification Email Addresses for details.

- **Scripts (3):**
  Scripts are used to automate recurring tasks and activities. They are managed and stored in the *Scripts & Templates Administration* (see section Admin Tool Scripts). You can assign:
  - **Email script**
    Choose a script from the list if outgoing emails for this queue should be modified by the script, e.g., to contain default values like the sender or address fields. The email script indicated here is the last script that processes an outgoing email, so all former settings will be overwritten if a variable has been set before. All scripts of type *Email* that are stored in the script section are available for selection, so please make sure to pick the correct one.

When you work with the configuration of the Reply-To email address, please note the following technical behavior of ConSol CM and adapt your system accordingly!

The technical background:

There are four potential Reply-To addresses which you deal with:

1. The **Reply-To address which is set with the system property `mail.reply.to`**. If it is set, it will be displayed in the Ticket Email Editor in the Web Client. If it is really the effective Reply-To address in an email depends on the configuration in the queue-specific outgoing email script. See next item. If the Page Customization attribute *showReplyTo* for the type *mailTemplate* is set to “false”, no Reply-To address will be displayed in the Ticket-Email-Editor, but if the property `mail.reply.to` is set, this address will be used anyway - unless an outgoing mail script sets another address, see next item.
2. The **Reply-To address which is set in a queue-specific outgoing email script**. Since the outgoing email script is the last instance which processes an outgoing email, the Reply-To address set in this script will always be the effective Reply-To address which is used. In case the `mail.reply.to` property is set, this mail-reply.to-address will not really be used (but it will be displayed in the Ticket Email Editor which might cause some confusion! What that means for your system configuration is explained in the next section).
3. The **email address which is set in the system property `mail.from`.** If this is set and neither `mail.reply.to` nor a queue-specific Reply-To address is set, most email clients will set the From address as Reply-To address.

4. The **email address of the current engineer** (the engineer who is logged in to the Web Client). This personal email address is used as Reply-To address for emails from the Web Client if neither the `mail.reply.to` property is set nor a queue-specific outgoing email script is configured nor the `mail.from` property is set.

In the Web Client, in the ticket history, the Reply-To address which was really used is always displayed for outgoing emails. So even in case there should be a difference between the address which was displayed in the Ticket Email Editor (the `mail.reply.to` property) and the Reply-To address which was really used (the Reply-To in the queue-specific outgoing email script), the effective address is displayed. This would be the one from the script in this case.

What we recommend:

A system Reply-To address should always be set! You can decide if you

- work with the Reply-To address in the queue-specific outgoing email script

or

- use the `mail.reply.to` system property.

However, since the email communication should take place via ConSol CM and not using personal email addresses, one of the two system settings mentioned above should be used to prevent CM from using personal email addresses as Reply-To. The latter would automatically lead to customer emails being sent to an engineer's personal email account instead of CM.

What that means for your system configuration:

1. The simplest way to set a Reply-To address is by using the `mail.reply.to` system property. It will be displayed in the Ticket Email Editor and will be the effective Reply-To address.
2. If queue-specific Reply-To addresses are required, we recommend to write one outgoing mail script where queue names are mapped to specific Reply-To addresses. This can then be extended for Bcc, Cc or other addresses.
   You can combine the `mail.reply.to` property and queue-specific Reply-To addresses: for all queues without a specific outgoing mail script, the `mail.reply.to` address will be used, for all queues which have a queue-specific outgoing mail script that contains a Reply-To address, this will be used.

What that means when you work with workflow scripts which send emails:

(A detailed explanation is provided in the *ConSol CM Process Designer Manual*!)

- Use the object and method `configurationService.getValue("cmweb-server-adapter","mail.reply.to")` to retrieve the value of the system property and set it as Reply-To address in the outgoing email.
- Use the `Mail` object when the queue-specific script should be used: e.g. `mail.useDefaultScript()`. This will overwrite the `mail.reply.to` property!

If neither the system property nor the queue-specific outgoing email script is used, i.e. when the Reply-To address is not set, usually the From address will be used as Reply-To by the email client.

- **Default values script**
  Here you can select a script to preset values of list boxes when creating a ticket for this queue in the Web Client. The script has to be present in the *Scripts & Templates Administration* of the Admin Tool and has to be of type *Default values*. Please refer to section Scripts of Type Default Values for details about this topic.
- **Clone script**
  Here you can select a script which is executed when a ticket in this queue is cloned (duplicated) using the Web Client (*Clone* option in the ticket menu). The script has to be present in the *Scripts & Templates Administration* of the Admin Tool and has to be of type *Clone*. The clone script sets default values for a ticket which is created using the *Clone* operation. Please refer to section Scripts of Type Clone for details.

- **Other (4)**
  - **Description:**
    You can enter a free-form description in this field, e.g., to document the purpose of the queue. This information is shown in the Admin Tool only.
- **Tabs for assignment of fields, customer groups, classes of text, and projects (5):**
  - **Tab *Ticket fields*:**
    In order to show data fields (ticket fields) in tickets of the queue, you have to assign the respective ticket field groups here. See section Ticket Field Administration (Setting Up the Ticket Data Model) for details about the definition of ticket fields.

    Tab *Customer groups*:
    Tickets in the queue can only be created for customers from the selected customer groups. Please make sure that the engineers who are supposed to work with tickets of the queue also have the respective access permissions to the customer (group) data.

    Tab *Classes of text*:
    Here you can assign the classes of text which should be available in tickets of this queue. Please see section Classes of Text for an explanation of the text class definition.
  - **Tab *Projects*:**
    Here you can assign projects to the queue. Engineers who work on a ticket in the queue

can book times on the projects that have been assigned to the queue. Projects are defined on the Projects page.

On each tab you can assign a selected entry by clicking the *Assign* button and remove it by clicking the *Unassign* button.

Click *Save* afterwards to create the queue. The details of the new queue are displayed on the right-hand side of the page.

E.3.2.3 Edit a Queue

If you want to edit a queue, select it in the list and click the *Edit* button or double-click the name of the queue. Modify the queue details and click *Save* to store your modifications.

> ⚠ You cannot change the workflow of a queue once the queue has been saved for the first time!

E.3.2.4 Delete a Queue

Select the queue you want to delete in the list and click the *Delete* button. If you confirm the following dialog with *Yes*, the queue will be deleted and is no longer available in the system.

> ⚠ If there are still tickets for a queue it cannot be deleted. You have to move the tickets to another queue before you can delete it.

E.3.2.5 Copy a Queue

The *Copy* button allows you to save time when creating a queue. The selected queue will be copied. The new queue has the same name as the copied queue. Double-click the name or click the *Edit* button to open the edit window where you can modify the name and details of the queue. Click *Save* to store your modifications.

> ⚠ You cannot change the workflow of a queue once the queue has been saved for the first time!

E.3.2.6 Enable or Disable a Queue

You can disable a queue to prevent that new tickets can be opened in this queue. This allows a queue to be made temporarily unavailable without having to delete it. To disable a queue, select the queue in the queue list and click the *Deactivate* button. The entry in the list is now shown in italics. Just click the *Activate* button at the bottom of the page to enable the queue again.

What engineers in the Web Client can do in a disabled queue, provided they have the required permissions:

- execute workflow activities thereby moving the ticket through the business process
- edit the ticket
    - edit ticket fields
    - add comments, emails, attachments

What engineers in the Web Client can NOT do in a disabled queue:

- open / create new tickets
- move tickets from another queue to this queue using the drop-down menu in the ticket

E.4 Projects

This chapter discusses the following:

E.4.1 Introduction

With ConSol CM you can book working time to projects. Please see the Time Booking Using ConSol CM section for a detailed explanation.

E.4.2 Managing Projects Using the Admin Tool

Projects are managed via the navigation item *Projects* in the navigation group *Global Configuration*. For a project to be active and available in the Web Client, it has to be assigned to a queue (see section Queue Administration). In the Web Client you can then book working time to tickets that are in one of the queues where the project has been assigned. Engineers can see their time bookings on the engineer profile page.

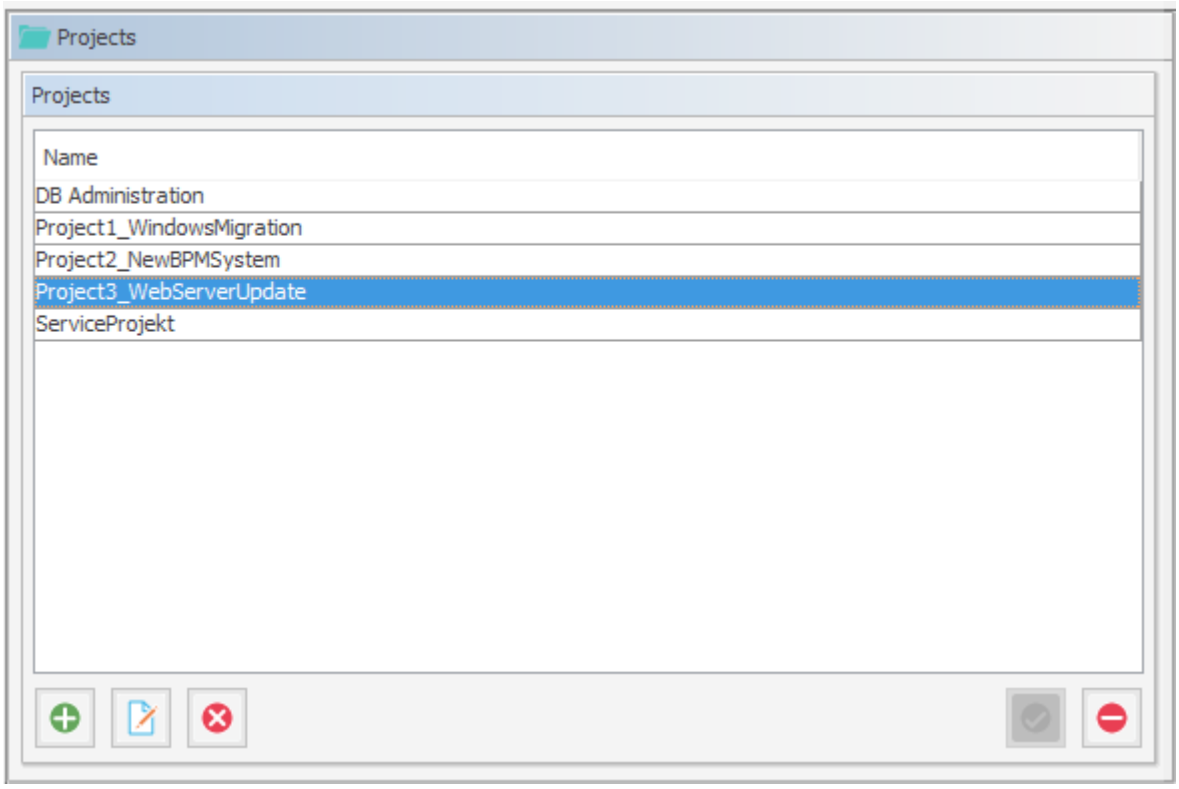


Figure 356: *ConSol CM Admin Tool - Global Configuration, Projects*

E.4.2.1 Create or Edit a Project

A project is defined by its name. Click the *Add* button to open a pop-up window where you can enter the name. Using the *Localize* button next to the name field you can localize the name (see Localize a Project). The checkbox *Enabled* is pre-selected to activate the project in the system (see alsoDisable or Enable a Project). You will get the same window when you click the *Edit* button to edit a project.

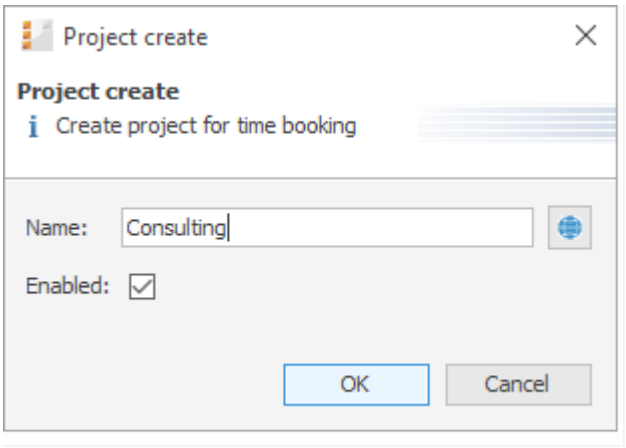


Figure 357: *ConSol CM Admin Tool - Global Configuration, Projects: Create or edit a project*

E.4.2.2 Delete a Project

A project can only be deleted if it is not assigned to any queues and has not been used for any time bookings. Otherwise you get a warning and can only disable this project (see below).

In order to delete a project, select it in the list and click the *Delete* button. After choosing *Yes* in the confirmation dialog the project will be removed from the list and the system.

E.4.2.3 Disable or Enable a Project

If a project is still assigned to a queue or has been used for a time booking in a ticket, but is not needed anymore, you can disable it. To do this select the project and click the *Deactivate* button. The entry in the list is shown in italics afterwards. The project is not available for new time bookings anymore. Just click the *Activate* button at the bottom of the page if you want to enable the project again.

You can also enable or disable a project in the window used for editing projects by selecting or deselecting the checkbox *Enabled*. When you create a project this checkbox is automatically selected.

E.4.2.4 Localize a Project

Click the Localize button in the create or edit window to enter a localized name for a project. See section Localization of Objects in General, Type 1 for a details explanation of the localization mechanism.

E.5 Working with Calendars

In ConSol CM, there are two configuration options for calendars:

1. You can define business calendars which can be used to manage business hours for one or more teams which work with ConSol CM. This is covered in section Business Calendars.
2. You can configure ConSol CM to access Microsoft Exchange Server calendars. This is explained in section Microsoft Exchange Calendar Integration.

E.5.1 Business Calendars

E.5.1.1 Introduction

A business calendar defines working hours. This can be used, for example, to represent the business hours of the Service Desk team to avoid system escalations being fired during non-business hours.

In ConSol CM, you can define as many business calendars as your company's environment requires. In this way you can configure specific working hours for each team.

> Tickets which have not been assigned to an engineer within an hour of opening the ticket are automatically moved to an escalation level. If a calendar defines working hours from 8 a.m. to 5 p.m. and a ticket arrives at 4:45 p.m., the ticket will not escalate at 5:45 p.m. but at 8:45 a.m. the next day. This time is calculated as follows: 15 minutes between ticket arrival and end of the working hours plus 45 minutes from next beginning of the working hours until the full hour given by the escalation limit is reached.

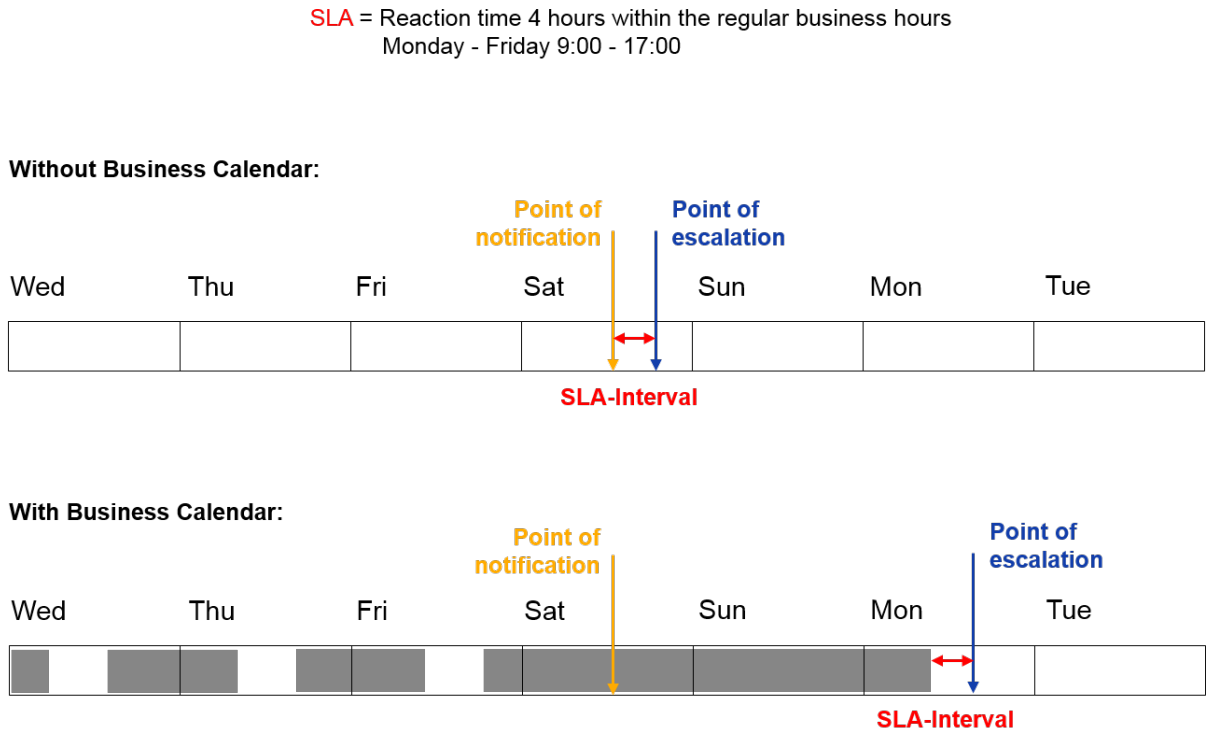


Figure 358: *ConSol CM Principle - Business calendar*

Aside from working hours, you can define holidays, too. On these days the automatic escalation pauses entirely. Holidays have to be defined per calendar. It is not possible to define a holiday that is valid for all existing calendars simultaneously.

If you want to work with times which are defined in a business calendar (e.g., use active time for a timer trigger in a workflow for an escalation), you have to perform three steps:

- create the business calendar with its active/inactive times (Admin Tool, see explanation below)
- assign the business calendar to all queues where it should be in operation, see section Queue Administration (Admin Tool)
- assign the use of a calendar to every single workflow element where the calendar should be used as the basis for time calculations (Process Designer), as explained in the *ConSol CM Process Designer Manual*.

E.5.1.2 Configuration of Business Calendars Using the Admin Tool

In the Admin Tool, business calendars are defined via the navigation item *Business Calendars* in the navigation group *Global Configuration*.

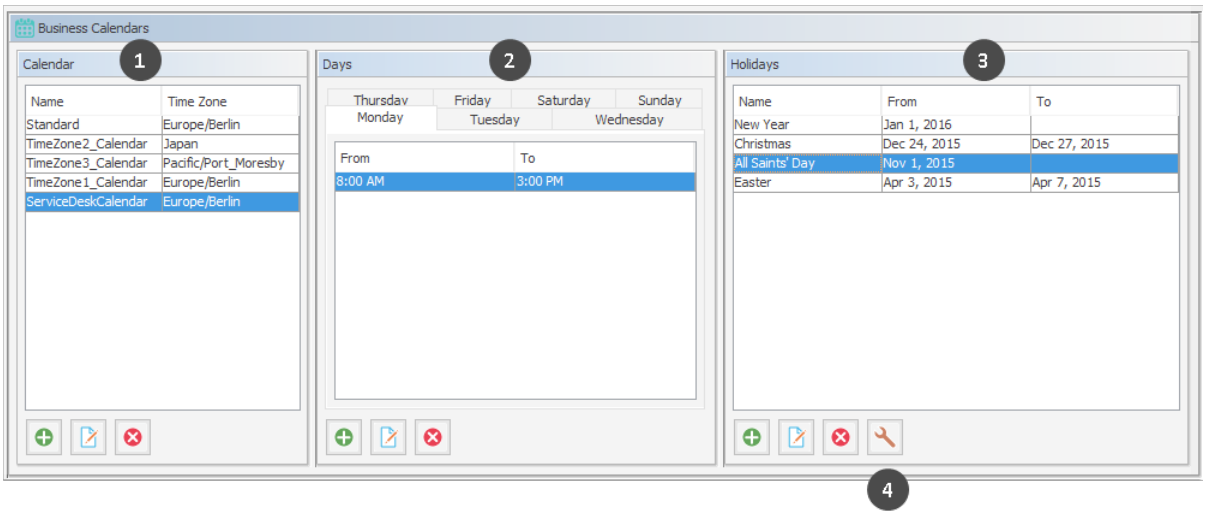
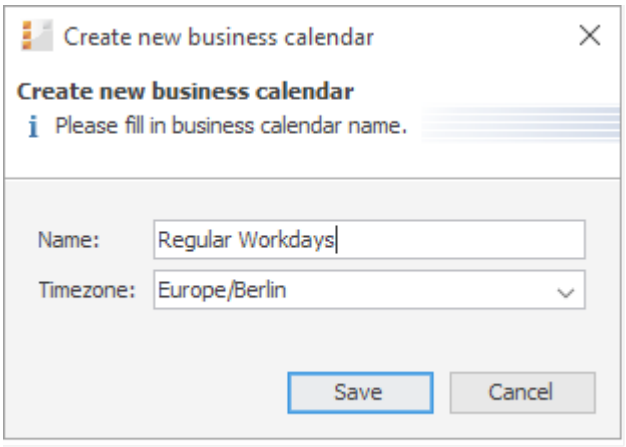


Figure 359: *ConSol CM Admin Tool - Global Configuration, Business Calendars*

- List of available business calendars (1)
- Configured working hours for the selected calendar (2)
- Holidays in the selected calendar (3)
- Button to import holiday definitions from an external csv file (4)

Creating a New Calendar

Click the *Add* button in the left part of the page to create a new calendar. The following window appears:



Regulre

Figure 360: *ConSol CM Admin Tool - Global Configuration, Business Calendars: New calendar*

- **Name:**
  Enter a unique name for the calendar.
- **Timezone:**
  Choose the time zone to be used for the calendar.

> ⚠ This field only describes to which time zone the defined hours refer. The calendar itself is valid worldwide for the respective workflow!
>
> **Example:**
>
> The ConSol CM server is located in Detroit, MI, USA. In the business calendar, Europe/Berlin is set as time zone. A time trigger which uses the business trigger would fire based on Berlin time, not Detroit time.

Click *Save* afterwards to create the calendar.

By clicking the *Edit* button, you can modify a selected calendar in the same way. Click the *Delete* button to delete the selected calendar.

Defining the Working Hours for a Calendar

Select a calendar on the left and click the *Add* button in the middle part of the page to create the days and hours for this calendar. The following window appears:

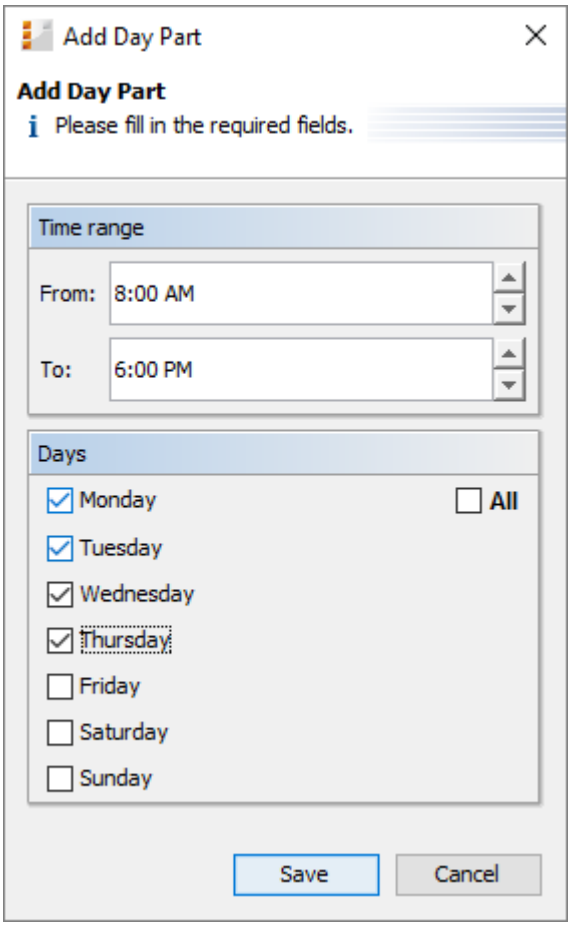


Figure 361: *ConSol CM Admin Tool - Global Configuration, Business Calendars: Working hours of a calendar*

- **Time range**
  Enter the time range for which the automatic workflow escalations shall be active.
- **Days**
  Select the checkboxes of the days for which the time range shall be valid. It is possible to choose individual days or all days at once (checkbox *All*).

> If the system detects an inconsistency between the time defined here and an already existing time, you will get a corresponding message.

Click *Save* afterwards to create this time range for the marked days.

If you want to edit the time range later, you have to do it separately for each day. Select the respective day, click the *Edit* button and change the time range in the window that appears. Or click the *Delete* button to delete the time range for a selected day. It is not possible to edit or delete the time range for multiple days at once.

Defining the Holidays for a Calendar

You can define the dates and time periods for holidays using one of two approaches:

- Defining the holidays manually.
- Importing the holidays from an Excel file.

Defining the Holidays for a Calendar Manually

Select a calendar and click the *Add* button in the right side of the page to create a new holiday entry. The following window appears:

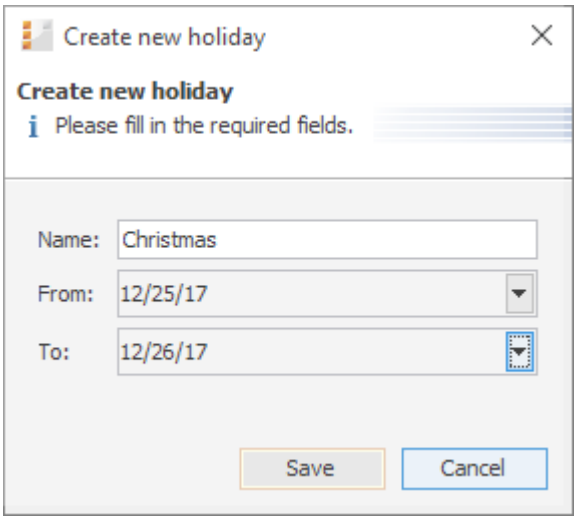


Figure 362: *ConSol CM Admin Tool - Global Configuration, Business Calendars: Holidays of a calendar*

- **Name:**
  The name of the holiday.
- **From:**
  The date of the holiday.
- **To:**
  If it is a multi-day holiday (e.g., Christmas), you can enter the last date of the holiday here.

  > It is not possible to define holidays that last only half a day.

Click *Save* afterwards to create the holiday.

If you want to edit a selected holiday entry just click the *Edit* button. Clicking the *Delete* button deletes one or more selected entries.

Importing Holidays for a Calendar from a .csv File

Holiday data can be imported from a .csv file conforming to the following format:

- First column: title/name of the holiday
- Second column: start date
- Third column: end date (use the same date as start date if it's a one-day holiday)
- Separator: comma (no comma at the end of the line)
- Slashes for the dates

```
Christmas,24/12/2015,27/12/2015
New Year,01/01/2016,01/01/2016
Easter,03/04/2015,07/04/2015
```

Please note that all dates for the holiday import must be written in the following format (as shown in the example): DD/MM/YYYY!

In the Admin Tool navigation group *Global Configuration*, navigation item *Business Calendars*, select a calendar and click the *Import holidays* button and enter the path to the .csv import file.

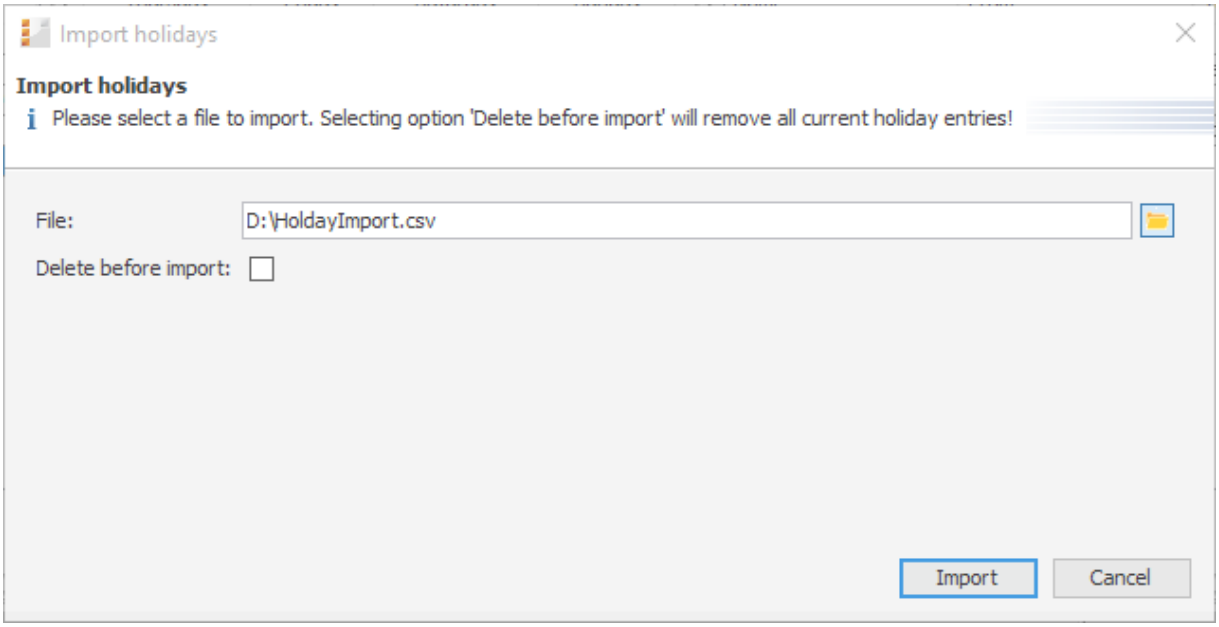


Figure 363: *ConSol CM Admin Tool - Global Configuration, Business Calendars: Importing holidays*

The new holidays will be imported in the holiday list of the selected business calendar.

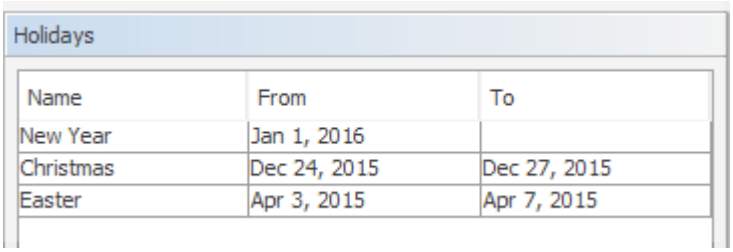


Figure 364: *ConSol CM Admin Tool - Global Configuration, Business Calendars: Newly imported holidays*

E.5.2 Microsoft Exchange Calendar Integration

E.5.2.1 Introduction

Starting with CM version 6.10, it is possible to include a Microsoft Exchange calendar view in the Web Client.

Please refer to the ConSol CM System Requirements of your CM version for a list of supported Exchange Server versions.

The calendar view can be offered ...

- on the ticket page
- on the customer page, i.e.,
    - on the contact page
    - on the company page

The calendar will be displayed in a distinct section of the ticket / customer page.

An engineer who works with the calendar view can ...

- display the monthly or weekly view
- move existing appointments using drag-and-drop
- create new appointments (if full access has been configured)

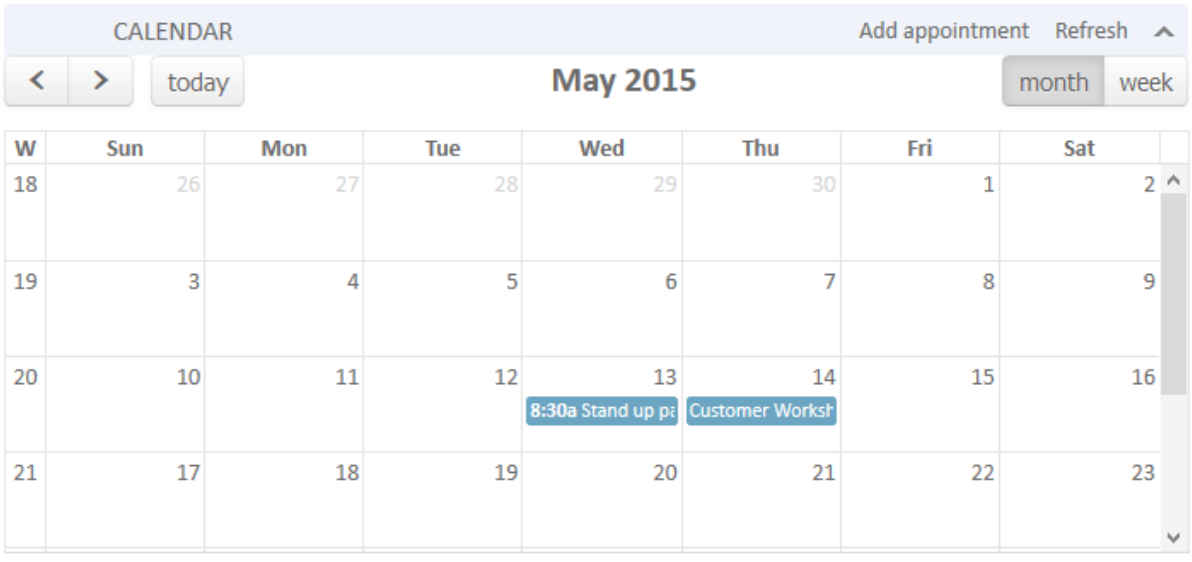


Figure 365: *ConSol CM Web Client - Ticket with Calendar section (monthly view)*

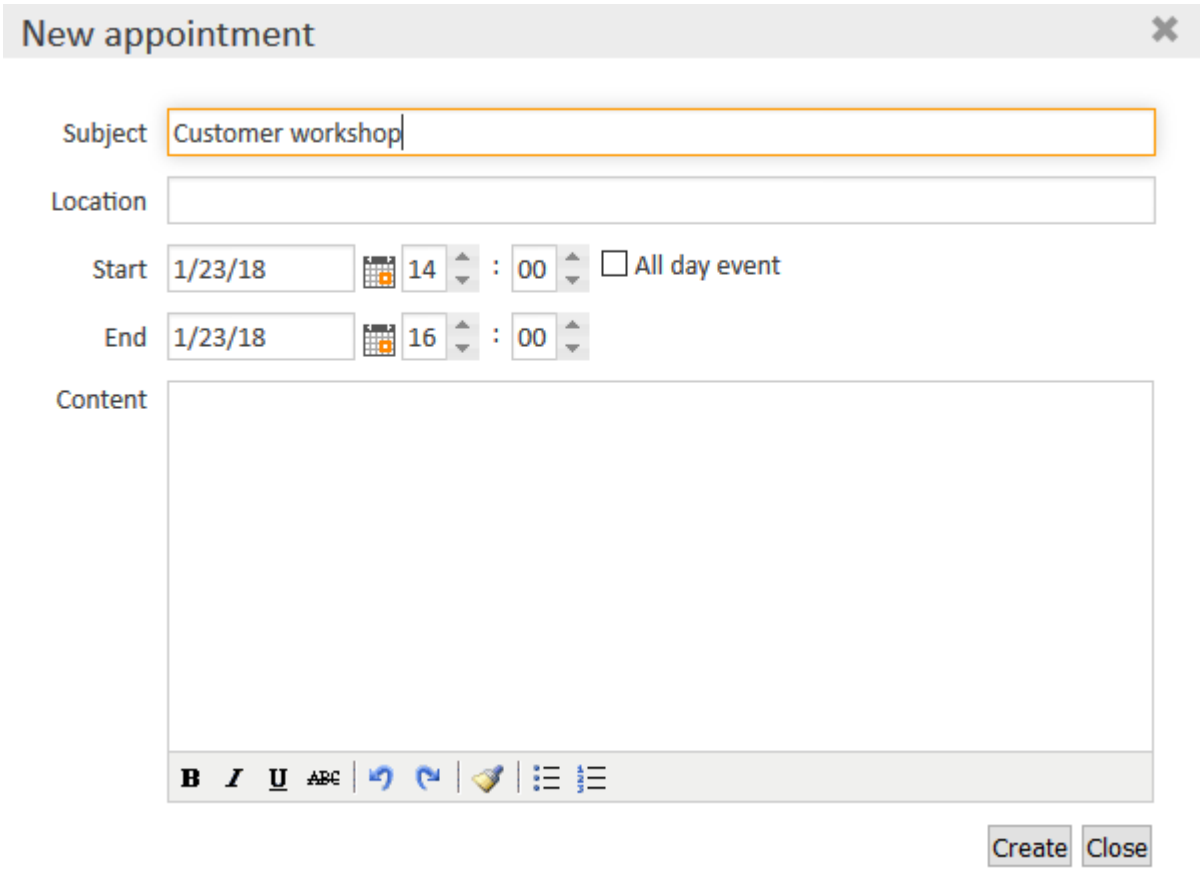


Figure 366: *ConSol CM Web Client - Adding an appointment in the calendar*

E.5.2.2 Configuring the Microsoft Exchange Calendar Integration

Basic Configuration

The integration of a Microsoft Exchange server to provide calendar data is done based on *page customization*. For a detailed introduction to this topic, please read the section about Page Customization. Here, only the calendar-specific configuration is explained.

Perform the following steps:

1. **Make the calendar section visible (example for the Ticket Edit page):**

   Log in as an administrator and open a ticket. Select *Enable page customization* in the main menu. Since the calendar section is not yet displayed, you cannot mark the element you want to configure, but instead have to select it in the page customization tree. Select *calendar/ticketEditPage/calendarSection* and set the attribute *state* from “hidden” to “expanded”. Alternatively, you can set “collapsed”. This will initially display a collapsed *Cal-*

endar section and the engineer can expand it manually. In both cases, the calendar section of the ticket will be visible. As header *No Calendar* will be displayed. The configuration of the calendar follows in step 2.

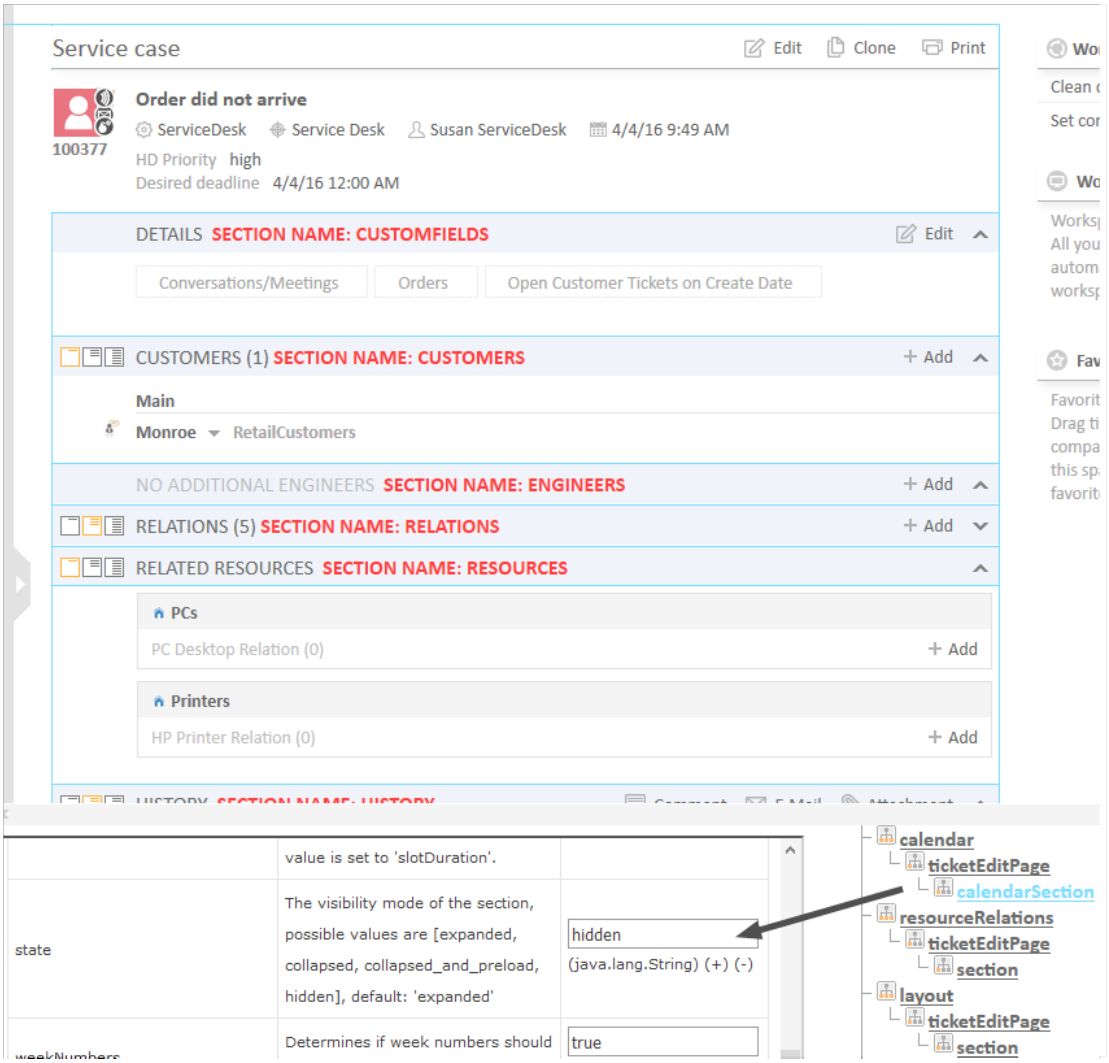


Figure 367: *ConSol CM Web Client - Using page customization to make the Calendar section of a ticket visible*

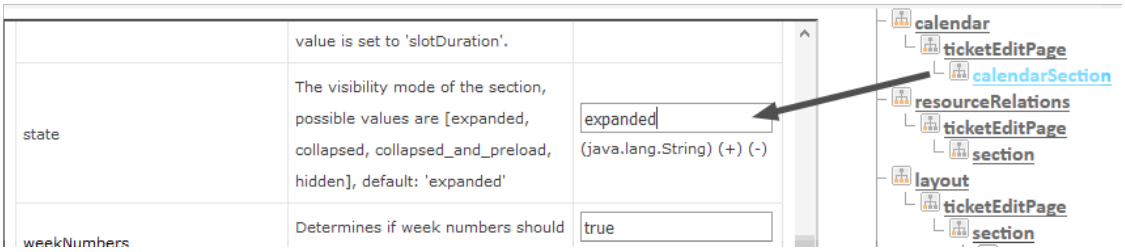


Figure 368: *ConSol CM Web Client - Page Customization Definition Section (PCDS) for the Calendar section of tickets*

2. **Configure the calendar script:**

   The connection to a Microsoft Exchange server calendar is implemented using an Admin Tool script. The script has to be stored in the Admin Tool script section, i.e., navigation group *System*, navigation element *Scripts and Templates*, and must be of script type *Calendar integration*. The name of this script has to be entered as the value for the attribute *calendarInitializationScript* in the page customization (see step 1). This Admin Tool script is used to initialize the calendar source and has to return a map which contains all parameters describing the source. If the returned map is empty or is null, no calendar will shown (the section will be empty with the label *No calendar*).

   The script has to be coded as shown in the following example:

```
return [
   name: 'Exchange Source',
   'access.type': 'EXCHANGE',
   'access.url': 'https://exchange1.server.net/EWS/Exchange.asmx',
   'access.username': 'exchange-user',
   'access.password':'exchange-password',
   'access.domain': 'SSO',
   'access.impersonation':'somebody@sso.server.net',
   'access.version': '2013'
]
```

   Code example 59: *Example calendar integration script*

   The following parameters can be used in the script:

   - **name**
     String. The name of the source. For now it has no functional use, but in future CM versions, when multiple sources can be used in the calendar section, it will be used to identify them in the user interface.
   - **color**
     String. Background color for appointment entries. Format: HTML colour format (e.g., red or #D80000)

- **editable**
  Boolean. Indicates whether appointment creation/editing/removal should be enabled. If this value has not been set explicitly, the value from the section customization will be used, which by default is set to “false”. NOTE: appointments are editable only if the current user has write permissions to the ticket in which the calendar section is shown.
- **access.***
  Properties describing access to the calendar: credentials, connection details, URLs ... etc.
  a. **access.type**
     String parameter. Possible values EXCHANGE (MS Exchange Server), RANDOM (randomly generated appointments for testing purposes).

     For access.type EXCHANGE:

     i. **access.url**
        URL. The url of the .asmx file (Active Server Method File) on the Exchange server. The file is usually located in the EWS (Exchange Web Services) directory and provides the exchange access web service.

     ii. **access.username**
        String. The name of a technical user under which the access/login at the Exchange server is performed.

     iii. **access.password**
        String. The password of the technical user under which the access/login at the Exchange server is performed.

     iv. **access.domain**
        String. The Windows domain of the Exchange server. The technical user (used for access.username and access.password, see above) has to be a member of this domain.

     v. **access.impersonation**
        String. Email account of an Exchange calendar user. The calendar on the ticket page or on the customer page will be based on the view of this Exchange calendar user. For example, *somebody* could be the Exchange login of the ConSol CM engineer who is currently logged in. If the name of an engineer is the same in Exchange and ConSol CM, you can simply pass the current engineer's name to Exchange using the method `workflowApi.getCurrentEngineer().getName()` or `engineerService.current.name`.

     For access.type RANDOM:

     i. **access.calendar**
        Name of the calendar file. Random calendar provider stores generated appointments on disk (files will be automatically removed when CM server is stopped). Using same name in configuration ensures that user will have the same set of appointments.

Additional variables available in the script:

- Ticket context: `ticket`
- Contact/customer context: `unit`

3. **Enable edit mode for the calendar:**
   To allow full access (i.e., to create and/or edit appointments) for engineers, set the page customization attribute *editable* to “true”.

Basic principle of Exchange server calendar access

The configured Exchange server is contacted, namely the Web Service indicated in the url. The login at this server is performed using the technical user (access.username, access.password). Then the user is changed to the person/user indicated under *access.impersonation*. The latter is performed using the Exchange server function *Impersonation*. The calendar of this impersonated user is the displayed on the ticket or customer page.

> The impersonation function can only be used by an account which has been granted the *ApplicationImpersonation* role by the Exchange administrator. Please make sure all security aspects are taken into consideration when you set up this role!

Advanced Configuration: Page Customization Parameters for the Microsoft Exchange Calendar Integration

Short Background Information about the Microsoft Exchange Calendar Integration

The integration (more precisely, the *display*) of Microsoft Exchange calendars in ConSol CM is based on the jQuery `fullcalendar` plugin. For complete details on that API, please refer to the fullcalendar web site.

Pages for the Configuration of the calendarSection

The *Calendar* section can be configured on these pages:

- **Ticket page**
  Use *calendar/ticketEditPage/calendarSection*.
- **Customer pages**
  - **Contact page**
    Use *calendar type/contactEditPage/calendarSection*.
  - **Company page**
    Use *calendar/companyEditPage/calendarSection*.

calendarSection

The following attributes can be used to configure the appearance and behavior of the integrated Exchange calendar.

Attributes:

- **allDaySlot**
  Boolean. Determines if the "all-day" slot is displayed at the top of the calendar. Default “true”.

- **appointmentBackgroundColor**
  java.lang.String. Sets the background color for all appointments in the calendar. You can use any CSS color format, such as “#f00”, “#ff0000”, “rgb(255,0,0)”, or “red”.
- **appointmentBorderColor**
  Sets the border color for all appointments in the calendar. You can use any CSS color format, such as “#f00”, “#ff0000”, “rgb(255,0,0)”, or “red”. (java.lang.String)
- **appointmentColor**
  java.lang.String. Sets the background and border colors for all appointments in the calendar. You can use any CSS color format, such as “#f00”, “#ff0000”, “rgb(255,0,0)”, or “red”.
- **appointmentConstraint**
  java.lang.String. Limits appointment dragging and resizing to certain windows of time.

  Possible values:

  - **<appointment_ID>**
    Appointments that are being dragged or resized must be fully contained by at least one of the appointments linked to by the given appointment ID.
  - **businessHours**
    Appointments being dragged or resized must be fully contained within the week's business hours (default: Monday-Friday 9am-5pm), see *businessHours* attribute for details.
  - **<start-time>-<end_time>;<days_of_week>**
    A custom time window in the same format as the *businessHours* attribute. Days of week are optional.
    Examples: “10:00-18:00; 1,2,3,4” or “10:00-18:00”
- **appointmentDurationEditable**
  Boolean. Allows appointments' durations to be editable through resizing. Default “true”.
- **appointmentOverlap**
  Boolean. Determines if appointments in the calendar, when dragged and resized, are allowed to overlap each other. Default “true”.
- **appointmentStartEditable**
  Boolean. Allows appointments' start times to be editable through dragging. Default “true”.
- **appointmentTextColor**
  java.lang.String. Sets the text color for all appointments in the calendar. You can use any CSS color format, such as “#f00”, “#ff0000”, “rgb(255,0,0)”, or “red”.
- **aspectRatio**
  java.lang.String. Determines the width-to-height aspect ratio of the calendar. Default value “2.8”. If empty calendar component will use internal default value 1.35.
- **businessHours**
  java.lang.String. Emphasizes certain time slots in the calendar.
  Format: <start-time>-<end_time>;<days_of_week>
  Example: 10:00-18:00; 1,2,3,4 (from 10am to 6pm, Monday-Thursday)

- **calendarEventHandlerScript**
  java.lang.String. Name of the script which handles calendar events. Besides standard context variables like `ticket`, there are additional ones:
  - **eventType**
    enum (values: CREATE, UPDATE, DELETE)
  - **appointment**
    with appointment data (uid, subject, location, etc.).

  See documentation for details. (java.lang.String)
- **calendarInitializationScript**
  java.lang.String. Name of the script which produces the calendar configuration. If value is empty or the script returns “null” the calendar won't be shown.
- **contentHeight**
  java.lang.String. Makes the calendar's content area this many pixels tall. By default, this option is not set and the calendar's height is calculated by *aspectRatio*.
- **defaultAllDayAppointmentDuration**
  java.lang.String. A fallback duration for all-day appointments without a specified *end time* value. Default value “1” (one day).
- **defaultDate**
  java.lang.String. The initial date displayed when the calendar first loads. Accepts an ISO8601 date string like “2014-02-01”.
- **defaultTimedAppointmentDuration**
  java.lang.String. A fallback duration for timed appointments without a specified *end time* value. If not set default value will be “02:00:00” (2 hours). This attribute also affects default duration of appointments during creation.
- **DefaultView**
  java.lang.String. Default calendar view.
  Possible values: month, basicWeek, basicDay, agendaWeek, agendaDay. Default *agendaWeek*.
  View examples at Available Views.
- **editable**
  Boolean. Whether the appointments can be created, dragged and resized. This value overwrites the source configuration. Default “false”.
- **firstDay**
  java.lang.String. The day that each week begins with. (Sunday=0, Monday=1, Tuesday=2, etc.). If empty, value will be based on browser's locale.
- **forceAppointmentDuration**
  Boolean. A flag to force calculation of an appointment's end if it is unspecified. Default “false”.
- **handleWindowResize**
  Boolean. Whether to resize the calendar automatically when the browser window resizes. Default “true”.

- **headerCenter**
  java.lang.String. Defines the buttons and title at the top/center of the calendar. See *headerLeft* description for details. Default “title”.
- **headerLeft**
  java.lang.String. Defines the buttons and title at the top/left of the calendar. Comma- or space-separated list values (values separated by a comma will be displayed adjacently). Default “prev,next today”.

  Possible values:

  - **title**
    Text containing the current month/week/day.
  - **prev**
    Button for moving the calendar back one month/week/day.
  - **next**
    Button for moving the calendar forward one month/week/day.
  - **prevYear**
    Button for moving the calendar back on year.
  - **nextYear**
    Button for moving the calendar forward one year.
  - **today**
    Button for moving the calendar to the current month/week/day.
  - **<view name>**
    Button that will switch the calendar to any of the available views (see *defaultView* description for available views).

  Header will disappear if all three header* attributes (Center, Left, Right) are empty.
- **headerRight**
  java.lang.String. Defines the buttons and title at the top/right of the calendar. See *headerLeft* attribute description for details. Default “month,agendaWeek”.
- **height**
  java.lang.String. Sets the height, in pixels, of the entire calendar (including header). By default, this option is not set and the calendar's height is calculated by *aspectRatio*.
- **hiddenDays**
  java.lang.String. Excludes certain days of the week from being displayed. Comma separated list of day-of-week indices (Example: “1,3,5”). Each index is zero-based (Sunday=0) and ranges from 0-6.
- **lazyFetching**
  Boolean. Determines when appointment fetching should occur. See detailed documentation. Setting this attribute to “false” makes sense when there are a lot of external changes to the user's calendar. Default “true”.

- **maxTime**
  java.lang.String. Determines the end time (exclusively) which will be displayed, even if the scrollbars have been scrolled all the way down. Default value is “24:00:00”.
- **minTime**
  java.lang.String. Determines the starting time that will be displayed, even if the scrollbars have been scrolled all the way up. Default value is “00:00:00”.
- **nextDayThreshold**
  java.lang.String. When an appointment's end time spans into another day, the minimum time it must be in order for it to render as if it were on that day. Default: “09:00:00” (9am). Only affects timed appointments that appear on whole days. Whole day cells appear in *month view*, *basicDay*, *basicWeek* and the *all-day slots* in the agenda views.
- **rightToLeftMode**
  Boolean. If enabled, displays the calendar in right-to-left mode. Default “false”.
- **scrollTime**
  java.lang.String. Determines how far down the scroll pane is initially scrolled. Default is “06:00:00” (6am).
- **slotAppointmentOverlap**
  Boolean. Determines whether timed appointments in agenda view should visually overlap. Default “true”.
- **slotDuration**
  java.lang.String. The frequency for displaying time slots. Default is “00:30:00” (30 minutes).
- **snapDuration**
  java.lang.String. The time interval at which a dragged appointment will snap to the agenda view time grid. Also affects the granularity at which selections can be made. Default value is set to *slotDuration*.
- **state**
  java.lang.String. The visibility mode of the section, possible values are [*expanded, collapsed, collapsed_and_preload, hidden*]. Default: “expanded”.
- **weekNumbers**
  Boolean. Determines if week numbers should be displayed in the calendar. Default “true”.
- **weekends**
  Boolean. Whether to include Saturday/Sunday (i.e., weekend) columns in any of the calendar views. Default “true”.

Defining Event Handlers for Appointment Events

Using the Page Customization attribute *calendarEventHandlerScript*, you can define actions which should be triggered in case a certain event has occurred. The events which can be used are:

- a new appointment has been created (event type CREATE)
- an existing appointment has been edited (event type UPDATE)
- an existing appointment has been removed (event type DELETE)

For example, you can define that when a new appointment is made, an email is automatically sent to all contacts of the ticket.

The following variables are available in a calendarEventHandlerScript:

- **ticket** (only in ticket context)
- **unit** (only in contact/company context)
- **eventType** - type of event, possible values are: CREATE, UPDATE or DELETE. It is an enum of type com.consol.cmweb.server.common.model.calendar.AppointmentEventType.
- **appointment** - appointment object (class `com.consol.cmweb.server.common.model.calendar.AppointmentVo`). Properties (Some properties may not be available. It depends on the type of calendar server and the respective version):

  Basic:

  - **subject**
    String. Subject/title of the appointment.
  - **startDate**
    Date. Start date/time of the appointment.
  - **endDate**
    Date. End date/time of the appointment.
  - **allDayEvent**
    Boolean. Defines whether an appointment is an all day event: which means that is lasts all day (or many days).
  - **location**
    String. Location/place of the appointment.
  - **meeting**
    Boolean. Is “true” when an appointment is a meeting. (MS Exchange Server specific property: means that attendees were invited appointment became a meeting)
  - **cancelled**
    Boolean. Indicates if an appointment is marked as *cancelled*.
  - **recurring**
    Boolean. Whether appointment is a part of recurring set.
  - **busyStatus**
    AppointmentVo.BusyStatus. Possible values: FREE, TENTATIVE, BUSY, OUT_OF_OFFICE, WORKING_ELSEWHERE or NONE.
  - **body**
    String. Body, content of the appointment. It can be text or HTML (depends on *bodyType* property)
  - **bodyType**
    AppointmentVo.BodyType. Possible values TEXT, HTML or NONE

Advanced:

- **uid**
  String. Unique identifier of an appointment within calendar server.
- **start**
  org.joda.time.DateTime. Start date/time (joda)
- **end**
  org.joda.time.DateTime. End date/time (joda)
- **timeZone**
  org.joda.time.DateTimeZone. Timezone of the appointment (used to correctly show all-day appointments because days starting at different time in each timezone.

The following script shows an example of a `calendarEventHandlerScript`.

```
import static
 com.consol.cmweb.server.common.model.calendar.AppointmentEventType.*;
import com.consol.cmas.common.model.customfield.meta.UnitDefinitionType;
// Check if you are on ticket page or on customer page:
def inTicket = false
def inUnit = false
if ( binding.variables.containsKey("ticket") ) {
   log.info "Context: Ticket" inTicket = true
}
if ( binding.variables.containsKey("unit") ) {
   log.info "Context: Unit" inUnit = true
}
def recip def mailField
// if you are on ticket page: write e-mail to engineer
if (inTicket){
   recip = ticket.engineer?.email
// if you are in unit context, CONTACT, write to contact
} else if (inUnit) {
def unitDefName = unit.definitionName
def unitDefType = unit.definition.type

log.info ' Definition is now ' + unitDefName
if (unitDefType == UnitDefinitionType.CONTACT){
   switch (unitDefName) {
      case 'DirCustCustomer': mailField = "dir_cust_email";
         break
      case 'customer': mailField= "email"
         break
      case 'PartnersContact': mailField = "email"
         break
      case 'ResellerCustomer': mailField = "email"
         break
   }
} else if (unitDeftype == COMPANY) {
   log.info "No email for Company!"
}
```

```
recip = unit.get(mailField) }
log.info 'mailField is now ' + mailField
log.info 'recip is now ' + recip


// EXAMPLE! log.info only :
log.info "Appointment '${appointment.subject}' has been "
if (eventType == CREATE) {
   log.info "created"
} else if (eventType == UPDATE) {
   log.info "modified"
} else if (eventType == DELETE) {
   log.info "removed"
}


// send mail here ...
```

Code example 60: *Admin Tool script, example of a calendarEventHandlerScript*

E.6 Classes of Text

This chapter discusses the following:

E.6.1 Introduction

A **class of text** is a classification that you assign to a ticket entry. This entry can be:

- a comment
- an email that was sent from the ticket
- an email that was received in the ticket
- an attachment

Assigning a class of text can serve one or more of the following purposes:

- Highlighting the text in the ticket with a special color to make it easier to find (e.g., an important note, as shown in the following figure). An icon can also be used for each class of text.
- Marking a ticket entry to make it visible in CM/Track, i.e., to make it available for customers who log in to the ConSol CM customer portal.
- Marking the entry to control the process flow, e.g., a ticket can only be finished when exactly one entry has been marked as *solution*.
- Marking the entry for hand-off to another process, e.g., the entries marked *question* and *answer* are automatically used for an FAQ ticket.

Thus, with classes of text you can organize ticket information within the ticket and can also control the process flow and the availability of information.

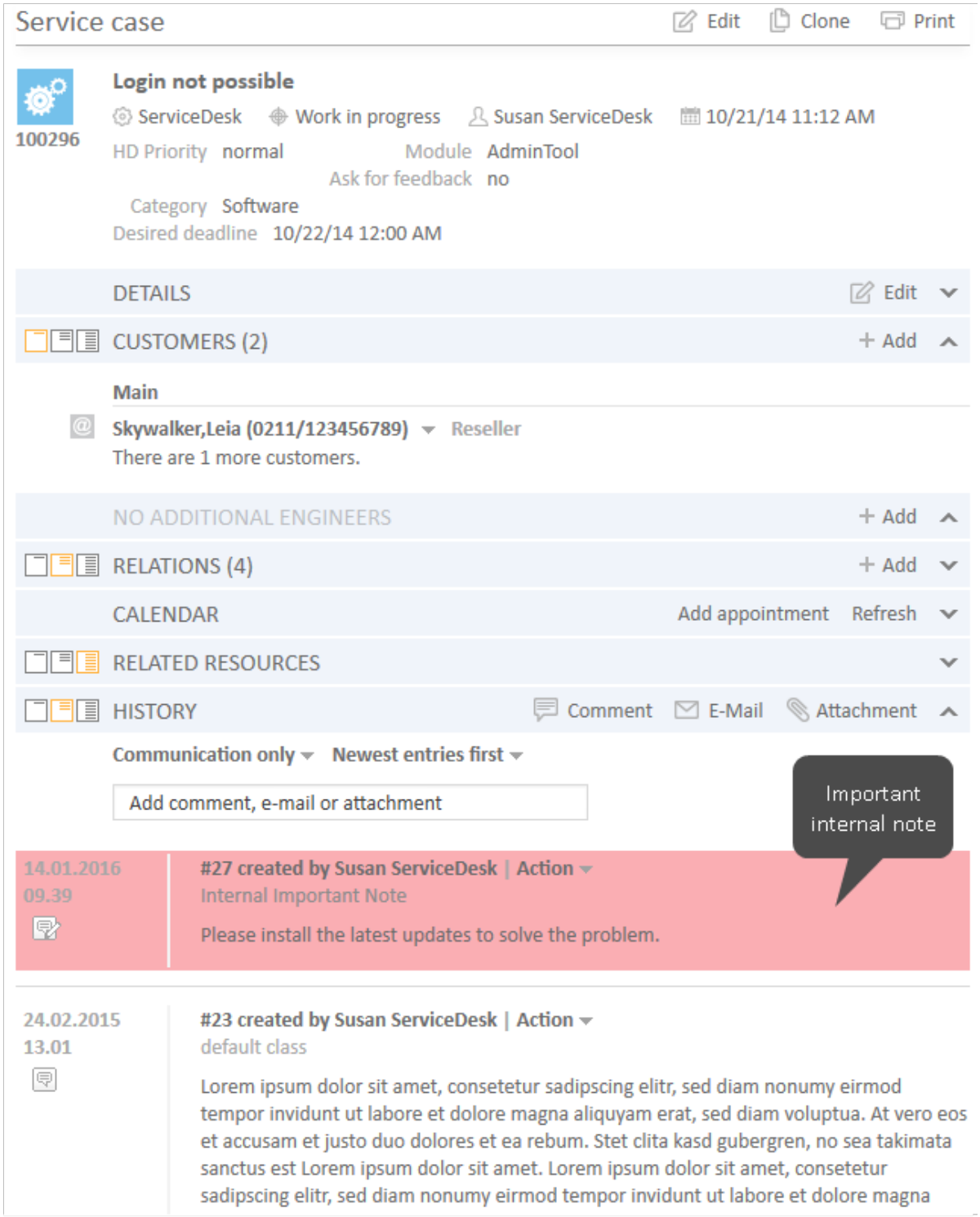


Figure 369: *ConSol CM Web Client - Using a class of text for an internal important note*

E.6.2 Managing Classes of Text Using the Admin Tool

E.6.2.1 Installing a New Class of Text

Two steps are required to install a new class of text for tickets in a certain queue:

1. Defining the class of text in the navigation item *Classes of text*.
2. Assigning the class of text to the queue where it should be available for tickets (see section Queue Administration for more information).

Defining a Class of Text

Classes of text are defined and managed in the corresponding navigation item in the navigation group *Global Configuration* in the Admin Tool (see the following figure).

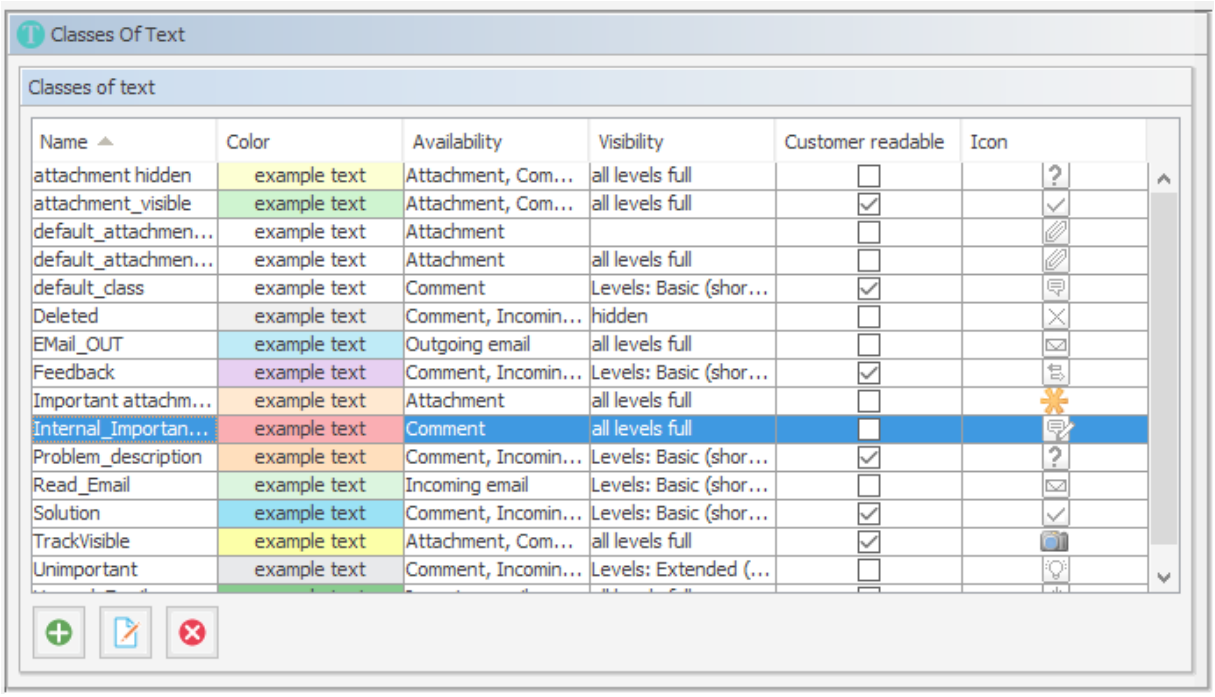


Figure 370: *ConSol CM Admin Tool - Global Configuration, Classes of Text*

You define a new class of text by clicking the *Add* button below the list. The following pop-up window appears:

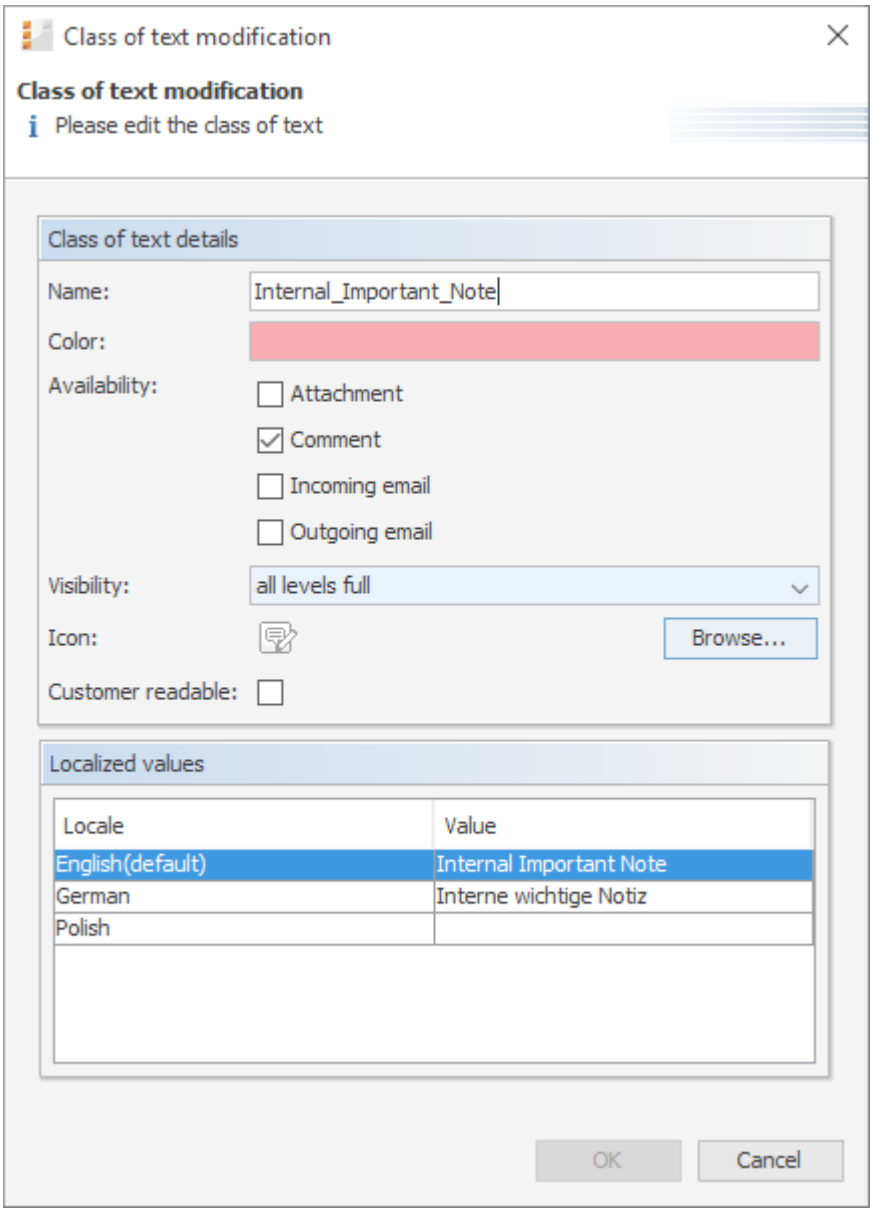


Figure 371: *ConSol CM Admin Tool - Global Configuration, Classes of Text: New Class of Text*

Here, you have to define the details of the class of text:

- **Name**
  Enter a name for the new class of text. The name must be unique.
- **Color**
  When you click into the *Color* field a pop-up window appears. It contains a range of colors from which you can choose the desired color for the class of text. You can see the selected color in the *Preview* area. Click *OK* to save your choice. Click *Reset* to return to the last saved color.

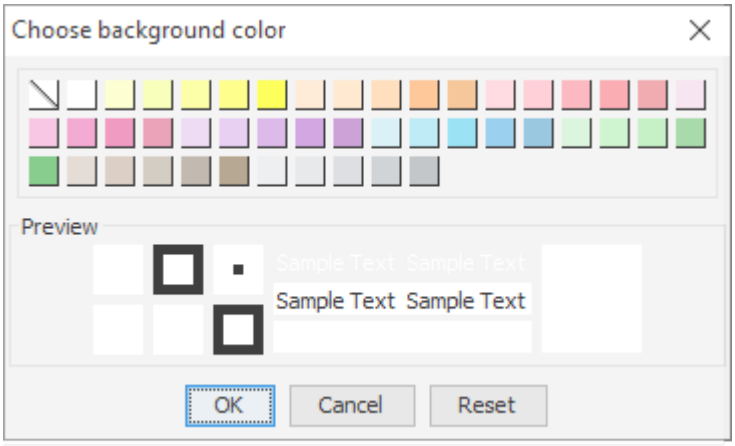


Figure 372: *ConSol CM Admin Tool - Global Configuration, Classes of Text: Choose a color*

- **Availability**
  You can choose here for which ticket information the class of text shall be available. Mark one or several of the following options:
  - Attachment
  - Comment
  - Incoming mail
  - Outgoing mail
- **Visibility**
  Sets the visibility level for comments and emails, is not applied to attachments. See also General Information about the Visibility of Ticket History Entries in the Web Client

  There are three ticket history levels in the Web Client:

  - Basic (1st level)
  - Extended (2nd level)
  - Detail (3rd level)

  Thus, the configuration of a class of text concerning one or more of those levels determines whether a text entry marked with this class of text is displayed at a given level.

The addition *short* and *full* indicates whether entries are shown **within** the chosen visibility level in full length (*full*) or are truncated after a certain number of characters (*short*):

- **short** - shortened
- **full** - full length

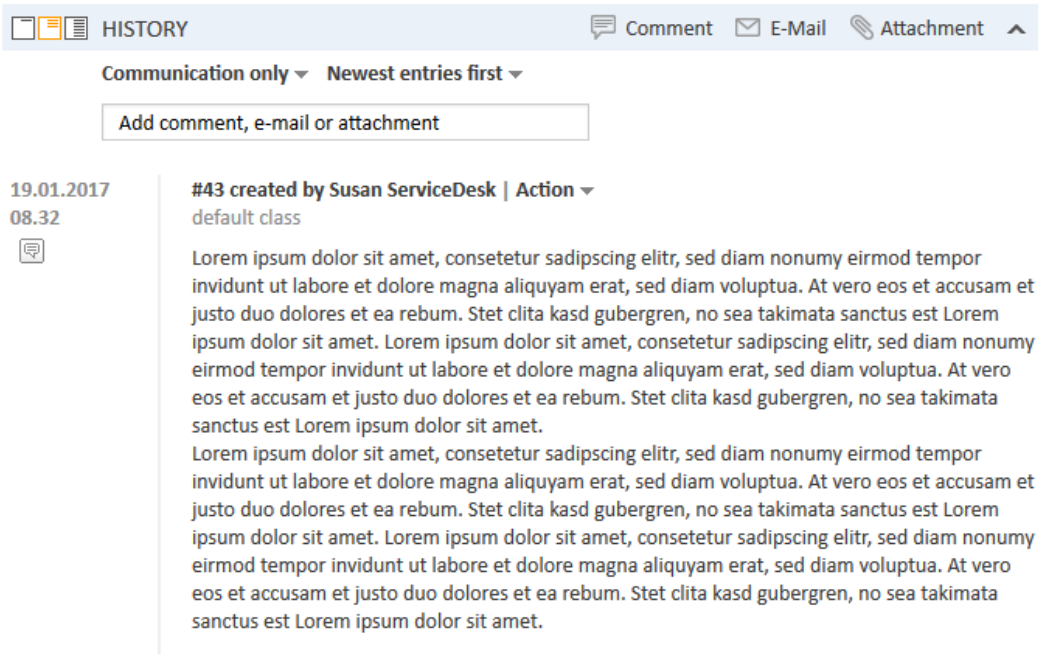


Figure 373: *ConSol CM Web Client - Example: Visibility level "Extended" (2nd level) shows entries in full length ("full")*

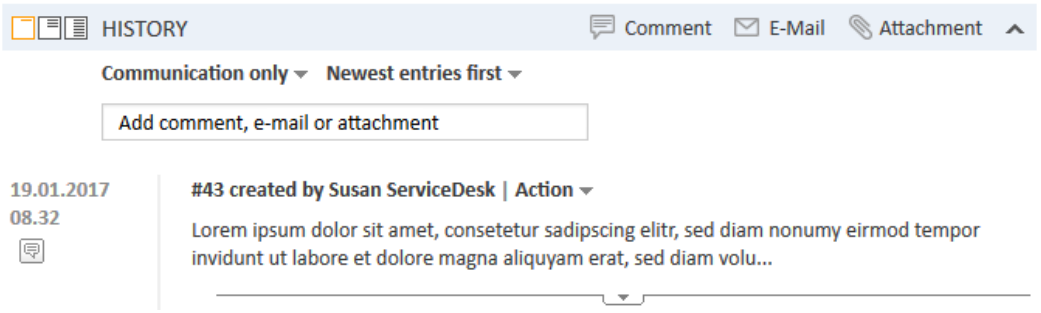


Figure 374: *ConSol CM Web Client - Example: Visibility level "Basic" (1st level) shows entries shortened ("short")*

Select, in the drop-down menu, the history levels at which the marked ticket information should be visible (see picture below).

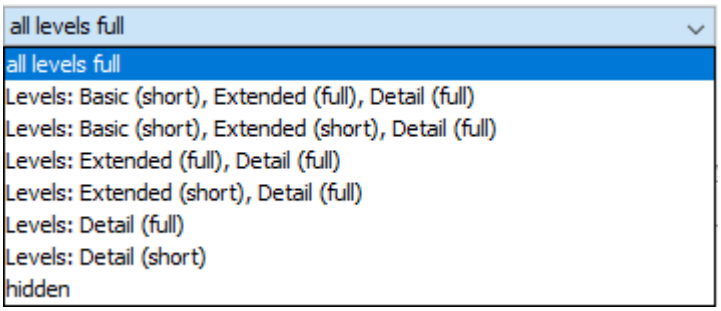


Figure 375: *ConSol CM Admin Tool - Global Configuration, Classes of Text: Choose a visibility level*

If you choose *hidden*, the ticket History entry marked by this class of text will not be visible in the ticket history. For a detailed explanation of visibility level and display mode for ticket History entries, please refer to section General Information about the Visibility of Ticket History Entries in the Web Client.

- **Icon**
  When you click the box next to *Icon* you will get a selection of standard ConSol CM icons. Select one of these icons for the new class of text or load your own individual icon by clicking the *Browse...* button. An icon for a class of text must have the size 16 (height) * 24 (width) px to be correctly placed and aligned in the drop-down menu in the Web Client.

Figure 376: *ConSol CM Admin Tool - Global Configuration, Classes of Text: Choose an icon*

- **Customer readable**
  Select this checkbox if ticket information marked with this class of text shall be visible for customers in CM/Track, the ConSol CM customer portal.
- **Localized values**
  You can localize the name of a class of text. Enter the corresponding class name in the *Value* field for each additional language. A detailed explanation of the localization mechanism is given in section Localization of Objects in General, Type 2.

Click *OK* to save the details of the new class of text and to close the window.

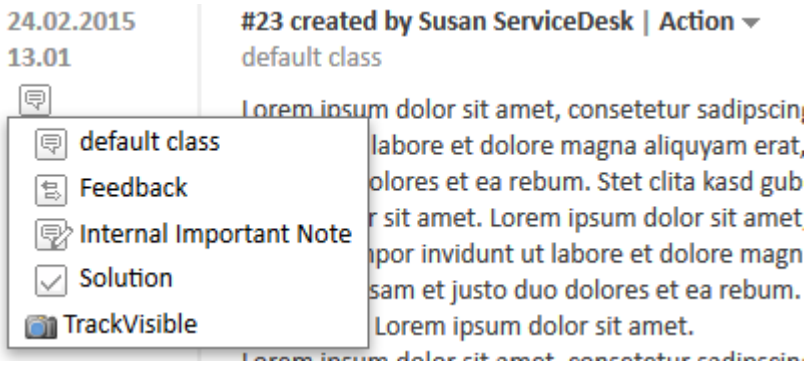


Figure 377: *ConSol CM Web Client - Context menu for classes of text in ticket history section*

Assigning the Class of Text to a Queue

After assigning the class of text to a queue within the Queue Administration it will be available for tickets of this queue in the Web Client.

E.6.2.2 Editing a Class of Text

If you want to edit a class of text, select it in the list and click the *Edit* button. The same window as described above for creating a class of text will appear. You can modify all details and save your changes by clicking *OK*.

E.6.2.3 Deleting a Class of Text

You can only delete a class of text if it is not used within any tickets and if it is not assigned to a queue. In order to delete a class of text select it in the list and click the *Delete* button. If you confirm the following dialog with *Yes*, the class of text will be removed from the list and the system.

E.6.2.4 Setting the Default Class of Text

To define the default class of text, use the system property cmweb-server-adapter, defaultContentEntryClassName (see System Properties). The default class of text will be applied to any ticket entry which is not explicitly marked with another class of text.

Depending on the visibility configuration of this class of text, the regular comments in the ticket (i.e., the comments which have the class of text *default class*) will be displayed, see section visibility.

E.6.2.5 Working With Classes of Text in Scripts

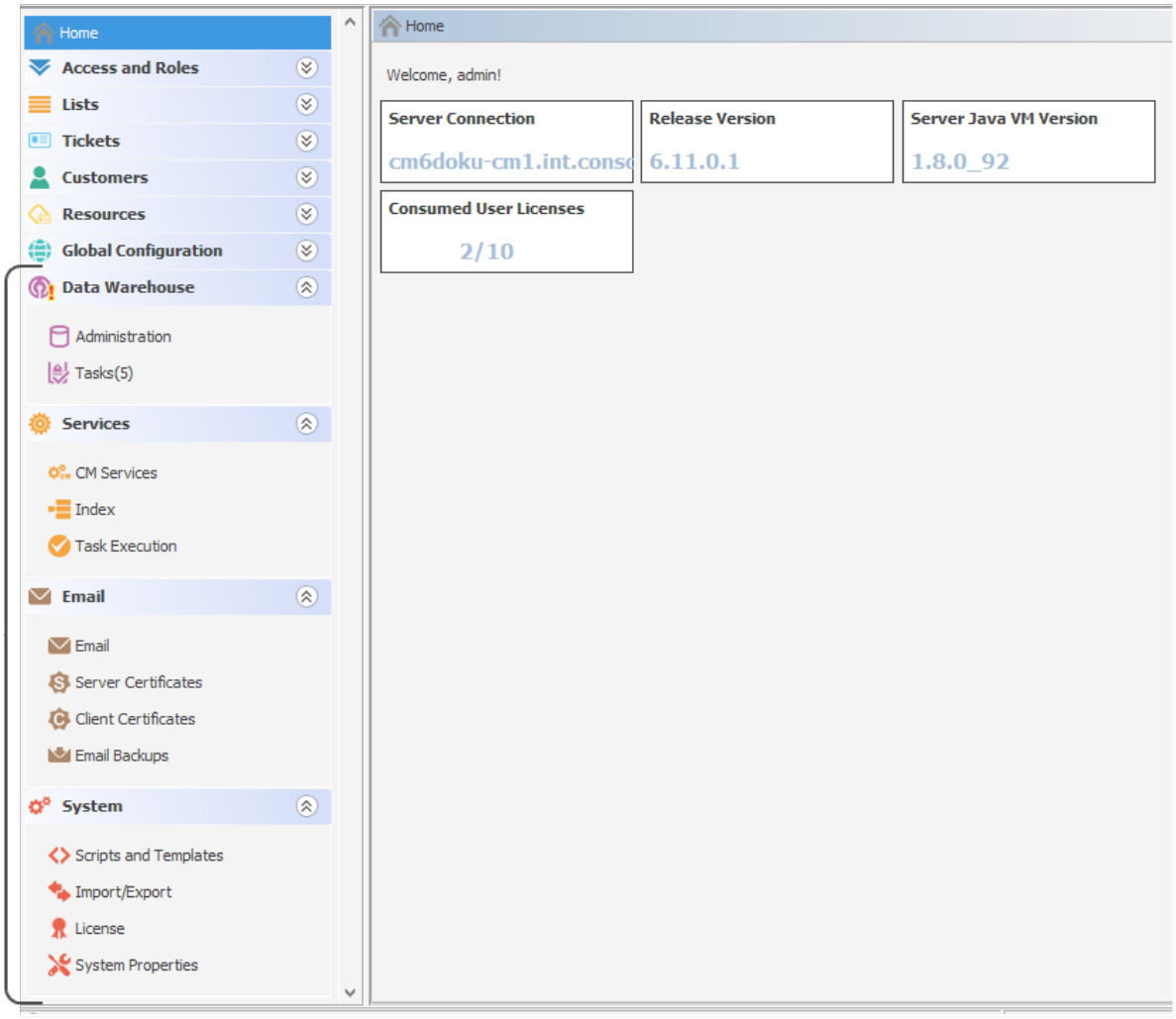
You can work with classes of text in scripts which are used in workflow activities or which are located in the *Script* section of the Admin Tool. For details about programming, please read the *ConSol CM Process Designer Manual*.

In scripts, you can assign a class of text to a TextEntry even if the class of text is not assigned to the queue! This can help automate certain processes.

E.6.2.6 Transferring Classes of Text to the DWH

All classes of text (default classes and special ones) used in history entries (comments, emails) are transferred from ConSol CM to the data warehouse (DWH) automatically in all transfer modes (LIVE and ADMIN). There is no configuration required for this feature, and the installation is upgraded automatically. A new initialization and initial transfer are required, however.

F - Expert Section



In this section, you will learn how to configure some advanced settings in ConSol CM:

The following topics are covered:

- Ticket Administration: You will learn how to delete or re-open tickets.
- Data Warehouse (DWH) Management: You will learn how to configure CM to initialize and work with the Data Warehouse, the basis for ConSol CM Reporting.
- CM Services: You will learn which services work within the ConSol CM application.
- Search in ConSol CM: You will learn all about searching in ConSol CM, e.g., how to manage the indexer or how to implement search actions for the Action Framework.
- The Task Execution Framework (TEF): You will learn what the TEF is and how you can configure it in your ConSol CM system.

- Email Configuration: You will learn all about the email functionality in ConSol CM and how to configure the respective modules.
- Script and Admin Tool Template Administration: You will learn which script and template types are managed with the Admin Tool and how to code these scripts and templates yourself.
- Deployment (Import/Export): You will learn all about ConSol CM scenarios which are used for ConSol CM system export and import.
- License Management: You will learn how to install a new license in your ConSol CM system.
- System Properties: You will learn what system properties are and what you can do with them to configure basic settings of ConSol CM.
- Working with Text Templates: You will learn all about text and document templates which can be used as integral part of your ConSol CM system.
- Time Booking Using ConSol CM: You will learn what you have to do to prepare your ConSol CM system for time booking.
- Authentication Methods in ConSol CM: You will learn various configurations for the engineer authentication in the Web Client.
- System Architecture: You will get a first glimpse of the ConSol CM system architecture. More information on this topic is provided in the *ConSol CM Setup Manual* and the *ConSol CM Operations Manual*.

F.1 Ticket Administration

This chapter discusses the following:

F.1.1 Introduction to Ticket Administration

Ticket administration allows you to:

- **Delete tickets**
  e.g., if a ticket was created by mistake.
- **Reopen tickets**
  e.g., if a ticket has been closed too early.

⚠ Please keep in mind that a ticket which is reopened starts the process at the start node of the respective workflow. So when a ticket has passed through nodes where events are triggered that should be performed only once (e.g., the ticket is passed to an approver) it might be better to open a new ticket. An alternative is to modify the workflow to contain a shortcut which allows such tickets to bypass steps to avoid that these steps are run multiple times for that ticket.

F.1.2 Ticket Administration Using the Admin Tool

In the Admin Tool, you can manage tickets using the navigation item *Administration* in the navigation group *Tickets*.

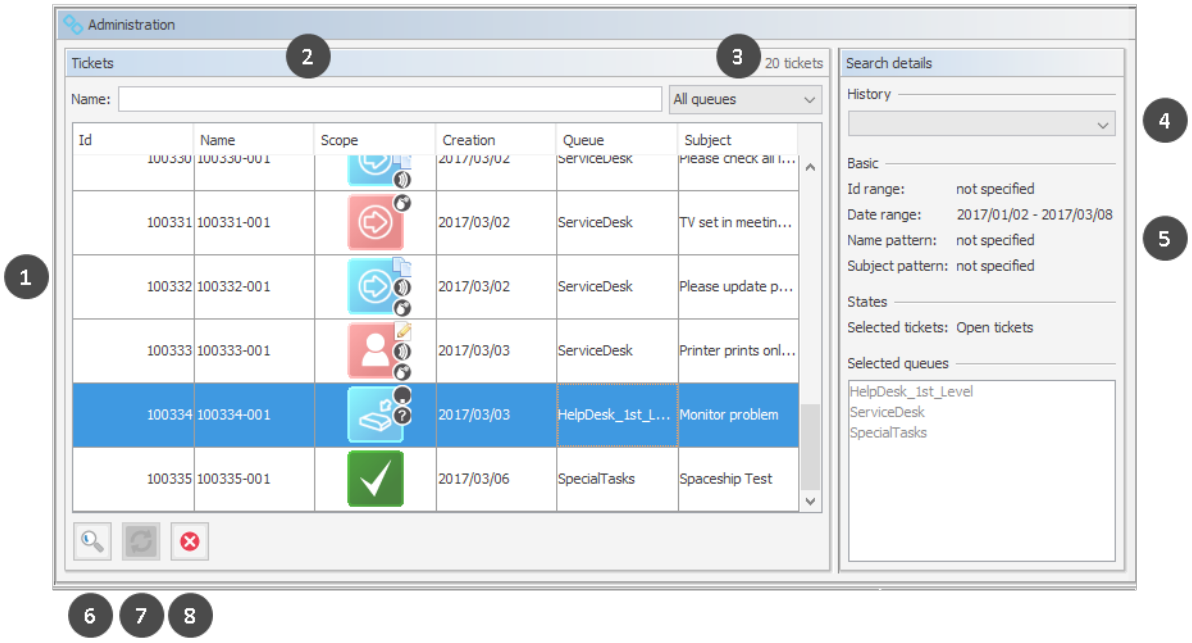


Figure 378: *ConSol CM Admin Tool - Tickets, Administration: List of tickets after search*

The ticket administration screen consists of the following elements:

- List of all tickets which match the search criteria (1)
- Filter ticket by name (2)
- Filter tickets by queue (3)
- List of all recently executed searched (4)
- Details of the current search (5)
- Button to define the search criteria to start a new search (6)
- Button to reopen the selected tickets (7)
- Button to delete the selected tickets (8) - only present if enabled in the CM system

F.1.2.1 Deleting or Reopening Tickets

For these operations you can either use the buttons below the list or you can use the context (right-click) menu.

- **Buttons:**
  Select the desired tickets in the list and click the *Delete* button to delete tickets or click the *Reopen* button to reopen tickets. If you confirm the following dialog with *Yes*, the corresponding action will be executed.
- **Context menu:**
  Select the desired tickets in the list and use the right mouse button to open the context menu. Select the desired operation.

F.1.2.2 Switching off the Delete Functionality Using a System Property

The delete functionality can be disabled system-wide using the system property cmas-app-admin-tool, delete.ticket.enabled. This is a boolean property. When it is set to “false” the *Delete* button is no longer displayed and the delete functionality is no longer available in the context menu.

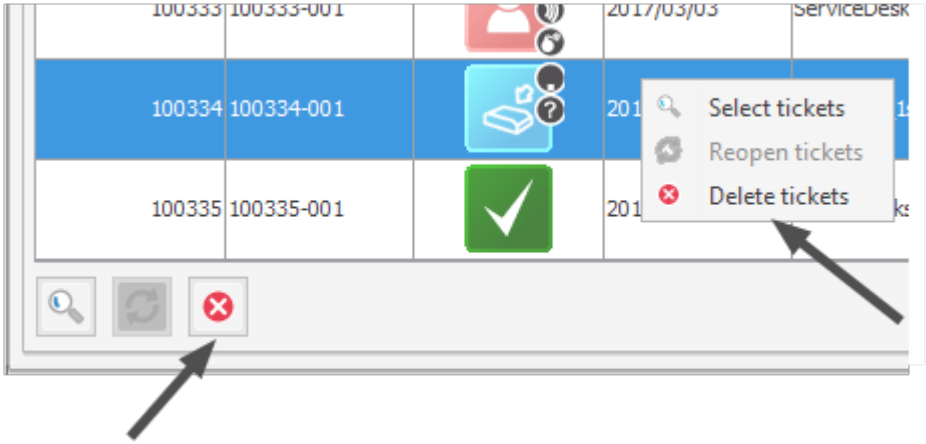


Figure 379: *ConSol CM Admin Tool - Tickets, Administration: cmas-app-admin-tool, delete.ticket.enabled = true*

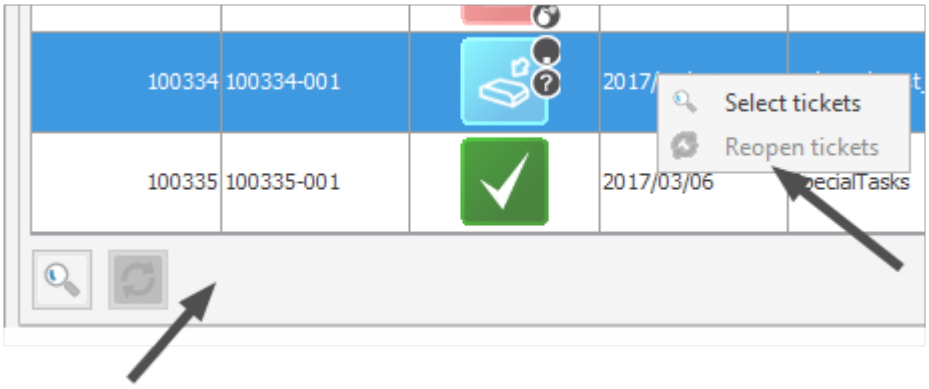


Figure 380: *ConSol CM Admin Tool - Tickets, Administration: cmas-app-admin-tool, delete.ticket.enabled = false*

F.1.2.3 Searching for Tickets

To search for tickets you want to delete or reopen click the *Search* button in the bottom left corner of the page or use the context menu. A pop-up window appears where you can enter the search criteria.

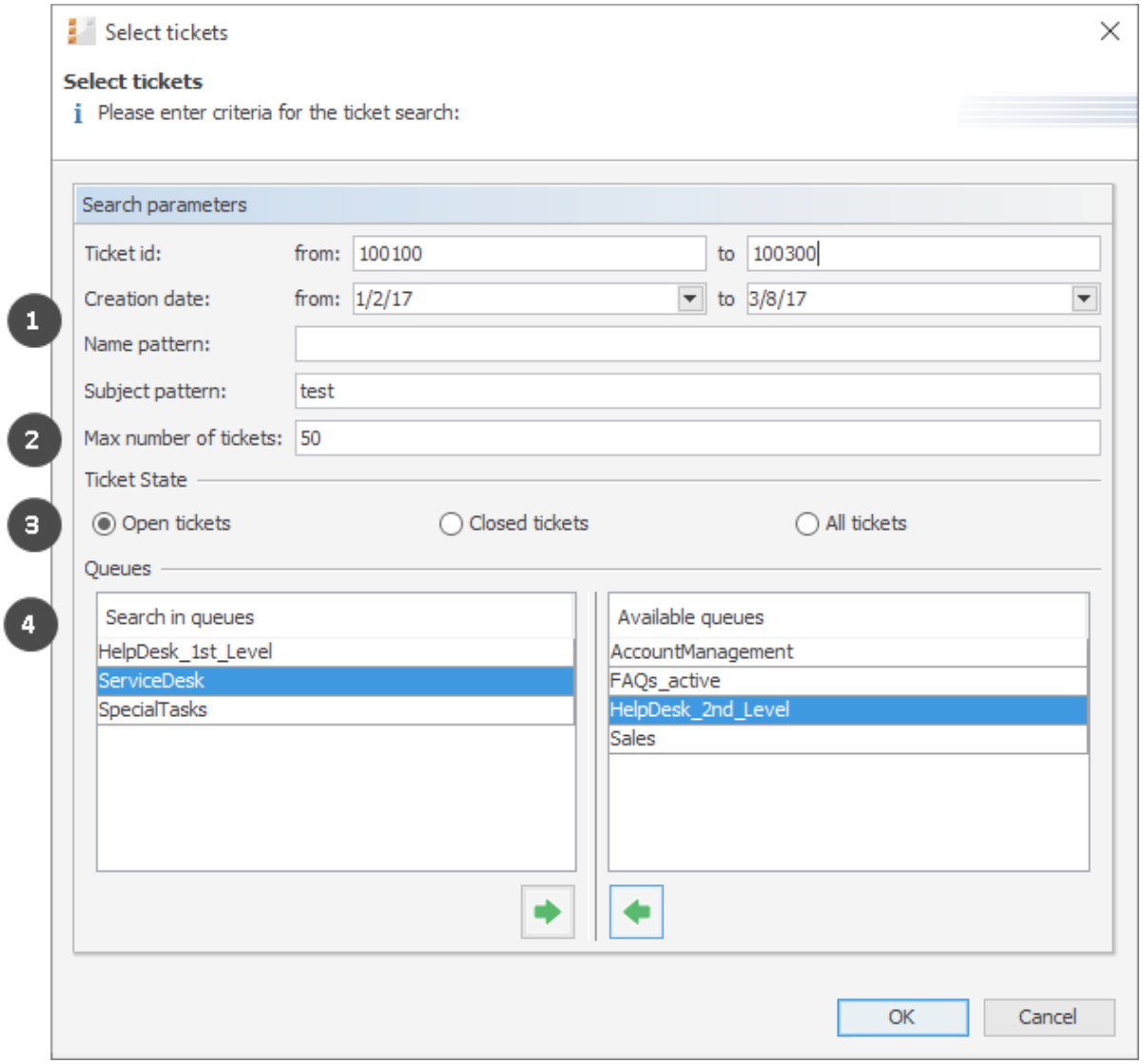


Figure 381: *ConSol CM Admin Tool - Tickets, Administration: Ticket search*

The following parameters can be used for searching:

- **Search criteria (1)**
  - **Ticket id**
    You can enter an ID range for the tickets here.
  - **Creation date**
    Via calendars you can restrict the search to tickets opened within a given period of time.
  - **Name pattern**
    Here you can enter keywords or search patterns for the ticket name.
  - **Subject pattern**
    In this field you can enter keywords or search patterns for the ticket subject.

- **Max number of tickets (2)**
  Here you can specify the maximum number of tickets to be displayed in the list.
- **Ticket State (3)**
  Using the radio buttons you can determine if you want to search for *open*, *closed*, or *all* tickets.
- **Queues (4)**
  The list on the right shows the available queues. Select the queues to search in and click the *Assign* button to move them to the search list on the left. If you do not choose any queues the search will be extended to all available queues.

Please note that there is a difference between the ticket id and the ticket name! The ticket id is an internal ID in the CM database which is usually not displayed on the GUI. The ticket name is displayed below the ticket icon and is - in everyday life - often called the ticket id, even though this is not correct. The ticket name might contain a prefix (e.g., SUP), depending on the queue-specific configuration. The ID will never contain a prefix!

See the example in the figure above:

The ticket id is *100103*, the ticket name is *SUP-116*. To search the CM database for this ticket you could either use a criterion like *ticket id* is between *10.000* and *200.000*, or a criterion like *ticket name pattern* should be *SUP-*1**.

Click *OK* to start the search. The result will be displayed on the *Ticket Administration* page. If the list is too long, you can limit the display using the name and queue filters above the list.

In the area next to the ticket list on the right you can find an overview of the search criteria you have chosen. The list box *History* above this area contains your most recent searches. If you click on an entry in this list a pop-up window with the criteria of the selected search will open. You can modify the search here or run it as-is.

F.2 Data Warehouse (DWH) Management

This chapter discusses the following:

⚠ To set up a DWH, a running ConSol CM Reporting Framework (CMRF) instance is required. If your system does not include a CMRF yet, please consult your ConSol CM manager or contact ConSol Software.

F.2.1 Introduction

F.2.1.1 Data Warehouse

A data warehouse (commonly known as a ***DWH***) is a collection of data from one or more systems and/or databases that provides a basis for reporting and data analysis. Often, imported data is combined or rearranged (integrated) to make it more suitable for reporting and analysis purposes. A more detailed introduction to the data warehouse principle and to the ConSol CM data warehouse is provided in the *ConSol CM DWH Manual*.

F.2.1.2 ConSol CM Data Warehouse and ConSol CM Reporting Framework

A ConSol CM default installation comprises all modules that are required to build a CM data warehouse. One of the two core components is the ***ConSol CM Reporting Framework (CMRF)***.

This is a Java EE application which synchronizes the data between a ConSol CM database and a DWH database. The following picture shows an overview of the system architecture of a typical DWH/CMRF installation. We recommend that you use separate servers for the ConSol CM and CMRF instances. Please refer to the current *System Requirements* for information about the supported application servers and RDBMS.

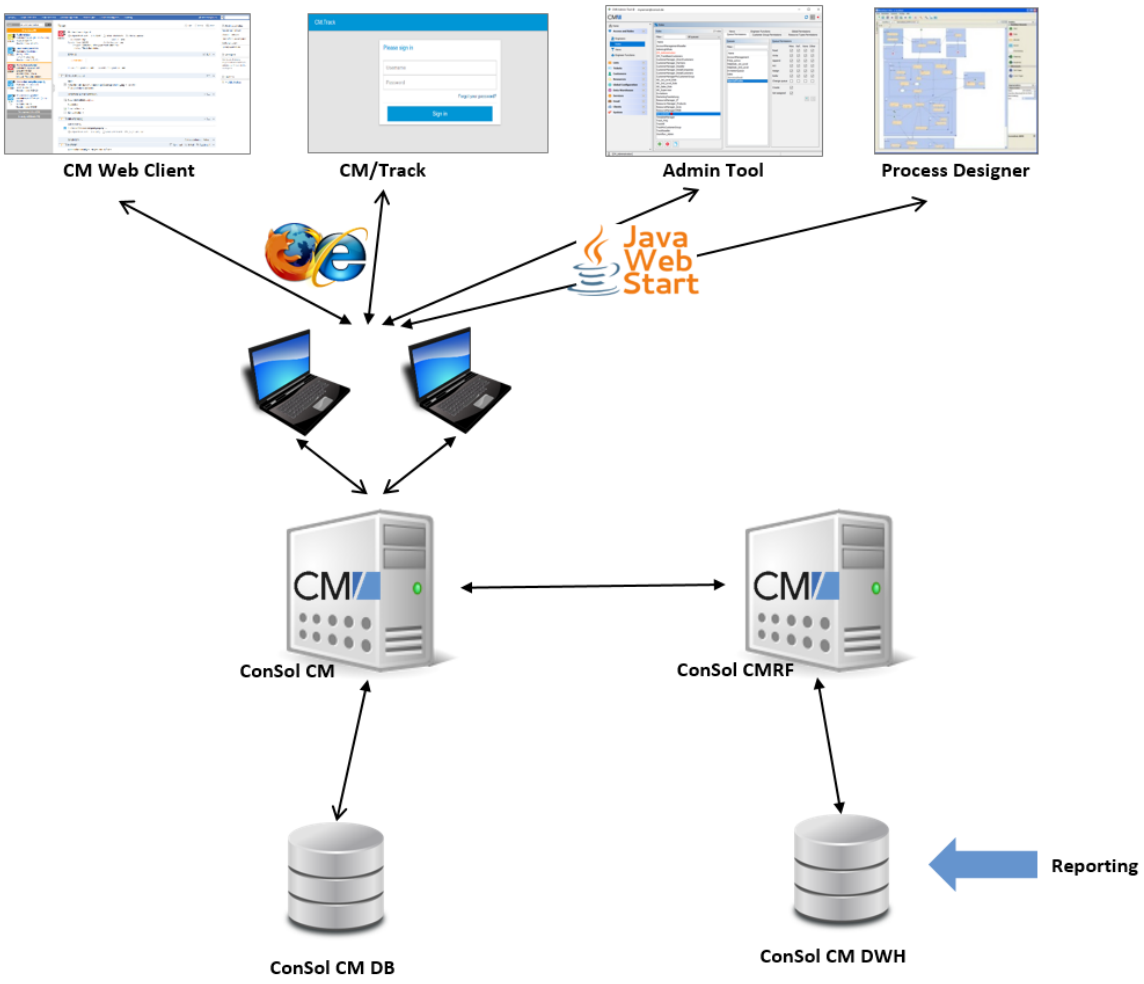


Figure 382: *ConSol CM - System architecture with DWH and CMRF (2 servers)*

There are two different synchronization modes for transferring data from ConSol CM to a DWH database:

- **LIVE mode**
  In this mode, every change that is submitted to the ConSol CM database is immediately synchronized with the DWH.
- **ADMIN mode**
  In this mode, the administrator has to trigger the synchronization manually.

⚠ Only data from ticket fields, customer fields, and resource fields with the annotation `reportable` = “true” will be synchronized with the DWH. In addition, all data which will be transferred as default will be transferred to the DWH. For a detailed explanation, please refer to the *ConSol CM DWH Manual*.

F.2.2 DWH Management Using the Admin Tool

To manage the DWH, use the navigation items in the navigation group *Data Warehouse*:

- Administration
- Tasks

F.2.2.1 Administration

For the administration of all DWH operations, open the navigation group *Data Warehouse*, navigation item *Administration*.

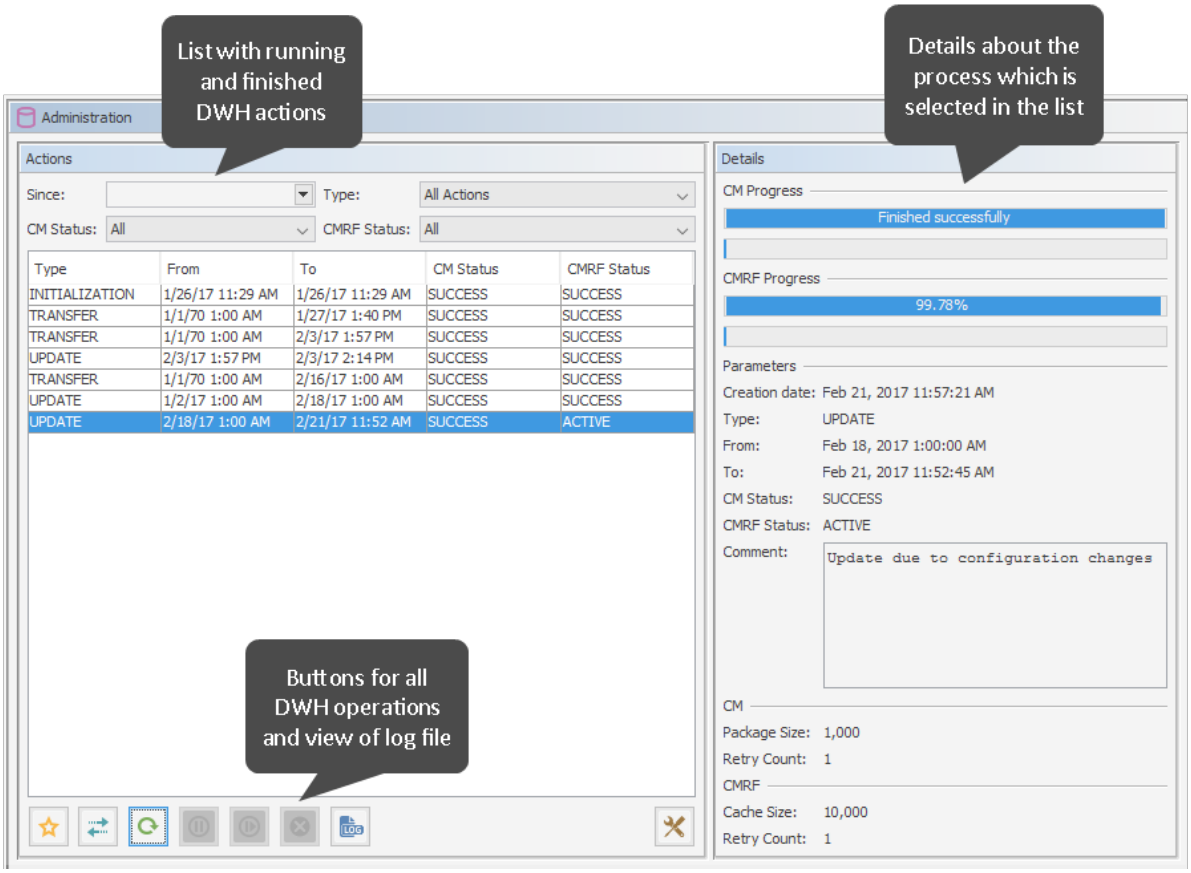


Figure 383: *ConSol CM Admin Tool - DWH, Tasks: Configuration and logs*

On the left hand side, the table *Actions* is displayed which lists all DWH operations which have been performed or which are still running.

You might want to filter the list by one or more of the filters above the list. The filter criteria will be combined by an AND.

- **Since**
  offers a date picker menu. Only the DWH actions which were started after the selected date will be displayed.

- **Type**
  only the DWH actions of the selected type will be displayed
  - **All actions**
    all available actions
  - **Already executed**
    Actions which are finished. The CM and CMRF status is SUCCESS or ERROR
  - **Currently executing**
    Actions which are currently running. The CM and/or CMRF status is ACTIVE.
  - **Planned to execute**
    Actions which have been configured and saved and entered into the list but which are not yet executed. The CM and/or CMRF status is NEW.
- **CM Status**
  only the DWH actions which have the selected CM status will be displayed.
  - All
  - NEW - first status after creation
  - ACTIVE - data is sent to CMRF
  - PAUSED - active state is stopped until the *Resume* button is clicked
  - SUCCESS - sending of data to CMRF has been finished successfully
  - ERROR - sending of data to CMRF has been finished unsuccessfully (see log files for information)
- **CMRF status**
  only the DWH actions which have the selected CMRF status will be displayed
  - ALL
  - NEW - first status after creation
  - ACTIVE - data is processed in CMRF
  - PAUSED - active state is stopped until the *Resume* button is clicked
  - SUCCESS - processing of data has been finished successfully
  - ERROR - processing of data has been finished unsuccessfully

A very elaborated explanation of the ConSol CM DWH transfer processes is provided in the *ConSol CM Operations Manual*, section *CMDB / CMRF/ Data Warehouse Synchronization Process*. In the following section in the current manual, only short information will be provided.

The list contains columns and values which are retrieved from the CM database table `cmas_dwh_synchronization`.

- **Type**
  - **INITIALIZATION**
    The DWH is new/empty. In the initialization step, the database structure is built.
  - **REINITIALIZATION**
    A new initialization of an already existing DWH
  - **TRANSFER**
    Initial data transfer after set-up. For a detailed explanation, please see section First DWH Synchronization (Transfer).
  - **UPDATE**
    Only new data is transferred to an already existing DWH. For a detailed explanation, please see section DWH Synchronization During System Operation (Update).
- **From**
  The start date of the DWH action
- **To**
  The end date of the DWH action
- **CM status**
  During a DWH action, data is transferred from the CM database to the DWH. This column indicates the status of the CM part of the action, i.e. the status of the transfer from CM to the DWH control tables. For an explanation of the status, please refer to the list above.
- **CMRF status**
  During a DWH action, data is transferred from the CM database to the DWH. This column indicates the status of the CMRF part of the action. For an explanation of the status, please refer to the list above.

Below the list, the following buttons provide all possible DWH actions:

- **Initialize**
  - case a) Create tables during DWH set-up. See Initialization of the DWH (Initialize).
  - case b) If the DWH already exists, you can here start a re-initialization. In order to do this, select the *Delete existing data* option. The database will the be rebuilt from scratch.
- **Transfer**
  Start initial data transfer after set-up. For a detailed explanation, please see section First DWH Synchronization (Transfer).
- **Update**
  Transfer new/additional data to the DWH. For a detailed explanation, please see section DWH Synchronization During System Operation (Update).
- **Pause**
  Pause a running DWH action
- **Resume**
  Continue a DWH action which has been paused

- **Delete**
  Deletes an unfinished data warehouse operation from the list, works for scheduled unstarted and previously paused operations. The most common case for this happens when queuing an initialize, transfer and update operation in a row without waiting.
- **Log**
  Open a panel which displays the `cmrf.log` file. This always provides the entries from the log file, the selection in the list does not have any influence.
- **Configuration**
  Opens the DWH configuration pop-up menu. See Basic DWH Configuration.

> ⚠ **IMPORTANT BACKGROUND INFORMATION ABOUT DWH OPERATIONS**
>
> Please be aware of the ConSol CM/CMRF behavior with regards to operations for the DWH!
>
> When you click one of the buttons *Initialize*, *Transfer* or *Update*, an operation with this type will be created as an entry in the ConSol CM database table `cmas_dwh_synchronization`. Each click on a button creates a new operation of the respective type. The operations are then executed one after another in the order of their creation, i.e., in FIFO (first-in-first-out) order! You can monitor all these actions in the *Administration* panel.

On the right hand side of the *Administration* panel, the *Details* section is displayed which lists all details of the action which is selected in the list.

- **CM Progress**
  Progress bars which indicate the progress of the transfer of CM data to the DWH control tables:
  - upper row: the overall progress for the operation on the sending side
  - lower row: the progress for the current sub-operation
- **CMRF progress**
  Progress bars which indicate the progress of CMRF updating the DWH database.
  - upper row: the overall progress for the operation on the receiving side
  - lower row: the progress for the current sub-operation
- **Parameters**
  All parameters of the selected DWH action:
  - **Creation date**
    Set automatically. Timestamp of the creation of the DWH action
  - **Type**
    DWH action type (INITIALIZATION | UPDATE etc.), see list above.
  - **From**
    The start date of the DWH transfer action. Might differ from the creation date when an action has been planned for a time in the future.
  - **To**
    The end date of the DWH action. Only set for finished actions.

- **CM status**
  see above
- **CMRF status**
  see above
- **Comment**
  The comment which has been set when the DWH action was defined. Might also be empty.
- **CM**
  - **Package size**
    Number of transferred objects per transaction from CM to the DWH control tables
  - **Retry count**
    Number of retries which are performed if an error occurs. After this number of retries has been reached, the operation is finished with the CM status ERROR.
- **CMRF**
  - **Package size**
    Number of transferred objects per transaction from the DWH control tables to the DWH tables
  - **Retry count**
    Number of retries which are performed if an error occurs. After this number of retries has been reached, the operation is finished with the CMRF status ERROR.

Basic DWH Configuration

Before you can set up a ConSol CM DWH you have to prepare a database (or database schema) which will contain the DWH data. The respective database server has to be available for the CMRF server. For a detailed description of those topics, please refer to the *ConSol CM Setup Manual*. Once the database (or database schema) for the DWH has been prepared and the CMRF is up and running, you can continue with the following steps.

In order to configure the DWH synchronization mode and the DWH-relevant notifications, open the navigation group *Data Warehouse*, navigation item *Administration*. Click on the *Configuration* button and enter all required values.

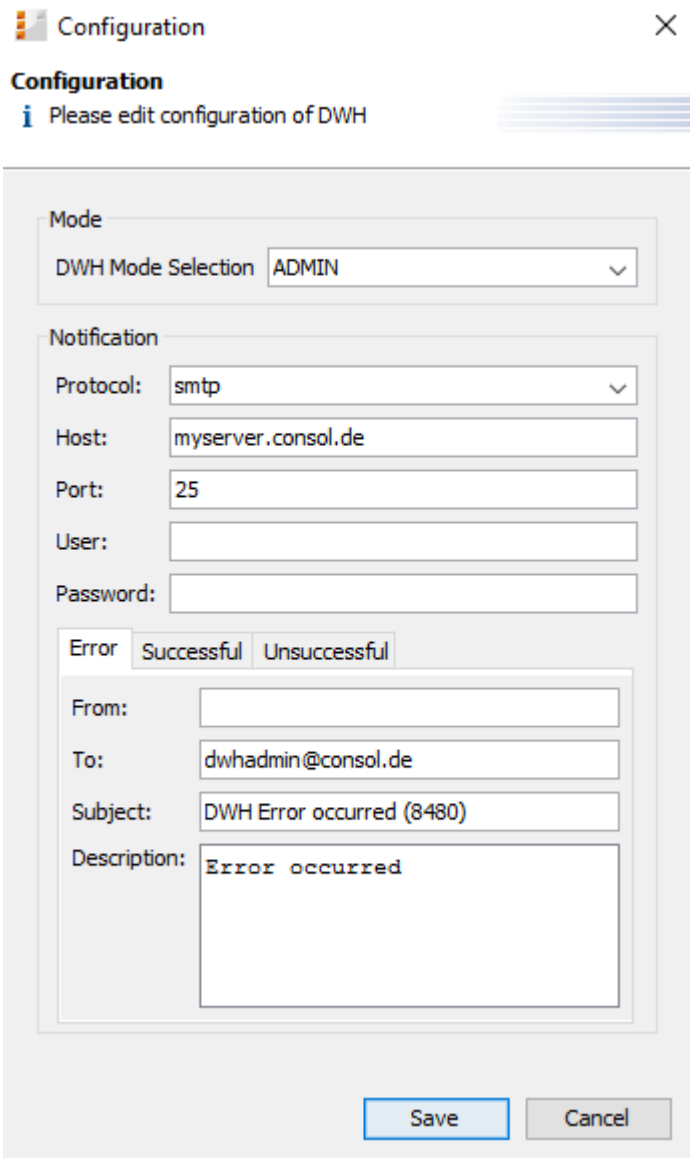


Figure 384: *ConSol CM Admin Tool - Data Warehouse, Administration: DWH configuration*

- **DWH Mode Selection**:
  - **LIVE**
    In this mode, every change that has been submitted to the ConSol CM database is immediately synchronized to the DWH.
  - **ADMIN**
    In this mode, the administrator has to trigger the synchronization manually.
  - **OFF**
    No data is transferred to the DWH.

You can also see the current DWH mode by looking at the corresponding DWH system property cmas-dwh-server, dwh.mode (see System Properties).

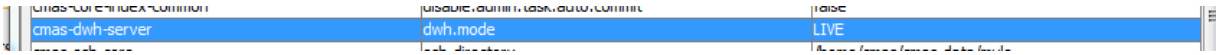


Figure 385: *ConSol CM Admin Tool - System property for DWH mode*

- **Notification**
  Here, the parameters for the emails which are sent due to DWH events, are configured:
  - **Protocol**
    Required - The protocol that is used to send the message. This is usually SMTP.
  - **Host**
    Required - The email server. You can enter a name (DNS-resolvable) or an IP address.
  - **Port**
    Required - The port on the email server where the mail daemon is listening.
  - **User**
    Optional - User name, if user authentication is required by the email server.
  - **Password**
    Optional - Password of the email user if user authentication is required by the email server.
  - Tabs **Error/Successful/Unsuccessful**
    Here the email parameters for emails that are sent by the system regarding DWH operations can be configured. There are three types of messages: in case of an error, in case of a successful operation, and in case of an unsuccessful operation.
    - **From**
      The From email address for messages (this may differ from the From address used for emails to customers and to engineers).
    - **To**
      The email address of the recipient of the DWH messages. Initially this will be the ConSol CM administrator's email address (CM system property cmas-core-security, admin.email).
    - **Subject**
      The (email) subject of the error/success/unsuccessful message.
    - **Description**
      The body (text) of the message.

Initialization of the DWH (Initialize)

When the basic configuration has been performed, the DWH initialization can be started. Click the *Initialize* button. A new DWH action of type INITIALIZATION will be entered in the list.

During this step, the database structure in the DWH is created with all tables and relations. No data will be transferred yet.

If the DWH has been in operation and has to be set-up a second time, a reinitialization has to be performed. Check the *Delete existing data* option in order to delete the old database structure and create a new one.

First DWH Synchronization (Transfer)

To fill the data warehouse with the ConSol CM data for the first time, click *Transfer*. The initial transfer is started. Depending on the number of tables, this might take some time (even several hours). You can follow the log entries by opening the `cmrf.log` file using the *Log* button.

For each Transfer operation, several parameters can be set, see section Parameters for Transfer and Update Operations.

> ⚠ The action *Transfer* should only be used directly after initialization, as this deletes all existing data from the standard tables and deletes all custom tables! For large databases this (transfer) can lead to problems concerning the available space on the database server due to a very large transaction log volume.

DWH Synchronization During System Operation (Update)

For each Update operation, several parameters can be set, see next section (Parameters for Transfer and Update Operations).

System running in ADMIN mode

If the DWH is running in ADMIN mode, the DWH administrator has to start the update manually by clicking *Update*. When the *Update* button is clicked, a new task of type “Update ”will be created for the operation. This task will appear in the list in the *Tasks* panel (navigation item *Tasks*). Then all data that are supposed to be transferred, i.e., data from fields with the `reportable` = “true” annotation that have been added or changed since the end date of the last Transfer or Update action, are transferred.

If a ticket field, customer field or resource field did not have the `reportable` annotation at the time of the last transfer and has it now, the corresponding content of the field from all tickets, customers and resources is transferred.

System running in LIVE mode

In case the system is running in DWH live mode, the task of type “Update ”is created and managed as mentioned for the ADMIN mode. Additionally, the LIVE mode will be suspended for the time of the update and will be reactivated automatically when the update is finished.

Notes for ADMIN and LIVE mode

> ⚠ Do not remove the annotation `reportable` = “true” for any field without being absolutely sure that the data is not required in reports any longer! If you remove a field that is used in reports and/or data cubes, the reporting will fail at run-time!

The DWH Update can also be started via command line or script. In this case you have to use a tool which can access the ConSol CM MBeans via command line, e.g. ***Twiddle*** for JBoss. The MBean to use is `consol.cmas.global.dwh.synchronizationService`. The command (method) is `update`.

The following command line shows an example command with Twiddle. Twiddle as standalone installation.

```
$TWIDDLE_STANDALONE_HOME/bin/twiddle.sh -s service:jmx:remoting-
 jmx://127.0.0.1:9999 invoke
 consol.cmas:type=admin,topic=global,name=dwh.synchronizationService update
```

Please note that the CM package size, the CMRF cache size, and the retry counts cannot be set using the MBean. The default values will be used.

Parameters for Transfer and Update Operations

For each *Transfer* or *Update* operation, several parameters can be set in the pop-up menu which is opened when the *Transfer* or the *Update* button has been clicked.

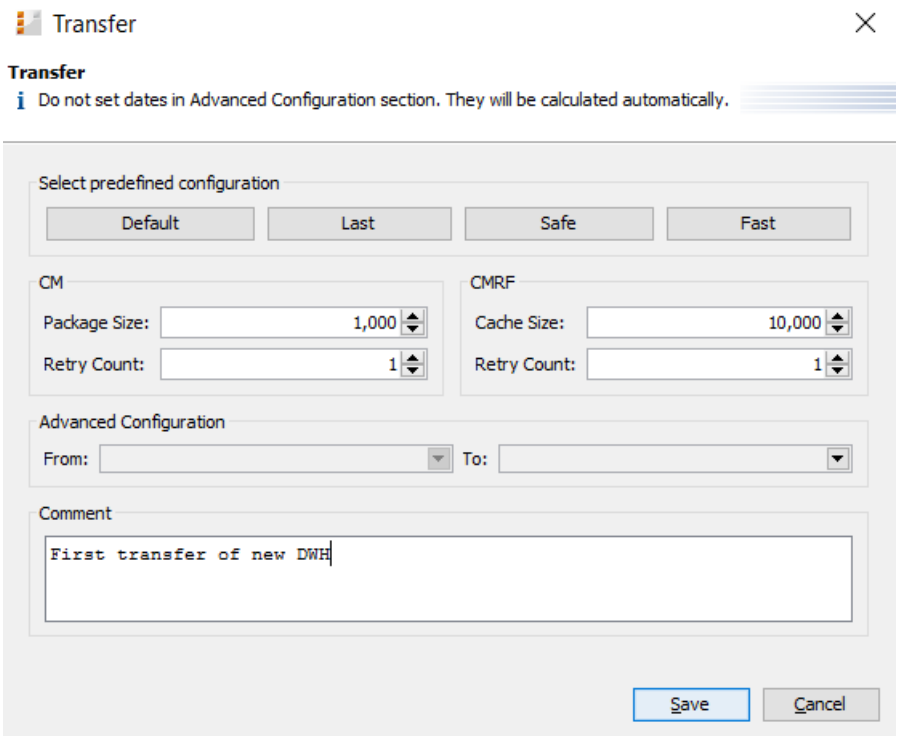


Figure 386: *ConSol CM Admin Tool - Data Warehouse, Administration: Configuration of a transfer operation*

Select predefined configuration for the individual job to start:

- **Default**
  run the job with the default settings, just like without any modification in the dialog.
- **Last**
  run the job with the same configuration like the previous run.
- **Safe**
  run the job with cautious settings, which may cause it to run longer, however, the settings ensure that no `OutOfMemory` exception will occur.
- **Fast**
  run the job with settings optimized for fast completion, however, `OutOfMemory` exceptions can occur, if the Java Virtual Machine is not well-tuned.

When you click one of the buttons, the parameters for the other fields (e.g. *package size*) are loaded into the fields from the respective configuration. No operation is started yet. You can further modify the values for the planned operation.

The following fields are available:

- **CM** (sending side)
  - **Package Size**
    Number of objects sent in one DWH message. Higher values mean better transfer performance and bigger memory usage. If the system has enough RAM, the value can be 1000 or even more. (In CM versions lower than 6.11, this was covered by the CM system property cmas-dwh-server, batch-commit-interval, but this was a system-wide parameter. Starting with CM version 6.11, it is possible to set this for every DWH operation individually.)
  - **Retry Count**
    Number of retries until an ERROR is thrown.
- **CMRF** (receiving side)
  - Cache Size
  - **Retry Count**
    Number of retries until an ERROR is thrown.
- **Advanced Configuration**
  - From
  - To

Please do not use the Advanced Configuration options unless clearly instructed by a ConSol representative to do so!

The parameters offered in the section *Advanced Configuration* for setting a date interval to be covered by the job should not be used without very clear understanding of the consequences. It could lead to an inconsistent data warehouse. They should be only used when specifically advised with detailed instructions by ConSol support or ConSol CM consulting.

- **Comment**
  Optional. A comment which characterizes the DWH action. Will be written into the database

and displayed in the *Details* section.

F.2.2.2 DWH Tasks

If there are due or active DWH tasks, this will be indicated in the Admin Tool, see following figure. The navigation group *Data Warehouse* shows an exclamation mark so that even when this navigation group is closed you will know that there are active tasks. The number of active tasks is indicated at the navigation item *Tasks*.

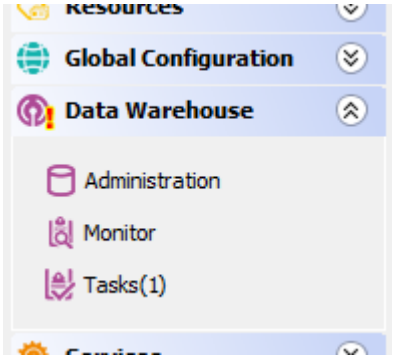


Figure 387: *Indication of active DWH tasks in the Admin Tool*

The list *Current tasks* contains columns and values which are retrieved from the CM database table `cmas_dwh_task`.

In the list of *Current tasks*, you will find entries (one entry per task) if ...

- the DWH is running in ADMIN mode and the administrator has started an update: all tasks that have to be performed are listed.
- the DWH is running in LIVE mode but the check box *Automatic commit of administrative changes* has not been checked.
- Ticket field, customer field or resource field annotations have been changed to `reportable` = “true” and the checkbox *Automatic commit of administrative changes* has not been checked.
- Ticket field, customer field or resource field annotations have been added and changed to `reportable` = “true” and the checkbox *Automatic commit of administrative changes* has not been checked.
- Ticket field, customer field or resource field annotations have been changed from `reportable` = “true” to “false” and the checkbox *Automatic commit of administrative changes* has not been checked.

If the checkbox *Automatic commit of administrative changes* has not been checked, the tasks will remain in the status "ready for execution".

You can mark one or more tasks in the list and execute them manually (*Run tasks* button).

If the checkbox *Automatic commit of administrative changes* has been checked, the tasks will be run automatically by the system. The execution can take several minutes.

There are some special cases to consider:

- Removing the annotation `reportable` = “true” or `reportable group` = “true” does not create a DWH task. These column and the respective DWH tables are only removed by the next DWH update.
- In ADMIN mode, a task for changing `reportable` to “false” is ignored! A task is created and is removed instantly. The DWH table is not changed! This change is performed by the next DWH update.
- In LIVE mode, the task is executed immediately, even when the checkbox *Automatic commit of administrative changes* is not checked.
- With the DWH update action the table structure is extended by the newly annotated field from the open tasks, but the data is only initially filled in during the execution of the respective DWH task!

F.2.2.3 DWH Monitor

To open the DWH Monitor, use the navigation group *Data Warehouse*, navigation item *Monitor*.

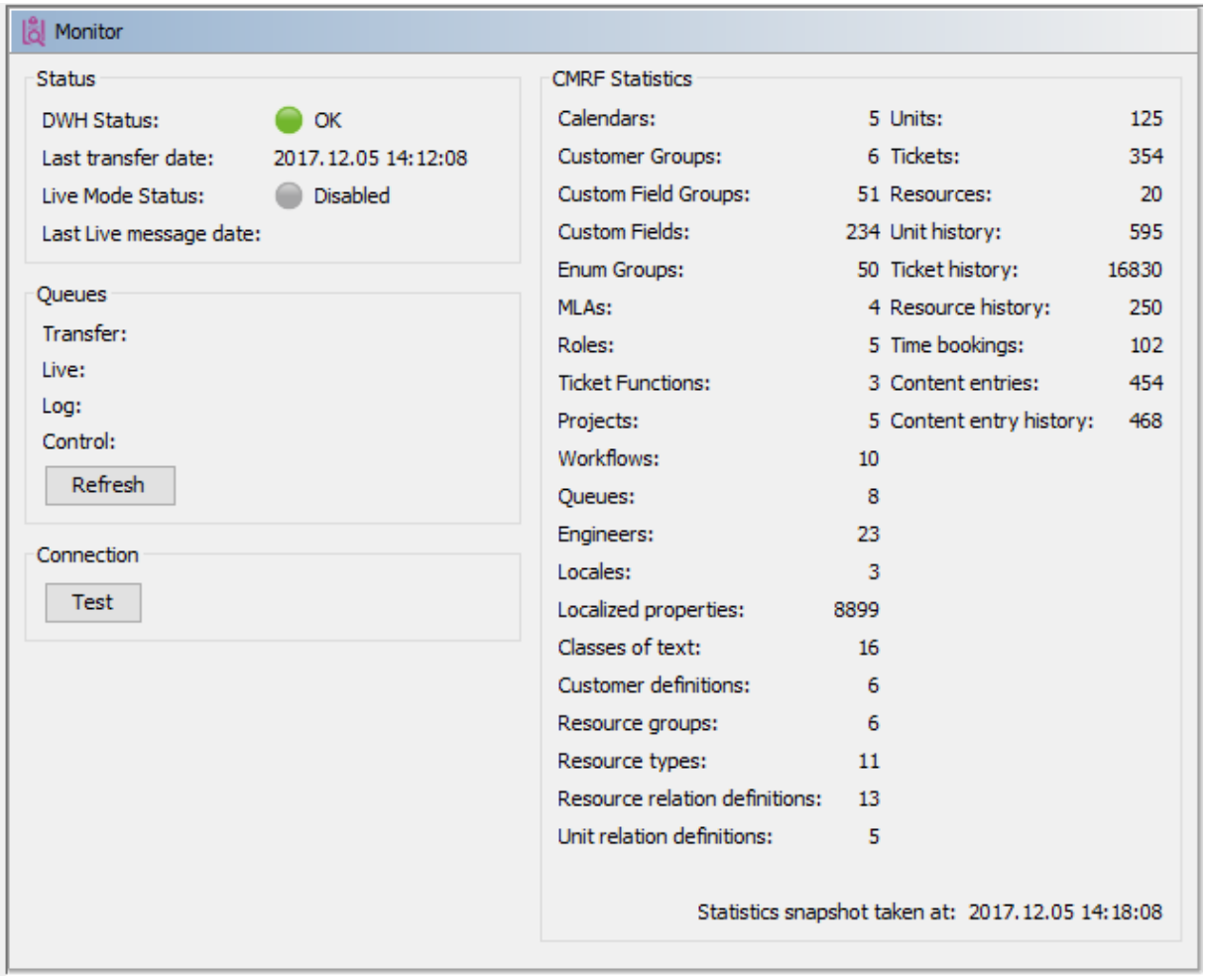


Figure 388: *Consol CM Admin Tool - Data Warehouse, Monitor: DWH Monitor*

The DWH monitor provides a quick overview of all DWH parameters. The date of the current snapshot is indicated at the bottom right of the screen.

The following parameters are displayed:

- **Status**
  A quick overview of the most important parameters.
  - **DWH status**

Color of status symbol	DWH status message
Green	OK
Yellow	Uninitialized
	Action is processed
	Action is paused
	Action is planned
Red	Error
Grey	Disabled

  - **Last transfer date**
    Date of the last transfer or update operation
  - **Live mode status**

Color of status symbol	Live mode status message
Green	OK
Yellow	Uninitialized
	Action is processed
	Action is paused
	Action is planned
	Started (update is needed)
Red	Error
	Error (update is needed)
Grey	Disabled

- **Queues**
  Informs about the number of message entries in the internal data warehouse message queue tables. Use the *Refresh* button to update the values.
  - Transfer: Number of messages in the data warehouse transfer table (INT_TRANSFER_QUEUE)
  - Live: Number of messages in the LIVE messages table (INT_LIVE_QUEUE)
  - Log: Number of messages in the log messages table (INT_LOG_QUEUE)
  - Control: Number of messages in the control messages table (INT_CONTROL_QUEUE)
- **Connection**
  Test the connection to the DWH database using the *Test* button.
- **CMRF statistics**
  Provides a list of the number of objects which have been transferred.

F.2.2.4 DWH Troubleshooting and Repair

If any errors have occurred during initialization, transfer, or update, the log entries are displayed in the log panel which is opened with the *Log* button .

You can also check the original log file under the following path:

- **JBoss 7 (single server):**

```
<JBOSS7_HOME>/standalone/log/cmrf.log
```

- **Weblogic:**

```
<ORACLE_HOME>\Middleware\user_projects\domains\consolcm6_domain\cmrf-
 logs\cmrf.log
```

Please note that these are the standard paths. They may be configured to use different paths in the file `cm6-cmrf.xml` file. A detailed description of CM logging and log files is provided in the *ConSol CM Setup Manual* and the *ConSol CM Operations Manual*.

Usually the log file and/or log panel entries give good hints regarding the initial reason for a transfer failure. If you run into a problem you cannot resolve and you have a maintenance contract with ConSol, please contact our support team.

For more details on Troubleshooting and Repair see section *CM / CMRF / DWH Synchronization: FAQs and Tips for Troubleshooting* in *ConSol CM Operations Manual*

F.2.3 DWH-Related System Properties

A list of all system properties which are relevant for a specific DWH configuration can be found in section CMRF & DWH Configuration of the System Properties chapter.

F.2.4 Transfer Mode

All data which has been annotated as `reportable` = “true” has to be transferred to the Data Warehouse. This is performed by ConSol CM and the CMRF.

In ConSol CM versions 6.11. and up, there is one transfer mode:

- DIRECT mode

The mode is set using the system property cmas-dwh-server, communication.channel. Possible values are:

- “DIRECT”: database communication channel, only available value since version 6.11.

In DIRECT mode, the CM server sends messages to the CMRF by using direct access to database tables in the DWH database:

- INT_CONTROL_QUEUE
- INT_LIVE_QUEUE
- INT_TRANSFER_QUEUE

CM is able to do that because in DIRECT mode, the CMRF datasource is made accessible for ConSol CM as well, i.e.

- in overlay mode:
  ConSol CM and CMRF use the same application server
- in standalone mode:
  the CMRF database configuration file is also located on (copied to) the ConSol CM application server

The CMRF server reads the entries from the tables and writes the data into the DWH.

F.2.5 Expert DWH Information

F.2.5.1 Internationalization of Static DWH Tables

Starting with CM version 6.10.1, database fields for the localized values in static DWH tables were added. This means, the localized descriptions of parameters will also be transferred to the DWH. The following example shows the fields in the Admin Tool and in the DWH for projects in ConSol CM.

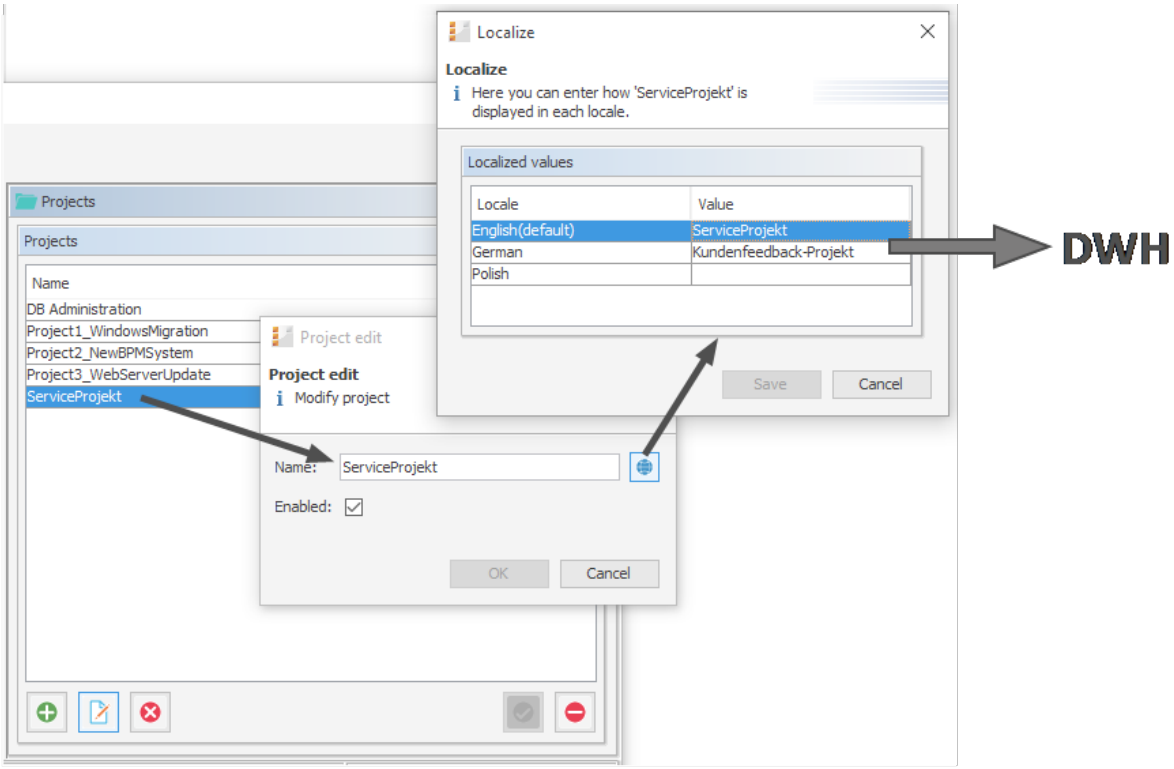


Figure 389: *ConSol CM Admin Tool - Global Configuration, Projects: Localized values of a project*

table *dim_project*

project_id	project_uid	name	name_en	name_de
46	2fc3d1c9-2df4-11e4-b9c4-ad888261acc9	ServiceProjekt	ServiceProjekt	Kundenfeedback-Projekt
47	4bb09eda-9e5f-4142-b33e-41e9b03d1e8f	Project3_WebServerUpdate	Project 3	Projekt 3
48	6db541f8-8a05-42e9-8eff-64620369ee9c	Project1_WindowsMigration	Project 1 Windows Migration	Projekt 1 Windows-Migration
49	aa570a20-3322-4696-9ffc-fd0750aebe25	Project2_NewBPMSystem	Project 2 New ERP System	Projekt 2 Neues ERP-System
50	5fce3e53-c8d1-11e5-998a-67528c2f9cca	DB Administration	DB Administration	NULL
NULL	NULL	NULL	NULL	NULL

Figure 390: *DWH table with internationalized values*

The use of those fields can increase the DWH update time. Thus, to prevent an update from running too long, you can start CM with the Java system property `cmrf.localization.enabled`. This property can be used to switch the transfer of localized values to the DWH on or off.

Example start command:

```
nohup $JBOSS_HOME/bin/standalone.sh --server-config=cm6-cmrf.xml -b=0.0.0.0 -
 Dcmrf.localization.enabled=false
```

For a detailed explanation, please see also the *ConSol CM DWH Manual*.

F.2.5.2 MBeans for DWH Management

Starting with version 6.10.5.4, the MBean `dwh.admin.service` (to be found `consol.cmas.admin.global`) provides access to the DWH mode (OFF |ADMIN | LIVE) . The two methods `getMode()` and `setMode()` are available.

The MBean can be accessed using graphic tools, e.g., JConsole for JBoss, or it can be used via REST API. This is described in the *ConSol CM REST API Documentation*.

F.3 CM Services

CM Services are managed on the navigation item *CM Services* in the navigation group *Services*. This navigation item contains the list (1) of the available CM services. Stopped or disabled services are displayed in italics. In this navigation item you can start (2) or stop (3) the individual services of the ConSol CM system, e.g., data indexing or DWH transfer.

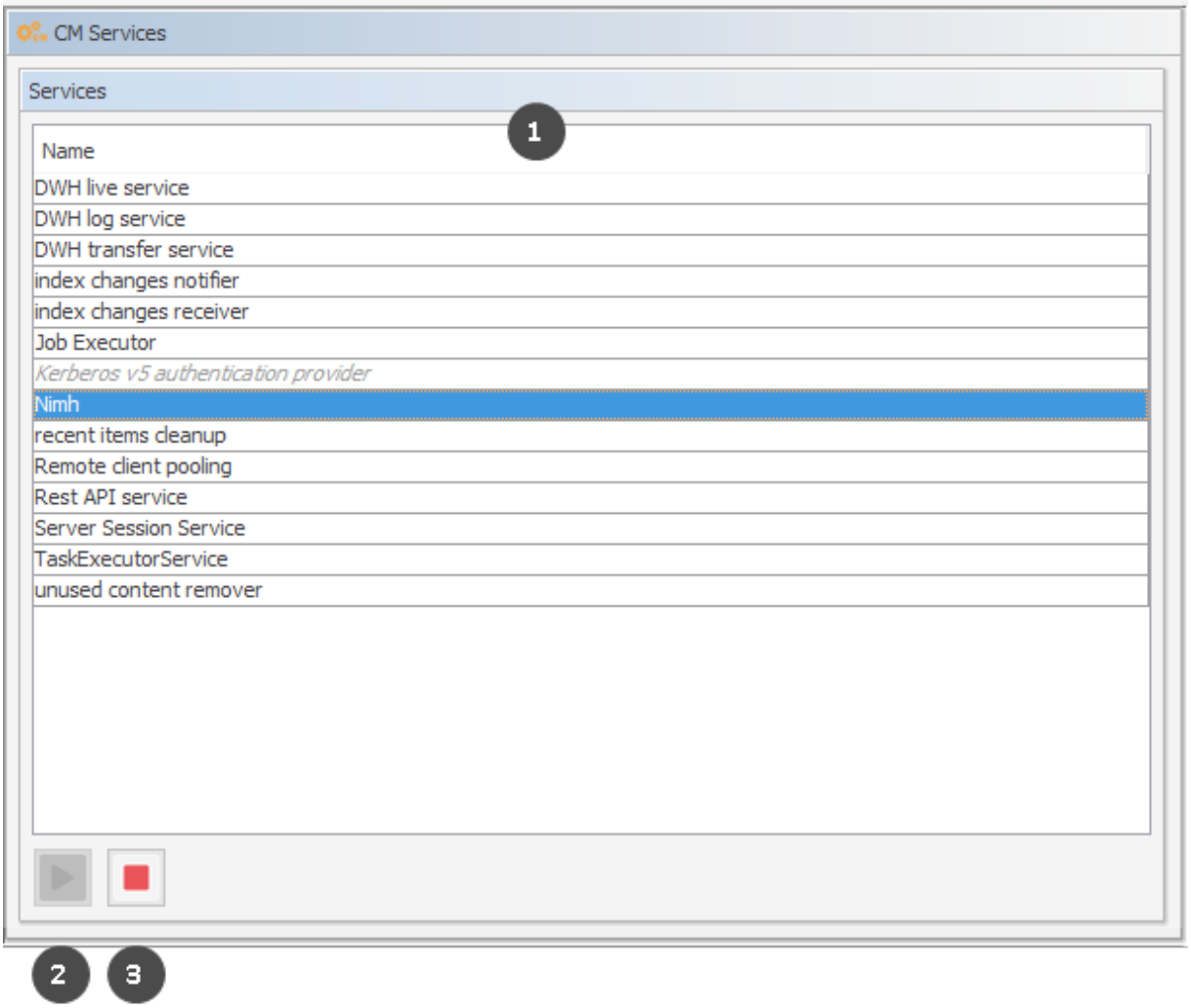


Figure 391: *ConSol CM Admin Tool - Services, CM Services*

⚠ The status of a service should only be changed by an experienced ConSol CM consultant or by a member of the ConSol CM support team! ConSol CM core functionalities might not work when a service is not running!

List of services:

- **DWH live service**
  Controls just-in-time DWH update in LIVE mode.
- **DWH log service**
  Reads and processes CMRF/DWH log messages for the Admin Tool and stores them in the ConSol CM database. The entries are used for the log protocol in the Admin Tool. See section Data Warehouse (DWH) Management.
- **DWH transfer service**
  Controls DWH transfers.
- **Job Executor**
  Controls escalations for processes resp. workflows.
- **Kerberos v5 authentication provider**
  Required if Kerberos authentication is enabled.
- **NIMH**
  Retrieves incoming emails when the server is running in NIMH mode (only mode in 6.11.1).
- **Remote client pooling**
  Controls that Web Clients get changes from the Admin Tool.
- **Rest API service**
  Activates or deactivates the REST (*Representational State Transfer*) interface.
- **Server Session Service**
  Checks sessions and stops sessions when the lifetime of a client or Admin Tool session has expired. See, for example system, properties admin.tool.session.check.interval and server.session.timeout.
- **TaskExecutorService**
  The engine for task execution in the Task Execution Framework. It comprises a main processing thread (with watchdog attached) which scans the database for tasks with the status *NEW*, and a second component which controls a dedicated thread pool used for tasks execution.
- **Index changes notifier**
  Creates JMS (*Java Message Service*) messages with notifications when changes occur that concern in the index.
- **Index changes receiver**
  Reads a JMS queue and starts update in Indexer.
- **Unused content remover**
  Removes attachments and comments which have been marked as *deleted* in the Web Client (in the protocol section of a ticket).

F.4 Search in ConSol CM

ConSol CM offers a powerful search engine.

Learn how to configure ConSol CM for your search requirements in section Search Configuration. Section ConSol CM Indexer contains information about the indexer, which is needed for search.

Starting with ConSol CM version 6.10.1, it is possible to perform actions on search results. This feature is part of the ConSol CM Action Framework and is described in the Action Framework - Search Actions section.

Result sets retrieved via the Web Client can be exported to a CSV list. This feature was added in ConSol CM version 6.10.1 and is explained in section CSV Export of Search Results.

F.4.1 Search Configuration

ConSol CM provides a powerful search mechanism for all objects involved in the business processes (customers, tickets and resources). Technically, the search is based on the **Indexer**, a module of ConSol CM.

The following paragraphs explain the entire topic *Search in ConSol CM* from an administrative point of view. Please refer to the *ConSol CM User Manual* for a detailed explanation about how to use the search as an engineer.

F.4.1.1 Search Modes

A ConSol CM engineer can use several search modes:

Quick Search

This is performed using the Quick Search field in the upper right-hand corner of the Web Client. The display of the results (i.e., the fields and the order of the fields in the result list) can be formatted using templates, as detailed in sections Templates for Customer Data and CM/Resource Pool - Templates for Resource Data. Please keep in mind that you can adapt the length of the result set using the system property cmweb-server-adapter, globalSearchResultSizeLimit. The result set will also be influenced by the Page Customization attribute *appendWildcardAutomatically*, see section Page Customization, appendWildcardAutomatically for details.

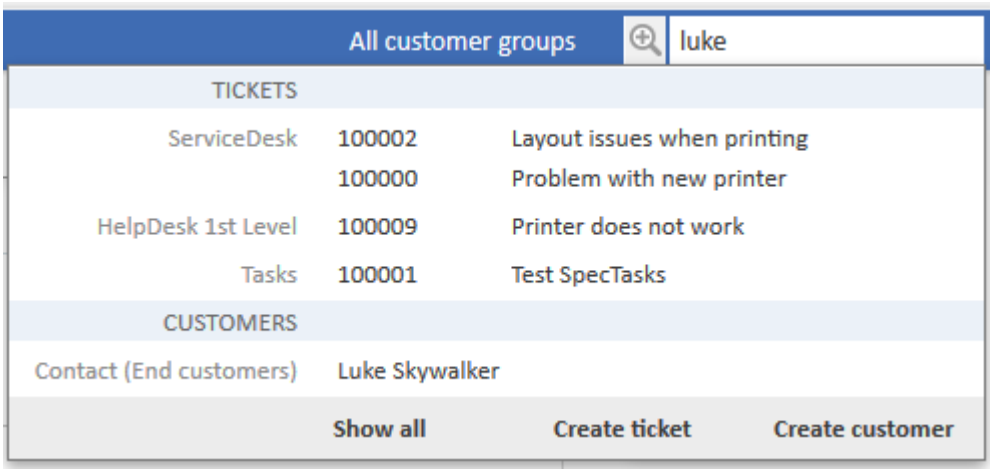


Figure 392: *ConSol CM Web Client - Quick Search*

Phonetic Search

For the Quick Search, a phonetic search can be configured for String fields. Please see section Configuring the Phonetic Search for the explanation of how to prepare your ConSol CM system for this feature.

With a phonetic search, the engineer cannot only find exact matches in the search results but also results which sound similar even if the spelling differs from the entered search term. The implementation is based on the Apache Commons Codec libraries.

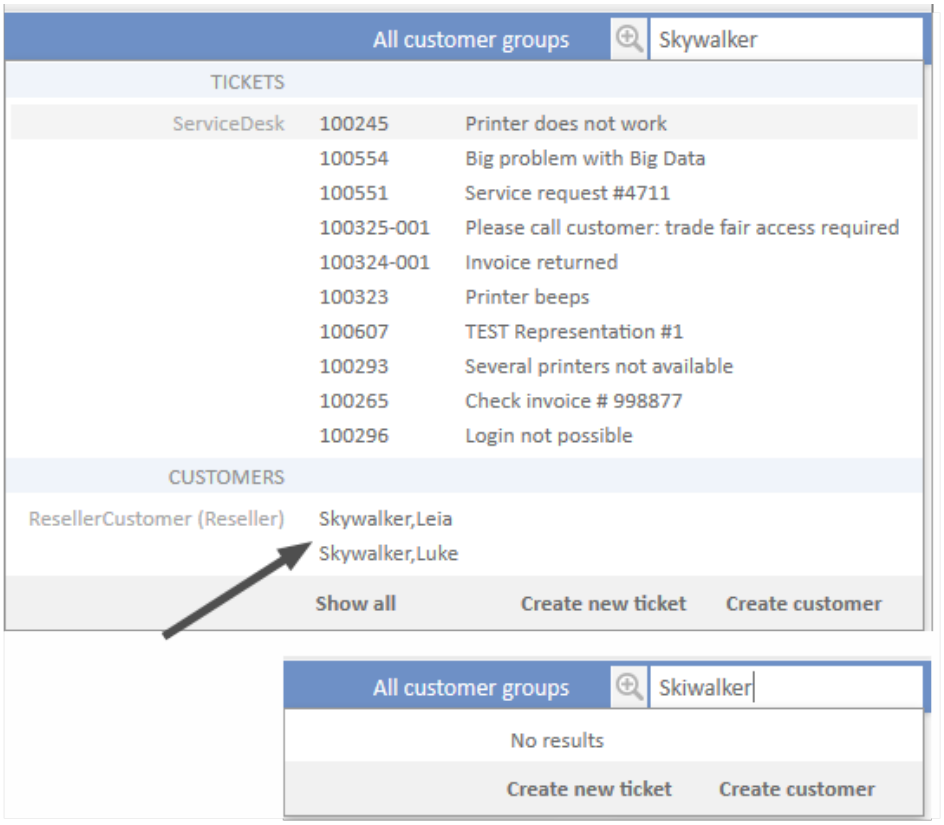


Figure 393: *ConSol CM Web Client - Search results without phonetic search*

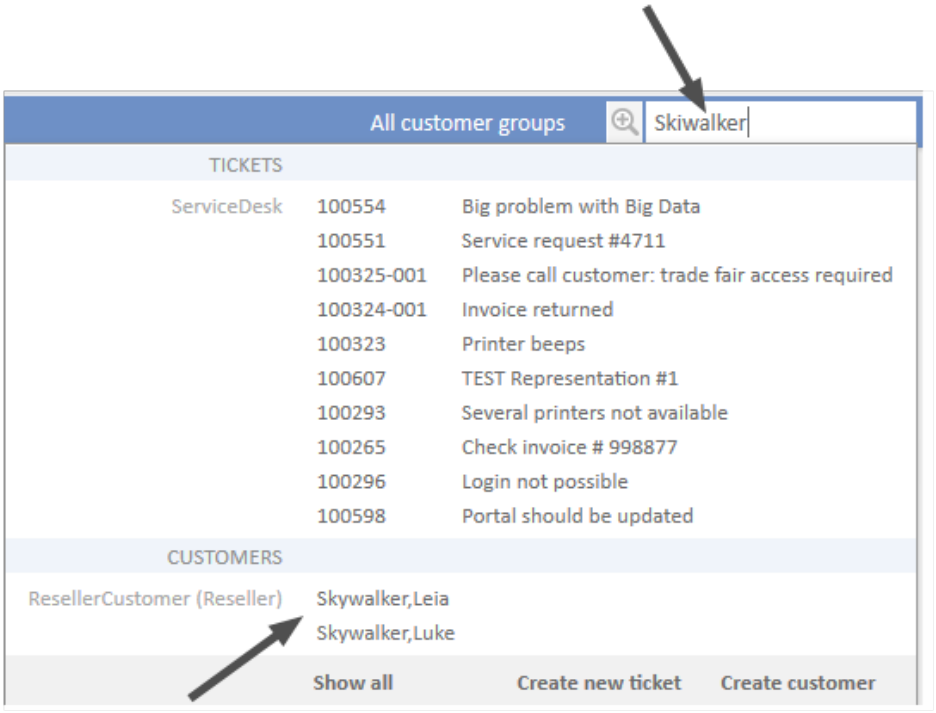


Figure 394: *ConSol CM Web Client - Search results with phonetic search*

Detailed Search

This is performed using the *Detailed Search* page. To open this page, click on the magnifier icon next to the Quick Search entry field.

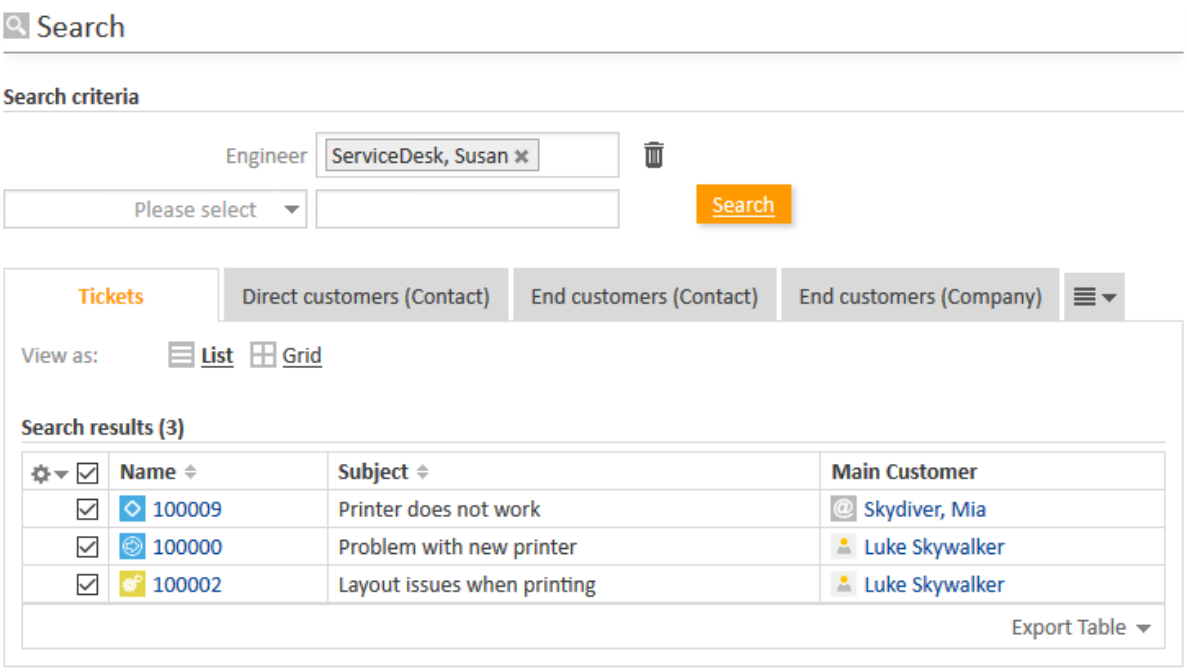


Figure 395: *ConSol CM Web Client - Detailed Search*

Please keep in mind that the result list length and paging of the result list for the Detailed Search can be configured using the system properties cmweb-server-adapter, searchPageSize and cmweb-server-adapter, searchPageSizeOptions.

For the engineer search, you, as an administrator, can define engineer names which should not be offered in the drop-down menu in the search criteria. This is configured using page customization. Please see section excludedUserNames for details.

Please refer to the *ConSol CM User Manual*, section *Searching for Tickets, Customers and Resources* to learn how to use the search functionality.

Important note about search with fields of lists of structs

Please note that in ConSol CM versions 6.10.6.x and older, two fields which are part of the same list of structs (i.e. of a table) are combined by OR in the Detail Search. In CM versions 6.10.7.0 and up, the two fields are combined by AND. This will lead to different search results in the different CM versions!

Please refer to the *ConSol CM User Manual* for a detailed example.

Configuring the Search Result List

You, as an administrator, can configure the layout of the search result list using the annotation `order-in-result`. This annotation influences the columns of object-specific search result sets. For example, the customer field group *ResellerCompanyData*, belonging to the customer data model *ResellerModel*, contains the following customer fields:

- *company_name*: order-in-result = 1
- *company_number*: order-in-result = 2

Thus, in a search result list, e.g., a Detailed Search which shows a result set with companies of a customer group which uses the ResellerModel will contain the two columns *company_name* and *company_number*.

For a field (i.e., column in the result table) which should not be displayed in the Web Client (i.e., in the field selector for the result list and as column in the table) at all, set the value of the `order-in-result` annotation to "0" (zero).

The following figure shows the `order-in-result` annotation of a customer field. You reach this screen by opening the navigation item *Data Models* in the navigation group *Customers*.

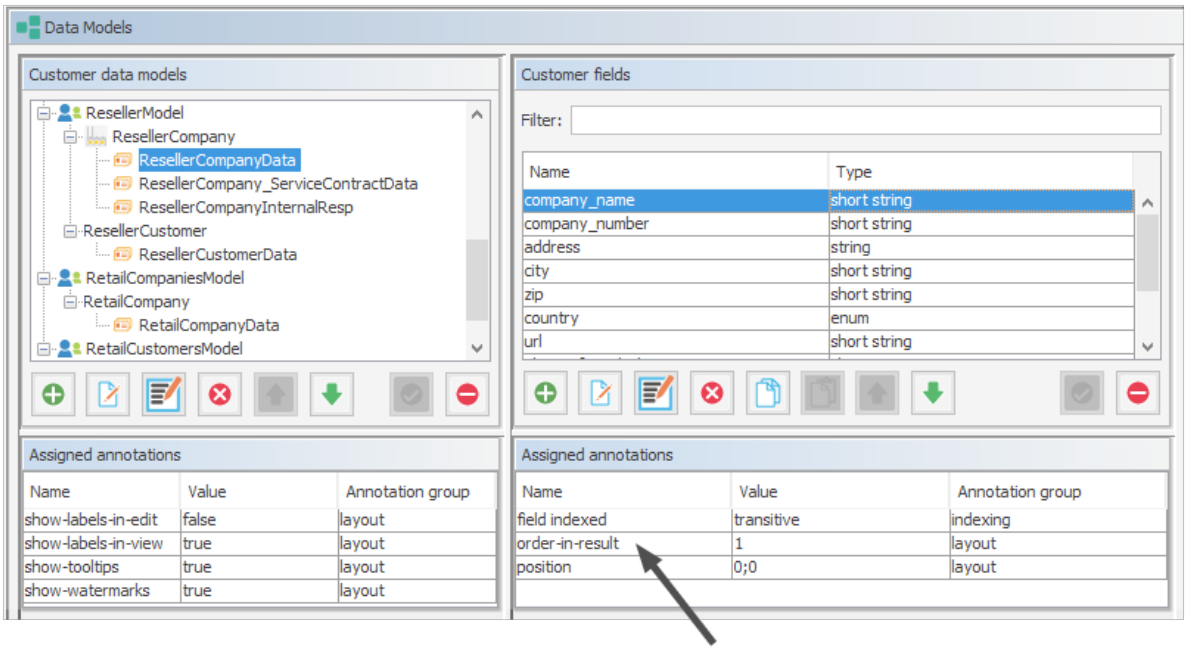


Figure 396: *ConSol CM Admin Tool - Customers, Data Models: Setting the annotation order-in-result for a customer field*

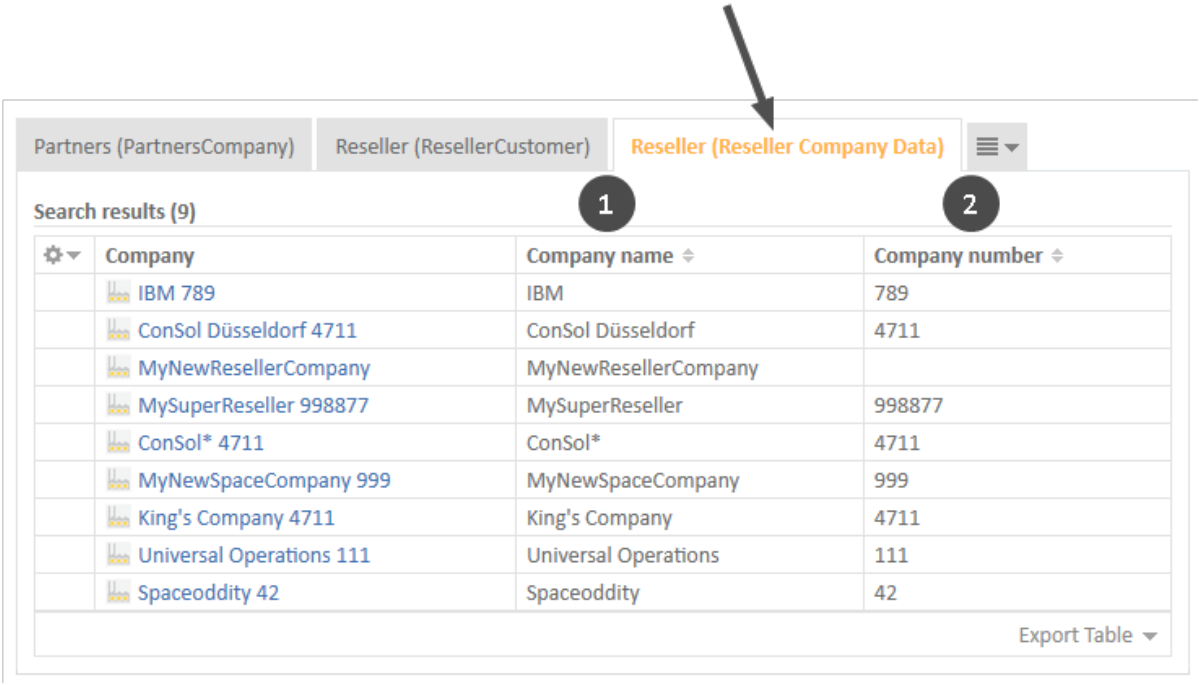


Figure 397: *ConSol CM Web Client - Search result set with the two annotated columns (order-in-result)*

Notes about the Order of Search Result Columns

Since the annotation `order-in-result` is not the only parameter which influences the order of columns in search results (tables), please find here a detailed explanation.

In the following section, *user preferences* means the layout of a search result table which the engineer has configured manually, e.g., the columns which should be displayed, the order of those columns or the sorting.

Rules:

1. The columns of a search result table can display different fields for each of the CM object types:
    a. for tickets: ticket fields as well as internal fields like queue, creation date, engineers
    b. for customers (contacts or companies): customer fields as well as internal fields like object name of a customer
    c. for resources: resource fields as well as internal fields like object name of a resource
2. The `order-in-result` annotation organizes only the order of the ticket fields, customer fields, and resource fields, not of internal fields.

3. The internal field types have the following visibility and order by default:
    a. for ticket lists:
        i. Engineer
        ii. Main customer
        iii. Ticket name
        iv. Ticket subject
    b. for customer (contact/company) lists:
        i. object (i.e. contact or company) name
    c. for resources lists:
        i. object (=resource) name
4. The data fields (ticket field, customer fields, and resource fields) are primarily ordered by the user preferences. Those will overwrite any other configuration.
   When, e.g., as default configuration, a ticket field has the annotation `order-in-result = 5` and is visible, it will (case a) not be displayed if the engineer has deselected the column, and (case b) it will be displayed as column #2 if the engineer has placed it there.
5. If no order is defined by the user preferences, the order is the ascending order of numbers from the data field annotation values `order-in-result`. Those numbers are evaluated globally across all data fields within all ticket field groups, all customer field groups, and all resource field groups. Identical values of the annotation `order-in-result` of different data fields are theoretically possible.
   For an exactly defined order, use some convention for an entire CM installation and apply unique numbers, e.g., use high four digit values.
6. If there are still two fields with identical values for `order-in-result` which should both be displayed, the data fields are alphabetically ordered based on their localized names. The locale of the engineer's browser is used.

Configuring Detail Search Behavior for Enum and MLA Values

The values within an enum (a sorted list) or an MLA might change over time, e.g. a new software version is added to a list, an old one should no longer be available. In this case, the deprecated/old value will be deactivated. This means that there might be tickets which contain enum or MLA values which are deactivated in the current system. Of course it is indispensable that engineers can still search for tickets which contain this value. Therefore deactivated enum or MLA values are offered in the Detailed Search as shown in the following figure for the deactivated enum value *Resource Pool*.

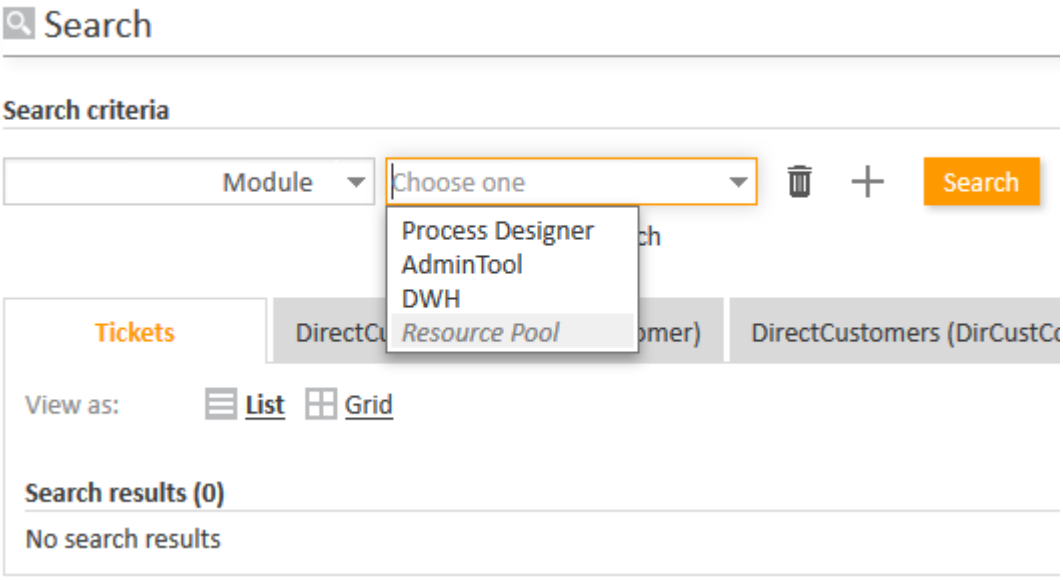


Figure 398: *ConSol CM Web Client - Detailed Search for modules, one enum value deactivated*

The ConSol CM behavior concerning the Detailed Search of deactivated enum and MLA values can be configured using a Page Customization attribute: enableSearchForDeactivatedEnums .

Autocomplete Search (Search by Using Intelligent Fields)

The Autocomplete Search is performed implicitly when you start entering a word in an Autocomplete field, e.g., in company data or customer data when you create a ticket (see figures below).

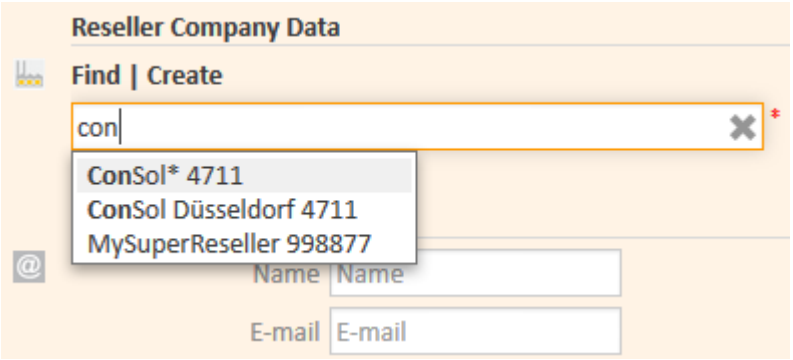


Figure 399: *ConSol CM Web Client - Autocomplete Search*

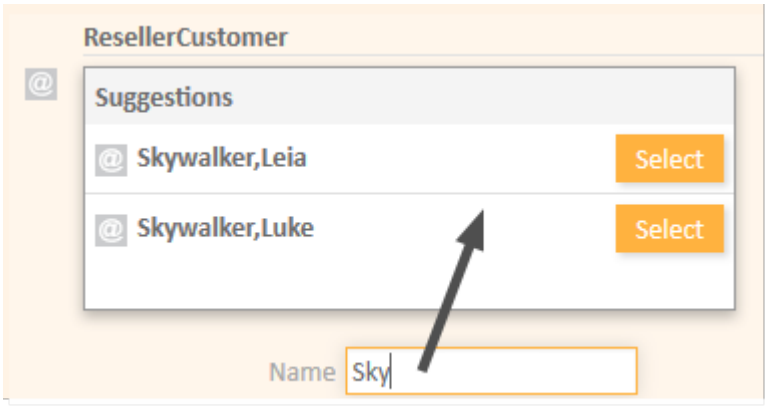


Figure 400: *ConSol CM Web Client - Suggestions for an Autocomplete Search*

Starting with ConSol CM version 6.10.2, a search in all contacts of the ConSol CM database is also performed in the Ticket Email Editor when an engineer starts typing in the To, Cc or Bcc field:

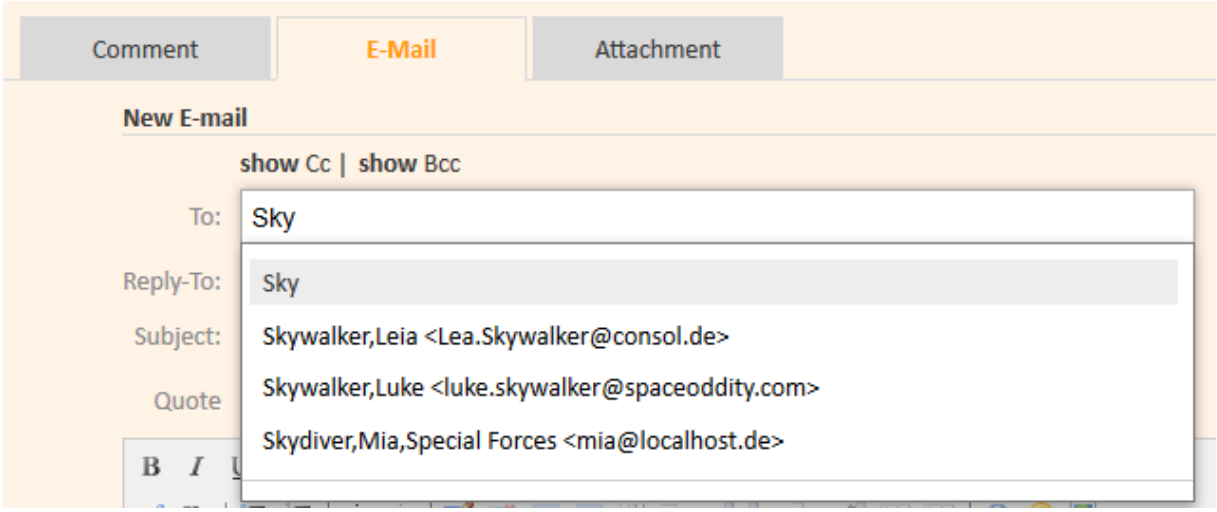


Figure 401: *ConSol CM Web Client - Ticket Email Editor: Automatic search for contacts as email receivers*

The number of characters which must be typed before automatic search is activated can be configured using the page customization attribute *minMailInputLength* (Integer, Default 1) in the scope *mailTemplate*.

F.4.1.2 Fields Which Can Be Searched

For ticket fields, customer fields and resource fields which should be searched, the annotation `field indexed` has to be set. See section Search Configuration. This makes the field available for the Quick Search and for the Detailed Search.

Two types of fields are processed by the search engine:

1. data fields and content which are/is indexed by default
2. data fields which are annotated using the annotation `field indexed`

Data Fields and Content Which Are Indexed by Default

- Engineer data
  - Email
  - First name
  - Last name
  - ID
- Ticket data
  - Attachment content (this is controlled by the CM system property cmas-core-index-common, index.attachment which is “true” per default)
  - Creation date
  - Engineer
  - History
  - ID (technically the ticket name)
  - Queue
  - Referenced engineers
  - Subject
  - View

Data Fields Which Are Annotated Using the Annotation `field indexed`

For all data fields which have been added and configured by the ConSol CM administrators, the annotation `field indexed` has to be set to make the fields searchable. A field which is indexed is available in

- the Quick Search
- the Detailed Search
- all searches in workflow and Admin Tool scripts which use, for example, criteria classes/objects (e.g., `TicketCriteria`, `UnitCriteria`, `ResourceCriteria`). This includes
  - scripts which have been manually added by the administrator
  - scripts which are present by default (e.g., the script `createTicket.groovy`. In this script, the correct contact for a new ticket can only be set if the customer field which is used to retrieve the correct contact or company is indexed)

Please note that if values of a table, i.e. a list of structs, should be available for the search, all elements of the data structure have to be annotated with `field-indexed` = “true”. This means that the list, the struct and every single data field within the struct have to be annotated that way!

You, as an administrator, can define the system's behavior with regards to search results. Depending on the values of the `field indexed` annotation, the search results for a contact might include all the tickets of the contact as well. The following table shows the implications of all possible values of the `field indexed` annotation for the different object types. The `field indexed` annotation has to be set for each individual data field, e.g., *name* and *email* for a contact, *zip* and *address* for a company, *priority* and *software module* for tickets, or *name* and *model* for a resource type.

Please note that two distinct perspectives are addressed here:

1. the **search criterion**, i.e., the field for which you start the search, e.g., all companies that start with ConS*. A field is made available in the search by setting the `field-indexed` annotation.
2. the **result set**, i.e., the objects which are displayed in the result lists. The objects in the result set are defined by the value of the `field-indexed` annotation. Depending on this value, you might see only the objects you have searched for (e.g., only contacts) or also objects which are related to the objects you have searched for (e.g., companies of those contacts). Please see the following table for a detailed explanation.

Object type/ value of field indexed annotation	transitive	unit	local	not indexed
TICKET	• no difference between the three values • the ticket data will be available for searching • no other objects which are related to a ticket in any way will be retrieved • we recommend that you use the default value “transitive”			• the ticket data will be NOT available for searching

Object type/ value of field indexed annotation	transitive	unit	local	not indexed
CONTACT	• the contact data will be available for searching • the tickets of the contact will also be found when you search for the contact • searching by company by its contact field is NOT possible	• the contact data will be available for searching • searching for a company by its contact field is NOT possible	• the contact data will be available for searching • no other objects which are related to a contact in any way will be retrieved	• the contact data will be NOT available for searching
COMPANY	• the company data will be available for searching • the tickets of the company will also be found when you search for the company • the contacts of the company will also be found when you search for the company	• the company data will be available for searching • the contacts of the company will also be found when you search for the company	• the company data will be available for searching • no other objects which are related to a company in any way will be retrieved	• the company data will be NOT available for searching
RESOURCE	• no difference between the three values • the resource data will be available for searching • no other objects which are related to a resource in any way will be retrieved • we recommend to use the default value “transitive”			• the resource data will be NOT available for searching

Please note that if it should be possible to sort result tables (in the Web Client) according to a column (by clicking on the column header), the respective field has to be indexed!

Configuring the Phonetic Search

In order to activate the phonetic search for a data field, set the annotation `phonetic` to “true” for this field. The annotation can only be used for String fields.

F.4.2 ConSol CM Indexer

F.4.2.1 Introduction

ConSol CM stores most of its data in a relational database. In order to improve the performance of search operations, ***ConSol CM*** uses ***Apache Lucene*** indices. For each data field that should serve as search criterion, an index is created. The ConSol CM module which creates and manages the indices is called the ***CM Indexer***. The indexes are stored on the file system, in a subdirectory of the data directory that was indicated during system setup. The path of the data directory is stored as a CM system property: cmas-core-shared, data.directory.

F.4.2.2 Indexer Data Directory

The following picture shows an example of index files of a ConSol CM installation. `demo_Datadir` is the data directory specified during setup, and all subdirectories are created automatically by ConSol CM.

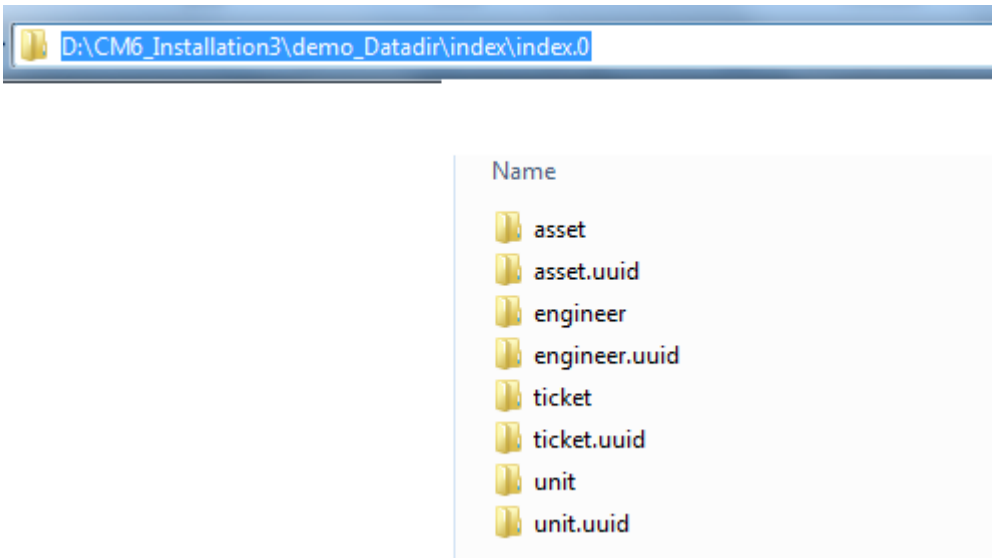


Figure 402: *ConSol CM Indexer - Directory demo_Datadir*

Please make sure that ...

- the data directory is always available for the ConSol CM server system if it was created on another server and is linked to (or mounted on) the application server.
- the data directory provides enough space for storing indexes.

If the index directory should be corrupted or the index is not available, it can be rebuilt or repaired. Please see the following sections for details.

F.4.2.3 Indexer Services

For indexing, two ConSol CM services are important:

- **Index changes notifier**
  This service creates messages in the persistent store with notifications when changes occur

that concern the index.
Stopping *index changes notifier* is **not** safe. If the Indexer module discovers that the notifier is stopped and there is a message that has to be sent to the persistent store, the Indexer will set the index status configuration property (cmas-core-index-common, index.status) to "RED", i.e., signal that index needs full synchronization.

- **Index changes receiver**
  The behavior of this service differs depending on the type of node it is running on:
  - On a master node, the *index changes receiver* loads tickets and creates the respective Lucene documents.
  - On a slave node (possible in a cluster only), it polls the master indexing server via http to download the Lucene files with the indexed data.
  - In any case, this service reads the persistent store and starts the update of the Indexer. Stopping *index changes receiver* is safe. After restart it will pick up all of the missing changes from the persistent store.

Please see also section CM Services.

F.4.2.4 Administrator Tasks Concerning the Indexer

It is important for the administrator to know how to configure ConSol CM in a way that ...

- all required fields can be searched.
- no overhead is produced (i.e., not too many fields are configured for searching).
- the search results are displayed in the desired way.
  - in the result table in the Detailed Search
  - as suggestions for Autocomplete Search

F.4.2.5 System Properties for the Indexer

There are several system properties which are used to configure the indexer. Please see Indexer for a list of the most important ones.

F.4.2.6 Indexer Management Using the Admin Tool

The Annotation `field indexed`

By default, the entire ticket text and all attachments are indexed. For all ticket fields, customer fields and resource fields, the fields which should be indexed have to have the annotation `field-indexed`. Please refer to the sections Ticket Field Administration (Setting Up the Ticket Data Model), Customer Field Management and GUI Design for Customer Data and CM/Resource Pool - Setting Up the Basic Resource Model for details about setting annotations to ticket fields (ticket data), customer fields (customer data) and resource fields (resource data). There are four possible values for this annotation:

- **local**
- **unit**

- **transitive**
- **not indexed**

A detailed explanation of the system behavior triggered by those values is provided in the table above.

Nested fields all have to have the same index type, otherwise they cannot be searched. For example, when you work with a list of structs, the list, the struct and all data fields in the struct which should be searched have to have the value “transitive” for the annotation `field-indexed`.

Indexer Management: Navigation Item *Index*

Usually, the index requires no manual maintenance. ConSol CM will handle everything regarding indexing automatically. There are only two cases where you have to perform manual administrative operations:

- You would like to change the configuration, e.g., by changing `field-indexed` annotations.
- Errors have occurred in the indexing process.

In the Admin Tool, open the navigation group *Services*, navigation item *Index* to configure and manage the Indexer. If Indexer tasks are still running, this will be indicated by an exclamation mark next to the name of the navigation group (*Services*) and by the number of open tasks next to the name of the navigation item (*Index*). In this way, even when the navigation group is not opened, you, as an administrator, can immediately notice that some tasks are open in the group *Services* and can then quickly identify the number of open tasks.

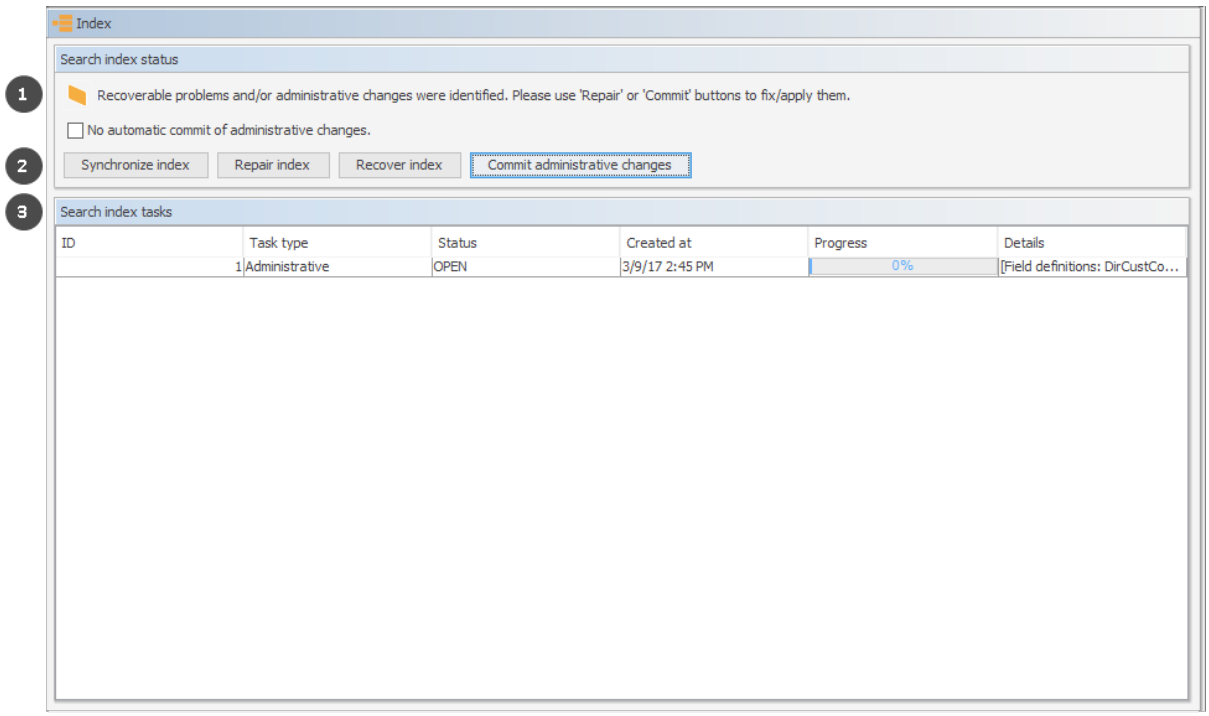


Figure 403: *ConSol CM Admin Tool - Services, Index*

The Index screen contains three main elements:

- Status of the indexer (1)
- Buttons to manage the indexer (2)
- List of open/active indexer tasks (3)

The number of open indexer tasks is also indicated next to the navigation item *Index*.

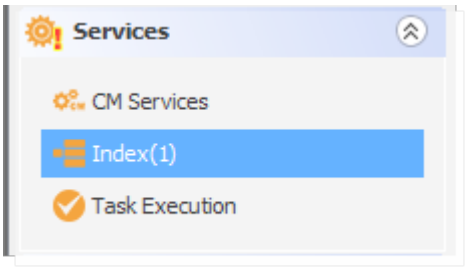


Figure 404: *ConSol CM Admin Tool - Services: Indication of open indexer tasks*

In the first line the current status of the Indexer is displayed (this is the value of the system property cmas-core-index-common, index.status):

- **GREEN**
  All Indexer tasks have run correctly, no action required. At the beginning of the synchronization process, the index status is set to green. If it completes successfully, it remains green. If there are any problems, it will change to yellow or red.
- **YELLOW**
  Fixable problems were identified, collected and persisted. The status is set when an administrative task (with auto-commit “off”) or a retry task is created.
- **RED**
  Errors have occurred. Please check. The index needs full synchronization.

The following operations can be performed:

- **Synchronize index**
  The index is rebuilt completely (from scratch), all open Indexer tasks are discarded. Please select one value for each of the following options to define the order of indexing. This will exert some influence on the period of time required until special search results are available.
  - **Tickets**
    Index open tickets of all queues, then closed tickets of all queues - Indexes all open tickets first, then all closed tickets.
    Index open and closed tickets of each queue - Indexes queue per queue.
  - **Attachments**
    Index tickets along with attachments - Complete indexing. Takes longer.
    Index tickets without attachments, then again with attachments - Provides quick indexing of all tickets in a way that ticket search is up-to-date quickly. Then attachments are indexed as well which takes some longer time.

- **Repair index**
  Indexer tasks which have not run successfully are restarted. The tasks can be selected in the Indexer task list.
- **Recover index**
  A time range can be selected. All changes which have been committed to ConSol CM during this period of time will be (re-)indexed.
- **Commit administrative changes**
  Click this button to commit the changes when you have set a ticket field, customer field or resource field to `field-indexed` that was not indexed before. This has to be used if the checkbox *No automatic commit of administrative changes* has been selected. If the checkbox is inactive, the changes will be committed automatically when you have set the new annotation (s).

  ⚠ Please note, that the automatic commit of administrative changes can affect the system performance.

ⓘ **For experienced administrators only:**

The operation *Commit administrative changes* can also be executed using the ConSol CM MBean `consol.cmas.admin.global.core.indexManagement`, method `commitAdministrativeChanges()`. You can use graphical tools like JConsole or use a command, e.g. with REST and Jolokia. If you need help with this topic, please ask your ConSol CM consultant.

If there are open tasks in the Indexer task list, the following data is displayed for each task:

- **ID**
  Task ID
- **Task type**
  Three types are available:
  - **Synchronization**
    - Recreates the whole index.
    - Triggered manually using the Admin Tool, *Synchronize index* command.
    - Before starting, all other index tasks are removed.
  - **Administrative changes**
    - Created automatically when one of the following is updated:
      scope, queue, enum value, ticket function, ticket engineer, supported locale, role.
    - Processed automatically if the *No automatic commit of administrative changes* option is unchecked.
    - The *Commit administrative changes* command will start all administrative changes tasks.

  - **Retry**
    - Created automatically when errors are encountered during the index update process.
    - Holds information about entities which caused problems.
    - The *Repair index* command will start all retry tasks.
- **Status**
  E.g., RUNNING
- **Created at**
  Time stamp when the task was created.
- **Progress**
  A progress bar that indicates the tasks progress as a percentage.
- **Details**
  A list of objects which are (re-)indexed by the task.

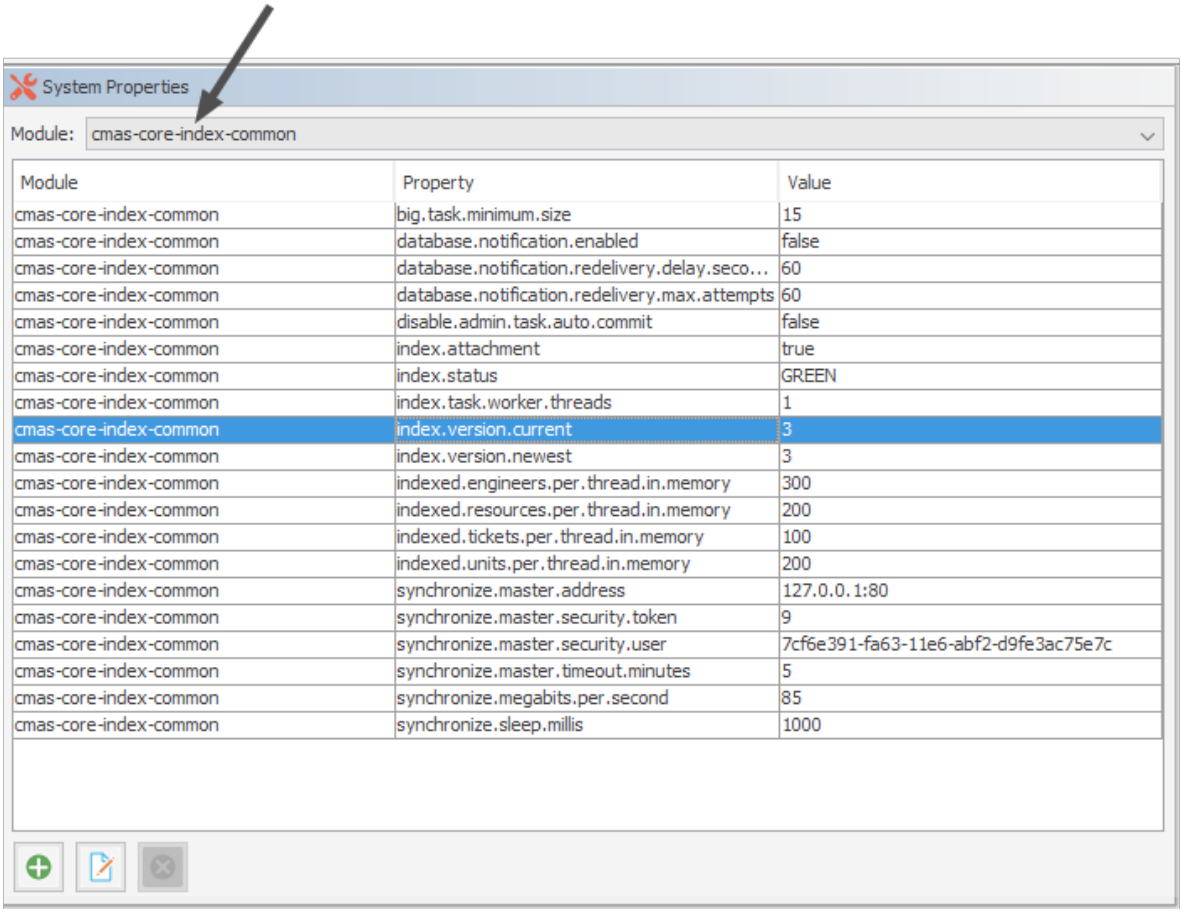
Please note that data in the index is always synchronized with the ConSol CM database, i.e. during an index update no data is deleted/removed from the index files. The index is fully usable during the synchronization process, i.e. the search operations (Detailed Search and Quick Search) can be used with their full functionality and the complete data set. Changes made after the synchronization was started are immediately reflected in the index, because these data have a higher priority than the data which is synchronized due to an index update triggered manually using the Admin Tool.

When an admin has started the index update manually (*Synchronize index*), all other index tasks are removed, before this update starts.

Indexer and Index-Relevant System Properties

The following system properties are also relevant for the Indexer, see following figure. Please refer to System Properties for a detailed explanation of the Indexer system properties and the Indexer and Search Configuration section of List of System Properties by Area.

The following figure shows the properties in the module `cmas-core-index-common`. You reach this screen by opening the navigation item *System Properties* in the navigation group *System* and selecting the module.



Module	Property	Value
cmas-core-index-common	big.task.minimum.size	15
cmas-core-index-common	database.notification.enabled	false
cmas-core-index-common	database.notification.redelivery.delay.seco...	60
cmas-core-index-common	database.notification.redelivery.max.attempts	60
cmas-core-index-common	disable.admin.task.auto.commit	false
cmas-core-index-common	index.attachment	true
cmas-core-index-common	index.status	GREEN
cmas-core-index-common	index.task.worker.threads	1
cmas-core-index-common	index.version.current	3
cmas-core-index-common	index.version.newest	3
cmas-core-index-common	indexed.engineers.per.thread.in.memory	300
cmas-core-index-common	indexed.resources.per.thread.in.memory	200
cmas-core-index-common	indexed.tickets.per.thread.in.memory	100
cmas-core-index-common	indexed.units.per.thread.in.memory	200
cmas-core-index-common	synchronize.master.address	127.0.0.1:80
cmas-core-index-common	synchronize.master.security.token	9
cmas-core-index-common	synchronize.master.security.user	7cf6e391-fa63-11e6-abf2-d9fe3ac75e7c
cmas-core-index-common	synchronize.master.timeout.minutes	5
cmas-core-index-common	synchronize.megabits.per.second	85
cmas-core-index-common	synchronize.sleep.millis	1000

Figure 405: *ConSol CM Admin Tool - System, System Properties: Indexer properties*

Indexer Services

For indexing, two ConSol CM services are important:

- **Index changes notifier**
  This service creates messages in the persistent store with notifications when changes occur that concern the index.
  Stopping *index changes notifier* is **not** safe. If the Indexer module discovers that the notifier is stopped and there is a message that has to be sent to the persistent store, the Indexer will set the index status configuration property (cmas-core-index-common, index.status) to “RED”, i.e., signal that index needs full synchronization.
- **Index changes receiver**
  The behavior of this service differs depending on the type of node it is running on:
  - On a master node, the *index changes receiver* loads tickets and creates the respective Lucene documents.
  - On a slave node (possible in a cluster only), it polls the master indexing server via http to download the Lucene files with the indexed data.

- In any case, this service reads the persistent store and starts the update of the Indexer. Stopping *index changes receiver* is safe. After restart it will pick up all of the missing changes from the persistent store.

Please see also section CM Services.

F.4.2.7 Indexer Update

This section describes the use of the indexer in a single-server environment. Please refer to the *ConSol CM Cluster Manual* for information regarding multi-server environments.

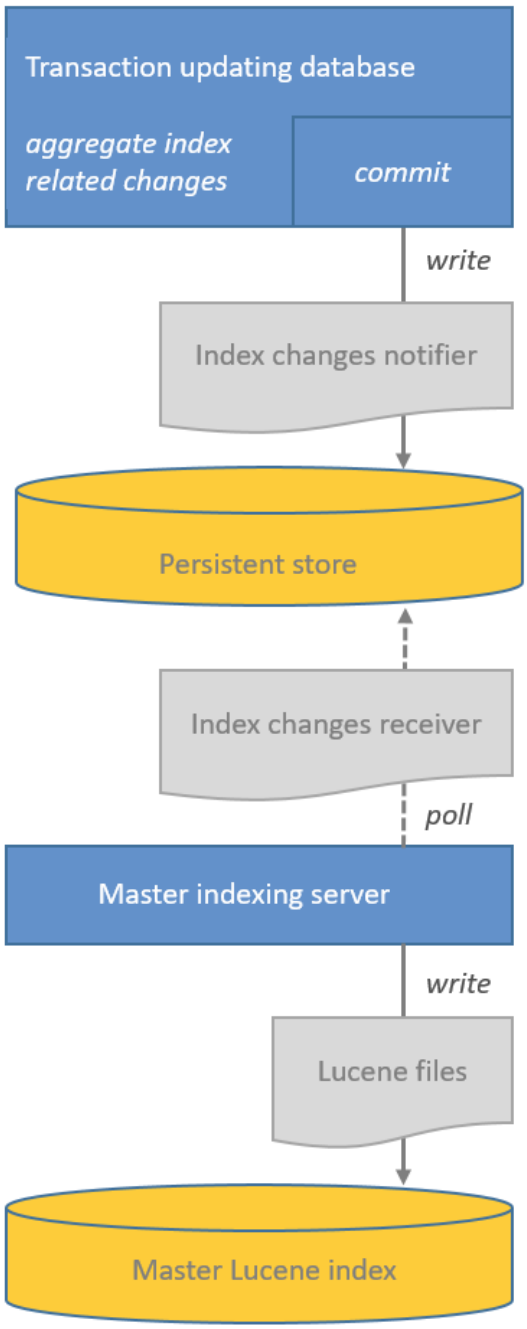


Figure 406: *Indexer architecture in a single-server environment*

Changes which Require Index Updates

There are two kind of changes relevant for the indexer:

- **Operative changes**

  These are changes to the following content:

- **Data field values:** values of fields (ticket, customer and resource fields) which are configured for indexing using the `field-indexed` annotation
- **Engineer data:** email, first name, last name
- **Ticket data:** attachments (unless configured otherwise), creation date, engineer, history, name, queue, referenced engineers, subject, view

The index is automatically updated (triggered by the service `Index changes notifier`) when these changes are carried out.

- **Administrative changes**

  These are changes of certain configuration parameters:

  - scopes
  - queues
  - enum values
  - ticket functions
  - ticket engineers
  - supported locales
  - roles
  - `field-indexed` annotation of ticket, customer or resource fields

  The processing of administrative changes depends on the option *No automatic commit of administrative changes*:

  - If it is not checked, the changes are processed automatically.
  - If it is checked, the changes are stored in the table `cmas_index_administrative_task` for deferred processing. Processing can be started by clicking the *Commit administrative changes* button in the Admin Tool.

> **For experienced administrators only:**
>
> The operation *Commit administrative changes* can also be executed using the ConSol CM MBean `consol.cmas.admin.global.core.indexManagement`, method `commitAdministrativeChanges()`. You can use graphical tools like JConsole or use a command, e.g. with REST and Jolokia. If you need help with this topic, please ask your ConSol CM consultant.

Indexer Update Modes

The indexer stores entities which require an index update in a persistent store. The persistent store can either be a JMS queue (`queue/cm6-index`) or a database table (`cmas_index_update_serialized`).

Which of the two will be used is defined by the system property cmas-core-index-common, database.notification.enabled:

- **false**
  JMS mode, the JMS queue `queue/cm6-index` will be used.
- **true**
  Database mode, the database table `cmas_index_update_serialized` will be used for indexer transactions (as persistent store). The indexing server polls this database. (Should not be used with CM versions before 6.9.4.1.)

Indexer Update Principle (Single-Server Environments)

The `Index changes receiver` on the **indexing server** polls the persistent store for new entries. If there are new entries, indexer tasks are created in the tables `cmas_index_update_task` and `cmas_index_update_part`. The indexing server then executes the tasks from `cmas_index_update_task` and `cmas_index_batch_update_task`. In this way, the index is updated.

On the indexing server, administrative tasks are stored in the database table `cmas_index_administrative_task`.

Please note that data in the index is always synchronized with the ConSol CM database, i.e. during an index update no data is deleted/removed from the index files. The index is fully usable during the synchronization process, i.e. the search operations (Detailed Search as well as Quick Search) can be used with their full functionality and the complete data set. Changes made after the synchronization was started are immediately reflected in the index, because these data have a higher priority than the data which is synchronized due to an index update triggered manually using the Admin Tool.

When an admin has started the index update manually (*Synchronize index*), all other index tasks are removed, before this update starts.

Failure of Update Task

Failed execution of the `cmas_index_update_task` will create a new task `cmas_index_update_task` with type `REPAIR`. Such task will wait for the administrator to run via the Admin Tool, navigation group *Services*, navigation item *Index -> Repair index*. Repair task existence will set the “YELLOW” index status configuration property.

F.4.3 Action Framework - Search Actions

F.4.3.1 Introduction to Search Actions

It is possible to define actions for search results. These actions are presented like workflow activities for result lists of

- tickets
- customer (= units)
- resources

Search (result) actions operate on the result set of a search. You can configure search (result) actions to offer different bulk operations to the ConSol CM engineers, e.g., assigning all tickets in the result list to the current engineer (see the following figure), sending a specific email to all customers in the result list, or creating a new maintenance ticket for all PCs (resources) in the result list.

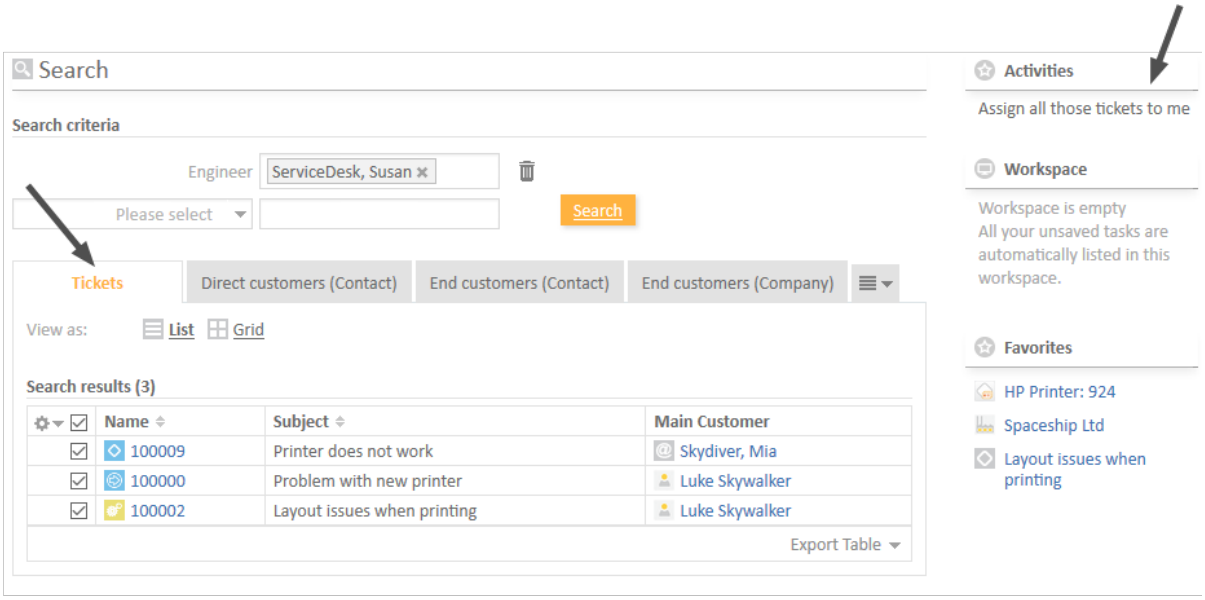


Figure 407: *ConSol CM Web Client - Search actions for tickets*

> A search (result) action will be displayed in the Web Client only if
>
> - the engineer has ACT permissions for all elements of the result set (if any)
> - the condition script (if any) has returned “true”

Each search action is implemented based on Admin Tool scripts of the following types.

- **Used for tickets:**
  - Search action for tickets
  - Search condition for tickets

- **Used for resources:**
  - Search action for resources
  - Search condition for resources
- **Used for customers:**
  - Search action for customers
  - Search condition for customers

Please note that the search action can be performed on

a) only the part of the result list which is displayed, e.g., on only 20 of 100 hits

OR

b) on the entire result list, e.g., on all 100 hits

The behavior of the search action depends on

- the implementation of the Admin Tool script. Please see the detailed explanation in section The Result Set in Search Action Admin Tool Scripts.
- the selection of rows. See Page Customization, enableRowSelection.

F.4.3.2 Configuring Search Actions Using the Admin Tool

Search actions are defined in the Admin Tool. You have to perform two or three steps to implement a search action, depending on the type of the search action.

Steps to Perform to Implement a Search Action for Tickets

1. Define and write the Admin Tool script (navigation group *System*, navigation item *Scripts and Templates*)
   - mandatory: a script of type *search action for tickets* (see following figure)
   - optional: a script of type *search condition for tickets*

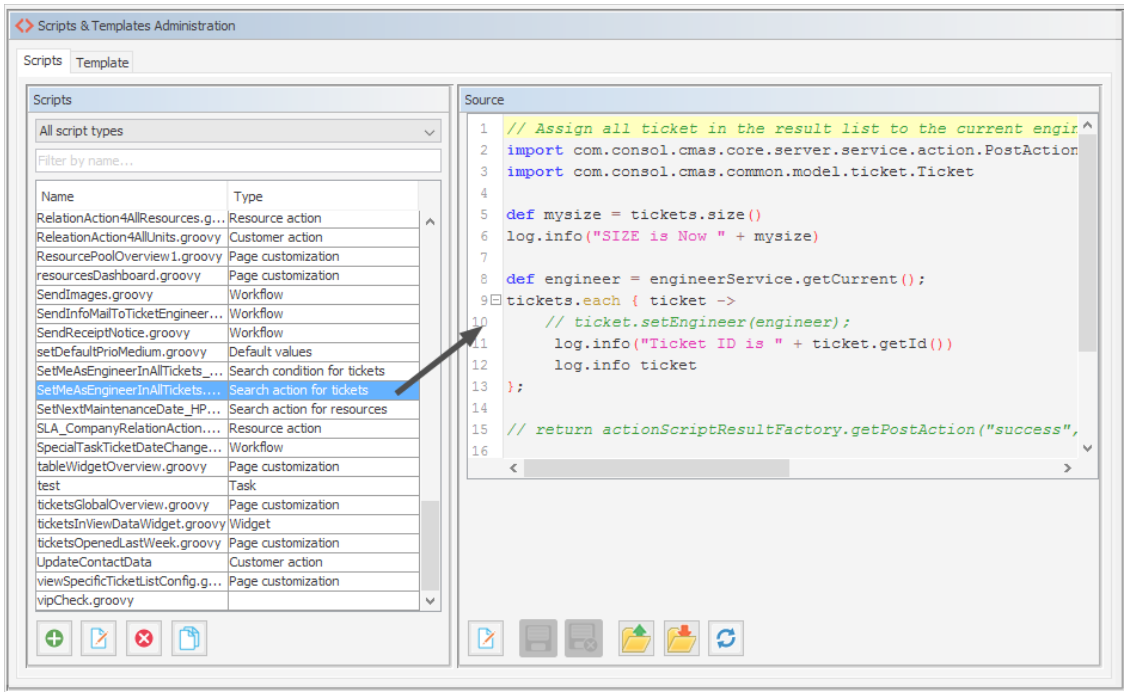


Figure 408: *ConSol CM Admin Tool - System, Scripts and Templates: Script for a search action for tickets*

Example script:

```
// Assign all tickets in the result list to the current engineer
import com.consol.cmas.core.server.service.action.PostActionType
import com.consol.cmas.common.model.ticket.Ticket

def engineer = engineerService.getCurrent();
tickets.each { ticket ->
   ticket.setEngineer(engineer);
};
return actionScriptResultFactory.getPostAction("success",
 "cmweb.search.assigned").withRefreshContent();
```

Code example 61: *Search action script for tickets*

> For a detailed explanation about how to write scripts for the ConSol CM Action Framework, please read section Scripts for the Action Framework.

2. Define the search action in the navigation item *Search Actions*, navigation group *Tickets*. During this step, the action script is assigned to the search action. A condition script might also be assigned, but this is optional. You can add the localized term for the Ticket Action using the *Localize* button. For a detailed explanation of the localization mechanism, please refer to

section Localization of Objects in General, Type 1.

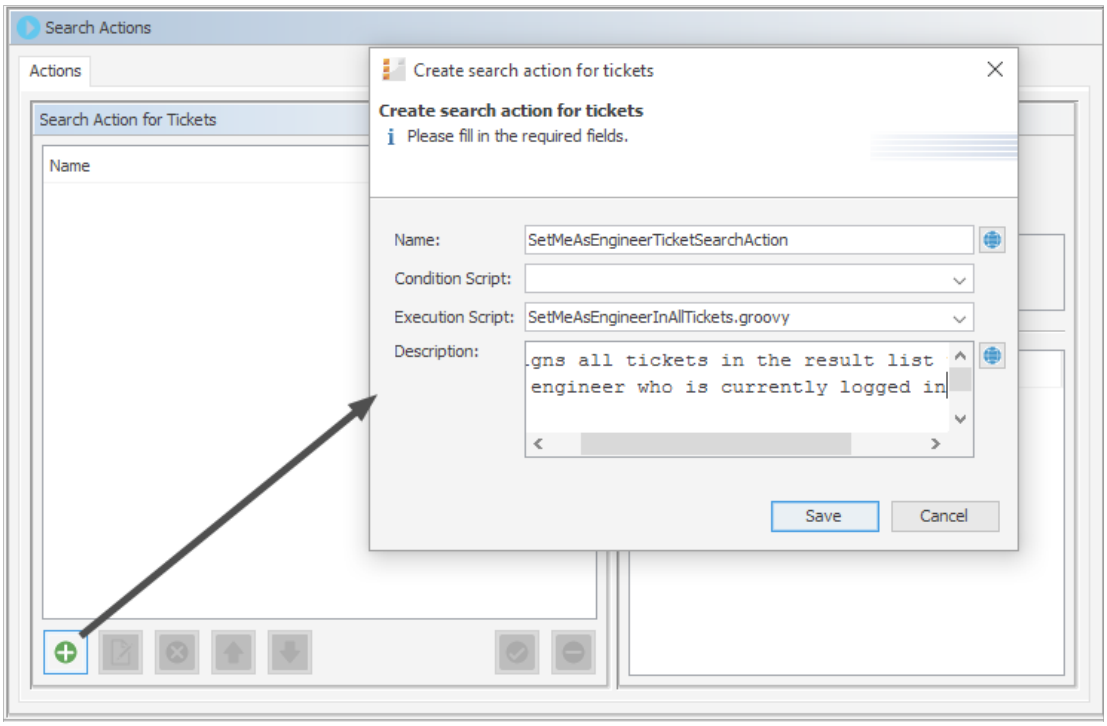


Figure 409: *ConSol CM Admin Tool - Tickets, Search Actions: Defining a new ticket search action*

Steps to Perform to Implement a Search Action for Resources

1. Define and write the Admin Tool script
   - mandatory: a script of type *search action for resources*
   - optional: a script of type *search condition for resources*

   Example script which sets the next maintenance date of all retrieved HP printers to a date in two weeks:

```
// Schedule maintenance date for all selected HP printers to a date in two
 weeks from today

import groovy.time.TimeCategory
import com.consol.cmas.core.server.service.action.PostActionType

println 'RUNNING Search action resources script!'

def now = new Date()

use(TimeCategory) {
   nextMaintenanceDate = now + 2.weeks
}

resources.each { res ->
   res.setFieldValue("HP_Printer_Fields_basic", "NextMaintenanceDate",
    nextMaintenanceDate)
}

return actionScriptResultFactory.getPostAction(PostActionType.SUCCESS,
 "cmweb.search.assigned").withRefreshContent();
```

Code example 62: *Search action script for resources*

> For a detailed explanation about how to write scripts for the ConSol CM Action Framework, please read section Scripts for the Action Framework.

2. Define the search action in the navigation item *Actions*, navigation group *Resources*. During this step, the action script is assigned to the search action. A condition script might also be assigned, but this is optional. You can add the localized term for the resource action using the *Localize* button. For a detailed explanation of the localization mechanism, please refer to section Localization of Objects in General, Type 1.

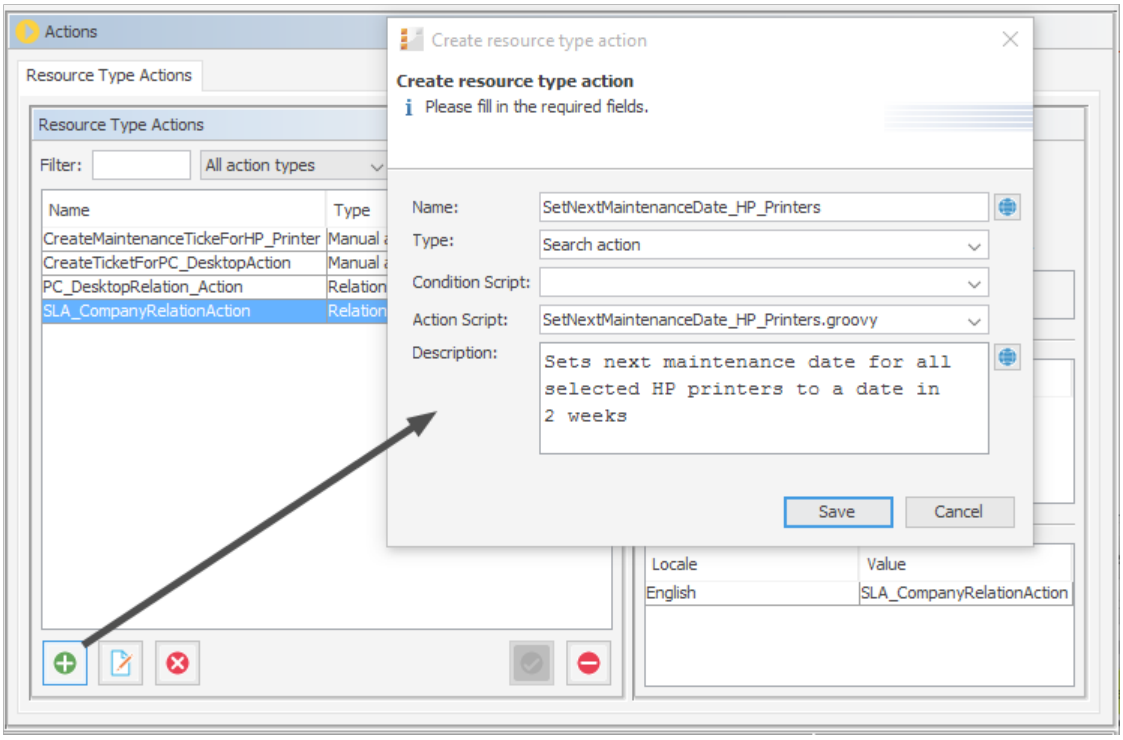


Figure 410: *ConSol CM Admin Tool - Resources, Actions: Defining a Resource Search (Result) Action*

3. Assign the action to one or more resource type(s), navigation group *Resources*, navigation item *Data Models*, tab *Search Actions* for each resource type.

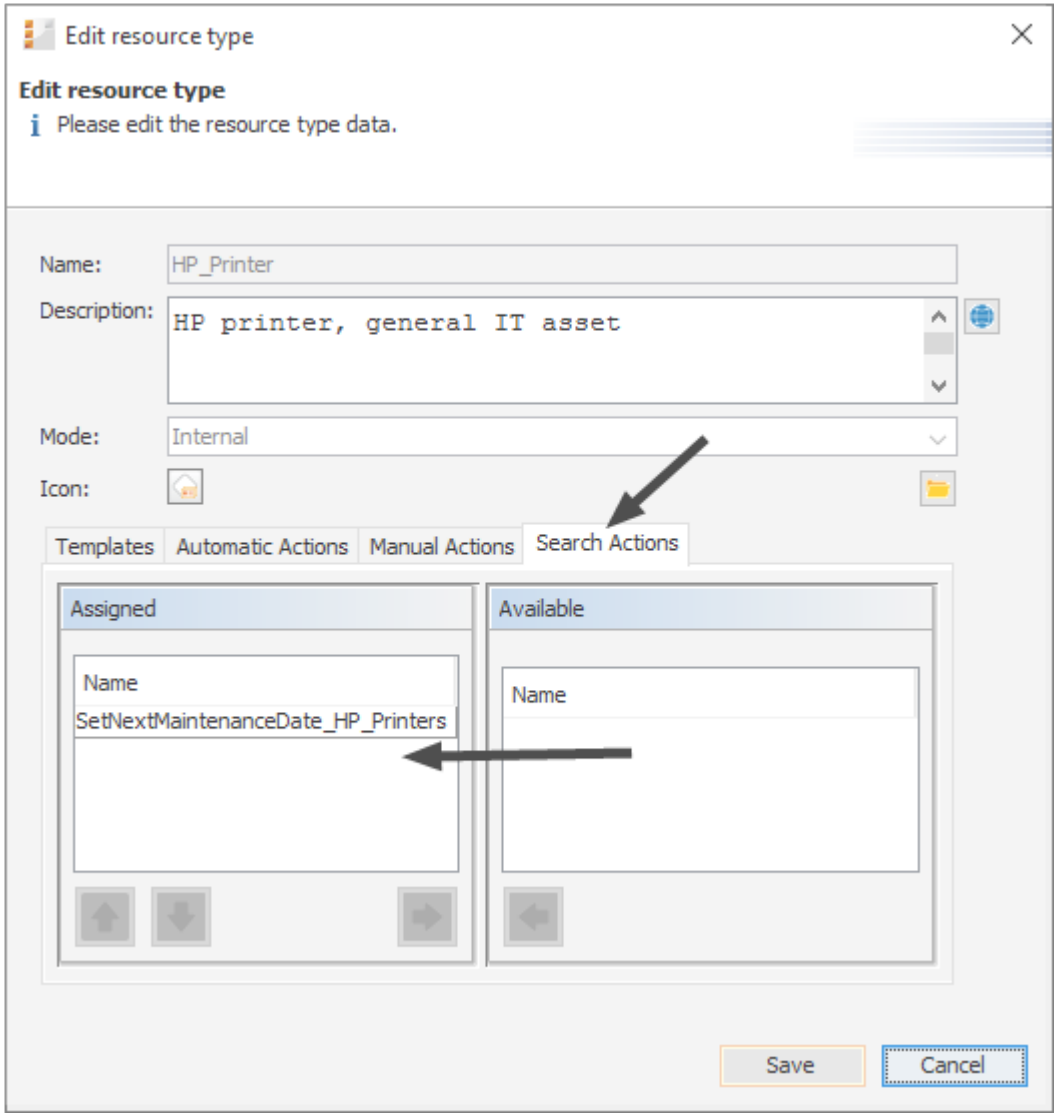


Figure 411: *ConSol CM Admin Tool - Resources, Actions: Assigning a search action to a resource type*

4. Ensure that it works in the Web Client.

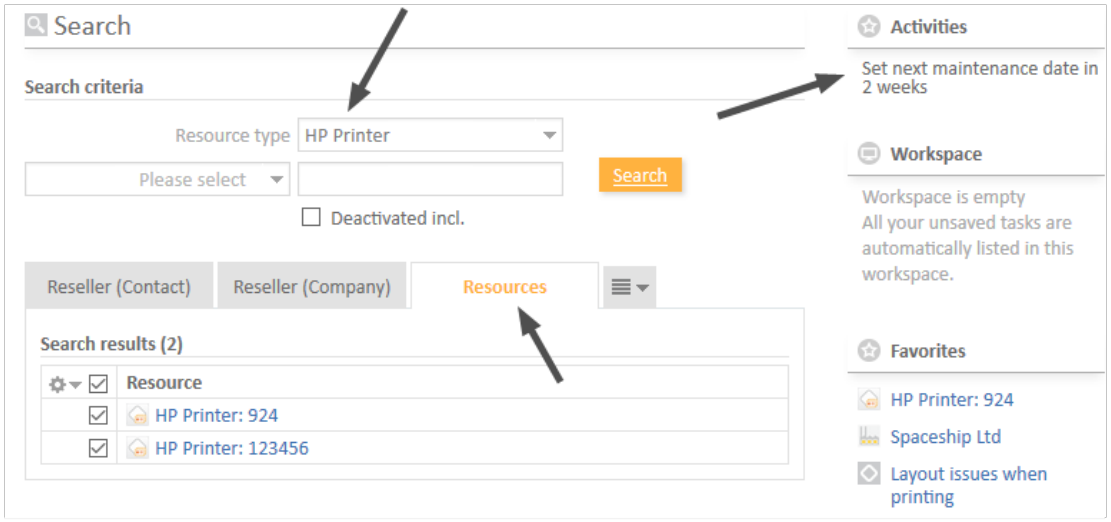


Figure 412: *ConSol CM Web Client - Search (Result) Action on resources, 1*

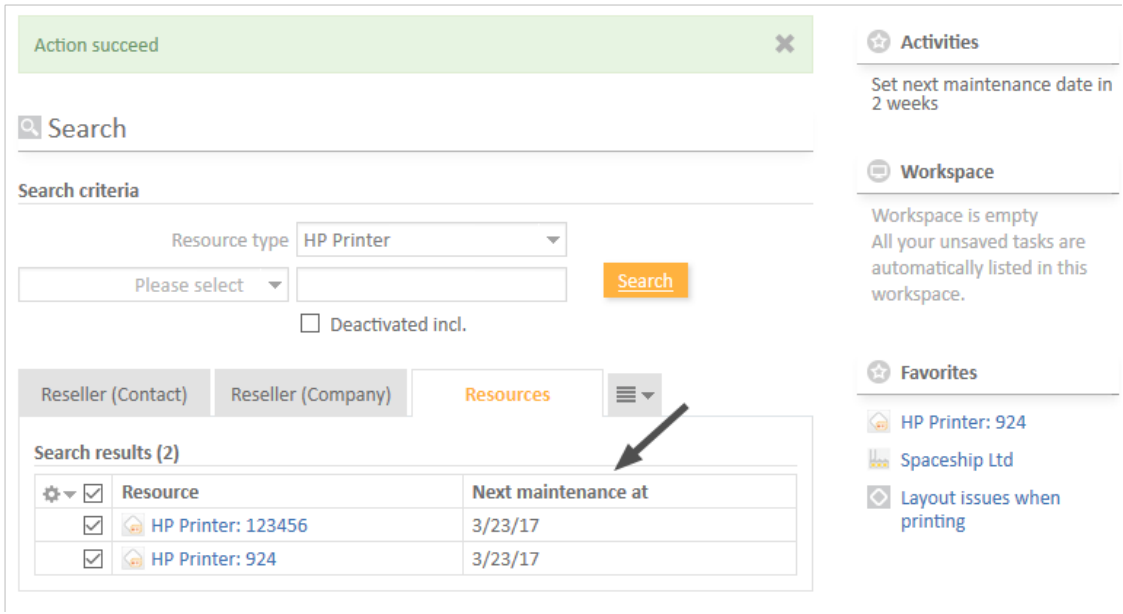


Figure 413: *ConSol CM Web Client - Search (Result) Action on resources, 2*

Steps to Perform to Implement a Search Action for Customers (= Unit)

1. Define and write the Admin Tool script
   - mandatory: a script of type *search action for customers*
   - optional: a script of type *search condition for customers*, in this example we will work with a condition script

   Example action script which creates a ticket for each contact in the result list:

```
// For all contacts in the result set, a new ticket in the queue Service Desk
 will be created
import com.consol.cmas.core.server.service.action.PostActionType
import com.consol.cmas.common.model.ticket.Ticket
import com.consol.cmas.common.model.customfield.Unit
import com.consol.cmas.common.model.resource.*
import com.consol.cmas.common.service.resource.*
import com.consol.cmas.common.model.ticket.Queue
import com.consol.cmas.common.model.resource.meta.*
import com.consol.cmas.core.server.service.action.*
import groovy.time.TimeCategory

println 'Search Result Action Script CreateAnSDTicketForAllSelectedContacts
 STARTED ... '
// deadline is a required field in Service Desk tickets!
def now = new Date()
def deadline
use(TimeCategory) {
   deadline = now + 2.weeks
}

Queue qu = queueService.getByName("ServiceDesk")
units.each{ cont ->
   def cont_name = cont.customer_name
   println 'Customer name is now : ' + cont_name
   Ticket newtic = new Ticket()
   newtic.setQueue(qu)
   newtic.set("serviceDesk_fields.desiredDeadline", deadline)
   newtic.setSubject("New Ticket due to Search Result for customer" + cont_
    name)
   newtic.set("helpdesk_standard.priority","low")
   ticketService.createWithUnit(newtic,cont)
   println 'New Ticket created for customer ' + cont_name
}
return actionScriptResultFactory.getPostAction("success",
 "cmweb.search.assigned").withRefreshContent();
```

Code example 63: *Search action script for units*

Please make sure that

- you only use customer fields in the script which are present in the customer data model of the customer group where the Search (Result) Action script will be used! For example: To reference the name of a contact, the exact field name has to be used (*customer_name* in the example above)! Hence, before implementing a Search (Result) Action script, it is recommended that you check the customer data model for the required fields.
- the customer group has been assigned to the queue where new tickets should be created.

Example condition script which causes the Search (Result) Action to only be displayed if the list has more than five entries.

```
if (units.size() >= 5){
   return true
} else {
   return false
}
```

Code example 64: *Search condition script for units*

For a detailed explanation about how to write scripts for the ConSol CM Action Framework, please read section Scripts for the Action Framework.

2. Define the search action in the navigation item *Actions*, navigation group *Customers*. During this step, the action script is assigned to the search action. A condition script might also be assigned, but this is optional. You can add the localized term for the customer action using the *Localize* button. For a detailed explanation of the localization mechanism, please refer to section Localization of Objects in General, Type 1.

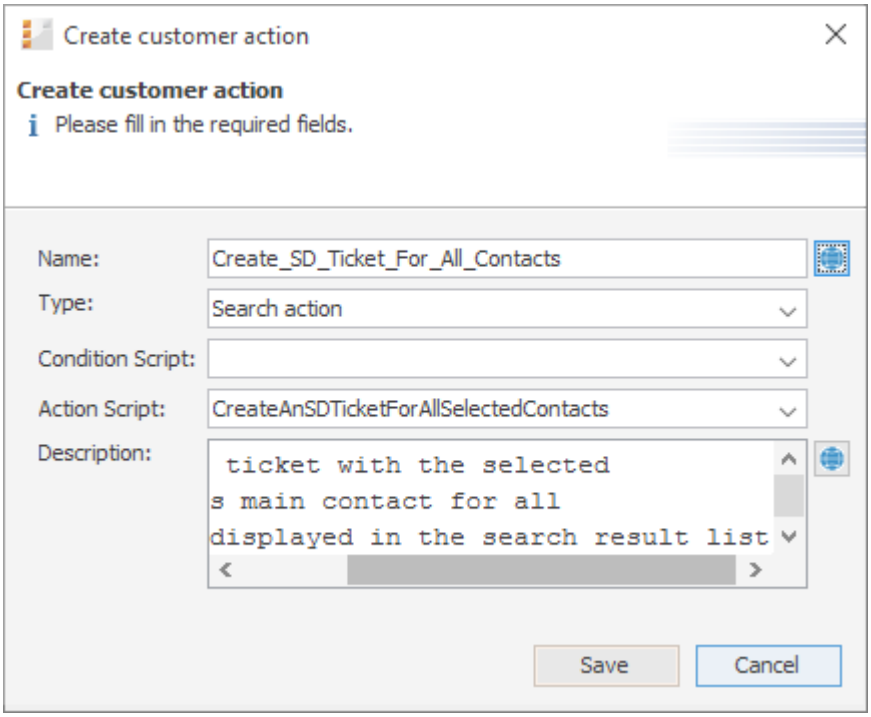


Figure 414: *ConSol CM Admin Tool - Customers, Actions: Defining a Unit Search (Result) Action*

3. Add the condition script.

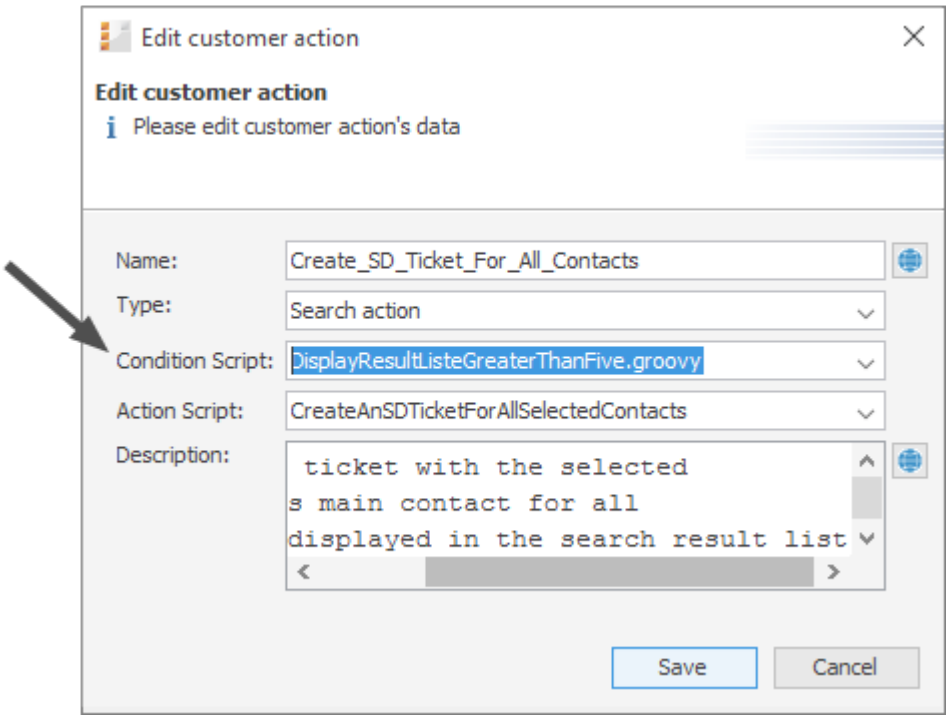


Figure 415: *ConSol CM Admin Tool - Customers, Actions: Defining a Search Script Condition*

4. Assign the action to one or more customer groups, navigation group *Customers*, navigation item *Customer Groups*, tab *Search Actions* for each customer group. Assign the action(s) to the contacts (*Contact Search Actions*) and/or the company/companies (*Company Search Actions*) of this customer group.
   Note that the action and condition scripts are listed together in one table.

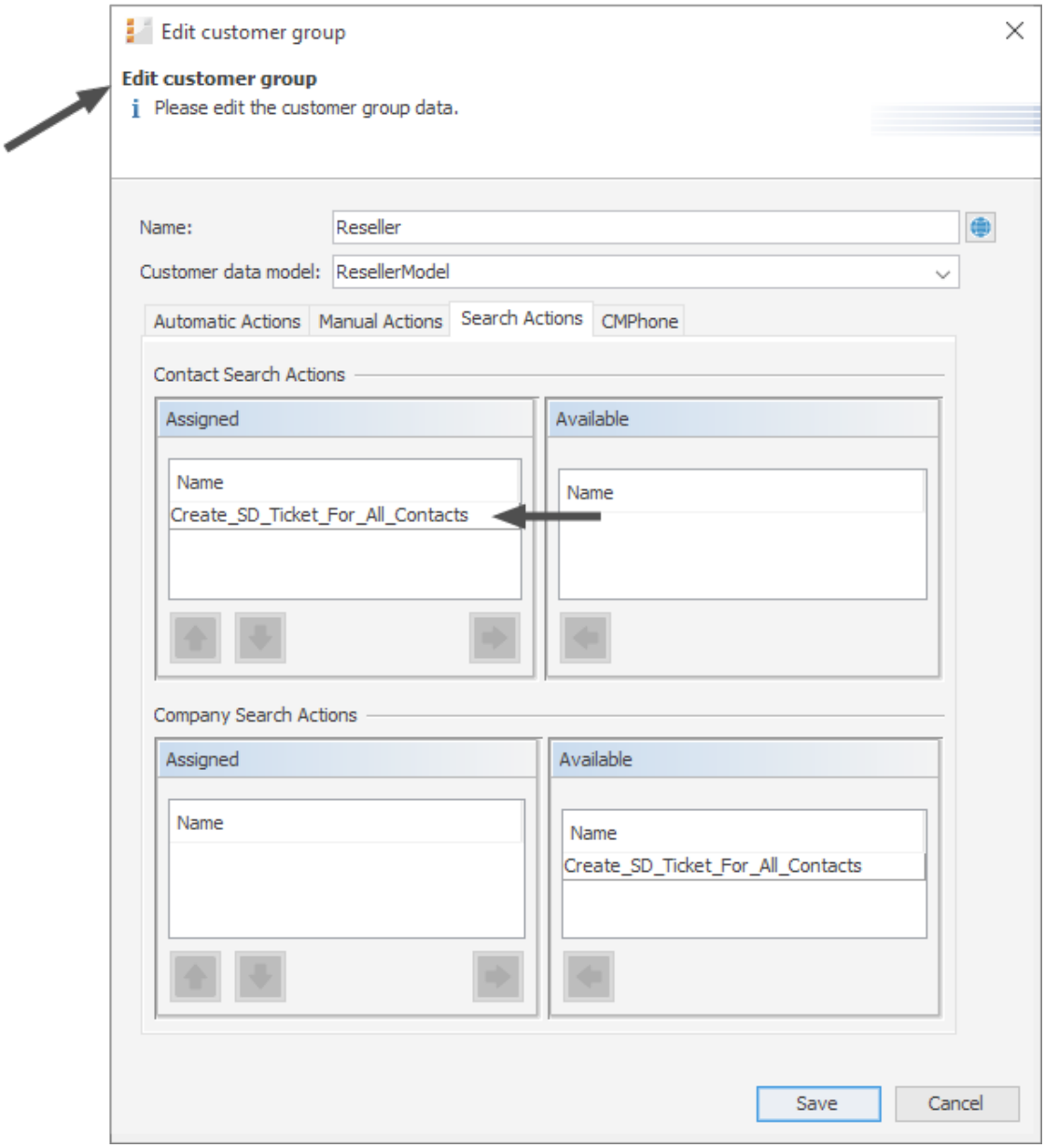


Figure 416: *ConSol CM Admin Tool - Customers, Customer Groups: Assigning a search action to a contact object within a customer group*

5. Ensure that it works in the Web Client.

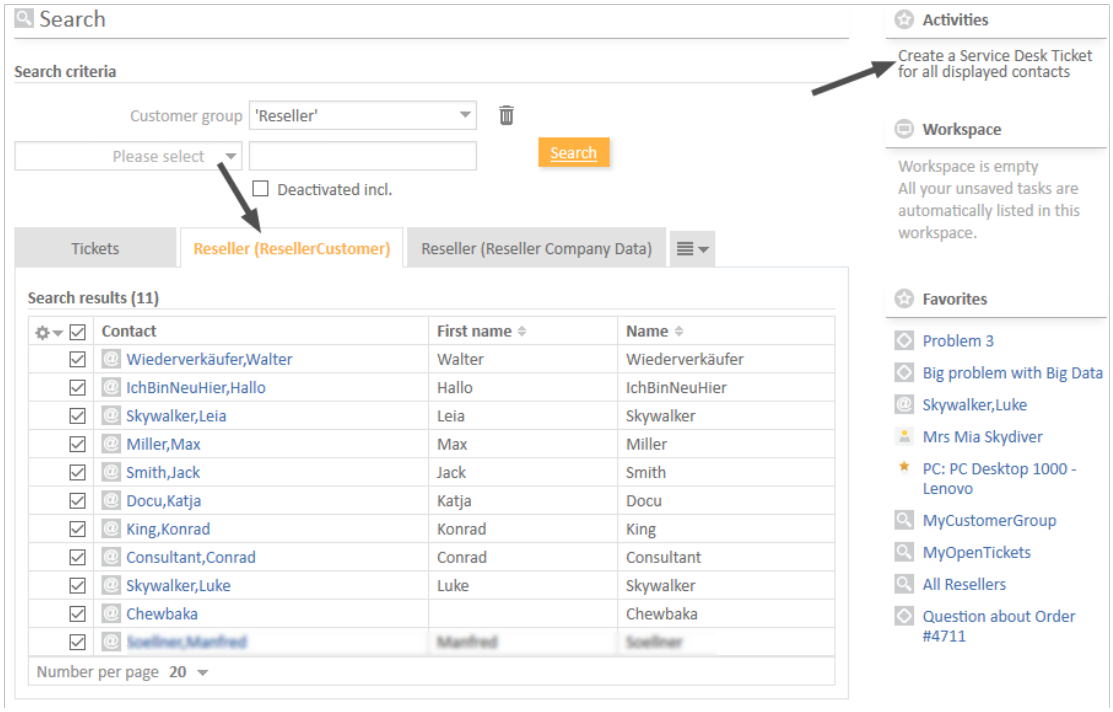


Figure 417: *ConSol CM Web Client - Search (Result) Action on units (here: contacts), 1*

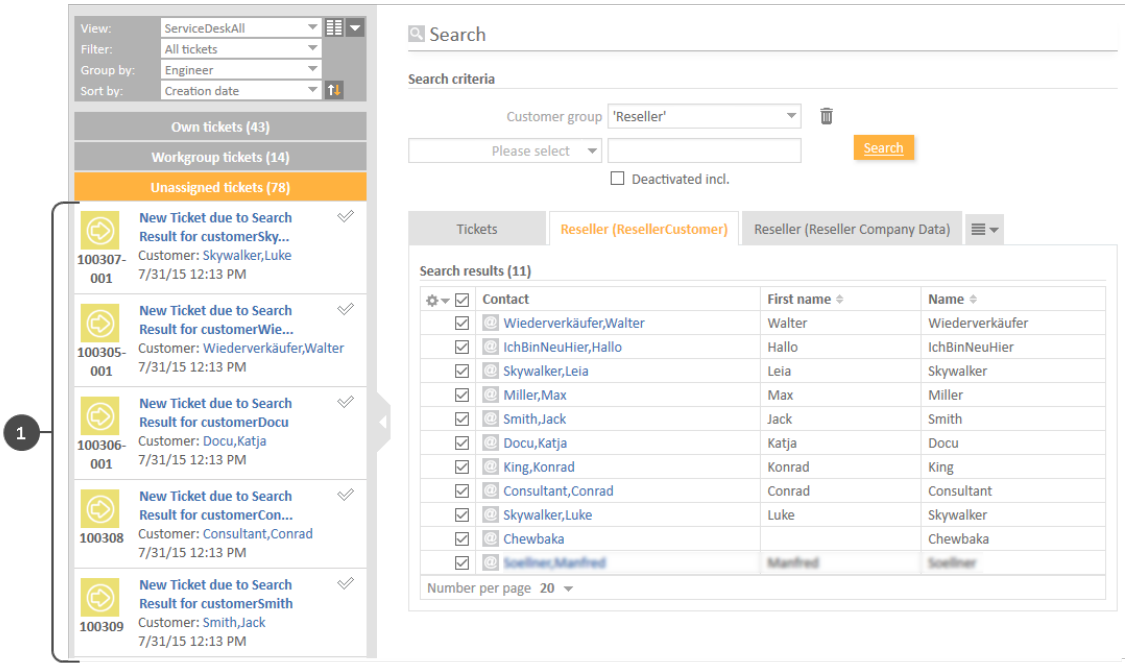


Figure 418: *ConSol CM Web Client - Search (Result) Action on units (here: contacts), 2*

The ticket list shows the new ServiceDesk tickets (1) which have been created in the Search Action script.

The following figure shows that the activity is not offered if the result list contains less than five entries (as defined in the condition script).

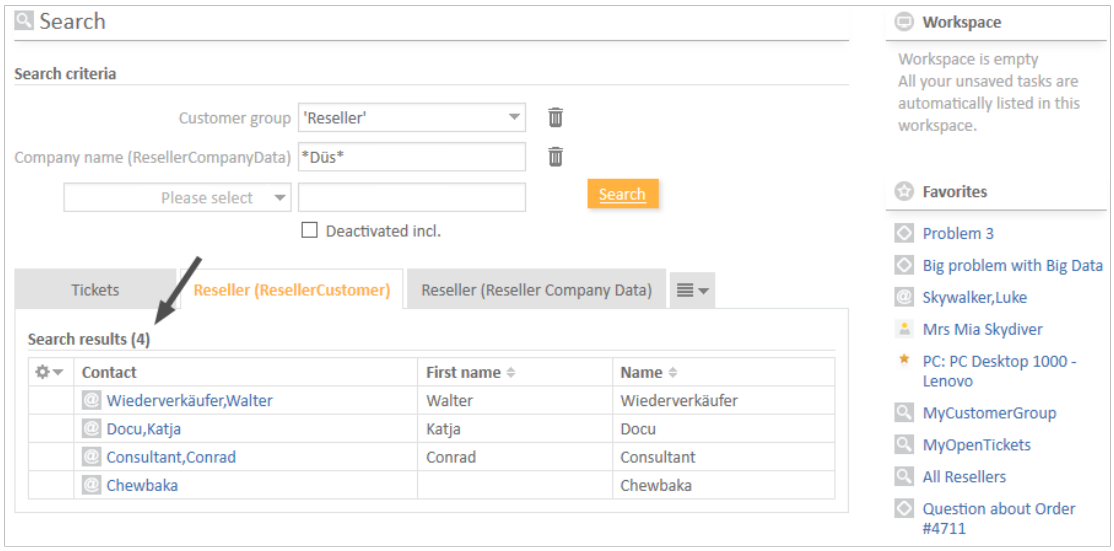


Figure 419: *ConSol CM Web Client - Search (Result) Action on units (here: contacts), 3*

F.4.3.3 Tips and Tricks for Search Action Admin Tool Scripts

Objects Implicitly Available for Search Action Admin Tool Scripts

Some objects are implicitly available in the Admin Tool scripts, i.e., you do not have to import specific Groovy classes, nor do you have to instantiate such objects from a class.

Objects which are implicitly (without an explicit import or instantiation) available in Admin Tool scripts:

All search action scripts

- pageSize (Integer)
- pageNumber (Integer)

Script types *search action for tickets* and *search condition for tickets*

- criteria (TicketCriteria)
- tickets (List<Ticket> for ticket search action)

Script types *search action for resources* and *search condition for resources*

- criteria (ResourceCriteria)
- resources (List<Resource> for resource search action)

Script types *search action for customers* and *search conditions for customers*

- criteria (UnitCriteria)
- units (List<Unit> for unit search action)

The Result Set in Search Action Admin Tool Scripts

The search action can be performed on different interpretations of the result set. The behavior of the search action depends on the implementation of the Admin Tool script.

The search action can be performed on

a) only the part of the result list which is displayed in the Web Client, e.g., on only 20 of 100 hits. So this depends on the paging the engineer has selected and on the rows the engineer has selected if the row selection is switched on (see section Page Customization, enableRowSelection for details).

In the Admin Tool script, this is represented by the list of objects which is implicitly available, i.e., `resources`, `tickets` or `units`.

OR

b) on the entire result list, e.g., on all 100 hits. This can be achieved by using the `criteria` object.

The Return Code in Search Action Admin Tool Scripts

Use the general feedback component to handle the two new `PostActionTypes`: SUCCESS and FAILURE.

- SUCCESS - green
- FAILURE - red

Example for a positive feedback:

```
return actionScriptResultFactory.getPostAction(PostActionType.SUCCESS,
 "cmweb.search.assigned").withRefreshContent();
```

Code example 65: *Positive feedback*

For a detailed explanation about how to write scripts for the ConSol CM Action Framework, please read section Scripts for the Action Framework.

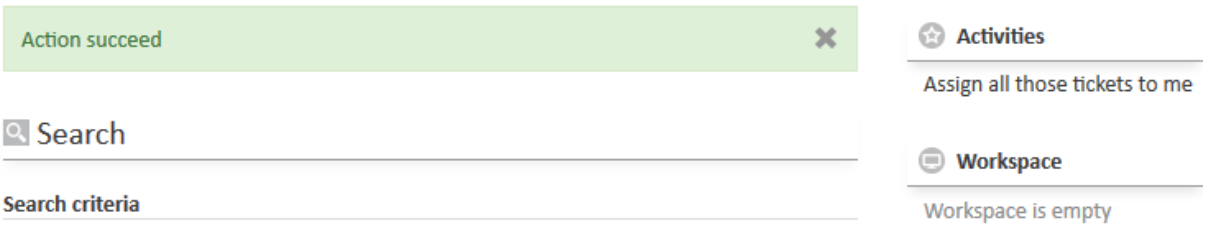


Figure 420: *ConSol CM Web Client - SUCCESS message after Search (Result) Action*

Example of a negative feedback:

```
return actionScriptResultFactory.getPostAction(PostActionType.FAILURE,
 "cmweb.search.assigned").withRefreshContent();
```

Code example 66: *Negative feedback*

For a detailed explanation about how to write scripts for the ConSol CM Action Framework, please read section Scripts for the Action Framework.

Figure 421: *ConSol CM Web Client - FAILURE message after Search (Result) Action*

F.4.4 CSV Export of Search Results

F.4.4.1 Introduction

Starting with ConSol CM version 6.10.1, search result lists can be exported in CSV format. This functionality is not activated by default but may be activated using page customization.

If CSV exporting is enabled, an engineer can use the *Export table* option which is offered on search result pages for lists of

- tickets
- contacts
- companies
- resources

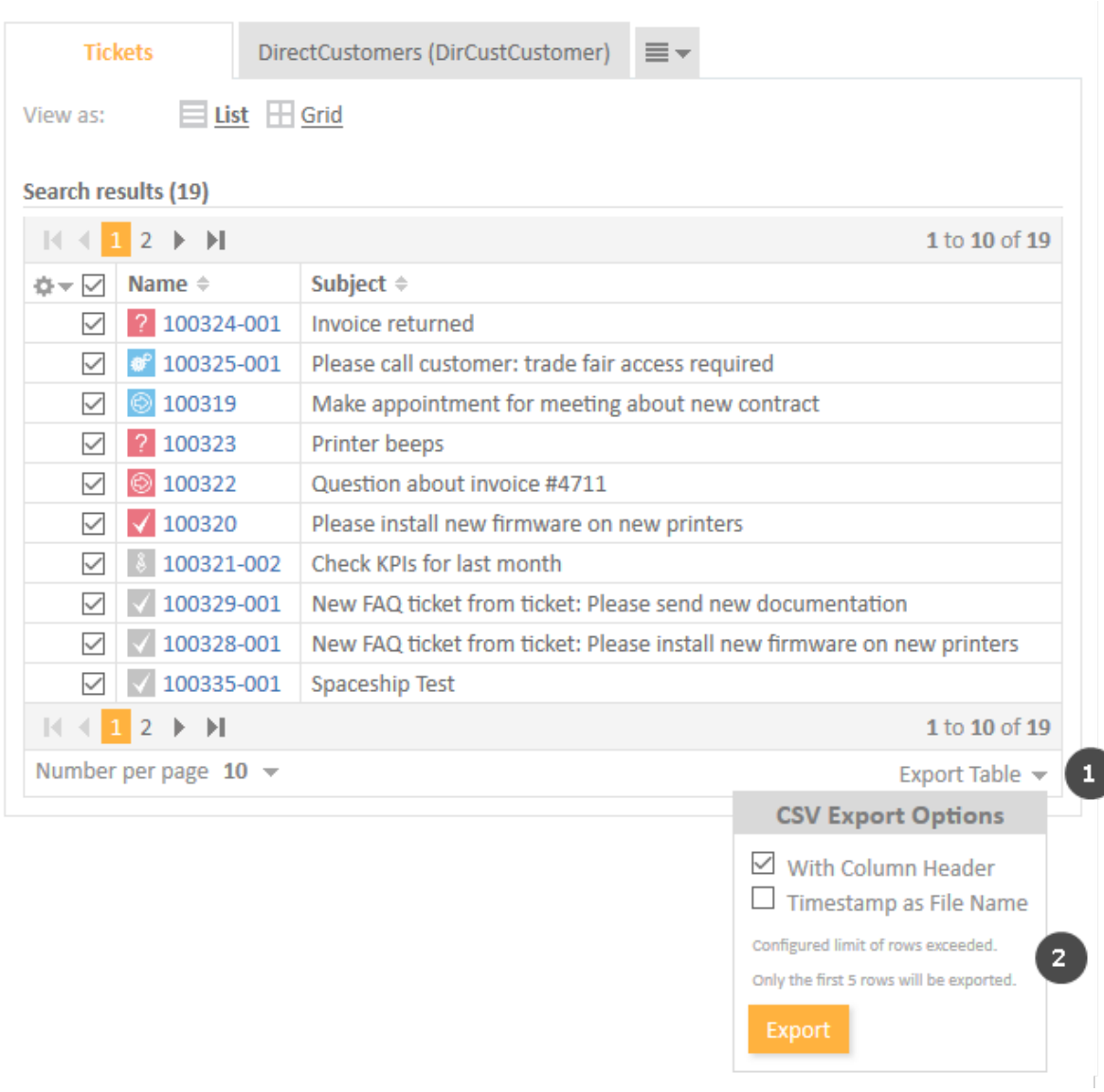


Figure 422: *ConSol CM Web Client - CSV export option for a ticket search result list*

- CSV export activated using the Page Customization attribute *enabled* (1)
- CSV export sub menu (2). In the example, the export would contain more rows than configured using the Page Customization attribute *exportRowsLimit.*

Available CSV export options in the Web Client:

- **With column header**
  This adds the column labels of the result table in the first line, as a content description.
- **Timestamp as File Name**
  If this is selected, the default file name will be the current date and time instead of the standard name “export.csv”.

Clicking on *Export* will automatically offer the option to open the newly created CSV file with the standard application configured to open CSV files on the client machine. In the CSV file there will be no pagination - the complete result set is included (possibly limited by the *exportRowsLimit* configuration option, as described below). Icons in the result display are not included in the CSV. Ticket field string values will be quoted by standard double quotes (ASCII value 34 decimal resp. 0x22 hex).

F.4.4.2 Activating the CSV Export Functionality

The CSV export functionality has to be activated selectively for distinct search result pages and is activated via page customization.

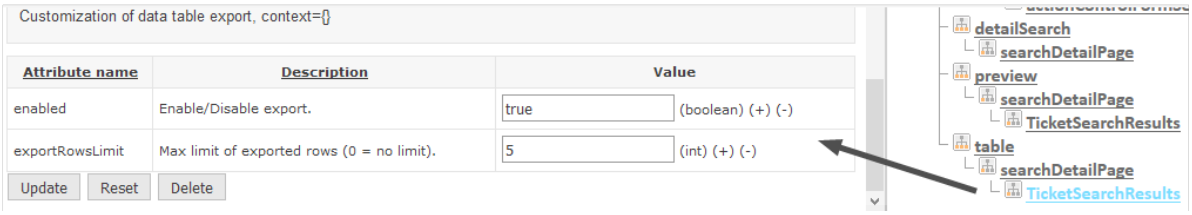


Figure 423: *ConSol CM Web Client - Page customization to activate CSV export for ticket search result lists*

Use the following sub-scopes:

- **for ticket search results** (see figure above)
  table/searchDetailPage/TicketSearchResults
- **for contact search results**
  table/searchDetailPage/<Customer Group name>
  This has to be defined for each customer group!
- **for company search results**
  table/searchDetailPage/<Customer Group name>
  This has to be defined for each customer group!
- **for resource search results**
  table/searchDetailPage/ResourceSearchResults

> Please note that you have to execute a search first to present a result table. Until a search result table has been produced, the *table* scope will not be available in the page customization tree!

The following attributes are available:

- **enabled**
  Enable/Disable the CSV export functionality for this search result page, Boolean. Default is “false”.
- **exportRowsLimit**
  Maximum number of rows to export in the CSV export. 0 means no limit. Default value is “0”.

F.5 The Task Execution Framework (TEF)

This chapter discusses the following:

F.5.1 Introduction

Using the ***Task Execution Framework*** (TEF), ConSol CM can perform various tasks which are not directly tied to or embedded in another script (like a workflow script, unit action, resource action, search action or another type of Admin Tool script) and which can be executed asynchronously. This can be used, e.g., for long-term system tasks which might cause a timeout when started within a regular ConSol CM script. The TEF tasks can be executed in an asynchronous manner. An API extension has been added to ConSol CM (in version 6.9.4.0) in order to provide the TEF functionality. TEF scripts can be started (i.e., the TEF API is available in):

- manually in the Admin Tool
- from workflow activity scripts
- from email scripts
- from action scripts (search actions, unit actions, resource actions)

A task is stored as Admin Tool script of type *Task* (see Admin Tool Scripts). This script type is explained in the respective section below.

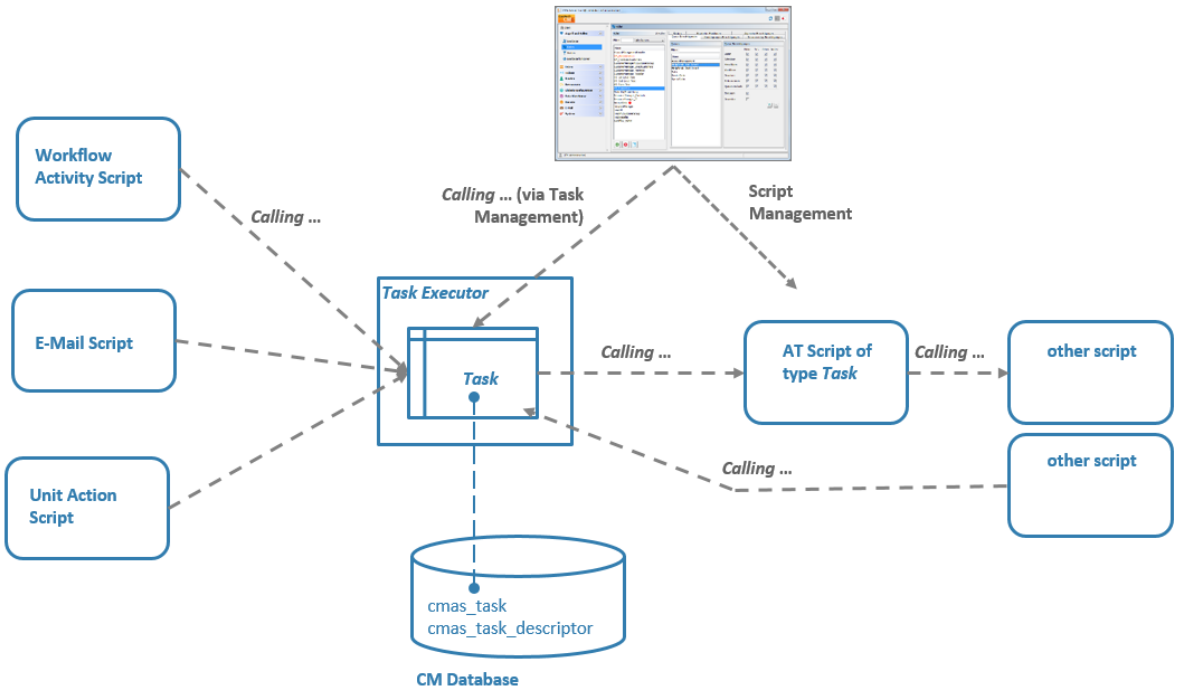


Figure 424: *ConSol CM Task Execution Framework*

The **Task Executor** is a ConSol CM module (a singleton with watchdog functionalities) which controls the execution of the tasks. The Task Executor scans the database for new scheduled tasks and uses its thread pool for the execution of the tasks which have to be executed at a certain point in time.

The **task definition** is stored in an Admin Tool script. Thus, for one task definition, usually one Admin Tool script is used.

A (scheduled) task, i.e., one single run of the task, can be started ...

- using the Admin Tool, navigation item *Task execution*, see section Execution of a Task Using the Admin Tool. Here, a task can be started immediately, it cannot be scheduled for another point in time.
- implementing the calling of the task script in another script of one of the following types. Here, a script can be executed immediately or can be scheduled for a later point in time.
    - workflow activity scripts
    - email scripts
    - action scripts (as part of the Action Framework, i.e., in unit action scripts, resource action scripts or search action scripts)

In scripts, for every execution of a task, a **task descriptor**, i.e., an object of the class `TaskDescriptor`, is available. This task descriptor provides information like the task's progress.

The start date of a task can be set using the **task execution specification** (object of the class `ExecutionSpecification`).

Please see section Programming with Tasks for programming details.

F.5.2 Admin Tool Scripts of Type *Task*

F.5.2.1 Introduction to Admin Tool Scripts of Type *Task*

Every Admin Tool script of type *Task* has to implement the following methods. The method signatures are inserted automatically when a script of this type is created.

```
def onInitialize(taskDescriptor) {}
def onExecute(taskDescriptor) {}
def onError(taskDescriptor) {}
def onCancel(taskDescriptor) {}
```

F.5.2.2 Example Admin Tool Script of Type *Task*

```
//Test
def onInitialize(taskDescriptor) {
   log.info("MyFirstTaskScript has been initialized!")
}

def onExecute(taskDescriptor) {
   log.info("MyFirstTaskScript is executed")
   try {
      Thread.Sleep(300000)
   } catch (Exception ex) {
      log.info("ztztzt ...")
   }
}

def onError(taskDescriptor) {
   log.info("MyFirstTaskScript has thrown an error!")
}

def onCancel(taskDescriptor) {
   log.info("MyFirstTaskScript has been cancelled!")
}
```

Code example 67: *Admin Tool Script of Type Task*

F.5.2.3 Execution of a Task Using the Admin Tool

In the Admin Tool, navigation group *Services*, navigation item *Task Execution*, you can start tasks which have been defined as Admin Tool scripts before.

To be able to execute tasks, the system property cmas-app-admin-tool, start.-groovy.task.enabled has to be set to the value “true”. This property is not present in a default installation and has to be added manually.

Please note that if the system property cmas-app-admin-tool, start.groovy.task.enabled is set to “true”, and thus if you, as an administrator, can execute tasks using the Admin Tool, you have to be absolutely sure what the task will be doing! Be aware of the risks involved with tasks which, for example, delete customer data or tickets and should only be executed via a workflow or Admin Tool script!

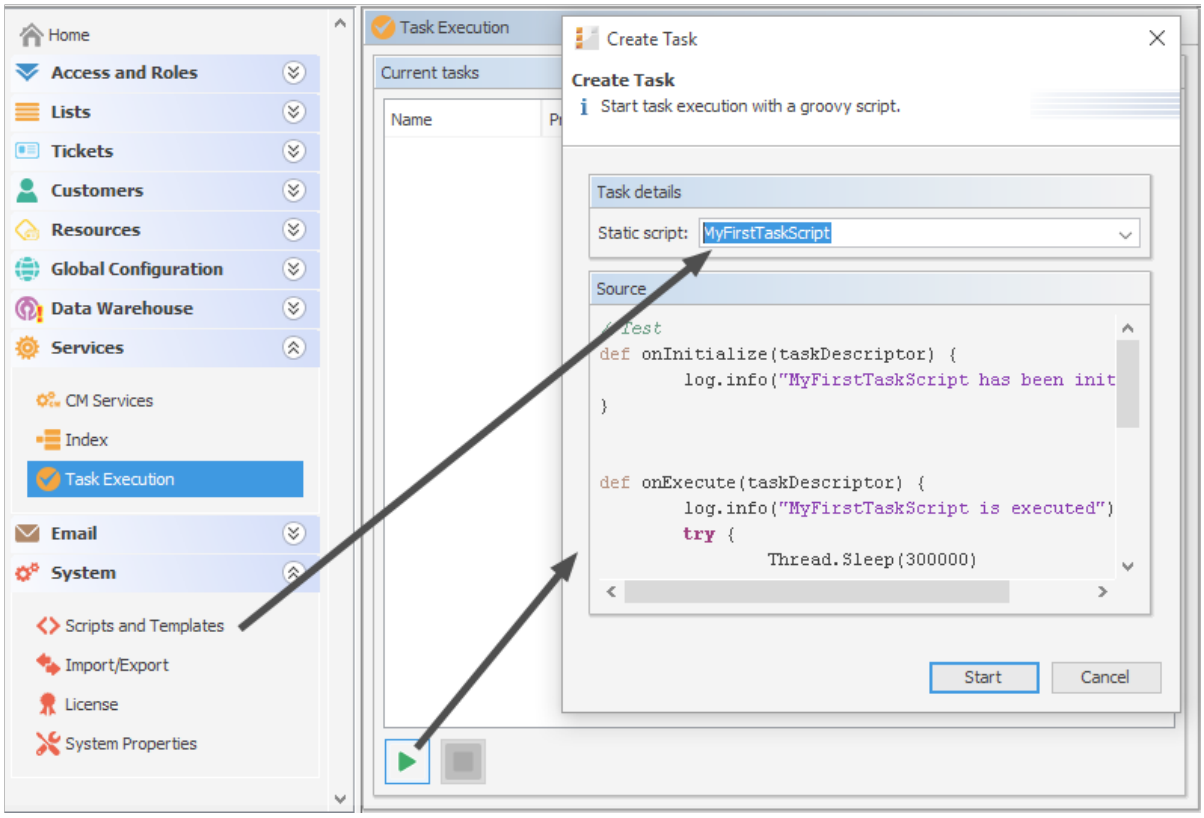


Figure 425: *ConSol CM Admin Tool - Services, Task Execution: Creating a task*

To start a script, click on the *Start* button and select the name of the Admin Tool script from the drop-down menu *Static script*. All Admin Tool scripts of type *Task* will be listed here. Click on *Start* to execute the script immediately. It is not possible to schedule a task using the Admin Tool GUI. If the start should be delayed, this has to be implemented within the script, see Defining the (First) Execution Date.

When a task is running, a progress bar is shown. You can stop (cancel) the task using the *Stop* button.

The update interval of the Task list is defined using the CM system property cmas-app-admin-tool, task.panel.refresh.interval.seconds. The property is not present by default but has to be added manually.

The following figure shows a task being executed. You reach this screen by opening the navigation item *Task Execution* in the navigation group *Services*.

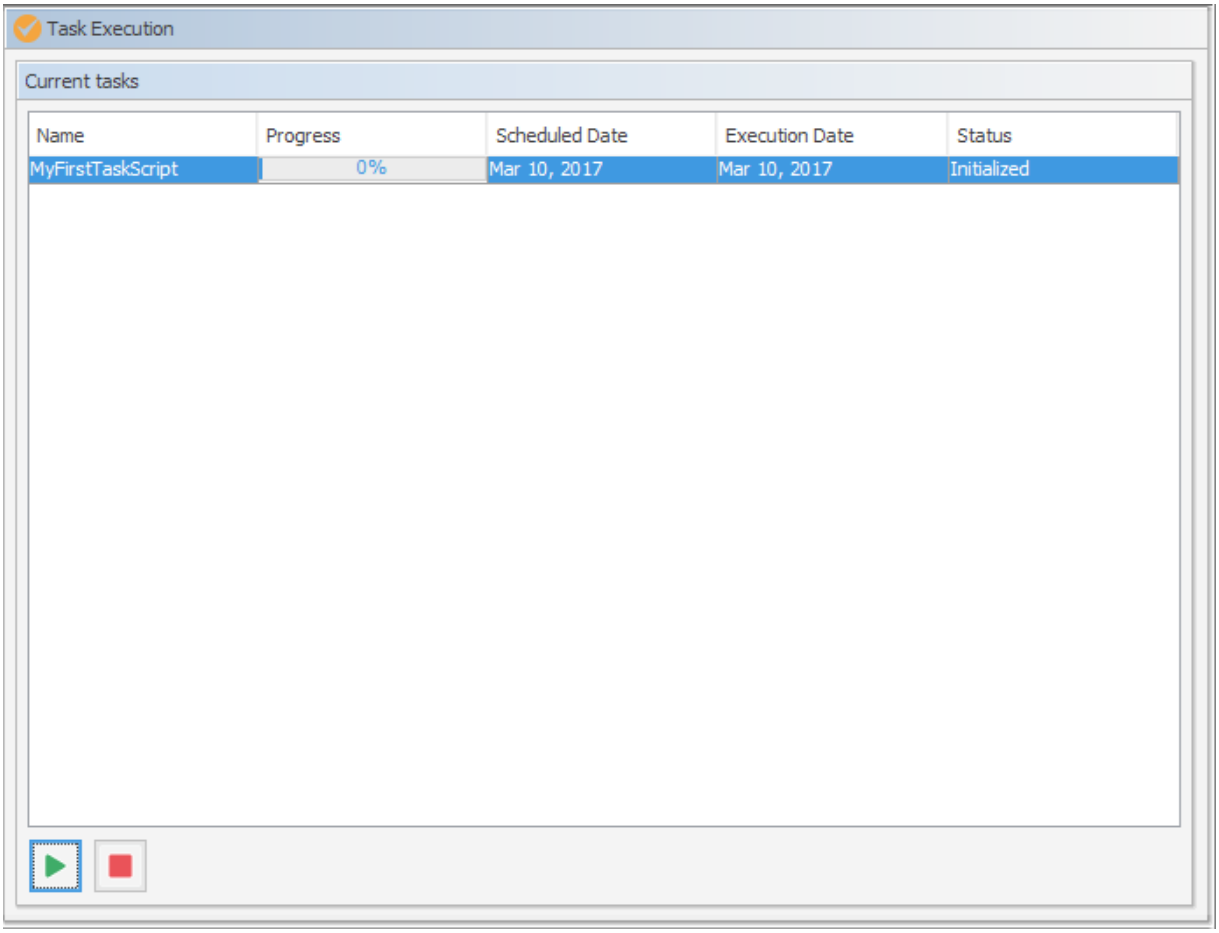


Figure 426: *ConSol CM Admin Tool - Services, Task Execution: Running task*

When a TEF script is executed manually using the Admin Tool, it will run on the machine where the Admin Tool is running. If you work in a clustered environment, where scripts are executed automatically, you can define the cluster node ID where TEF scripts should run using the CM system property cmas-core-server, task.execution.node.id.

F.5.3 Programming with Tasks

F.5.3.1 Introduction

In the current ConSol CM version, only one type of task is available, the **Groovy Task with a static script**. This refers to the Admin Tool script which defines the task, as described in the previous section.

The **Task Execution Service** (Groovy class `TaskExecutionService`, a singleton) runs in the background and scans the ConSol CM database for tasks (database table `cmas_task_descriptor`) with the status INITIALIZED. Like all ConSol CM services it is implicitly available as an object named `taskExecutionService` (see the following examples). When the start time of the task has been reached, the task is started.

Parameters for the new task have to be set either in the task descriptor (Groovy class `TaskDescriptor`) or in the task execution specification (Groovy class `ExecutionSpecification`). The task descriptor will also provide information about the running task, like the task's progress.

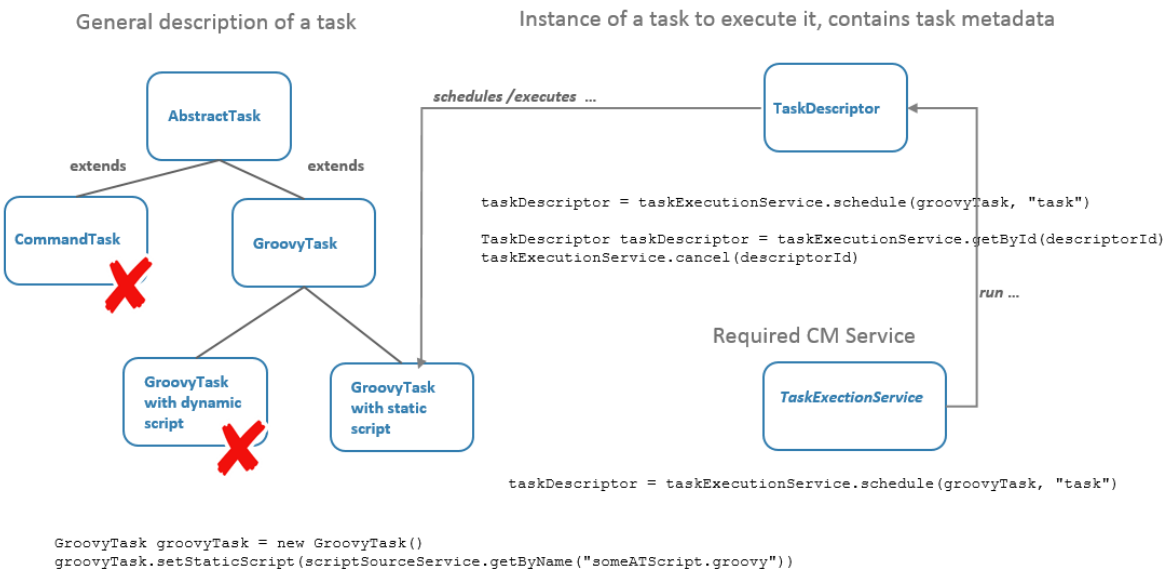


Figure 427: *Some TEF Groovy classes*

F.5.3.2 Coding Examples

Creating a Task

```
GroovyTask groovyTask = new GroovyTask();
groovyTask.setStaticScript(scriptSourceService.getByName("someATScript.groovy"));
taskDescriptor = taskExecutionService.schedule(groovyTask, "task");
```

Code example 68: *Creating a task descriptor*

Canceling (Killing) a Task

Part 1: Create the task descriptor and save its ID somewhere.

```
GroovyTask groovyTask = new GroovyTask();
groovyTask.setStaticScript(scriptSourceService.getByName("someATScript.groovy"));
taskDescriptor = taskExecutionService.schedule(groovyTask, "task");
def myTaskDescriptorId = groovyTask.getId()
//save this Id wherever it will be needed, e.g., in a different script which might
 be used to kill the task
```

Code example 69: *Cancelling a task*

Part 2: Potentially used during the execution of the task.

```
taskExecutionService.cancel(myTaskDescriptorId)
```

Repeating a Task

If you set another execution date for a task after its job has completed, it will be rescheduled. This is accomplished from within the Admin Tool Task script, as demonstrated here.

```
def onInitialize(taskDescriptor) {}

def onExecute(taskDescriptor) {
   //some code to execute
   ........................
   //here, we return the special steering object with the new execution date of the
    task
   return new ExecutionSpecification().setRetryRequested(true).setExecutionDate(new
    Date(new Date().getTime() + 15000));
}

def onError(taskDescriptor) {}
def onCancel(taskDescriptor) {}
```

Code example 70: *Repeating a task*

Defining the (First) Execution Date

For a script which should not be started immediately, you can define a start time in the `onInitialize()` method.

```
def onInitialize(taskDescriptor) {
   return new ExecutionSpecification().setExecutionDate(new Date(new Date().getTime
    () + 15000));
}
```

Code example 71: *Scheduling a task*

Repeating a Task after an Error Occurs

```
def onInitialize(taskDescriptor) {}

def onExecute(taskDescriptor) {}

def onError(taskDescriptor) {
return new ExecutionSpecification().setRetryRequested(true);
// this will reschedule the task for immediate re-execution, in case a future date
 is needed, this can be set as explained in the example above

def onCancel(taskDescriptor) {}
```

Code example 72: *Repeating a task after an error occurred*

Working with the ContextReference

Using the `ContextReference`, it is possible to determine which context a script has been called in. For example, a TEF script might be required to behave differently depending on whether it is called from a workflow script in a certain activity or in the process from another activity. In such cases, the `ContextReference` will help you to deduce which workflow script the TEF script was called from. You just set a different `ContextReference`, which is a simple string as identifier, in each workflow script. Then you can retrieve the `ContextReference` within the task. Thus the task "knows" where it was called from.

Use the methods

- `void GroovyTask.setContextReference(String ContRef)`
- `String TaskDescriptor.getContextReference()`

See also example #2 below.

F.5.3.3 Examples for the Use of Task Scripts

Example 1: Using a Simple Task Script

In this example, a task script will be executed from a workflow activity. No delay is set, i.e., the task is scheduled to be executed immediately when the engineer executes the workflow activity using the Web Client. The script might then run in the background and the engineer will only see the results (like new ticket entries or new customer data) when the script is finished. No action is required on the engineer's part in between the script's start and completion.

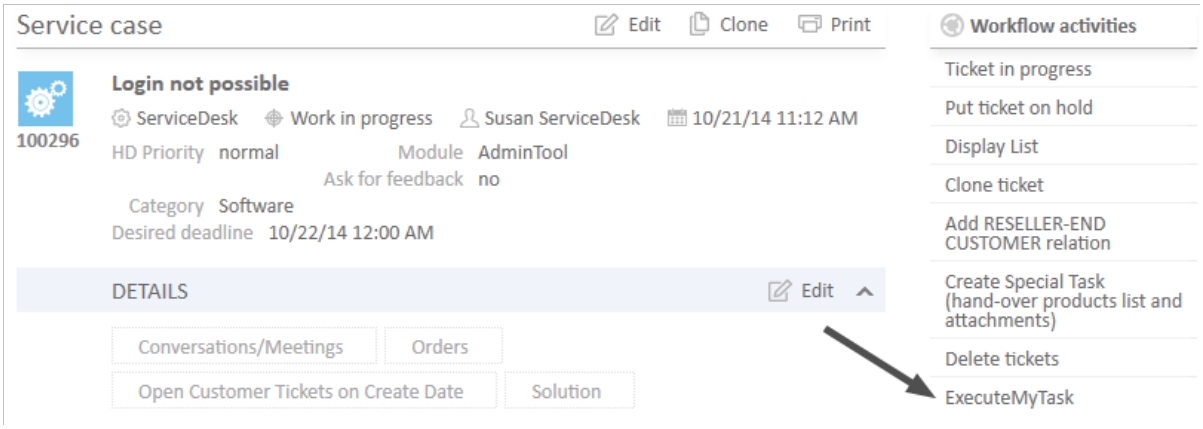


Figure 428: *ConSol CM Web Client - Workflow activity for task execution*

Figure 429: *ConSol CM Process Designer - Workflow activity for task execution*

```
def myNewTask = new GroovyTask()
myNewTask.setStaticScript(scriptSourceService.getByName("MyFirstTaskScript"))
def myTaskDescriptor = taskExecutionService.schedule(myNewTask, "myTaskGroup")
```

Code example 73: *Workflow activity script for task execution*

```
2015-02-20 11:54:24,742 INFO [rver.service.task.TaskExecutor] [task-executor-task-
 executor:10.0.6.200:0-] Task Executor task-executor:10.0.6.200:0 is executing
 task: TaskDesc-02-20 11:54:19.0, transactionTimeout (sec)=0, type=class
 com.consol.cmas.common.model.task.GroovyTask}
2015-02-20 11:54:24,747 INFO [ database_MyFirstTaskScript] [task-executor-task-
 executor:10.0.6.200:0-] MyFirstTaskScript is executed
2015-02-20 11:54:24,747 INFO [ database_MyFirstTaskScript] [task-executor-task-
 executor:10.0.6.200:0-] ztztzt ...
2015-02-20 11:54:24,748 INFO [rver.service.task.TaskExecutor] [task-executor-task-
 executor:10.0.6.200:0-] Task execution successful removing task : TaskDescriptor
 {group='myT, transactionTimeout (sec)=0, type=class
 com.consol.cmas.common.model.task.GroovyTask}
```

Example 2: Working with the ContextReference, Simple Example with Log Output

The Task script (Admin Tool script of type *Task*) will display the ContextReference from which it has been called.

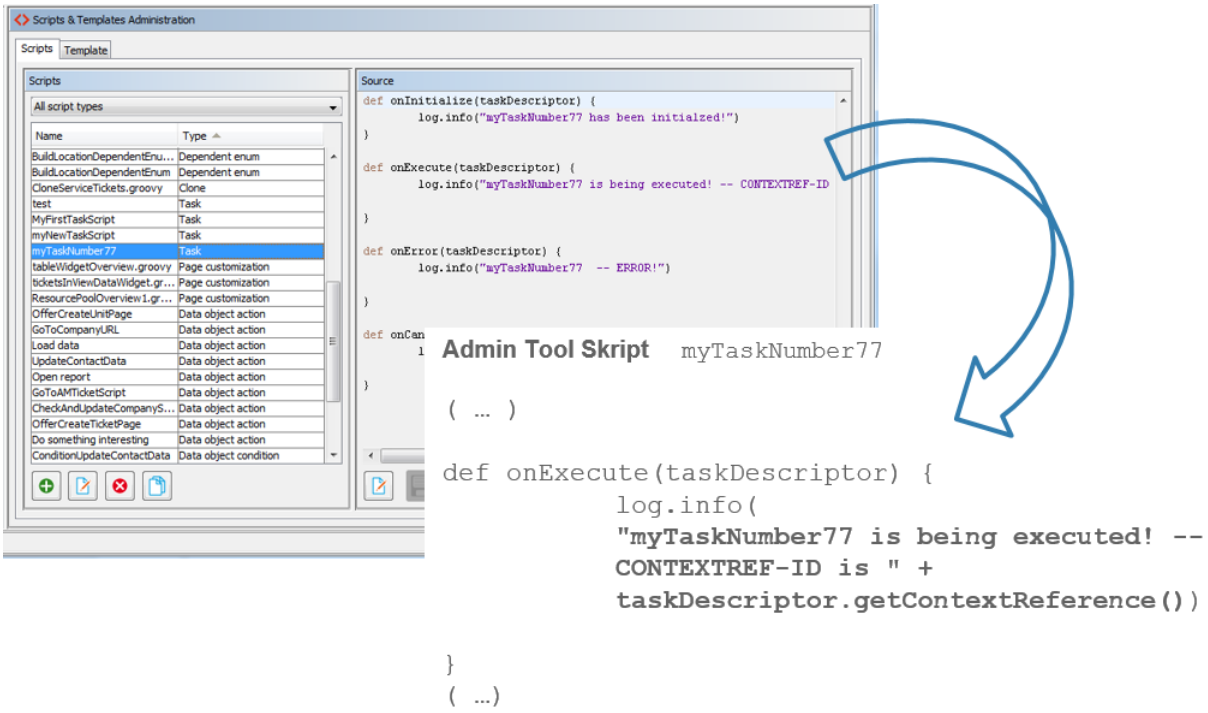


Figure 430: *ConSol CM Admin Tool - System, Scripts and Templates: Admin Tool script for a TEF task*

In different workflow activities, different ContextReferences are set as unique identifiers. When the TEF script is called, it will always display (in the log output) the ContextReference of the calling workflow activity.

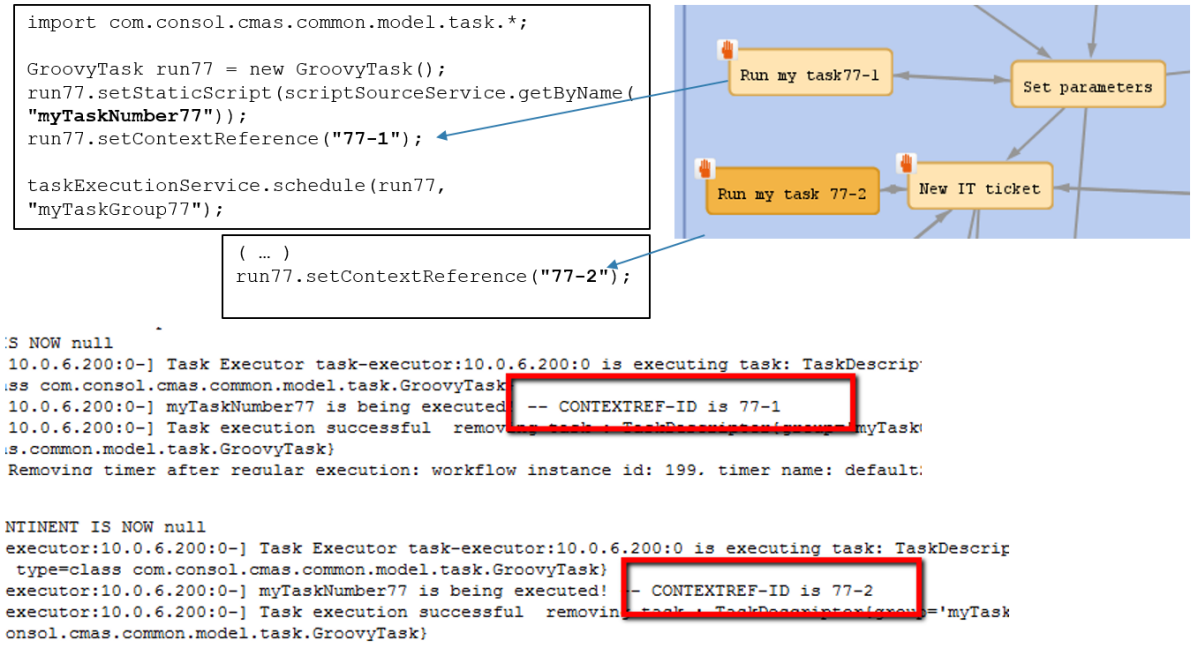


Figure 431: *Workflow activities and log output when one of the activities has called the TEF script*

F.5.4 System Properties Relevant for the TEF

Module	Parameter	Default Value	Description
cmas-app-admin-tool	start.groovy.task.enabled	false	Enables the "start task" button in the Admin Tool
cmas-core-server	transaction.timeout.minutes	60	Sets the transaction timeout for the task execution service, i.e., one run of a task must finish before this timeout is reached
cmas-core-server	number.of.tasks	1	Thread pool size, i.e., number of tasks executed in parallel
cmas-core-server	task.execution.interval.seconds	5	Time to wait between execution of two tasks, in seconds

F.6 The ConSol CM Action Framework

The ConSol CM Action Framework comprises several types of actions which are NOT workflow actions. Workflow actions are executed for tickets, either manually or automatically, whereas the Action Framework provides actions for other types of objects. Additional components like Control Forms extend the spectrum of the framework.

Please see the following overview for a first impression of all components of the Action Framework and read the respective sections for detailed information.

F.6.1 Actions

- **Search actions**
  Always manual actions. Available for
  - tickets
  - customer (= units)
  - resources

  Explained in section Action Framework - Search Actions
- **Customer actions**
  Can be manual or automatic actions. Available for
  - companies
  - contacts

  Explained in section Action Framework - Customer Actions
- **Resource actions**
  Can be manual or automatic actions. Available for
  - resources

  Explained in section CM/Resource Pool - Resource Actions

F.6.2 Additional Components

- **Control Forms**
  Forms, similar to Activity Control Forms, which allow gathering data during the execution of an action of the Action Framework. They are explained in section Control Forms.
- **Scripts for the Action Framework**
  Scripts used to implement several functionalities of the Action Framework. The scripts are explained in section Scripts for the Action Framework.

F.6.3 Scripts for the Action Framework

F.6.3.1 Introduction

The ConSol CM Action Framework offers the ability to start actions which are not related to workflow activities, i.e., they can be started or triggered in another context, not only in a workflow context.

The Action Framework consists (as of ConSol CM version 6.10) of three components which are treated in the respective manual sections in detail.

- **Customer actions (unit actions)**
  Actions based on contact or company objects, see section Action Framework - Customer Actions.
- **Resource actions**
  Actions based on resource objects, see section CM/Resource Pool - Resource Actions.
- **Search actions**
  Actions which are based on the result set of a search, see section Action Framework - Search Actions.

In this section here, you will learn more about Action Framework-specific programming.

F.6.3.2 Admin Tool Scripts for the Action Framework

Script Types

Each action is based on an Admin Tool script as described in the sections mentioned above. For each Admin Tool script, the correct script type has to be set.

Please note that for each **action script**, a condition script **can** be defined. It has to be associated with the action script in the action definition in the Admin Tool. This **condition script** will be executed before the action script.

Thus, the action script will only be executed if either

- no condition script is defined

or

- the condition script returns “true”.

The following Action Framework-related script types are available:

- For **customer actions (= unit actions)**:
  - Customer condition
  - Customer action
- For **resource actions**:
  - Resource condition
  - Resource action

- For **search actions**:
  - For actions based on ticket result lists:
    - Search condition for tickets
    - Search action for tickets
  - For actions based on customer (= unit) result lists, i.e., contact or company result lists:
    - Search condition for customers
    - Search action for customers
  - For actions based on resource result lists:
    - Search condition for resources
    - Search action for resources
- For **Control Forms** (see section Control Forms):
  - Control Form Condition
  - Control Form Prefill

Action Types

An action can be of one of the following types:

- **Automatic**
  - Create
  - Update
  - Delete
  - Relation
  - Search
- **Manual**

F.6.3.3 Programming with the Action Framework

Available and Deprecated Objects

In all of the available script types (see section Script Types) a class is available to implement the behavior of the system after the script has been executed. This is the class `OperationResponseBuilder`. An object of this class, named `client`, is implicitly available in all action scripts, in automatic as well as in manual scripts. This object is also available in workflow scripts - but this will be treated in detail in the *ConSol CM Process Designer Manual*.

The `client` object offers methods for operations like the following:

- open a certain URL
- open the create resource page
- open the create ticket page with a defined customer already assigned

Besides those actions, the `client` object also offers methods to display messages in the Web Client, for example:

- display the info that the customer action has been executed successfully
- display the info that the resource action has failed
- show a warning message if certain criteria are not met by the entered data

Furthermore, the `client` object offers a `rollback()` method.

Please note:

- The classes and respective objects of `ActionScriptResult`, `ActionScriptResultFactory`, `PostActionScriptResult`, `PostActionType` and `postActivityExecutionHandler` are deprecated in CM versions 6.11.1.0 and up! Use the new `client` (= `OperationResponseBuilder`) implementation!
- The method `workflowApi.addValidationError()`, which can be used to display messages in the Web Client, is still valid.

Overview of All `client` Methods (in Comparison to Old Methods)

The following methods are available in the `client` object, an implementation of class `OperationResponseBuilder`. For more details, please refer to the ConSol CM Java / Groovy API documentation. A mapping to the old/deprecated objects and methods is provided - this might make it easier for you to exchange the code in your scripts.

New class/method	Deprecated class/method	Note
client.failure()	PostActionType.FAILURE	Sets the OperationType = FAILURE.
client.goToCreateResource (resource)	actionScriptResultFactory.getPostAction (PostActionType.CREATE_RESOURCE, resource)	Opens the create resource page prefilled with data from `resource` object.
client.goToCreateTicket (ticket)	actionScriptResultFactory.getPostAction (PostActionType.CREATE_TICKET, ticket)	Opens the create ticket page prefilled with data from `ticket` object.
client.goToCreateTicket(tick-et).withCustomer(unit);	actionScriptResultFactory.getPostAction (PostActionType.CREATE_TICKET, ticket, unit)	Opens the create ticket page prefilled with data from `ticket` and `unit` object.

New class/method	Deprecated class/method	Note
client.goToCreateUnit(unit)	actionScriptResultFactory.getPostAction (PostActionType.CREATE_UNIT, unit)	Opens the create customer page prefilled with data from `unit` object.
client.goToPage("url")	actionScriptResultFactory.getPostAction (PostActionType.GOTO_PAGE, "url");	Opens the given URL.
client.goToResource (resource)	actionScriptResultFactory.getPostAction (PostActionType.GOTO_RESOURCE, resource);	Opens the resource page of the given `resource` object.
client.goToTicket(ticket)	actionScriptResultFactory.getPostAction (PostActionType.GOTO_TICKET, ticket)	Opens the ticket page of the given `ticket` object.
client.goToUnit(unit)	actionScriptResultFactory.getPostAction (PostActionType.GOTO_UNIT, unit)	Opens the customer page of the given `unit` object.
client.goToResource (resource).openActionForm ("actionformname");	-	Opens the given action form of the resource.
client.goToUnit(unit).openActionForm("actionformname")	-	Opens the given action form of the customer.
client.goToTicket(ticket) .openActivityForm ("acfpath")	actionScriptResultFactory.getPostAction (PostActionType.GOTO_TICKET, ticket, acfExecutionContext)	Opens the ACF of the current ticket.
client.rollback()	-	Rolls back the action. Optionally, an error message can be provided.
client.showDebugMessage ("messageOrLabel")	actionScriptResultFactory.getPostAction (PostActionType.FAILURE, "messageOrLabel")	Shows a red message, the message text can be entered as a string or using a label.
client.showErrorMessage()	actionScriptResultFactory.getPostAction (PostActionType.FAILURE, "messageOrLabel")	Shows a red message, the message text can be entered as a string or using a label.

New class/method	Deprecated class/method	Note
client.showInfoMessage()	actionScriptResultFactory.getPostAction (PostActionType.SUCCESS, "mes-sageOrLabel")	Shows a green message, the message text can be entered as a string or using a label.
client.showMessage("mes-sageOrLabel")	actionScriptResultFactory.getPostAction (PostActionType.SUCCESS, "mes-sageOrLabel")	Shows a green message, the message text can be entered as a string or using a label.
client.showWarningMessage ()	actionScriptResultFactory.getPostAction (PostActionType.SUCCESS, "mes-sageOrLabel")	Shows a green message, the message text can be entered as a string or using a label.
client.success()	PostActionType.SUCCESS	Sets the OperationType = SUCCESS.

Examples

Open the *Create Ticket* Page

- client.goToCreateTicket(ticket)
- client.goToCreateTicket(ticket).withCustomer(unit)

Example:

```
import com.consol.cmas.common.model.ticket.Ticket

Ticket ticket = new Ticket();
ticket.setQueue(queueService.getByName("Helpdesk"))
ticket.setSubject("sample subject")
ticket.set("queue_fields.string", "test")
ticket.set("queue_fields", "boolean", "true")
client.goToCreateTicket(ticket)
//to additionally set main contact use unit , has to be retrieved in code
//client.goToCreateTicket(ticket).withCustomer(unit)
```

Code example 74: *Open the Create ticket page*

Open the Ticket Page in Display-Only Mode

- client.goToTicket(ticket)

Example:

```
client.goToTicket(newtic)
```

Code example 75: *Open the ticket page in display mode*

Open the Ticket Page in Display-Only Mode and Show ACF Before

- `client.goToTicket(ticket).openActivityForm(executionContext)`
  Redirects to ticket page of given ticket and shows given activity control form obtained via `controlFormDefinitionService.getExecutionContext(Ticket,ACF-path)`.

Example:

```
(...)
def executionContext = controlFormDefinitionService.getExecutionContext(newtic,
 "defaultScope/TaskInProgress/AcceptTask")
if (!executionContext) {
   client.showErrorMessage("action.fail.wrong.activity")
}

// Modify entities from the execution context - not the original ones
// - since the user may still press cancel.

executionContext.ticket.add("SpecialTasks_Fields","Deadline", new Date());

client.goToTicket(newtic).openActivityForm(executionContext)
```

Code example 76: *Open the ticket page in display mode and show ACF before*

Open the Create Unit (Contact or Company) Page

- `client.goToCreateUnit(unit)`
  Redirects to unit create page with fields filled with unit data.

Example:

```
// used for companies in MyCustomerGroup to create new contacts easily
import com.consol.cmas.common.model.customfield.meta.*
import com.consol.cmas.common.model.customfield.*

def isAutoBindingEnabled() {
   return true;
}
def myunit = new Unit()
def mycustomergroup = customerGroupService.getByName("MyCustomerGroup")

myunit.setCustomerGroup(mycustomergroup)

def mycustomerdefinition = unitDefinitionService.getByName("customer")
myunit.setDefinition(mycustomerdefinition)

myunit.set("company()", unit)

def fkey_newFirstname = new FieldKey("customer","firstname")
def newFirstname = formFields.get(fkey_newFirstname).value
myunit.set("customer.firstname", newFirstname)
def fkey_newName = new FieldKey("customer","name")
def newName = formFields.get(fkey_newName).value
myunit.set("customer.name", newName)

client.goToCreateUnit(myunit)
```

Code example 77: *Customer action script (here: Company script -> company implicitly available) to open the Create contact page*

Open a URL

- `client.goToPage(url)`
  Redirects to page of given URL.

Example:

```
// opens company's web site
def url = unit.get("company.www")

if (!url) {
   client.showErrorMessage("error.script.no.url")
} else {
   client.goToPage(url)
}
```

Code example 78: *Open a URL*

Display Result Message in Web Client: SUCCESS, Green Messages

- `client.success()`
  Displays the message “Action succeeded” with green background. Sets operation type to OperationType.SUCCESS.
- `client.showInfoMessage(string or label)`
  Displays the string or the message coded by the label with a green background.
- `client.showMessage(string or label)`
  Displays the string or the message coded by the label with a green background.
- `showMessage(String pMessageKey, OperationMessage.Severity.INFO)`
  Displays the string or the message coded by the label with a green background. Another severity level would display the message background in another color. In the following example, a label is used.
- **Example**:

```
client.showMessage("controlForm.info.executionOK",
 OperationMessage.Severity.INFO)
```

Display Result Message in Web Client: FAILURE, ERROR, Red Messages

- `client.failure()`
  Displays the message “Action failed” with red background. Sets operation type to OperationType.FAILURE.
- `client.showDebugMessage(string or label)`
  Displays the string or the message coded by the label with a red background.
- `showMessage(String pMessageKey, OperationMessage.Severity.WARN)`
  Displays the string or the message coded by the label with a red background.

Working with Relation Actions

Relation actions are available for all three main object types and are defined in the Admin Tool for the navigation items. Please see the detailed explanations in the respective sections of this manual.

- **units / customers**
  explained in section Action Framework - Customer Actions
- **tickets**
- **resources**
  explained in section CM/Resource Pool - Resource Actions

A relation action is executed when the relation is

- created
- deleted

It is not executed if the comment of the relation has been changed.

> Please note that a relation action is always executed when the object type for which the relation is defined is involved. As a consequence, two actions will be executed in case the source as well as the target object belong to a type with a relation action!

In scripts which define the relation actions, the following objects are available:

- Unit-Unit Relation:
  - `unit` object
  - `relation` object (UnitRelation)
- Unit-Ticket Relation:
  - `unit` object
  - `ticket` object
  - `role` object (ContactTicketRole)
- Resource-Unit Relation:
  - `resource` object
  - `unit` object
  - `relation` object (ResourceUnitRelation)
- Resource-Ticket Relation:
  - `resource` object
  - `ticket` object
  - `relation` object (ResourceTicketRelation)
- Resource-Resource relation:
  - `resource` object
  - `relation` object (ResourceResourceRelation)

Starting with ConSol CM version 6.11.1.1, it is possible to distinguish between operations where a relation has been added and operations where a relation has been removed. Use the context parameter `actionType` for this purpose. It can be used in the action condition script, e.g. to control if the action script should be executed or not, or it can be used in the action script itself. The `actionType` can be either of two values:

- ADD
- REMOVE

In the following example, the person who is the contact person for an SLA should receive an email whenever a new relation for this SLA to a customer has been established or has been deleted.

Steps to perform:

1. Define a relation, in this example the SLA_to_Company_Relation, with source = resource type *SLAs* and target = *company*, several customer groups.
2. Write the resource action script, in this example: `SLA_CompanyRelationAction.groovy`.
3. Define a resource action (in this example: *SLA_CompanyRelationAction*) of type Relation which uses the script as action script.
4. Assign the resource action to a resource type, in the example to the type *SLAs*.
5. Test the desired use case in the Web Client.

Example relation action script `SLA_CompanyRelationAction.groovy`:

```
import com.consol.cmas.common.model.mail.MailSendHolder
log.info 'SLA to company relation action has been triggered!'
def myUnit = relation.targetUnit
def myUnitDef = relation.targetUnit.definition.name
log.info 'myUnitDef is ' + myUnitDef
def nameField
def groupField
switch(myUnitDef) {
   case "ResellerCompany": nameField = "company_name";
      break;
   case "company": nameField = "name1";
      break;
   case "DirCustCompany": nameField = "dir_cust_company_name"
      break;
}
def myCustName = myUnit.get(nameField)
log.info 'customer name is ' + myCustName
def contPersMail = resource.get("SLA_Fields_basic.responsible_person_email")
   if (contPersMail) {
   log.info ' Email about SLA will be sent to ' + contPersMail

   // Send an email asynchronuously. Mail object is not available outside workflow
    context!
   def pResName = resource.get("SLA_Fields_basic.SLA_Name")

   // alternative to fixed text: use text template, not shown here

// distinguish between ADD and REMOVE of relation
def relOp
if (actionType == "ADD"){
   relOp = "added to "
} else if(actionType == "REMOVE") {
   relOp = "removed from "
}
```

```
 def pText = "A relation has been " + relOp + "from resource " + pResName + " to
 company " + myCustName

  def pTo = contPersMail
  def pSubject = " new or deleted SLA relation - please take care"
  def pHtml = false
  def pFromEmail = configurationService.getValue("cmweb-server-
   adapter","mail.reply.to")

  def holder = MailSendHolder.createSelfSendHolder(pText, pHtml, null, null)
  holder.setSubject(pSubject)
  holder.setTo(pTo)
  holder.setFrom(pFromEmail)
  mailService.sendMailAsynchronous(holder)
} else {
  log.info 'No mail sent to SLA contact person, no email address found'
}
```

Code example 79: *Relation action script to send an email when a resource-company relation has been created or deleted*

Working With the Data of Control Forms

Control Form-Specific Objects

Two specific objects are available in scripts which deal with Control Forms (see section Control Forms), i.e., in

- Action scripts (of the action which is executed when the control form has been filled in)
- Control Form condition scripts
- Control Form prefill scripts

The two objects are:

- **formFields** (Map<FieldKey, AbstractField>)
  The map entries represent the data fields and the respective values of the form. In Control Form condition scripts, the `formFields` are only available, after the Control Form has been filled for the first time.
- **controlForm** (class `TicketActionControlForm`, `UnitActionControlForm` or `ResourceActionControlForm`, depending on the context)
  Represents the action form. It may be null if no action form is used. In Control Form condition scripts, the `controlForm` object is only available, after the Control Form has been filled for the first time.

Furthermore, new methods have been added to the class `ControlFormService` to retrieve Control Forms using several parameters as input values:

Figure 432: *ConSol CM API Doc - Methods of class ControlFormService which are specific for the work with Control Forms*

AutoBinding

The setting (on/off) of AutoBinding, e.g., in an action script of the action which is executed when a control form has been filled in, defines the behavior of the system concerning the values of the form:

- **AutoBinding off**
  The values of the form are available in the form only. They can be retrieved using the `formFields` object.
- **AutoBinding on**
  The values of the form are used to manipulate the values (properties) of the original object, because there is a binding between the data field and the object's property. The form values are also available in the `formFields` object, but the original object is changed as well.

Please see the following example which explains the difference between the two values.

The script `CheckAndUpdateCompanyServiceStatus` is a customer action script.

Example AutoBinding off:

Script:

```
def isAutoBindingEnabled() {
   return false;
}

log.info ' Getting FormFields ...'
log.info ' FormFields are ' + formFields

client.showMessage("controlForm.info.executionOK", OperationMessage.Severity.INFO)
```

Web Client:

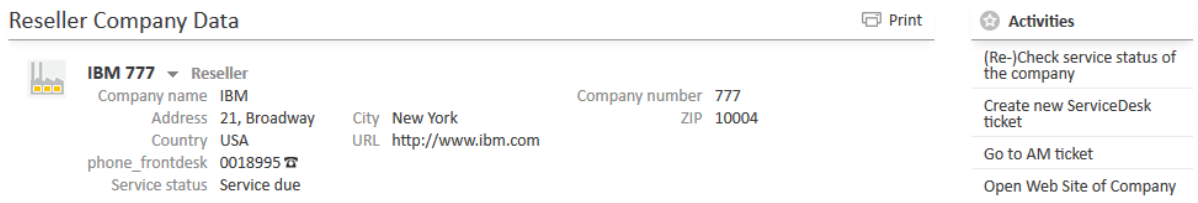


Figure 433: *Company page, before execution of customer action*

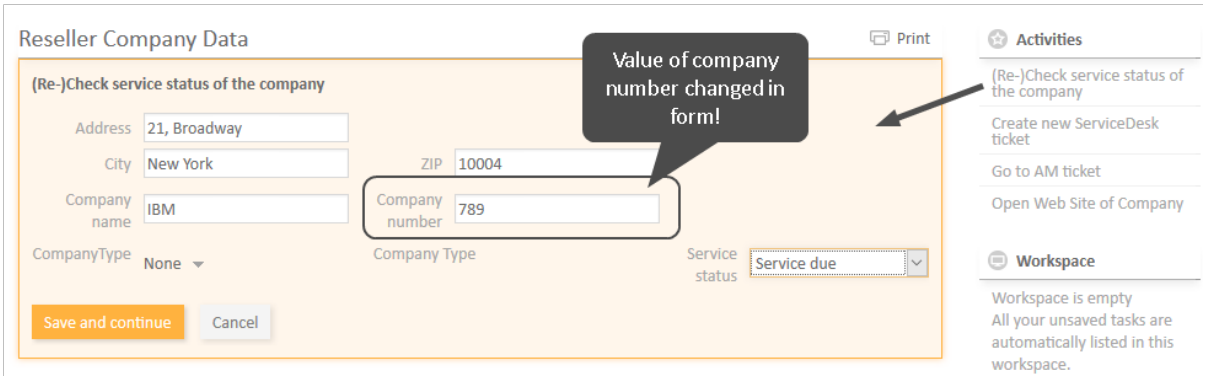


Figure 434: *Control Form of the customer action, value is changed in the form*

Output in `server.log`, changed value available in `formFields` object:

```
2018-01-04 14:46:31,965 INFO [heckServiceStatus1515073591944] [Susan-c4258118-
f127-11e7-af12-858479a7cfd7] Getting FormFields ...

2018-01-04 14:46:31,965 INFO [heckServiceStatus1515073591944] [Susan-c4258118-
f127-11e7-af12-858479a7cfd7] FORM FIELDS are a null

2018-01-04 14:46:31,966 INFO [heckServiceStatus1515073591944] [Susan-c4258118-
f127-11e7-af12-858479a7cfd7] FormFields are
[(address,ResellerCompanyData):AbstractField{key=(address,ResellerCompanyData),
value=21, Broadway}, (company_name,ResellerCompanyData):AbstractField{key=
(company_name,ResellerCompanyData), value=IBM}, (company_
number,ResellerCompanyData):AbstractField{key=(company_
number,ResellerCompanyData), value=789}, (city,ResellerCompanyData):AbstractField
{key=(city,ResellerCompanyData), value=New York},
(zip,ResellerCompanyData):AbstractField{key=(zip,ResellerCompanyData),
value=10004}, (service_status,ResellerCompanyData):AbstractField{key=(service_
status,ResellerCompanyData), value=EnumValue[name=serviceDue,orderIndex=2]}]
```

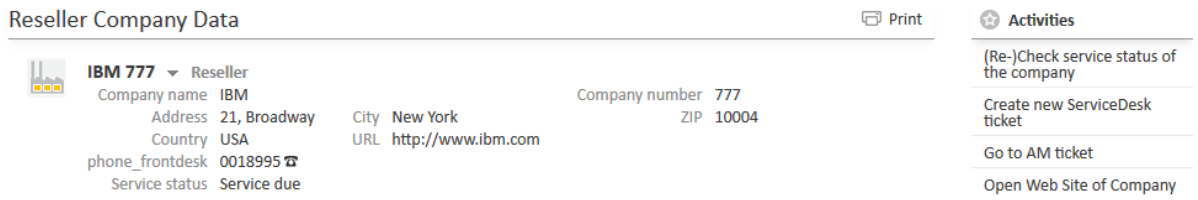


Figure 435: *Company page after execution of customer action. Nothing has changed, because AutoBinding is off*

Example AutoBinding on:

Script:

```
def isAutoBindingEnabled() {
   return true;
}

log.info ' Getting FormFields ...'
log.info ' FormFields are ' + formFields

client.showMessage("controlForm.info.executionOK", OperationMessage.Severity.INFO)
```

Web Client:

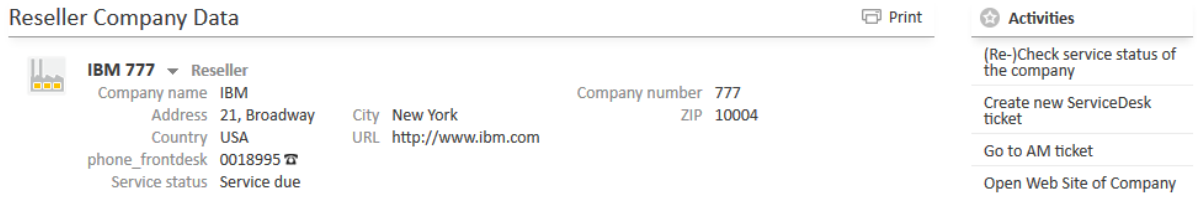


Figure 436: *Company page, before execution of customer action*

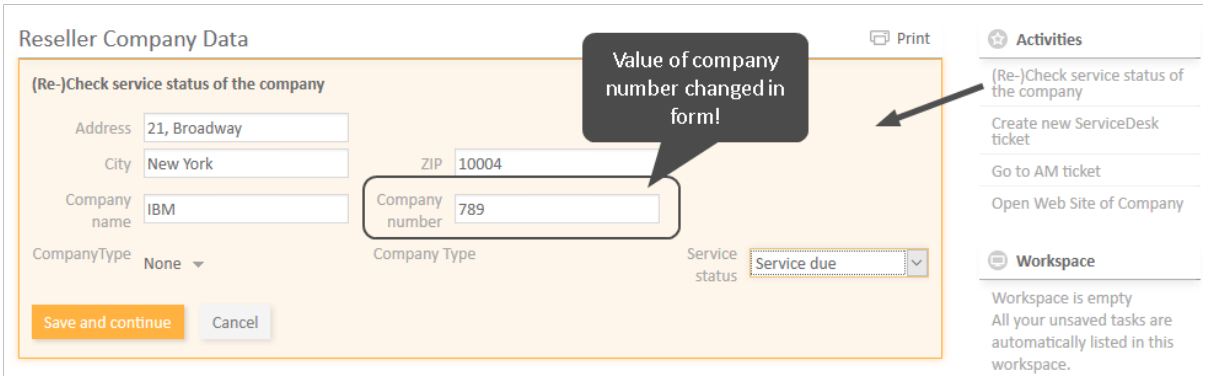


Figure 437: *Control form of the customer action, value is changed in the form*

Output in *server.log*, changed value available in `formFields` object:

```
2018-01-04 14:46:31,965 INFO [heckServiceStatus1515073591944] [Susan-c4258118-
f127-11e7-af12-858479a7cfd7] Getting FormFields ...

2018-01-04 14:46:31,965 INFO [heckServiceStatus1515073591944] [Susan-c4258118-
f127-11e7-af12-858479a7cfd7] FORM FIELDS are a null

2018-01-04 14:46:31,966 INFO [heckServiceStatus1515073591944] [Susan-c4258118-
f127-11e7-af12-858479a7cfd7] FormFields are
[(address,ResellerCompanyData):AbstractField{key=(address,ResellerCompanyData),
value=21, Broadway}, (company_name,ResellerCompanyData):AbstractField{key=
(company_name,ResellerCompanyData), value=IBM}, (company_
number,ResellerCompanyData):AbstractField{key=(company_
number,ResellerCompanyData), value=789}, (city,ResellerCompanyData):AbstractField
{key=(city,ResellerCompanyData), value=New York},
(zip,ResellerCompanyData):AbstractField{key=(zip,ResellerCompanyData),
value=10004}, (service_status,ResellerCompanyData):AbstractField{key=(service_
status,ResellerCompanyData), value=EnumValue[name=serviceDue,orderIndex=2]}]
```

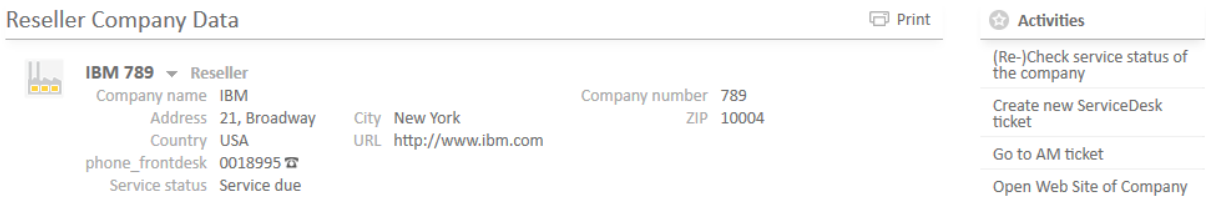


Figure 438: *Company page after execution of customer action. Value of company number has been changed, because AutoBinding is on. A page reload is available to display the new value!*

F.6.4 Control Forms

F.6.4.1 Introduction

What Are Control Forms?

Control Forms offer the possibility to collect data, before a certain action is executed. Just like ACFs (Activity Control Forms, see section Tab Activity Form Data) for workflow activities, Control Forms collect data for actions of the ConSol CM Action Framework, i.e., for the following action types:

- Ticket search actions
- Customer actions
    - Manual customer actions
    - Customer search actions
- Resource actions
    - Manual resource actions
    - Resource search actions

The Control Form is opened and has to be filled in, before the action, i.e., the action script, is executed. Please read section Scripts for the Action Framework for a detailed introduction to scripts for actions of the Action Framework.

Like ACFs, Control Forms can be fine-tuned using scripts. Each Control Form can have the following scripts. They are both optional - a Control Form can also work perfectly without them.

- Form condition script
  Only when the condition is met (“true”), the Control Form is displayed. See section Form Condition Script, Optional.
- Form prefill script
  Can fill in values in the Control Form, i.e., when the form is displayed, the respective fields are filled out. See section Form Prefill Script, Optional.

Use cases like the following can be covered using Control Forms:

- **Company action**
  Enter new contact data into the form - create a new contact for the company with one click.
- **Ticket search action**
  Enter the new contract expiry date for each asset in a list of IT assets, found in a search operation, with one click.
- **Company search action**
  Enter all dates concerning an upcoming conference in each company of a list of companies you want to invite.
- **Contact search action**
  Enter only the name of the email template you want to use and send an email to a list of recipients:

- inform job candidates about the start date of the assessment center
- inform all employees who have a certain type of software license about the necessity to update the license

When a Control Form has been displayed and the respective data have been filled in or modified, the action script which implements the ticket/customer/resource action is executed. Specific methods are available which deal with the Control Form data. This is explained in detail in section Working With the Data of Control Forms.

See the following example to get an impression how Control Forms work.

First Impression of Control Forms - a Control Form of a Resource Action

The resource page of a HP printer is open and the engineer executes the resource activity *Create maintenance ticket for HP printer*. A form is opened where the deadline and priority for the new ticket have to be entered. Only when the engineer has clicked *Save and continue*, the form data is saved and the resource action is executed. In the current example, it would not have been possible to enter nothing, since the form contains mandatory fields (marked by a red asterisk).

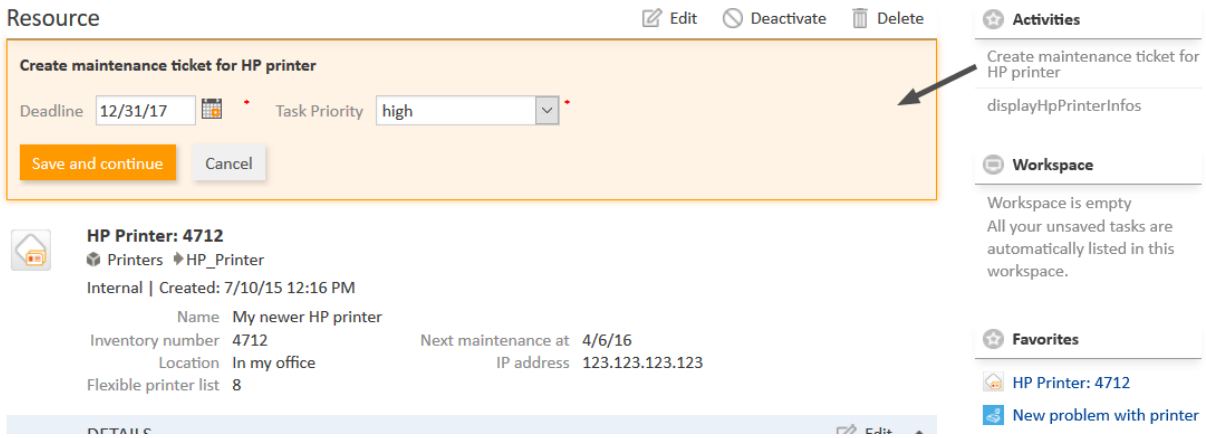


Figure 439: *ConSol CM Web Client - Control Form of a resource activity*

F.6.4.2 Defining Control Forms Using the Admin Tool

All components of Control Forms are defined in the Admin Tool.

A Control Form comprises at least one and can comprise three components:

- The Control Form Itself, Mandatory
- Form Condition Script, Optional
- Form Prefill Script, Optional

The Control Form Itself, Mandatory

It depends on the type of action to which the form should be assigned where in the Admin Tool you have to define the Control Form:

- For ticket search action:
  Navigation group *Tickets*, navigation item *Search Actions*, Tab *Control Forms*
- For customer/unit action:
  Navigation group *Customers*, navigation item *Actions*, Tab *Control Forms*
- For resource action:
  Navigation group *Resources*, navigation item *Actions*, Tab *Control Forms*

In the following example , we will explain a customer/unit action. The principle applies to all Control Forms.

Perform the following steps:

- Define the Data Fields for the New Control Form
- Assign the Control Form to the Action Where It Should Be Opened
- Test the New Control Form

Define the Data Fields for the New Control Form

In order to create a new Control form for a customer action, open the navigation group *Customers*, navigation item *Actions*, Tab *Control Forms*. Click *Add* and select the data fields which should be part of the new Control Form in the *Available* section and move them to the *Assigned* section.

Please note that, unlike in an ACF for tickets where only ticket fields are available, in a Control Form, all types of data fields are available:

- ticket fields
- customer fields:
  - company fields
  - contact fields
- resource fields

This implies that a Control Form can comprise data other than the data which belongs to the object of the action! For example: in a contact action, not only data of this contact, like email or phone, can be requested, but also every other data field like data of the ticket or of the company. Two things are of major importance here:

1. The action script decides what happens to the data of the form! So the developer of this script is responsible for the data processing!
2. The behavior of the action script concerning “auto-binding” is crucial! See section AutoBinding about this!

In the drop-down menus which work as filters above the list of data fields, you can select which type of fields you want to have displayed:

- ticket
- unit
- resource

In the second drop-down menu/filter, you can select the data field group of which the fields should be displayed in the *Available* list. If you have selected a certain field type in the first drop-down menu/-filter, only the matching field groups are displayed, e.g., only fields of the customer field group *ResellerCompanyData* (in our example).

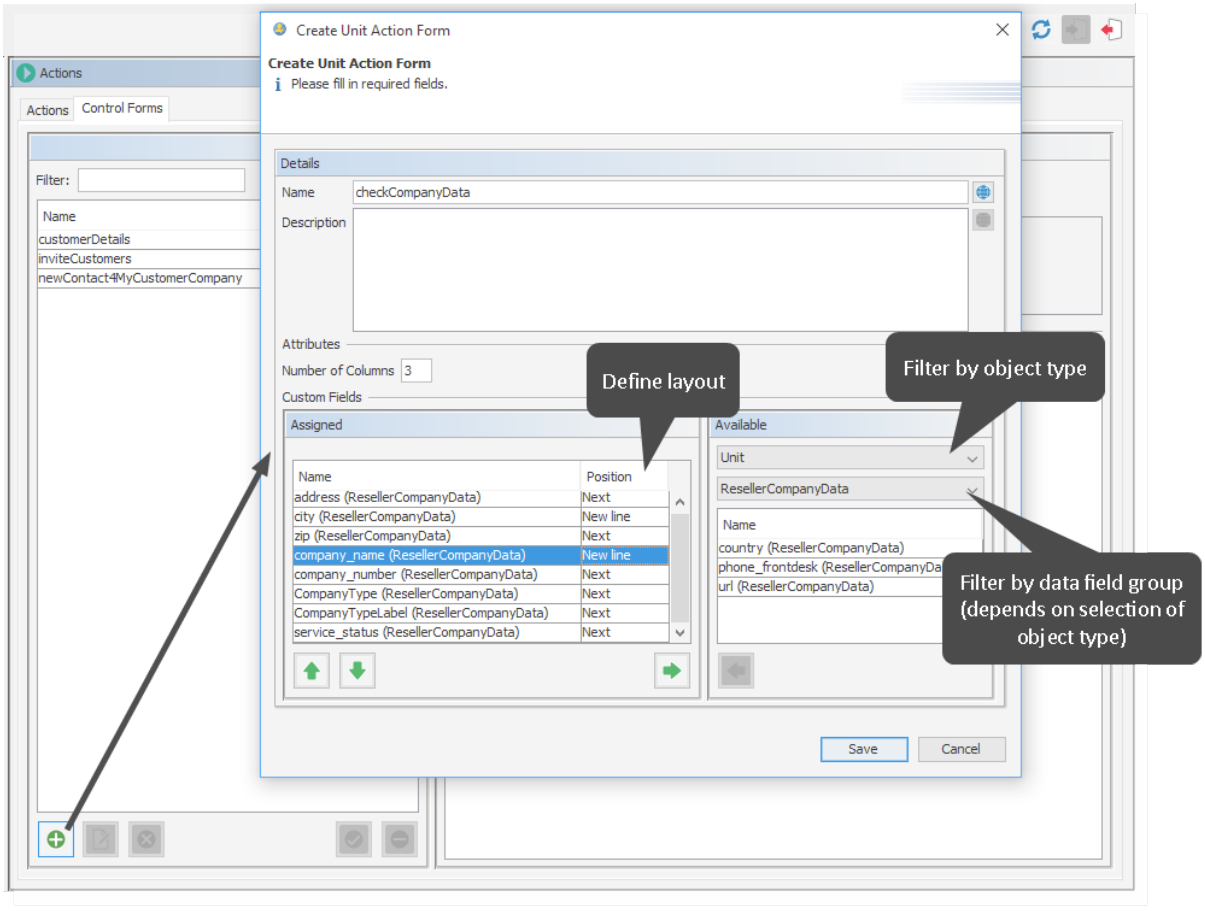


Figure 440: *ConSol CM Admin Tool - Customers, Actions: Creating a new Control Form for a customer action*

In the example, some company fields of the customer field group *ResellerCompanyData* have been selected.

Assign the Control Form to the Action Where It Should Be Opened

When the new Control Form has been defined, you have to assign the form to an action of the Action Framework. In our example, to a company action.

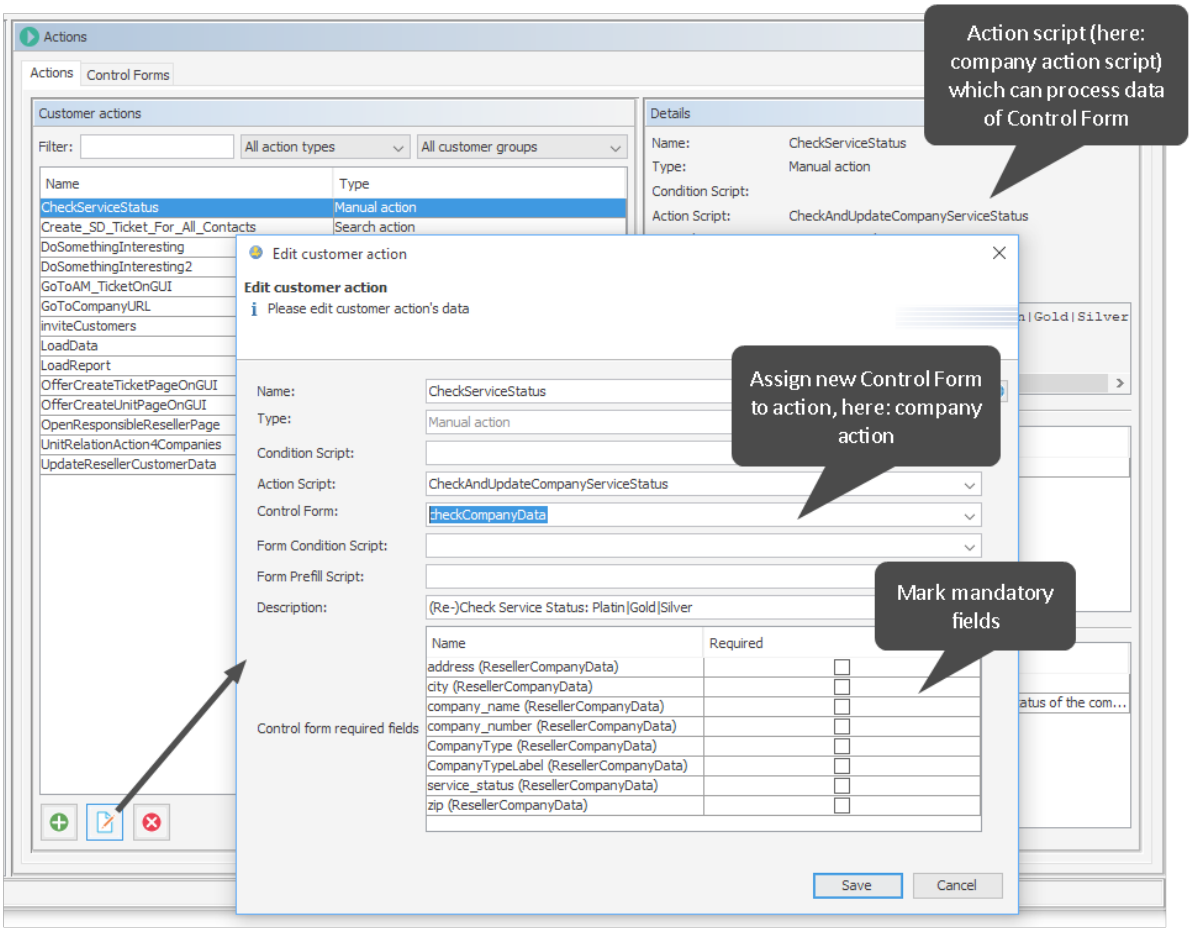


Figure 441: *ConSol CM Admin Tool - Customers, Actions: Assigning a new Control Form to a company action*

Test the New Control Form

Open a company page in the customer group where the manual action has been assigned and execute the company action. The Control Form is opened and can be modified.

In the current example, the fields are already filled in, because the company fields have been filled already and the data is simply displayed. It could be modified within the form.

> Please note the different behavior of the system concerning the saving of the form data depending on the “auto-binding” settings! See section AutoBinding about this!

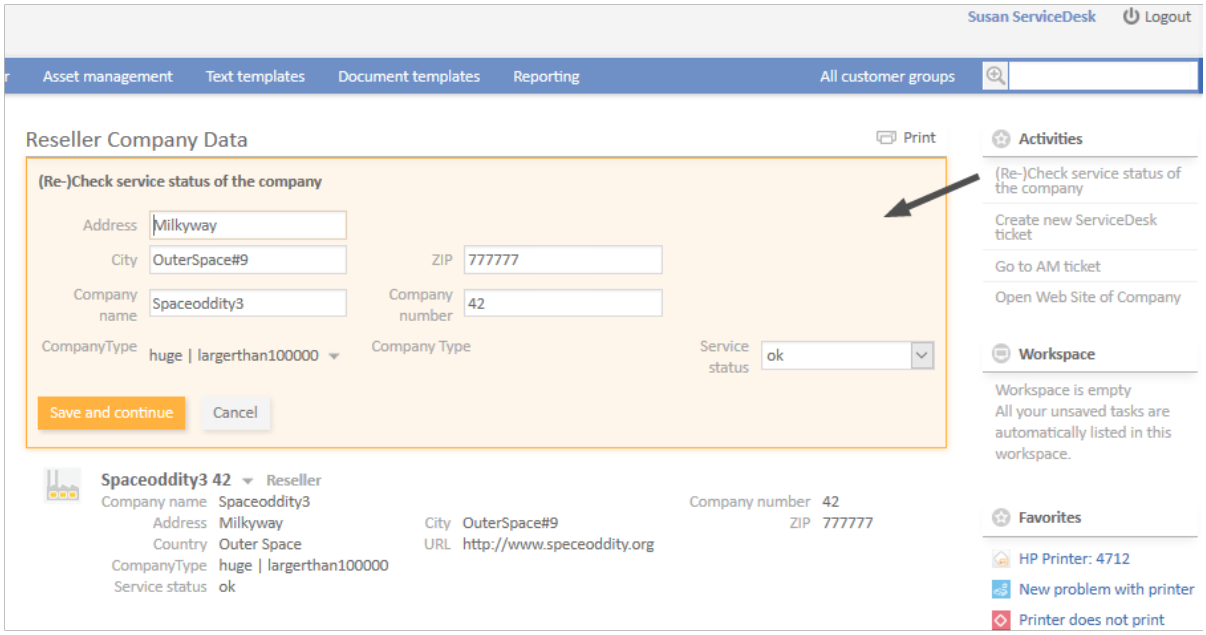


Figure 442: *ConSol CM Web Client - Control Form of a company action*

Form Condition Script, Optional

In the Web Client, a Control Form is only displayed if

- no condition script is present

OR

- a condition script is present and has returned “true”

The form condition script is stored in the Admin Tool, in the scripts section and has to be of type *Control Form Condition*. It has to return either “true” or “false”. In order to implement a form condition script, perform the following steps:

1. Write the Control Form condition script.
2. Assign the script to the action as *Form Condition Script*.

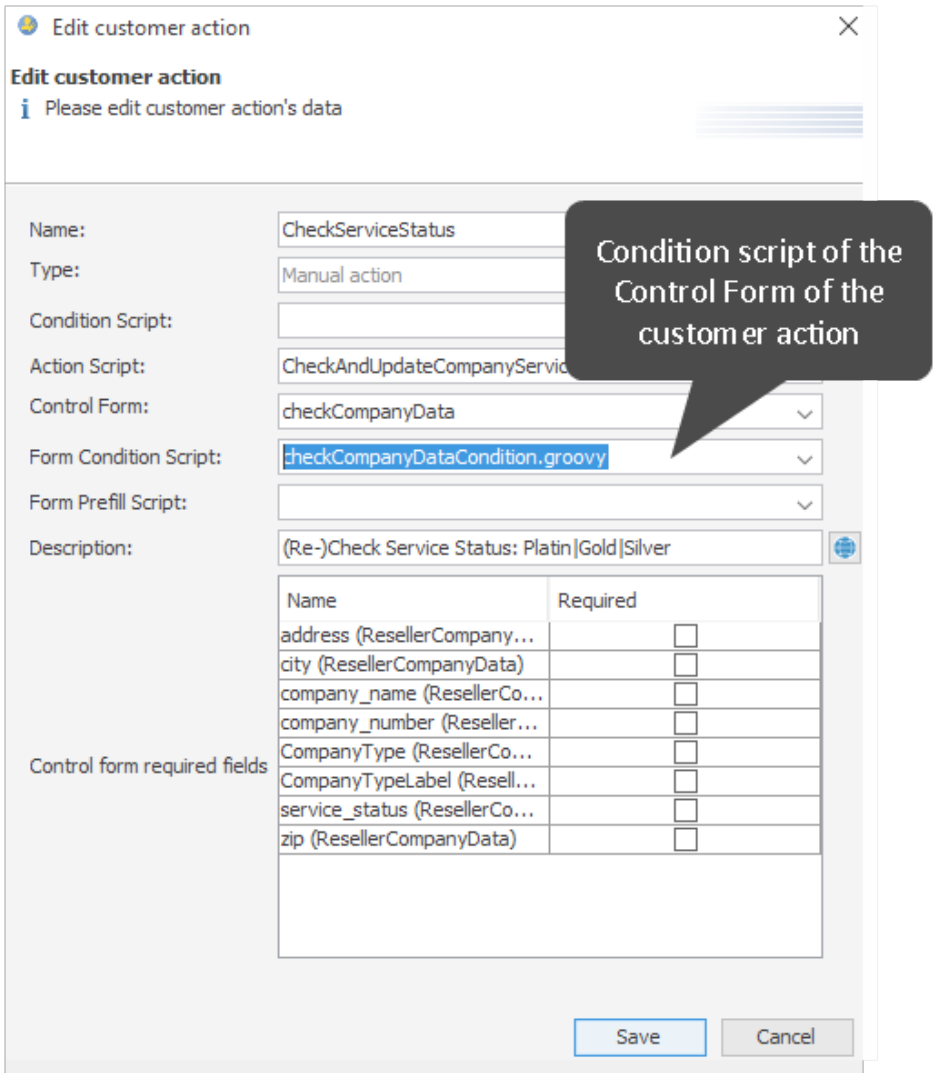


Figure 443: *ConSol CM Admin Tool - Customers, Actions: Assigning a condition script to a Control Form of a company action*

If the script returns “true”, the Control Form is displayed, if it returns “false”, the Control Form is not displayed, but the action is executed.

The following example shows a Control Form condition script which checks if there are empty fields in the resource. Only then the form is displayed in order to gather the missing data. If all fields are already filled, no form is displayed.

```
if(resource.get("product.quantity") == null || resource.get("product.price") ==
 null) {
   return true
}
return false
```

Code example 80: *Control Form condition script which checks if fields are filled*

Form Prefill Script, Optional

If a form should be opened with some fields already filled in with specific values, you have to assign a form prefill script to the respective customer/ticket/resource action. In the script the fields can be filled.

For example, the following Control Form prefill script could be used to make sure the value for service status is set to a current value by an engineer.

```
company.set("ResellerCompanyData.service_status","unknown")
```

In the Web Client, the respective field is pre-filled with the given value:

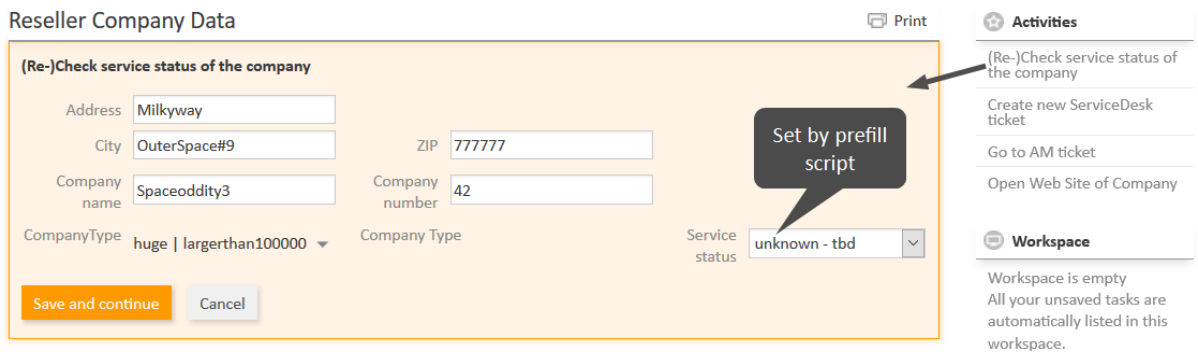


Figure 444: *ConSol CM Web Client - Customer action and Control Form where one field has been pre-filled by a script*

> Please note the different behavior of the system concerning the saving of the form data depending on the “auto-binding” settings! See section AutoBinding about this!

When the form is opened:

- It depends on the auto-binding definition (see info box!) if the fields which belong to the context object are filled or not.
- Values of the form prefill script overwrite the original values of the object.

When the form is saved:

The values which are set within a form prefill script are only used to fill out the form. The data is not persisted. If the data is really saved in the objects, depends on the action script, namely on

- the definition of the auto-binding (see info box!)
- the usage of the content of the `formFields` (see section Working With the Data of Control Forms about `formFields`)

F.7 Email Configuration

ConSol CM includes a sophisticated email interface. CM fetches emails from one or more mail servers and can send emails. Incoming as well as outgoing emails are included in the respective tickets, hence you always have access to all information regarding a case.

Basic CM email functionalities are explained in the following sections:

- Email
- Email Backups

In case your company wants to use email encryption, you can employ server and/or client certificates, see section:

- Email Encryption

F.7.1 Email

F.7.1.1 Introduction to Emails in ConSol CM

Sending and receiving emails is one of the core functionalities of ***ConSol CM***. The application interacts with one or more mail servers to fetch emails and send emails.

The following sections provide information about the following email-related topics:

- Email Functionalities in ConSol CM (sending and receiving emails, email duplication)
- Email Configuration Details
- Email Configuration Using the Admin Tool

For information regarding the setup of the email functionalities in a cluster, please refer to the *ConSol CM Cluster Manual*, section *Email*. Hints on email management during system operation are included in the *ConSol CM Operations Manual*, section *Email*.

F.7.1.2 Email Functionalities in ConSol CM

This section contains a short introduction on the subject “emailing with ConSol CM”. Sending and receiving emails is a core functionality of the application.

Sending Emails from ConSol CM

Manual Emails

Emails can be sent manually by an engineer or automatically by the system. Manual emails are sent using the Ticket Email Editor. In most systems, by default, the ticket's main customer is the receiver of the email, but the engineer can select or type any other email address. The system-wide default value can be changed by an administrator using Page Customization, using the attribute mailToSelection in the Page Customization section. Furthermore, the engineer can use email templates and/or quote ticket text. Please see the *ConSol CM User Manual* for a detailed introduction about working with the Ticket Email Editor.

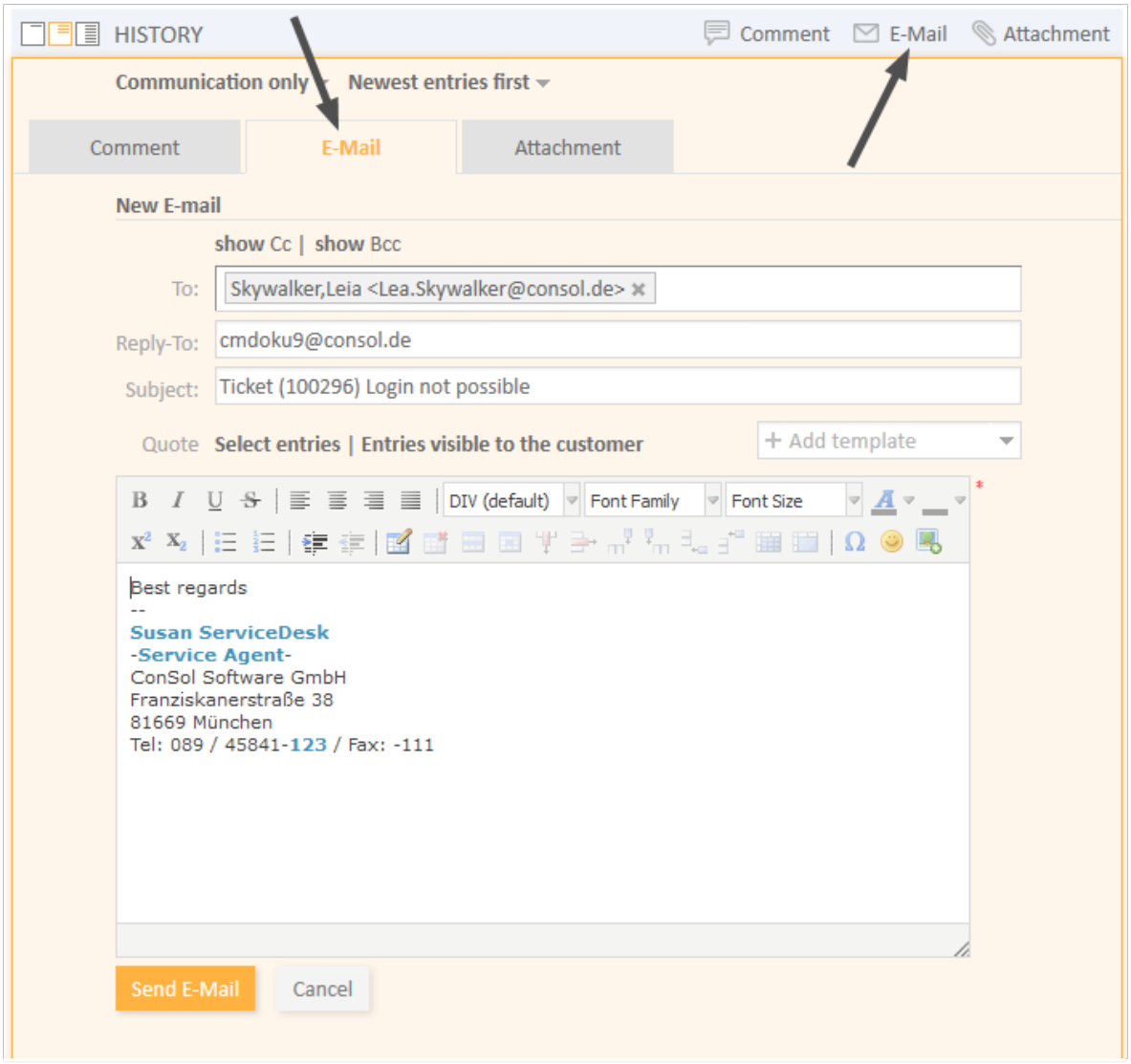


Figure 445: *ConSol CM Web Client - Ticket Email Editor*

Automatic Emails

Automatic emails may be sent by ConSol CM in situations like the following:

1. Initiated by the workflow engine, e.g.,
    a. when the engineer to whom the ticket is assigned should be reminded to attend to the ticket.
    b. when customers should receive an automatic confirmation that a ticket has been opened for them.

c. when customers should receive an automatic confirmation that a ticket of theirs has been closed.

d. when a supervisor or approver should receive a message that a new case has to be approved.

In any workflow activity, an email can be sent to any valid email address. Please see the *ConSol CM Process Designer Manual* for a detailed explanation of the methods to use.

2. Initiated by the system in case of an error or for a success message, e.g.,

   a. system error

   b. email error

   c. DWH synchronization (error or success)
   Usually, those emails are sent to the ConSol CM administrator. However, for most special error cases a special receiver email address can be configured using system properties. Please see section System Properties for details.

3. Initiated by the ConSol CM system to remind engineers

   a. When an engineer receives a ticket or a ticket is retrieved from the engineer, an email can be sent to this engineer. This can be configured on a per-queue basis, as described in section Queue Administration.

Receiving Emails with ConSol CM

The ConSol CM system can fetch emails from one or more mailboxes (= email accounts) on one or more mail servers. The mailboxes are configured in the Admin Tool (Email Functionalities in ConSol CM). Please keep in mind that ConSol CM works with mailboxes here. Each of the mailboxes can be reached by at least one email address. In certain cases, one mailbox might be used for more than one email address. This can be significant when writing Scripts of Type Email.

As far as the mail server is concerned, ConSol CM is just a regular email client fetching emails using a standard mail protocol: IMAP(S) or POP3(S). Depending on the mail server configuration and on the ConSol CM system property cmas-nimh, mailbox.default.task.delete.read.messages, the emails are deleted from the mailbox on the mail server after ConSol CM has picked them up. The default setting is **mails are not deleted after pick-up**.

> ⚠ If you do not want ConSol CM to delete emails from the mail server, please make sure to monitor the mailbox(es) to avoid a data overflow and server or performance problems.

All incoming emails are first stored in an incoming email pool in ConSol CM and are then processed in a chain of email scripts. Please see section Scripts of Type Email for a detailed explanation of those scripts. When an email cannot be processed, the administrator will receive a notification email. The unprocessed email is listed under Email Backups.

There are different possibilities concerning the default system behavior for an incoming email:

- The subject of the email does not contain any ticket number with a valid syntax (i.e., it does not contain the pattern which is defined as regular expression (RegEx) for the ticket subject):
  A new ticket is created.

- The subject of the email does contain a ticket number with a valid syntax (RegEx) and the ticket is still open:
  The email is attached to the existing ticket.
- The subject of the email does contain a ticket number with a valid syntax (RegEx), but the ticket is closed:
  A new ticket is created and a reference to the old ticket is established.

By modifying the email scripts (see section Scripts of Type Email), the default system behavior can be changed. However, this can corrupt core functionalities of the system and should not be done or only done by very experienced ConSol consultants!

Email Duplication in the ConSol CM Web Client

Please see explanations on the *Page Customization* page at showCloneOption and appendOrReplaceOnClone.

F.7.1.3 Email Configuration Using the Admin Tool

This section explains the navigation item *Email* in navigation group *Email* in the Admin Tool.

IMPORTANT INFORMATION

Since ConSol CM version 6.9.4, there are two modes to receive incoming emails:

- **Mule/ESB** - This mode has been available in ConSol CM since version 6. It is deprecated in CM versions 6.11 and up. Therefore, this mode is no longer treated in ConSol CM manuals. If you run an older CM system with Mule/ESB, please refer to older manuals. All ConSol CM 6.11 systems should run in NIMH mode. Please refer to older Administrator manuals to learn how to switch from Mule/ESB to NIMH.
- **NIMH** - *New Incoming Mail Handler*, available since version 6.9.4. This is the only available incoming email mode in ConSol CM version 6.11 and up.

The active email mode is set in the system property cmas-core-server, nimh.enabled:

- **true** - NIMH mode
- **false** - Mule/ESB mode

The sending of emails, i.e., the SMTP server configuration, is not influenced by the incoming email mode.

General Email Configuration (Navigation Item *Email*)

In this navigation item you can set the parameters for the email connection.

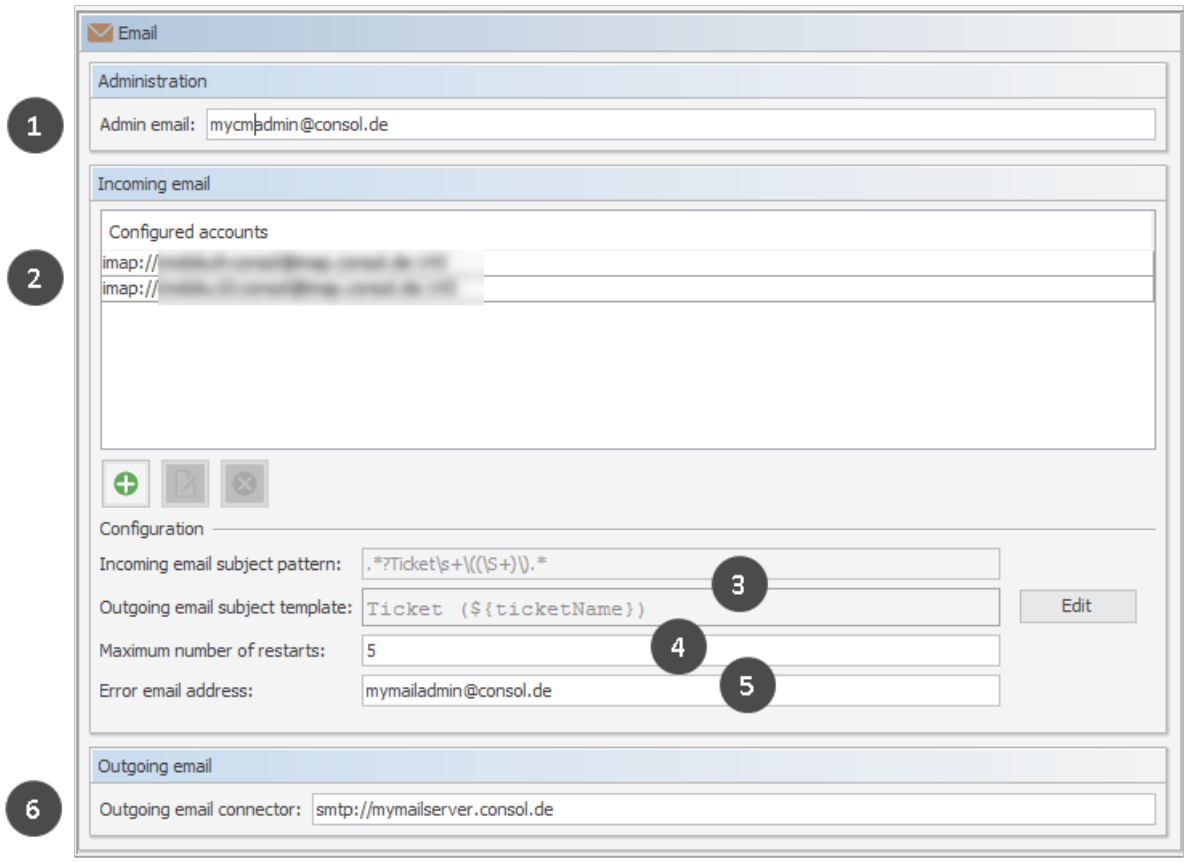


Figure 446: *ConSol CM Admin Tool - Email*

The *Email* screen shows the following information:

- General admin email address for error messages (1)
- Email accounts (not email addresses!) on one or more email servers where ConSol CM will fetch emails (2)
- Configuration of RegEx pattern for the recognition of existing tickets (incoming emails) (3)
- Number of restarts/retries after an error with email delivery (4)
- Email address for email-related error messages (5)
- Address of outgoing email server (SMTP) (6)

⚠ When using NIMH, the configuration of existing mailboxes (*Configured accounts*) can only be edited and new mailboxes can only be added while the NIMH service is stopped.

Administration

Admin email:

Enter the email address which should receive general messages or warnings from the system. Using the GUI, you can enter one admin email address. Multiple addresses separated by commas are possible, but they have to be entered directly in the navigation item System Properties. Use the system property cmas-core-security, admin.email. The total number of characters (for both GUI and property) must not exceed 72. If there are many recipients, we recommend using a mailing list on the mail server system.

Incoming Email

The configuration of incoming email is divided into two areas:

- **Configured accounts:**
  Here you can use a pop-up window to add or edit accounts from which emails are retrieved. The connection to the mailbox is checked during set-up, so it is not possible to configure an account that cannot be used when the system is in operation (provided the mail server has not changed etc.). The value(s) are saved in the system properties in the module `cmas-nimh`. Please see the email properties section in System Properties for detailed information. Required values are:
  - **Protocol**
    The protocol used to retrieve emails from the server. Supported protocols are IMAP4, IMAP4s, POP3, and POP3s. Please keep in mind that ConSol CM behaves like a regular email client. When the secure protocol version is used, the corresponding certificate is required! This has to be stored in the security store of the application server.
  - **Server**
    The DNS-resolvable name or IP address of the mail server.
  - **Port**
    The port on the mail server where the mail daemon/service is listening.
  - **User name**
    The user name of the email account.
  - **Password**
    The password of the email account.

    
    Please keep in mind that one email account can have more than one email address. So here, you are dealing with the account name, i.e., with the mailbox. When you edit the Admin Tool script(s) that process the incoming emails, it might be required to use the email address. The email address is also required when you configure the Reply-To address, the From address, and queue-specific email addresses! So be sure to use the correct parameter: mailbox or email address!

- **Configuration:**
  - **Incoming email subject pattern:**
    Describes the elements that the subject of an incoming email has to contain in order to assign this email to a certain ticket. The pattern is defined in form of a regular expression (RegEx).
    Example: *.*?Ticket\s+\((\S+)\).** would match every subject line that contains *Ticket (<Ticket number>)*.
  - **Outgoing email subject template:**
    Describes the pattern which is used to create the ticket ID in the subject of an outgoing email. The template should be matchable by the incoming email subject pattern. Via the *Edit* button on the right you can modify the incoming email subject pattern and outgoing email subject template and verify if they match.
    Example: *Ticket (${ticketName})* would match the example RegEx above.

    > ⚠ You can check if the pattern for the incoming email subject pattern and for the outgoing email subject template match by using the *Edit* button and the editor that is opened. Please make sure that the email subject has been set correctly at **all** locations, e.g., also in all workflow scripts and Admin Tool scripts!

  - **Maximum number of restarts:**
    Shows the maximum number of restarts after an error when ConSol CM fetches emails. Valid for all mail pollers.
  - **Error email address:**
    Email address to which messages and warnings of the mail sub-system are sent. This is usually the same as the general administrator address.

For the configuration of incoming email you might also want to check the email-related system properties, see System Properties. Particularly, the polling interval (the time interval for fetching emails from the mail server, system property mailbox.default.task.interval.seconds (module `cmas-nimh`) or the respective property of another mailbox (e.g. `mailbox.1.task.interval.seconds`) might be of interest.

Outgoing Email

The connection data for outgoing emails are set here:

Outgoing email connector

Use the following format:

```
smtp://<IP address of mail server>:<port>
```

Example with standard port:

```
smtp://123.123.123.123:25
```

Example for SMTP server with authentication:

```
smtp://test:testpassword@ConSolMailServer2:25
```

The example above uses the following parameters:

- Protocol: SMTP
- User name: test
- Password: testpassword
- Host/Mailserver: ConSolMailServer2
- Port: 25

Please note: If user name and/or password which have to be provided contain the character "@", the "@" has to be coded as %40.

To enable SMTPS for outgoing emails, set the system property cmas-core-server, mail.smtp.tls.enabled to "true".

F.7.1.4 Email Configuration Details

This section describes emailing in a single-server environment. Please refer to the *ConSol CM Cluster Manual* for information regarding multi-server environments.

IMPORTANT INFORMATION

Since ConSol CM version 6.9.4, there are two modes to receive incoming emails:

- **Mule/ESB** - This mode has been available in ConSol CM since version 6. It is deprecated in CM versions 6.11 and up. Therefore, this mode is no longer treated in ConSol CM manuals. If you run an older CM system with Mule/ESB, please refer to older manuals. All ConSol CM 6.11 systems should run in NIMH mode. Please refer to older Administrator manuals to learn how to switch from Mule/ESB to NIMH.
- **NIMH** - *New Incoming Mail Handler*, available since version 6.9.4. This is the only available incoming email mode in ConSol CM version 6.11 and up.

The active email mode is set in the system property cmas-core-server, nimh.enabled:

- **true** - NIMH mode
- **false** - Mule/ESB mode

The sending of emails, i.e., the SMTP server configuration, is not influenced by the incoming email mode.

Email Configuration Using NIMH

NIMH, the ***New Incoming Mail Handler***, is a proprietary module of ConSol CM. The following picture provides an overview of all components.

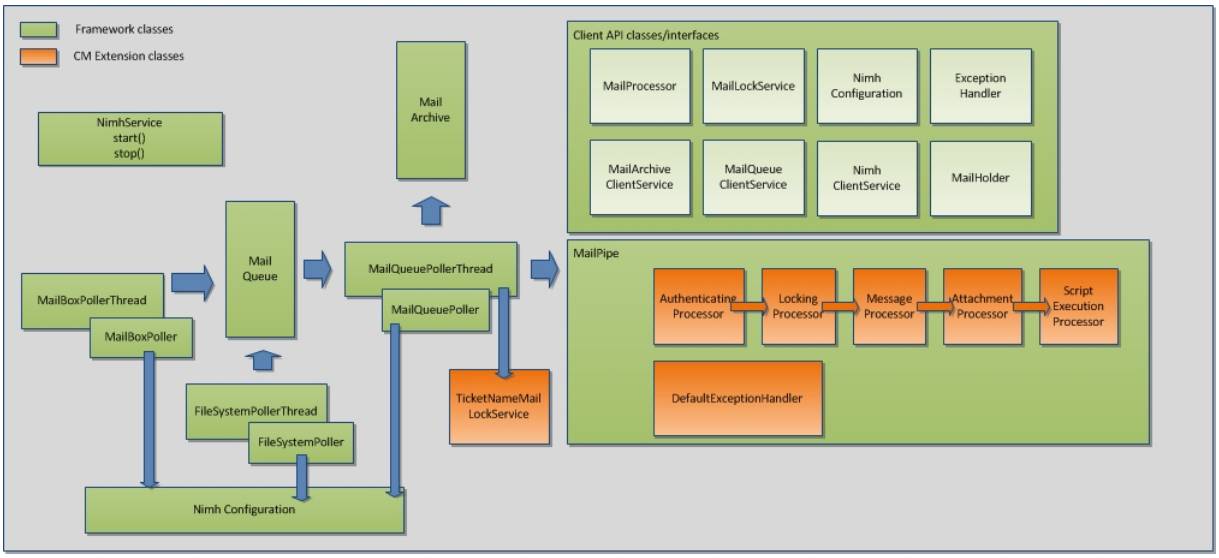


Figure 447: *NIMH components*

When NIMH is enabled, the following apply:

- NIMH runs as a (single) ConSol CM service, see section CM Services.
- Emails from mailboxes on a mail server are retrieved by ConSol CM using the MailBoxPoller.
- Emails can also be fetched from a data directory using the FileSystemPoller.
- All ConSol CM emails are stored in the ConSol CM database (nothing is stored in the file system).
- The MailQueuePoller retrieves the emails from the database and forwards them to the core ConSol CM system where the emails run through the email script pipeline.
- Emails which could not be processed are stored in a separate database table and can be re-sent to the ConSol CM system. See section Management of Email Backups with NIMH.
- NIMH-specific system properties are used (they are added automatically to the system configuration during an update to version 6.9.4 and up):
    - General NIMH properties
    - Default mailbox properties which are used when a property is not set as a mailbox-specific property
    - Mailbox-specific properties for each mailbox which has to be retrieved
    - FileSystemPoller properties
    - MailQueuePoller properties

The following figure provides an example of NIMH properties. Please also refer to the detailed explanation of NIMH system properties in List of System Properties by Area).

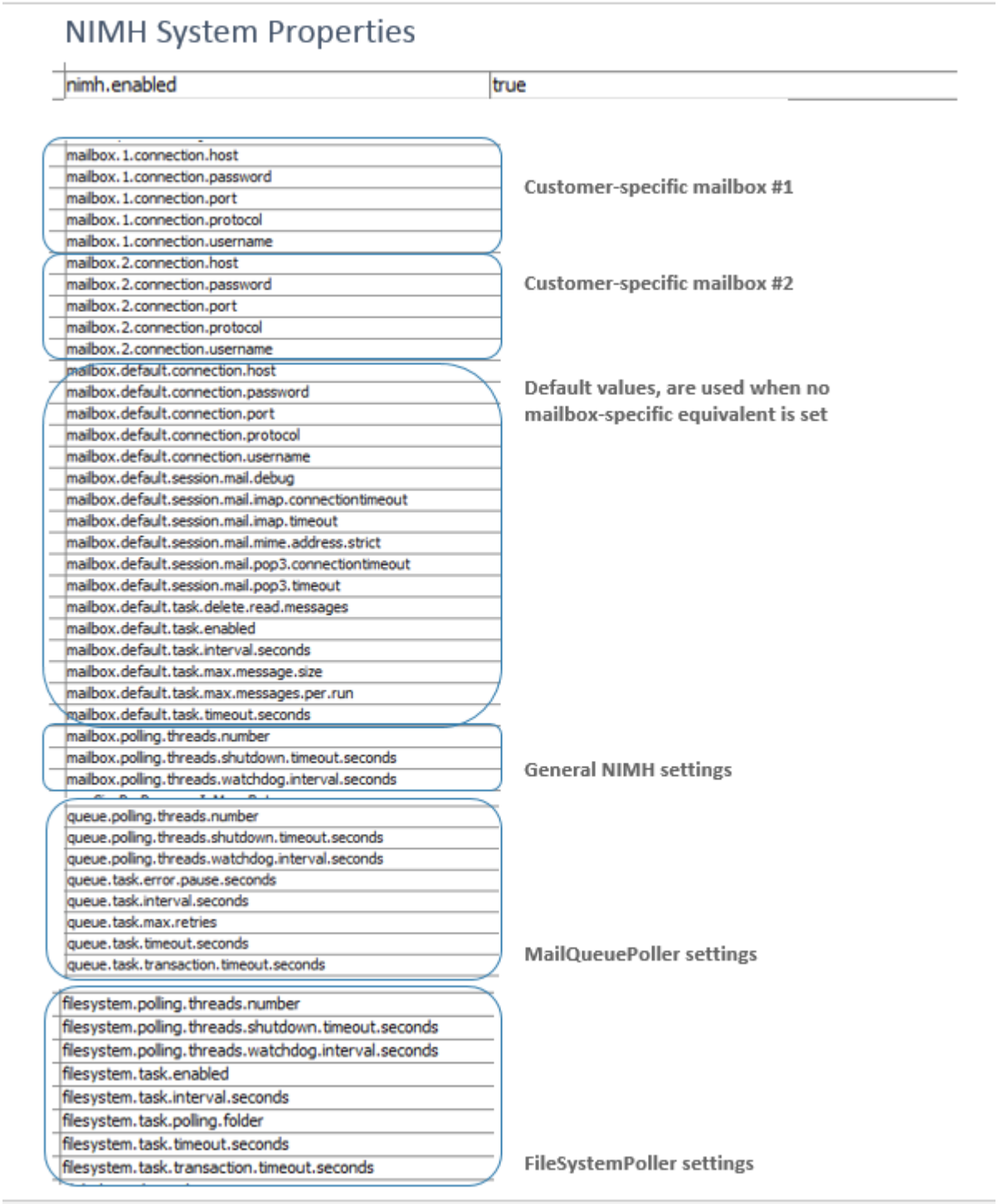


Figure 448: *System properties for NIMH, 1*

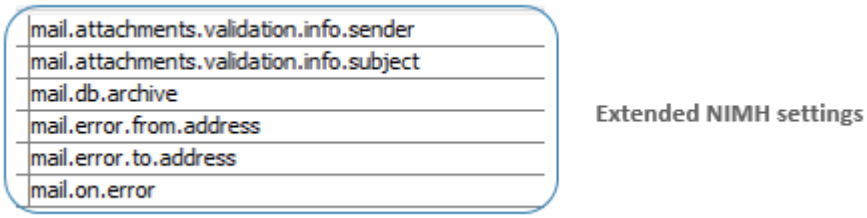


Figure 449: *System properties for NIMH, 2*

F.7.2 Email Backups

F.7.2.1 Introduction

Incoming emails which could not be processed are stored in the CM database.

You as an administrator can then try to re-send the emails to the system manually. The emails stored here can also be deleted, e.g., spam emails.

IMPORTANT INFORMATION

Since ConSol CM version 6.9.4, there are two modes to receive incoming emails:

- **Mule/ESB** - This mode has been available in ConSol CM since version 6. It is deprecated in CM versions 6.11 and up. Therefore, this mode is no longer treated in ConSol CM manuals. If you run an older CM system with Mule/ESB, please refer to older manuals. All ConSol CM 6.11 systems should run in NIMH mode. Please refer to older Administrator manuals to learn how to switch from Mule/ESB to NIMH.
- **NIMH** - *New Incoming Mail Handler*, available since version 6.9.4. This is the only available incoming email mode in ConSol CM version 6.11 and up.

The active email mode is set in the system property cmas-core-server, nimh.enabled:

- **true** - NIMH mode
- **false** - Mule/ESB mode

The sending of emails, i.e., the SMTP server configuration, is not influenced by the incoming email mode.

F.7.2.2 Email Backups in the Admin Tool

All emails which could not be processed are listed in the navigation item *Email Backups* in the navigation group *Email*.

Usually, when an email cannot be processed, this is caused by one of the following reasons:

- One of the email scripts (see section Scripts of Type Email) has a bug and therefore the email which has been fetched from the mail server cannot be processed further in CM.
- The attachment of the email is too large. For NIMH, the size limit is defined by the system property cmas-nimh, mailbox.default.task.max.message.size or by a mailbox-specific value, e.g., *cmas-nimh*, `mailbox.1.task.max.message.size`.

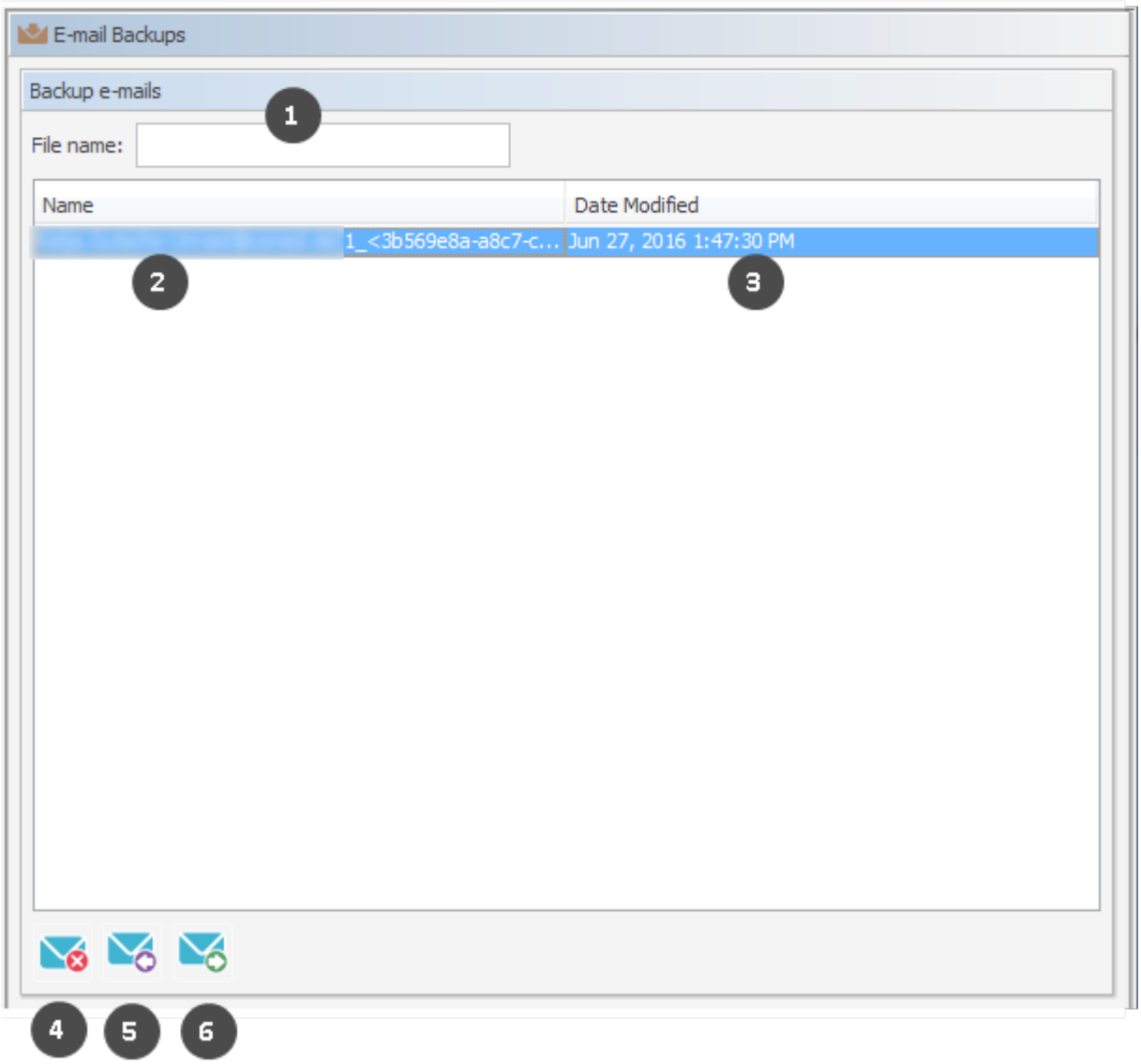


Figure 450: *ConSol CM Admin Tool - Email, Email Backups*

The list panel for unparsable emails contains the following elements:

- **File name (1)**
  This field provides a filter. When you enter the name or part of the name of email files, only the matching file names will be displayed in the list.
- **Name (2)**
  The name of the email file (usually with an .eml extension).
- **Date modified (3)**
  The last date when the file was modified. Usually the date when the email was stored on the ConSol CM server.

In an error-free ConSol CM system, the list panel for unparsable emails should be empty. If there are emails listed, an error has occurred in the processing of those incoming emails. Please see section Scripts of Type Email for a detailed explanation of the processing pipeline.

You can perform the following actions with email backups:

- **Delete (4)**
  To delete an email from the list, select the list entry and click *Delete*. Please keep in mind that the information will be lost! It will not be saved or transferred to ConSol CM in any way!
- **Re-Send (Import) (5)**
  You can also re-send (*import*) the email to the processing pipeline (e.g., when a script was not working correctly and has been fixed) by selecting it in the list and by clicking *Resend*.
- **Export to file system (6)**
  You can export an email as .eml file to the file system of the machine where the Admin Tool is running (e.g., for further analysis of an error-prone email).

F.7.2.3 Management of Email Backups with NIMH

Incoming emails which could not be processed are stored in the ConSol CM database, in the following table (as .eml files):

```
cmas_nimh_archived_mail
```

The system behavior concerning email backups with NIMH is controlled by the system property cmas-nimh-extension, mail.db.archive.

If the property is set to “true”, all incoming emails are archived in the database. If the property is set to “false”, the emails which can be processed are deleted after processing. The emails which cannot be processed will be stored in the database as long as they have neither been processed nor deleted (which can be achieved by reprocessing or deleting such emails, see section above). An email database entry will be deleted automatically after being successfully processed.

F.7.3 Email Encryption

F.7.3.1 Introduction

Due to security policies, it might be required to encrypt email traffic (including the emails which are sent and received by the ConSol CM installation) using standard S/MIME encryption. If the topic is new to you, you might want to read some articles about it, e.g. the Public-key cryptography article in Wikipedia.

In order to enable the use of encrypted emails with ConSol CM, you first have to enable the email encryption in the system:

1. **Mandatory:**
   Set the system property cmas-core-server, mail.encryption to “true”. It is set to “false” as default value. This is the basic configuration to enable email encryption for the entire system.
2. **Optional:**
   Set the page customization attribute mailEncryptionAvailable to “true”. This activates an option in the Web Client to choose whether an email should be encrypted.

General Explanation about Email Encryption in ConSol CM

There are two types of certificates:

- **Server certificates, mainly used for incoming emails**
  The public key (in the certificate) and the private key of the receiving email address can be manually imported into the CM system from a PKCS12 file (.p12 or .pfx). If the certificate is password-protected, the administrator has to provide this password when the certificate is imported.
- **Client certificates, used for outgoing emails**
  For outgoing emails, the certificates (public keys) of the receiving email addresses are required. These certificates can be imported into the CM system in two ways:
  - **Manually**
    By selecting (uploading) the respective X.509 file (.cer or .crt)
  - **Automatically**
    From an LDAP repository where it is stored in the correct format (i.e. in the same format which is required for the file import). This automatic retrieval of the requested certificate can be performed on-the-fly when the email is sent.

⚠ The certificates discussed here are used for email encryption only and **not** for the access of ConSol CM (as email client) to the email server! That has to be managed using certificates which are stored in the security store of the application server.

The following figure shows the basic principle of email encryption for incoming and outgoing emails in ConSol CM.

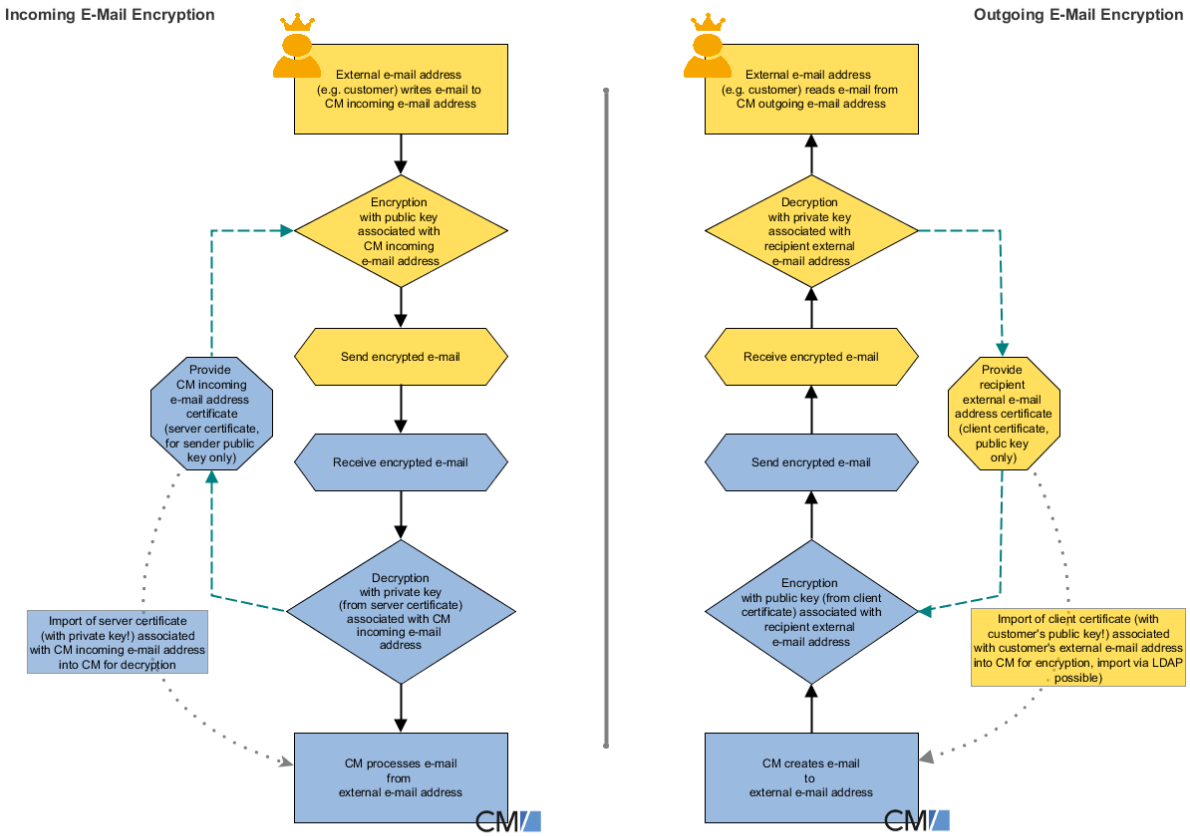


Figure 451: *Basic principle of ConSol CM email encryption*

Requirements

- The client certificate must contain the email address of the customer (receiver of the email) in the attribute *SubjectDN* (*E=* or *EMAILADDRESS=*) or the *X509v3 Subject Alternative Name* element from the *Extensions* section of the certificate.
- *Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files* has to be installed on the server and on the machine where the Admin Tool is started. This is required to enable the Admin Tool to import certificates.
- *X.509 Base64*-encoded certificates are supported.

Certificate Import from LDAP (Available for Client Certificates)

If LDAP is configured, ConSol CM will look up the client certificate for the requested contact in the LDAP repository. This is done as follows:

1. Someone tries to send an encrypted email.
2. The cryptography service looks for a client certificate (with the public key) of the recipient.
3. If a client certificate is found, the email is encrypted using the client's public key and sent.

4. If a client certificate is not found in the Admin Tool or it has expired, it is looked up in the LDAP repository.
5. If it is found, it is imported to ConSol CM and the email is encrypted and sent.
6. If it is not found, the email is sent unencrypted.

The following configuration properties have to be set to enable certificate lookup via LDAP:

- ldap.certificate.basedn
- ldap.certificate.searchattr
- ldap.certificate.content.attribute

Please see section *LDAP certificate parameters* in System Properties for details.

F.7.3.2 Certificate Management in the Admin Tool

In the Admin Tool, the navigation items *Server Certificates* and *Client Certificates* in the navigation group *Email* are used to configure the CM environment for email encryption.

Server Certificates

Server certificates are used to decipher **incoming** email messages. In some exceptional cases, they are also used to encrypt outgoing emails: if one of the recipients is the same as one of the incoming email accounts, the server certificate will be used to encrypt that message. Server certificates each contain the public and the private key for the given email address. If you define an incoming email account (see section above), you have to upload a server certificate for that email address (or for all email addresses covered by this mailbox) to be able to receive encrypted messages (because the server certificate contains the respective private key). If you have several incoming accounts, you either have to upload a server certificate for each of them or you can upload one certificate with all required email addresses.

When you open the navigation item *Server certificates*, a list of all existing server certificates is displayed. To add a new server certificate, click the *Add* button and use the file browser to find the required certificate. The certificate is validated before it is imported. If there are any incompatibilities, an error message is displayed and the certificate is not imported.

Supported formats for server certificates are:

- PKCS #12 archive file containing certificate (public) and private key (password protected). Supported filename extensions for PKCS #12 files are:
  - .p12
  - .pfx

Client Certificates

A client certificate contains only the public key of an external recipient (e.g., a customer). It allows encrypting messages which are sent to that user, i.e. client certificates are used for **outgoing** emails.

When you open the navigation item *Client certificates*, a list of all existing client certificates is displayed. To add a new client certificate, click the *Add* button and use the file browser to find the required certificate. The certificate is validated before it is imported. If there are any incompatibilities, an error message is displayed and the certificate is not imported.

Supported formats for client certificates are:

- X509 standard format.
  Supported file name extensions for X.509 certificates are:
  - .cer
  - .crt
  - .der
  - .pem

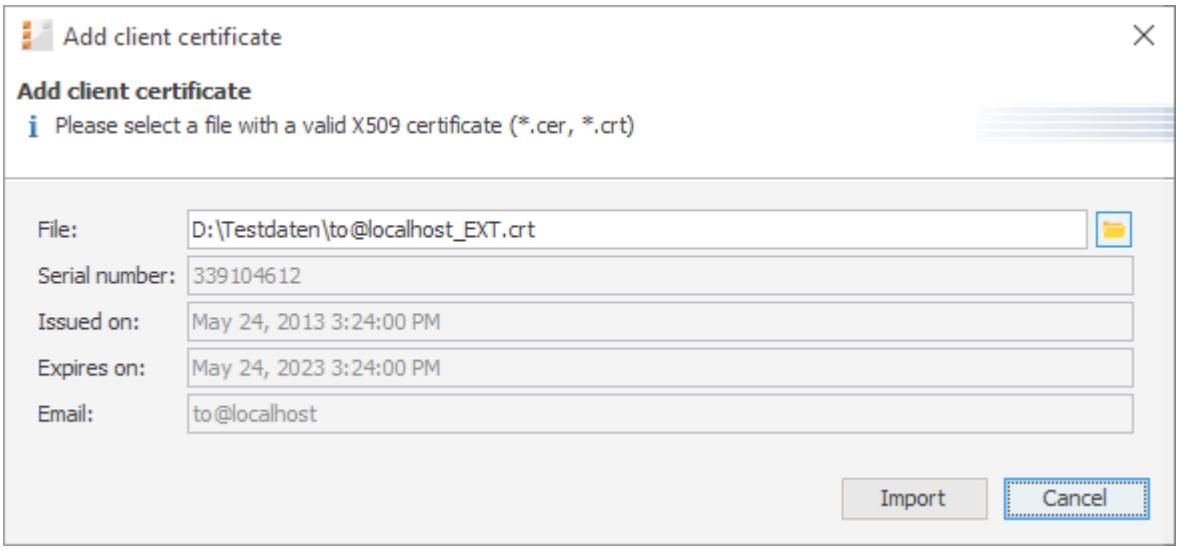


Figure 452: *ConSol CM Admin Tool - Email, Client Certificates: Pop-up window for adding a client certificate*

Here are some example use cases:

1. An engineer works in the ConSol CM Web Client and writes an encrypted email using the Ticket Email Editor. When the *Send* button is clicked, the ConSol CM system looks up the receiver address in the list of email addresses under *Client certificates* and uses the public key of the recipient to encrypt the outgoing email. If ConSol CM cannot find a matching certificate (the email address is not in the certificate list), the certificate for the email address is loaded from LDAP. If no compatible certificate is found there either, the email is sent unencrypted.
   If one of the recipients is one of the incoming email accounts, the server certificate (public key) will also be used to encrypt that message.
2. ConSol CM receives an email and checks the To address. If it is found in the list under *Server certificates*, ConSol CM uses the private key given in this certificate to decrypt the message and to either create a new ticket or append the message to an existing ticket.

Sending Encrypted Emails

Choosing if Emails Should Be Sent Encrypted from the Web Client

If the page customization property mailEncryptionAvailable has been set to “true”, a checkbox *Send encrypted* is available in the Ticket Email Editor in the Web Client. Thus, the user can choose whether the email should be sent encrypted.

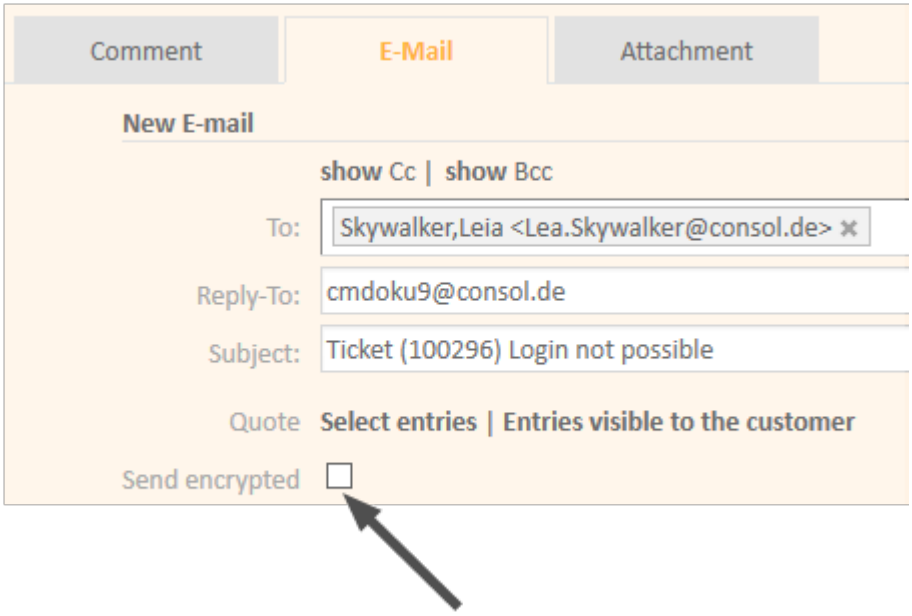


Figure 453: *ConSol CM Web Client - Send encrypted email*

Sending an Encrypted Email from the Workflow

An encrypted email can be sent by using the method `enableEncryption()`. Please see the *ConSol CM Process Designer Manual* for a detailed explanation.

Sending Encrypted Emails by Default

If the system property cmas-core-server, mail.encryption is set to “true”, all outgoing emails from the workflow and Web Client are encrypted by default.

If users would like to send selected emails unencrypted, they can uncheck the checkbox *Send encrypted* in the Web Client. For emails sent by the workflow the method `disableEncryption()` can be used for this purpose.

F.8 Script and Admin Tool Template Administration

In this chapter, you will learn how to work with scripts and templates that are stored in and managed with the Admin Tool.

Scripts are used in various contexts in ConSol CM, particularly in the Process Designer within workflows. Please see the *ConSol CM Process Designer Manual* for a detailed explanation of this topic. However, various scripts are also stored in the Admin Tool, in the *Scripts* section. This is explained in section Admin Tool Scripts.

Templates are also stored in several locations:

- Accessible via Web Client:
  - In the Text Template Manager (email templates)
  - In the Document Template Manager (Microsoft Word or OpenOffice templates in CM/Doc)
- Accessible via Admin Tool:
  - In the *Scripts and Templates* section
    - Some special email templates
    - Templates for customer data
    - Templates for resource data
    - Password reset template
    - CM/Phone templates
    - <more templates, depending on customization>

For an explanation of using email templates using the Text Template Manager, and for configuring CM/Doc, please refer to section Working with Text Templates. For an explanation of templates in the Admin Tool, please read section Admin Tool Templates.

F.8.1 Admin Tool Scripts

F.8.1.1 Introduction to Scripts in the Admin Tool

Scripts are stored in the *Scripts and Templates* section of the Admin Tool. They are written in Groovy and should only be edited by experienced ConSol CM consultants and administrators.

To work with scripts, open the navigation item *Scripts and Templates* in the navigation group *System* in the Admin Tool. The tab *Scripts* will be shown.

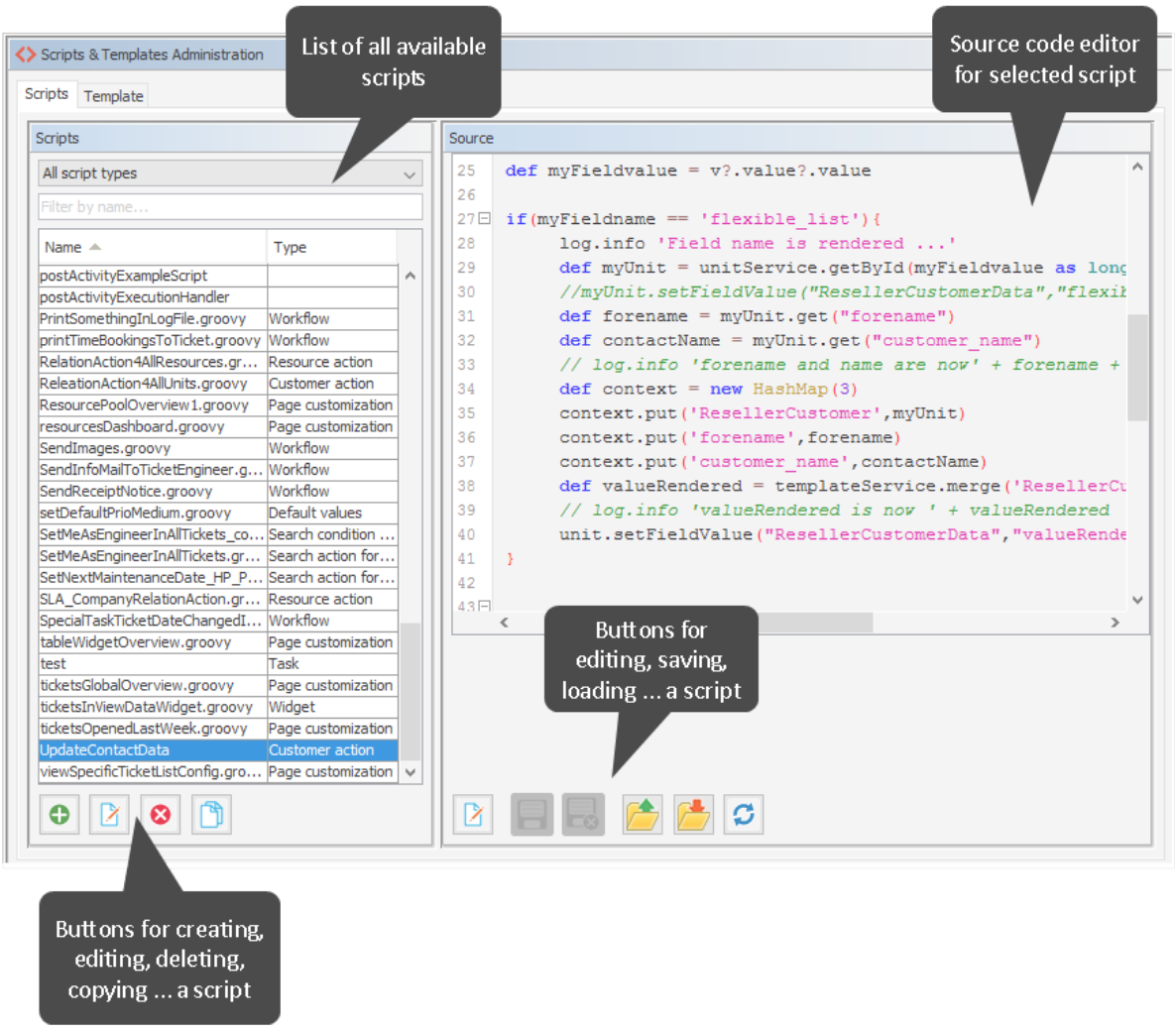


Figure 454: *ConSol CM Admin Tool - System, Scripts and Templates*

On the left you see the list of all scripts. The list can be filtered using

- the drop-down menu where the script type can be selected.
- the input field for a text filter for the script name

Two parameters have to be set for each script:

- **Name**
  This is the name by which the script will be referenced, e.g., from the workflow or from other objects like queues.
- **Type**
  The script type. One of the following possible script types has to be selected:
  - **Search action for customers**
    This script type is part of the Action Framework, namely the search actions, and is described in section Action Framework - Search Actions.
  - **Search condition for customers**
    This script type is part of the Action Framework, namely the search actions, and is described in section Action Framework - Search Actions.
  - **Search action for resources**
    This script type is part of the Action Framework, namely the search actions, and is described in section Action Framework - Search Actions.
  - **Search condition for resources**
    This script type is part of the Action Framework, namely the search actions, and is described in section Action Framework - Search Actions.
  - **Search action for tickets**
    This script type is part of the Action Framework, namely the search actions, and is described in section Action Framework - Search Actions.
  - **Search condition for tickets**
    This script type is part of the Action Framework, namely the search actions, and is described in section Action Framework - Search Actions.
  - **Calendar integration**
    Scripts of this type provide the connection information for the integration of Microsoft Exchange calendars. See section Microsoft Exchange Calendar Integration.
  - **Clone**
    Script which is executed when the *Clone* option is selected for a ticket. Has to be assigned to a queue. See section Scripts of Type Clone for details.
  - **Customer action**
    Script which is executed when a customer action has occurred, see section Action Framework - Customer Actions for details.
  - **Customer condition**
    Script which is executed to evaluate whether a customer action should be made available to the Web Client, see section Action Framework - Customer Actions for details.
  - **Default values**
    Scripts of this type are used to define default values, i.e., values that are (pre)set in data fields when a new ticket is to be created. Please see section Scripts of Type Default Values for details.

- **Dependent enum**
  Scripts of this type are used to define *dependent enums*, structures which provide hierarchical lists. Please see section Scripts of Type Dependent Enum for details.
- **Email**
  Scripts of this type are used to manage incoming and outgoing emails. Please see section Scripts of Type Email for details.
- **Field visualization**
  Scripts of this type are used to configure the visualization of data fields. Please see section Scripts of Type Field Visualization for details.
- **Integration**
  Scripts of this type are used to implement services for the CM *Webhooks* interface. See section Webhooks for details.
- **Page customization**
  Scripts of this type are referenced by page customization settings. Please see sections Page Customization for the Web Client Dashboard and CM/Resource Pool - The Resource Pool Dashboard for details. The script type *Widget* can be used interchangeably with this type. Both types work exactly the same way. However, by using scripts of type *Page customization* for scripts which are referenced by page customization scopes or types and using script type *Widget* for scripts which are used to implement dashboard widgets, you can distinguish the scripts easily in the list of scripts (by sorting by type).
- **PostActivityExecutionScript** (no specific type indicated)
  An (optional) script which is executed after every manual workflow activity. See section PostActivityExecutionScript for details.
- **Relation graph**
  Scripts of this type are used to implement the graph display of relations in either standard sections or the expert section in ticket, customer (contact/company) and resource pages. For a detailed explanation, please refer to section Configuration of the Graphical Representation of Relations.
- **Resource action**
  Script which is executed when a resource action has taken place, see section CM/Resource Pool - Resource Actions for details.
- **Resource condition**
  Script which is executed to evaluate whether a resource action should be made available to the Web Client. See section CM/Resource Pool - Resource Actions for details.
- **Task**
  Scripts of this type are used by the TEF (Task Execution Framework), please see section The Task Execution Framework (TEF).
- **Text Autocomplete**
  This script type is used to implement scripted autocomplete lists. Please see section Scripted Autocomplete Lists for details.

- **Widget**
  This script type is used for scripts which implement dashboards. This might be the dashboard on the Overview page (see section Page Customization for the Web Client Dashboard) or the resource pool dashboard (see section CM/Resource Pool - The Resource Pool Dashboard). The script type *Widget* can be used interchangeably with the script type *Page customization*. Both types work exactly the same way. However, by using scripts of type *Page customization* for scripts which are referenced by page customization scopes or types and using script type *Widget* for scripts which are used to implement dashboard widgets, you can distinguish the scripts easily in the list of scripts (by sorting by type).
- **Workflow**
  Scripts of this type are referenced from the workflow. Please see section Scripts of Type Workflow for details.

The buttons below the list offer the standard Admin Tool functionalities:

- Add a script
- Edit a script
- Delete a script
- Copy a script

On the right you see the *Source Code Editor*. The script which is selected in the list on the left is displayed. Here you can write/edit the script source code when using edit mode.

F.8.1.2 The Source Code Editor

The Source Code Editor provides an editing panel with

- syntax highlighting
- code completion (use CTRL+SPACE, the classes in the locally executed tool, i.e. in the CLASSPATH, will be offered, not all classes and methods in ConSol CM)
- code validation (see display in the *Compilation result* box, since the validation is performed on the server side, all classes and methods of ConSol CM will be checked).
  - The interval between two code checks can be configured using the CM system property cmas-app-admin-tool, script.validation.interval.seconds.
  - The code validation can be disabled per session using the checkbox *Disable validation* below the editor pane

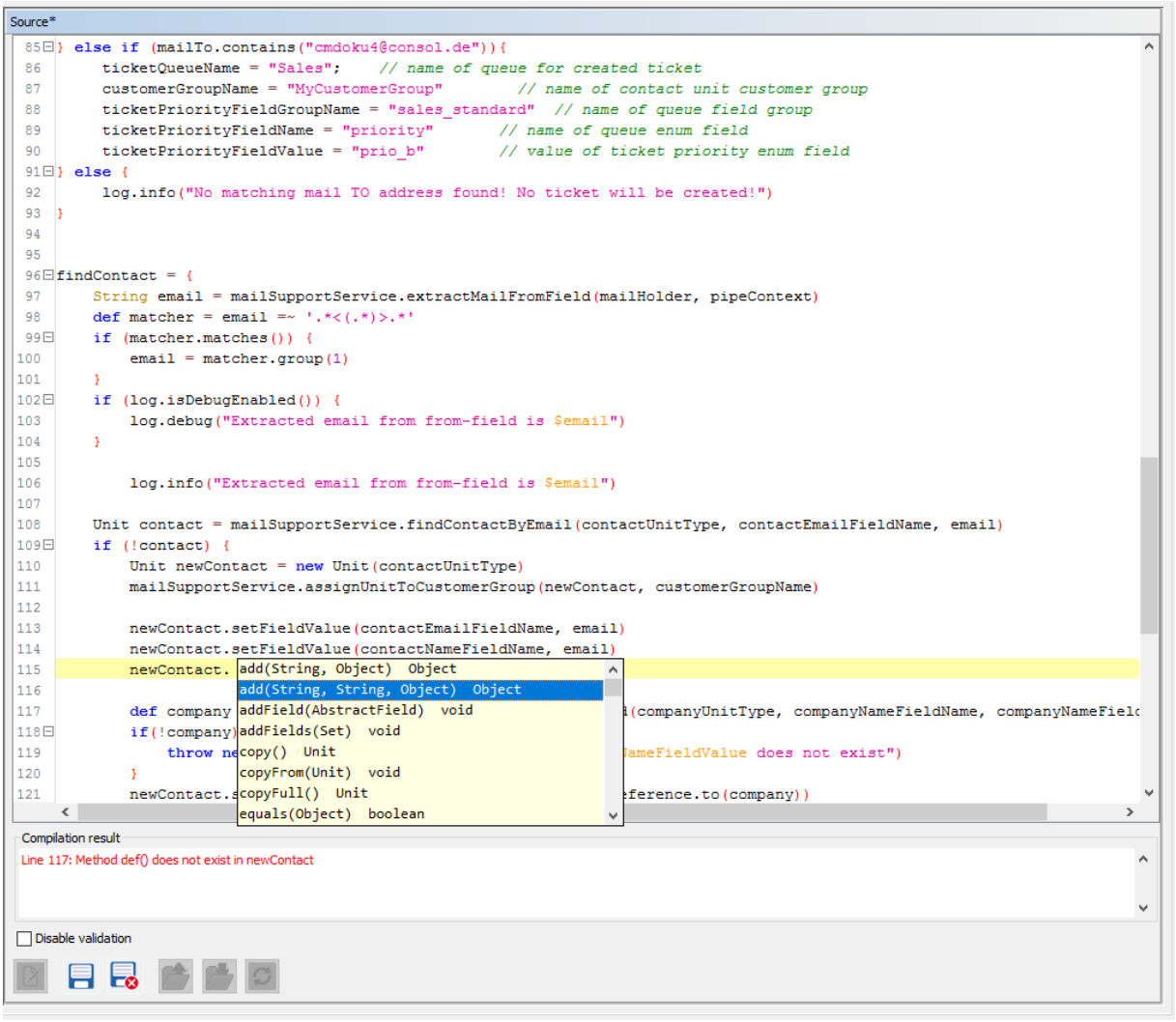


Figure 455: *ConSol CM Admin Tool - System, Scripts and Templates: Source Code Editor*

The lower section of the Source Code Editor has the following buttons:

- **Edit**
  Click this button to switch to edit mode in the Source Code Editor. When you open the navigation item *Scripts and Templates* in the Admin Tool, all scripts are in read-only mode to prevent an administrator from accidentally changing something.
- **Close edit mode and save changes**
  Save the script and quit edit mode, i.e., switch to read-only mode again.
- **Close edit mode without saving changes**
  Switch to read-only mode, without saving any changes you might have made to the source code.

- **Open script from file**
  This option opens a file browser. Two cases are possible:
  - The name of the selected file and the name of the script are identical.
    In this case, the code of the script (in the Source Code Editor) will be completely replaced.
  - The name of the selected file and the name of the script differ.
    In this case, a new script will be created with the name of the selected file. The new script will be displayed in edit mode in the Source Code Editor.
- **Save script to file**
  Here you can save the text of the script as a plain text file in the file system of the machine the Admin Tool is running on.
- **Update script from file**
  This option opens a file browser. The code in the Source Code Editor will be replaced by the text in the file, no matter if the names of the script and the file name are identical or not.

F.8.1.3 Script Types

In the following section, the available script types are explained. Some examples are provided to give you an idea of potential uses for scripts.

The following script types are explained here in this section. Please refer to the links provided in the list above for a complete overview of all script types with links to their respective documentation.

- Scripted Autocomplete Lists
- Scripts of Type Clone
- Scripts of Type Default Values
- Scripts of Type Dependent Enum
- Scripts of Type Field Visualization
- Scripts of Type Email
- Scripts of Type Workflow
- PostActivityExecutionScript

F.8.1.4 Scripted Autocomplete Lists

Please note that the implementation of scripted autocomplete lists has been modified from ConSol CM version 6.11.1.0 to version 6.11.1.1! Please refer to the *ConSol CM Administrator Manual* version 6.11.1.0 if you run this CM version. The current section only treats scripted autocomplete lists in CM version 6.11.1.1!

Introduction

Scripted autocomplete lists can be used to provide drop-down menus with dynamically generated content. Please see the following examples for a first impression.

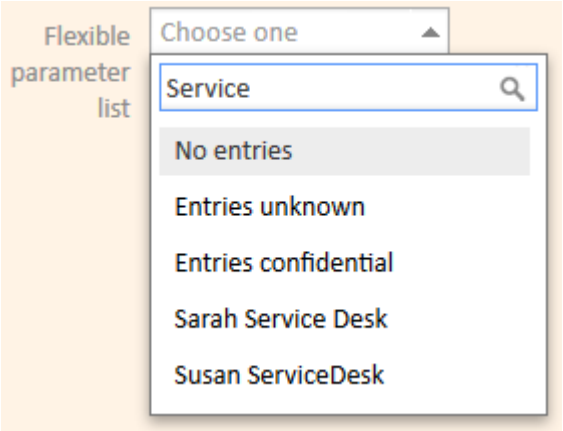


Figure 456: *ConSol CM Web Client - Scripted autocomplete list which offers fixed values plus engineer search result*

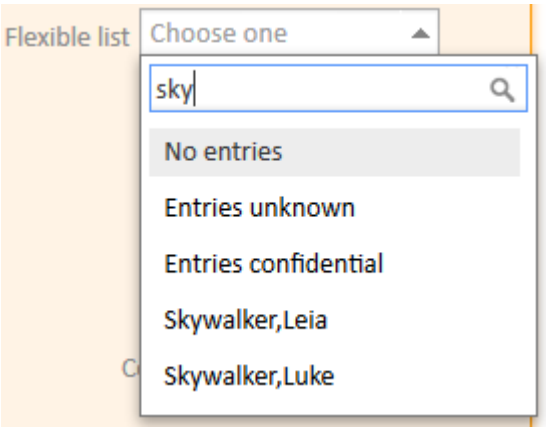


Figure 457: *ConSol CM Web Client - Scripted autocomplete list which offers fixed values plus customer search result*

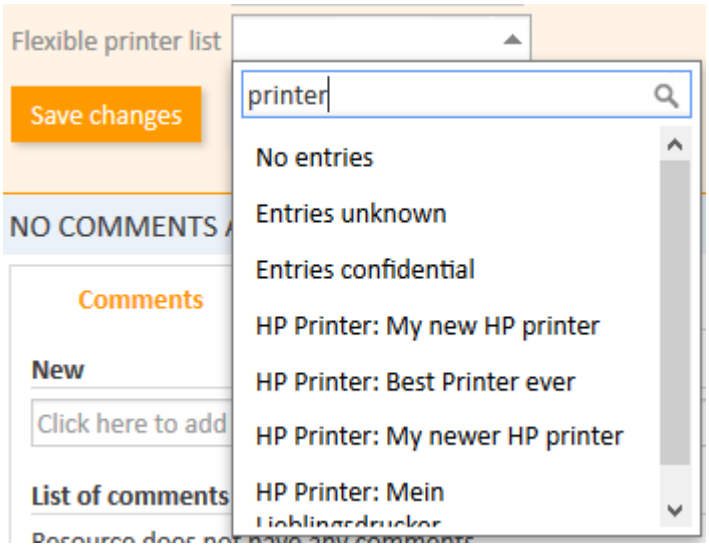


Figure 458: *ConSol CM Web Client - Scripted autocomplete list which offers fixed values plus resource search result*

This type of list can be very helpful to provide a subset of list entries for the engineer instead of providing a huge list which is rather user-unfriendly. In the examples shown above, scripted autocomplete lists are implemented for the following use cases:

1. In a ticket, an engineer should be set, but not in the standard field *engineer* (in the top part of the ticket) but for another reason, e.g., to send an info email to this engineer. Another use case for an engineer list would be an ACF where the next engineer in a process pipeline can be selected.
2. On a customer page, the responsible engineer should be set.
3. On a customer page, the resources/assets which are in use at this customer's site should be registered. A similar list can also be very helpful in an incident ticket where the resource/asset which caused the problem is linked.

The following **object types** can be used as entries for scripted autocomplete lists:

- Fixed strings
- Engineers (please note that this does not have any connection with assigning a ticket to an engineer!)
- Tickets
- Units (= customers, i.e., contacts and companies)
- Resources

The entries for engineers, tickets, customers and resources can be based on search results. In this way, you can customize CM to create the entries for the drop-down menu dynamically.

The display value of the object in the list depends on the type of label which has been defined, static vs. dynamic. Please see section explanation below.

Configuring a Scripted Autocomplete List Using the Admin Tool

Data Field Configuration

In order to implement a scripted autocomplete list, you have to perform the following steps:

Step 1: Define the autocomplete script, i.e., create a script of type *Text Autocomplete*. The script can be modified and extended later on, but you need to have an existing script to be able to assign it in step 2.

Step 2: Define the data field which should contain the list, i.e., define one of the following fields.

- Ticket field
- Customer field
- Resource field

The field has to be of type *autocomplete* and has to have a reference to the script which should implement the autocomplete functionality.

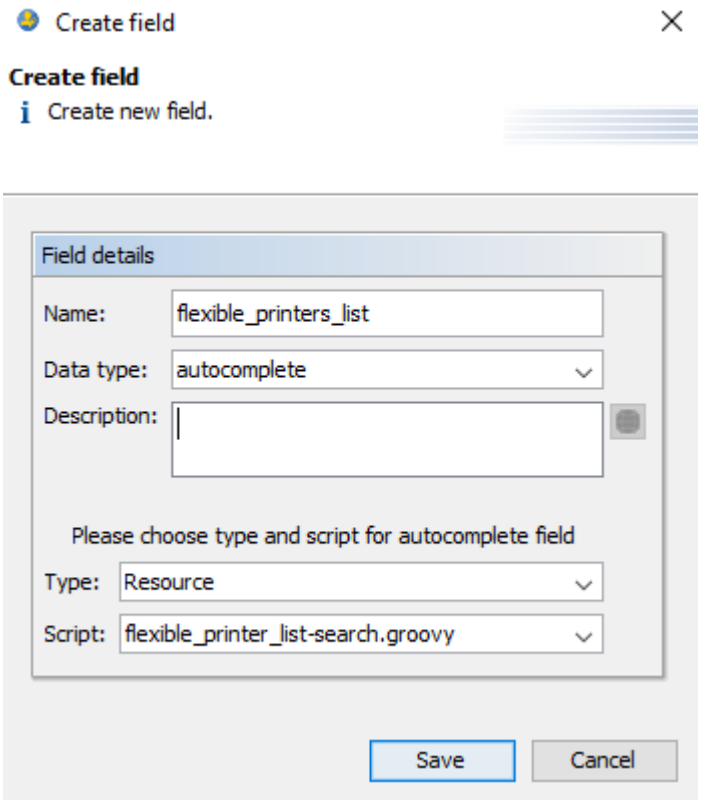


Figure 459: *ConSol CM Admin Tool - Resources, Data Models: Creating a new resource field of type autocomplete*

The following values are relevant for an entry in an autocomplete list, i.e., for one list value:

- **Internal value**
  For fields of type *Ticket*, *Unit*, *Resource*, or *Engineer*, this is the reference to the respective

object. For fields of type *String*, this is the string itself.

- **Display value**
  This is the value shown to the user in the Web Client or CM/Track V2. The display value can be:
  - **dynamic**
    The display value is retrieved from the referenced object itself.
  - **static**
    A string is defined which is always used as display value.

For dynamic display values, the following **templates** are used for rendering the entries in the drop-down menu:

- **Engineers:**
  - *engineer description template name*
- **Customers:**
  - *Customer search result template*
  - If customer search result is not present: Default template
- **Resources:**
  - *Search template*
- **Tickets:**
  - Fixed internal template with ticket name and subject

In order to define a dynamic or static display value, use the respective method for adding the value to the result set.

To define a static display value, i.e., the display value is **not** updated when there are changes to the referenced object, use the method signature with both the object and a string as parameters in order to add the object and a value from the referenced object or any string to the result set. For example:

```
resultset.add(ticket, ticket.getSubject());
```

```
resultset.add(ticket, "some description");
```

To define a dynamic label, i.e., the display value is retrieved from the referenced object and is updated in case of relevant changes in the referenced object, use the method signature with only the object as a parameter in order to add only the object itself to the result set. For example:

```
resultset.add(ticket);
```

The Text Autocomplete Script

A script of type *Text Autocomplete* has to implement the method `onSearchInput`. When you create a new script of type *Text Autocomplete*, the method template is inserted automatically and you only have to fill in the “intelligence” of your script.

```
import com.consol.cmas.common.model.autocomplete.script.*
/**
* This method is called by web client when the user clicks or types into the
 autocomplete field
* @param pSearchStr the search String types or NULL if user clicked into field
 without typing
* @param pKey fieldKey
* @param pContext (Ticket/Unit/Resource) holder containing current values set on
 the form
*/
ScriptAutocompleteResult onSearchInput(String pSearchStr, FieldKey pKey, Context
 pContext) {
   return ScriptAutocompleteResult.noResults('No Results')
}
/**
* For custom-fields in ListFields row index is provided as well
*/
ScriptAutocompleteResult onSearchInput(String pSearchStr, FieldKey pKey, int
 pRowIndex, Context pContext) {
   return ScriptAutocompleteResult.noResults('No Results')
}
/**
* This method controls the value displayed when this field enters edit mode on
 screen. Can be used to obtain the beautified value based on the current internal
 value
* for display purposes.
* @param pValue the stored value of the field
* @param pKey fieldKey
* @param pContext (Ticket/Unit/Resource) holder containing current values set on
 the form
* @return - the string which contain beautified label or one of the domain objects:
 Ticket/Unit/Resource,
* for domain object beautified label will be taken from standard template of the
 object.
* @deprecated - For new AutocompleteField this method is ignored. Labels are stored
 in DB so this method is not needed any more
* The old scripted autocomplete fields still support this method, but it will be
 removed with next release
*/
Object onEditDisplayEntered(String pValue, FieldKey pKey, Context pContext) {
   if (!pValue) return null; // display nothing (current behaviour)
      return "Beautified: " + pValue;
}
/**
* @param pValue the stored value of the field
* @param pKey fieldKey
* @param pContext (Ticket/Unit/Resource) holder containing current values set on
 the form
* @return - the string which contain beautified label or one of the domain objects:
 Ticket/Unit/Resource,
* for domain object beautified label will be taken from standard template of the
 object.
```

```
* @deprecated - For new AutocompleteField this method is ignored. Labels are stored
 in DB so this method is not needed any more
* The old scripted autocomplete fields still support this method, but it will be
 removed with next release
*/
Object onEditDisplayEntered(String pValue, FieldKey pKey, int pRowIndex, Context
 pContext) {
  if (!pValue) return null;
     return "Beautified:" + pValue;
}
```

Code example 81: *Template for Admin Tool scripts of type Text Autocomplete*

What you have to do is to fill the `ScriptAutoCompleteResult` object (called *resultset* in the examples) which will be returned. This object represents the drop-down menu. You can add objects of one type to a resultset, the type being defined by the result set type which has been entered during the definition in the Admin Tool. In the example above, the type of the result set is *Resource*, so only resources can be added to the resultset which is returned for the list. If you need other values, for example, because you want to display a message (i.e., string only) for which at least a minimum number of characters have to be entered, a separate resultset has to be returned. This can of course be done in the same script.

To display a message when no results have been found, use the `noResults(StringMessage)` method.

Please see the ConSol CM API Doc for a detailed overview of the methods of the Java class `ScriptAutoCompleteResult`.

The objects which are added to the resultset can be derived from CM-internal data (e.g., from an engineer search), or they can also originate from a search in external sources. In this way, you can, for example, offer a search list with names of an external database.

In order to work with input the engineer has provided (e.g., the engineer has started typing a customer name into the string field) without this input being saved yet, you have to work with the `pContext` object. Depending on the page which is open (i.e., depending on the context), the `pContext` contains different data and can return different objects.

- Unit
    - Available only on unit (contact/company) pages:
    - Example: myunit = pContext.getUnit() or myunit = pContext.unit
- Engineer:
    - Availability different for current and for ticket engineer.
    - Current engineer only via engineerService. Example: def myeng = engineerService.current
    - Ticket engineer via ticket: def myeng = pContext.ticket?.engineer
- Ticket

  available on/in

    - Ticket create page
    - Ticket edit page
    - ACF
- Resource
    - Available only on resource page
    - Example: myres = pContext.resource
- Strings
    - Always available, no connection with pContext
    - Example: "p1"

The following example script creates a list (i.e., drop-down menu) where one of the following objects is displayed:

- A resultset which informs the user that at least three characters have to be entered.
- A resultset which contains only the message that no results have been found, i.e., no matches could be retrieved based on the characters already entered by the user.
- A resultset which represents the search result in the resources based on the characters the user has already entered. This is the "main" resultset of the script.

```
import com.consol.cmas.common.model.autocomplete.script.*
import com.consol.cmas.common.model.resource.ResourceCriteria
/**
* This method is called by web client when the user clicks or types into the
 autocomplete field
* @param pSearchStr the search String types or NULL if user clicked into field
 without typing
* @param pKey fieldKey
* @param pContext (Ticket/Unit/Resource) holder. More about this later in the
 specification (Context section)
*/
log.info '##### Starting Autocomplete script ... #####'
ScriptAutocompleteResult onSearchInput(String pSearchStr, FieldKey pKey, Context
 pContext) {
log.info '##### Class of pContext is now: ' + pContext.class
log.info '##### Resource of pContext is now: ' + pContext.resource
ScriptAutocompleteResult resultset = new ScriptAutocompleteResult();

// Add matching customer when entering more than 2 characters
if (pSearchStr.length() < 3) {
   return new ScriptAutocompleteResult("NOTE: Please type at least 3 characters!");
} else
   log.info ' Building RESOURCE criteria ...'
   ResourceCriteria criteria = new ResourceCriteria();
   criteria.setPattern("*"+pSearchStr+"*")
   def resources = resourceService.getByCriteria(criteria)
   if(resources.size()<1){
      return ScriptAutocompleteResult.noResults('No matching resources found')
   } else {
      resultset.add(resources);
   }
      return resultset;
}
```

Code example 82: *Script of an autocomplete list for a resource field*

In the script above, the resource objects are added to the resultset, i.e., we work with dynamic display values. That means the display value of a resource object in the list always represents data of the original object, rendered by the respective template.

The list which is implemented by the script shown above is displayed in the Web Client as follows:

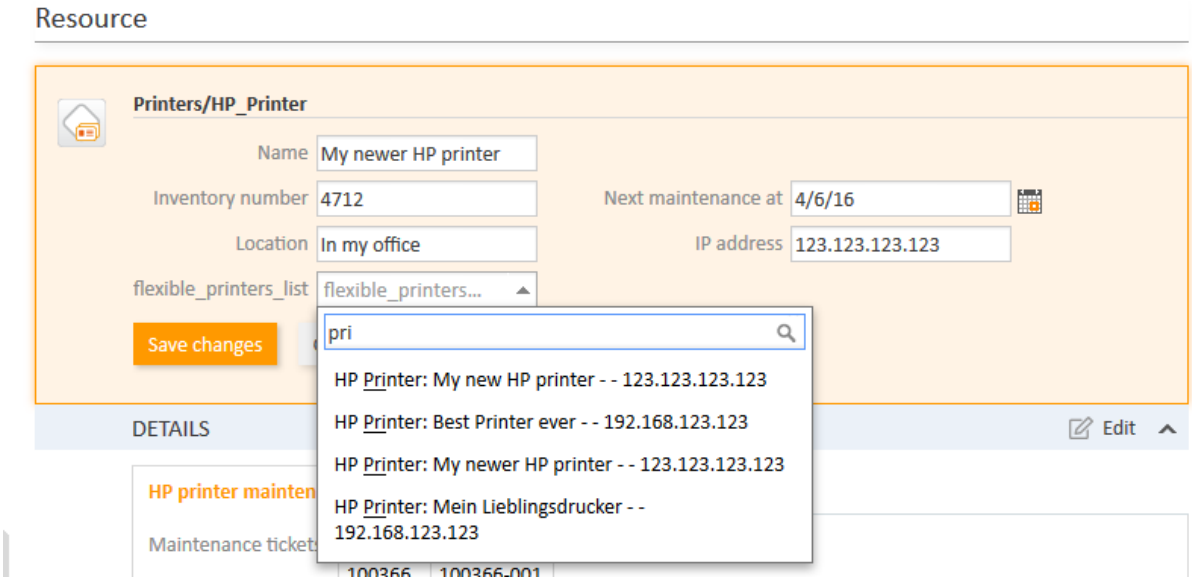


Figure 460: *ConSol CM Web Client - Scripted autocomplete list of type resource*

You can control the number of characters the engineer has to type before the search is initiated. This is done using the following if-statement, which you also find in the example above.

```
if (pSearchStr.length() < 3) { }
```

In this example, the engineer will have to type at least three characters into the string field before the search is initiated.

Retrieving and Setting Values of Autocomplete Fields

The value of an autocomplete field can be accessed later using the new class `AutocompleteValue`:

```
AutocompleteValue myAutocompleteValue = <object>.get("<field group>.<field>");
myAutocompleteValue.getInternalValue();
myAutocompleteValue.getDisplayValue();
```

Example script which can be used for test purposes. The script only displays the value of an auto-complete ticket field with engineer data in the log file.

```
// displayFieldValues.groovy
// Test display of filed values in log file ... for test purposes only!
// display engineer of ticket field (not engineer field in header!)
def ticket = workflowApi.ticket

AutocompleteValue myAutocompleteValue = ticket.get("serviceDesk_fields.flexible_
 engineers_list")
def intValue = myAutocompleteValue.getInternalValue()
log.info 'INT VALUE IS NOW ' + intValue
def dispValue = myAutocompleteValue.getDisplayValue()
log.info 'DISP VALUE IS NOW ' + dispValue
```

Code example 83: *Script which writes values of an autocomplete ticket field with engineer data into the server.log file*

It is also possible to set the values of an autocomplete field using the API. The following example sets a dynamic label. To set a static label, omit the second line of the code example.

```
AutocompleteValue myAutocompleteValue = new AutocompleteValue("<internal
 value>","<display value>");
myAutocompleteValue.setDynamicDisplayValue(true);
<object>.set("<field group>.<field>", myAutocompleteValue);
```

Access to Values of Scripted Autocomplete Lists in CM/Track

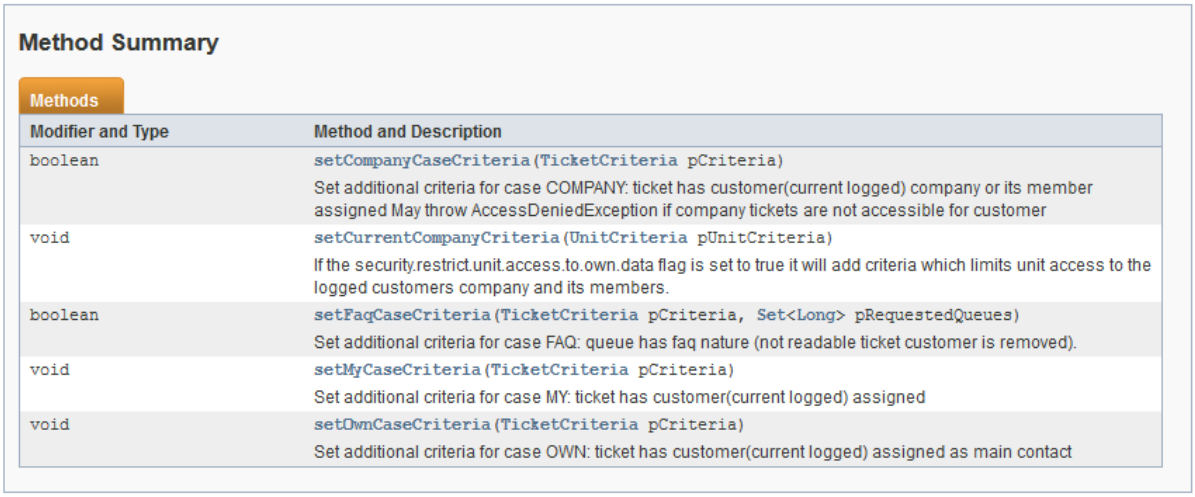
Permissions are checked during the execution of the autocomplete script when a field is edited in the Web Client and CM/Track V2, i.e., the user can only see and select entries referring to objects which he is allowed to view according to his assigned roles. When viewing autocomplete fields, permissions are not checked. Therefore, the user might see labels referring to objects for which he has no permissions in autocomplete fields and history entries.

Note regarding permissions in CM/Track V2:

Permissions related to scripted autocomplete fields work the same way as for the Web Client. CM/Track V2-specific restrictions, namely the limitation that a user only sees his own tickets (and tickets of other contacts of his company, if configured) and contacts of his company, are not part of the default configuration. This means that, by default, the user has permissions to see and select entries referring to tickets belonging to queues for which the Track user has at least read permissions and customers belonging to customer groups for which the Track user has at least read permissions. In short, the customer would see data belonging to other customers.

If this is not desired, the autocomplete script must be adapted using methods of the new class `customerSecurityCriteriaBuilder`, see script example below for more details.

The class `customerSecurityCriteriaBuilder` has been added to the ConSol CM API. It provides the following methods to control permissions for autocomplete results in CM/Track V2:



Method Summary

Methods

Modifier and Type	Method and Description
boolean	setCompanyCaseCriteria(TicketCriteria pCriteria) Set additional criteria for case COMPANY: ticket has customer(current logged) company or its member assigned May throw AccessDeniedException if company tickets are not accessible for customer
void	setCurrentCompanyCriteria(UnitCriteria pUnitCriteria) If the security.restrict.unit.access.to.own.data flag is set to true it will add criteria which limits unit access to the logged customers company and its members.
boolean	setFaqCaseCriteria(TicketCriteria pCriteria, Set<Long> pRequestedQueues) Set additional criteria for case FAQ: queue has faq nature (not readable ticket customer is removed).
void	setMyCaseCriteria(TicketCriteria pCriteria) Set additional criteria for case MY: ticket has customer(current logged) assigned
void	setOwnCaseCriteria(TicketCriteria pCriteria) Set additional criteria for case OWN: ticket has customer(current logged) assigned as main contact

Figure 461: *ConSol CM API Doc of class customerSecurityCriteriaBuilder*

Example script: The following autocomplete script limits the search results of an autocomplete field used in CM/Track V2 to tickets belonging to the user.

```
ScriptAutocompleteResult onSearchInput(String pSearchStr, FieldKey pKey, Context
 pContext) {
def result = new ScriptAutocompleteResult();
def criteria= new TicketCriteria();
criteria.setPattern(pSearchStr+ "*");
customerSecurityCriteriaBuilder.setMyCaseCriteria(criteria);
for (Ticket ticket : ticketService.getByCriteria(criteria)) {
   result.add(ticket);
}
return result;
}
```

Code example 84: *Script of an autocomplete list which can be displayed in CM/Track V2*

General Information About Scripted Autocomplete Fields in CM Version 6.11.1.1

The following limitations exist when using the new way of configuring scripted autocomplete fields:

- No support for CM/Track V1. Scripted autocomplete fields are not supported in CM/Track V1.
- No transfer to the DWH. Scripted autocomplete fields are not transferred to the DWH and cannot be used in reports.
- No support for ETL. Import and export of scripted autocomplete fields using ETL is not supported.
- Scripted autocomplete fields cannot be indexed. Therefore they are not available for search.
- If an object which is referenced in an autocomplete field is deleted, the field might point to a non-existing object. This has to be considered in case the field value is later used in scripts.
- Scripted autocomplete fields cannot be used in text templates and document templates.

Note about Scripted Autocomplete Fields in the Current CM Versions vs. Previous Versions

This new way of configuring autocomplete fields replaces the previous method, which used the annotation *text-type* = "autocomplete". Existing fields which use the old method will remain functional as of version 6.11.1.1. These fields will not be modified during an update. The previously used method `onEditDisplayEntered` still works for existing fields. Nevertheless, it is not needed anymore and will be removed in a future version of ConSol CM.

F.8.1.5 Scripts of Type *Clone*

In the Web Client, a ticket can be cloned (duplicated) using the *Clone* option in the ticket menu. You can do this with or without a script.

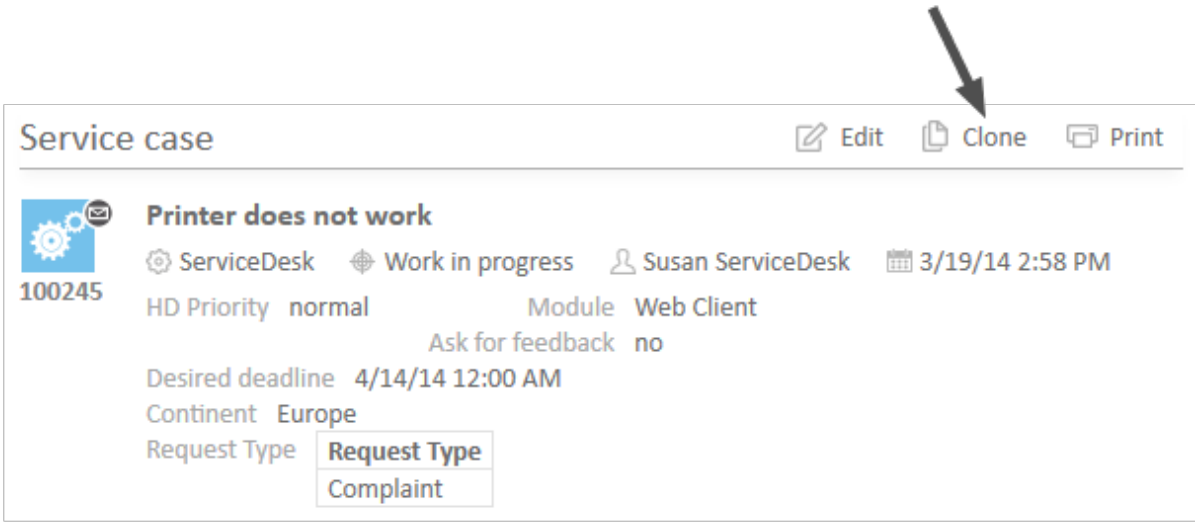


Figure 462: *ConSol CM Web Client - Clone option in ticket menu*

The following two pictures show the cloning of a ticket without a clone script.

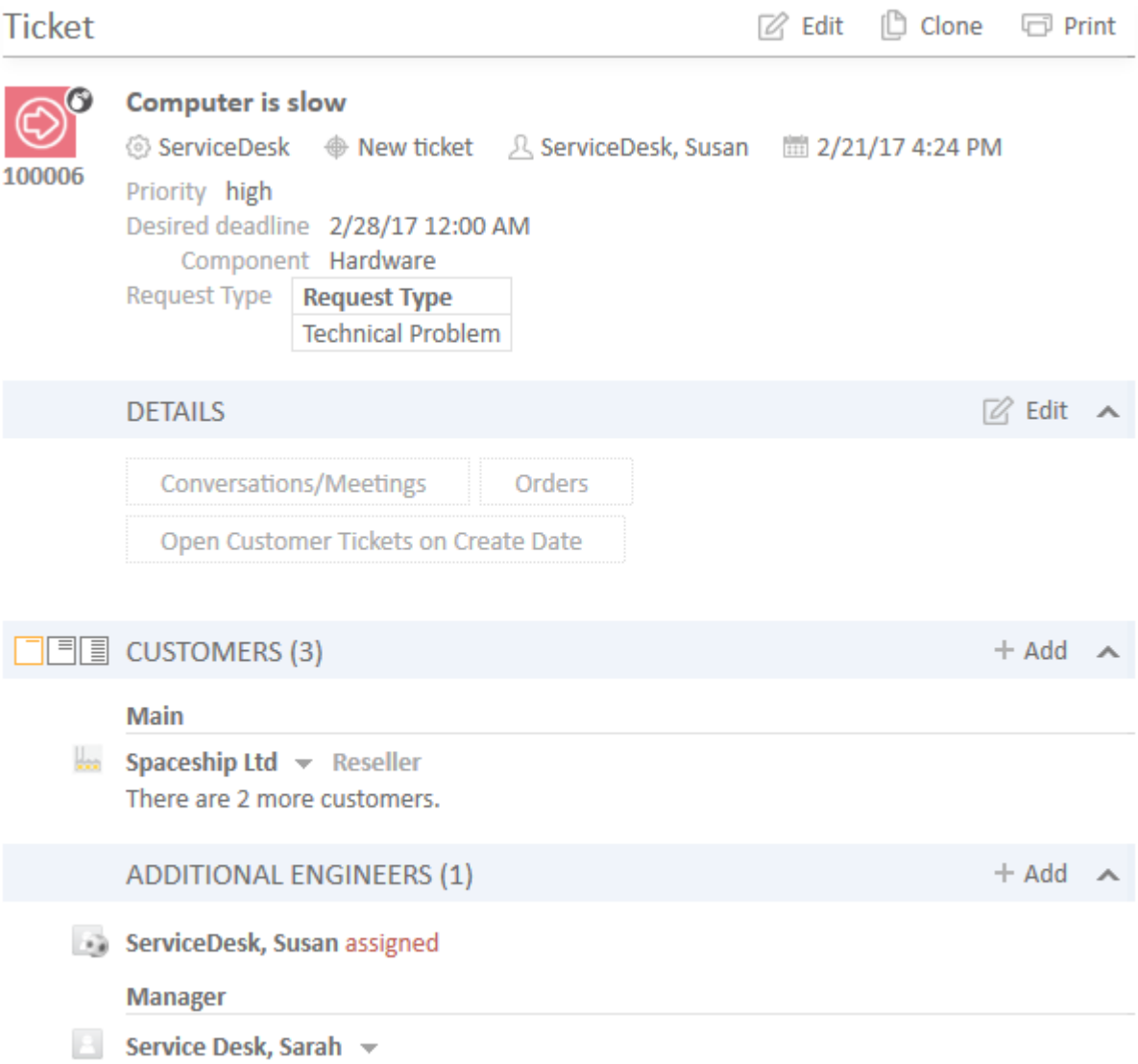


Figure 463: *ConSol CM Web Client - Original ticket*

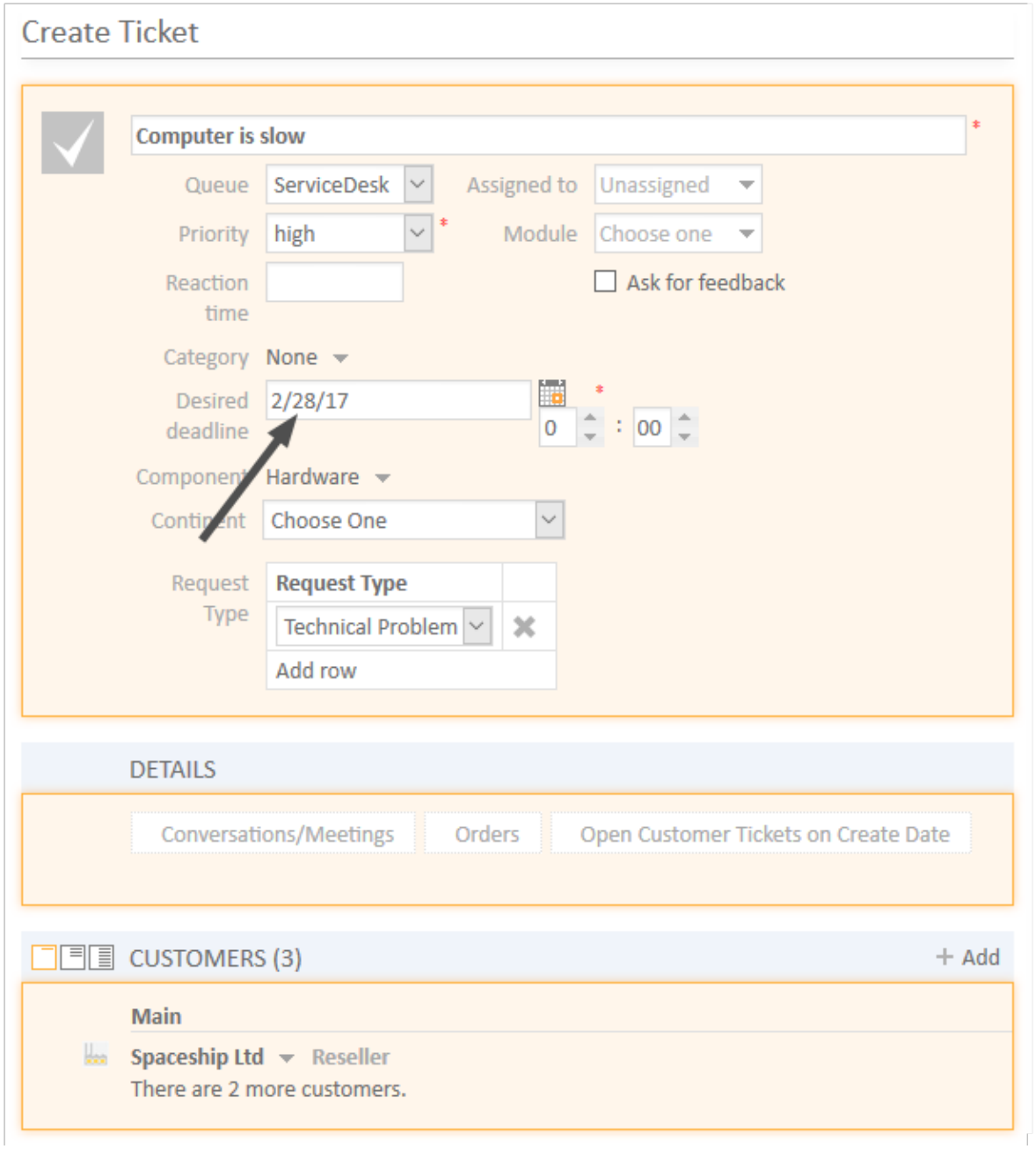


Figure 464: *ConSol CM Web Client - Cloning the ticket (without Clone script)*

When a ticket is cloned, the following data is transferred from the original ticket (compare the two images above):

- the ticket subject
- the queue
- the engineer (if one was set)
- the values of all ticket fields (ticket data section and Details section)
- the main customer
- additional customers

When a ticket is cloned, the following data is **not** transferred from the original ticket (compare the two images above):

- all ticket text:
  - comments
  - emails
  - attachments
- the ticket history
- additional engineers
- related tickets

If the queue where the ticket is located uses a *Clone* script (see section Queue Administration), this script will be executed when the engineer clicks on *Clone*. The script can be used similarly to a *Default values* script (see respective section below): you can preset values in the newly-created ticket. In the cloning process the values are pre-filled in the ticket fields in the Web Client, i.e., before the ticket is generated. The engineer can change the values if required.

Please keep in mind that in a Clone script, you do not work in the workflow context! That means the `workflowApi` object (implementation of `WorkflowContextService`) is not available!

In the following example, the *Clone* script is used to reset the data field *Desired deadline* to avoid having incorrect service dates in (cloned) *Service Desk* tickets.

```
ticket.set("serviceDesk_fields.desiredDeadline", null)
```

Code example 85: *Clone script to reset ticket field for desired deadline*

When the script is assigned to the queue (*ServiceDesk*, in the example), the field for the desired deadline is empty when the cloned ticket is opened by the engineer (see following image).

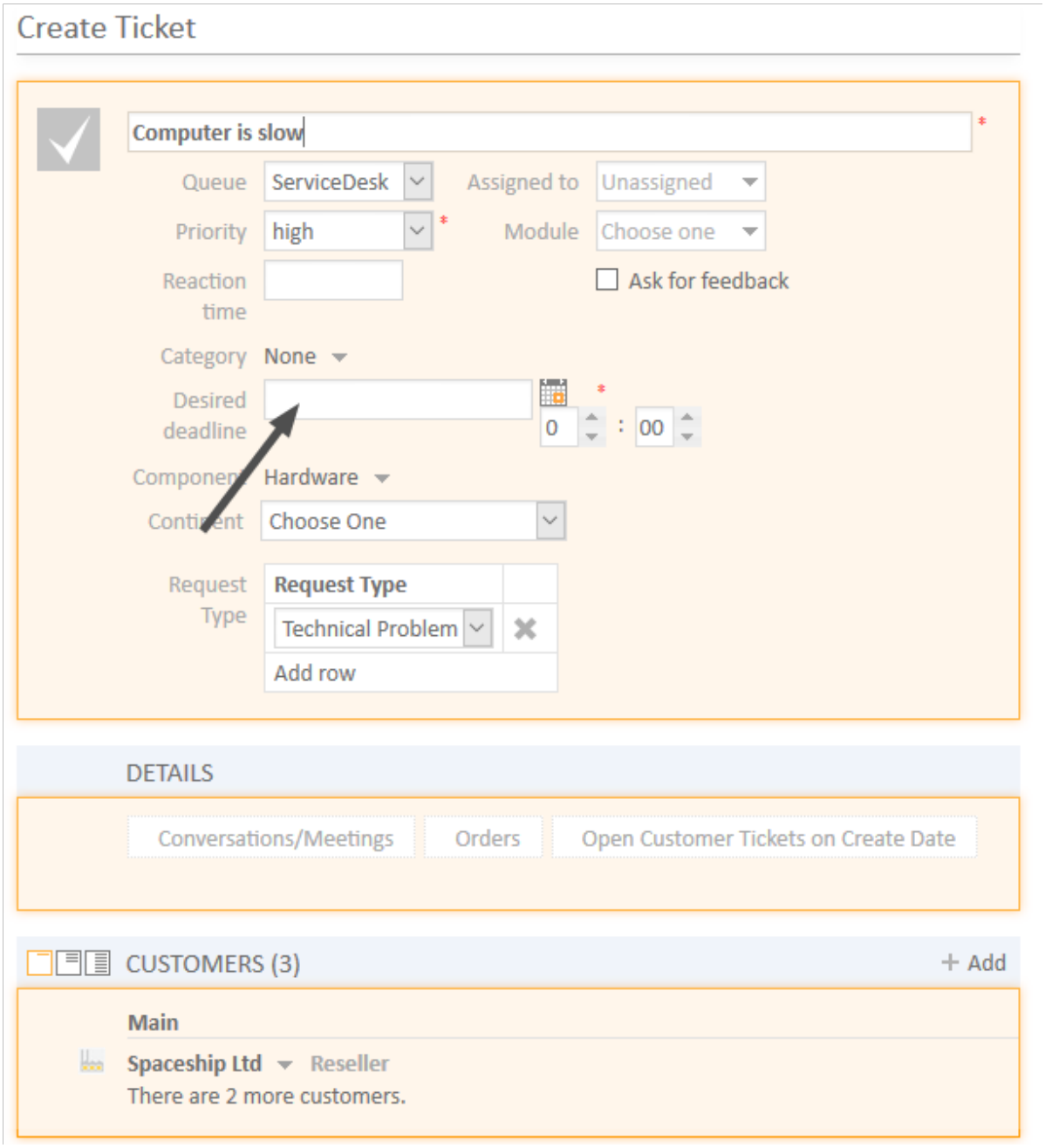


Figure 465: *ConSol CM Web Client - Cloned ticket (with Clone script assigned to the queue)*

F.8.1.6 Scripts of Type *Default Values*

Sometimes it is required that a data field of a ticket has a certain value when the ticket is initially created, i.e., when the engineer clicks *New ticket* and the respective form is opened in the Web Client, one or more values should be preset. This saves the engineer from having to set the value every time, e.g., when the default priority is *normal*, this can be preset. In case *high* or *low* is required, the engineer can switch to another value.

This ConSol CM behavior can be achieved by using one or more *Default values* scripts. The default values can be individually defined for each queue. For each queue, exactly one *Default values* script can be assigned. Please see the following example.

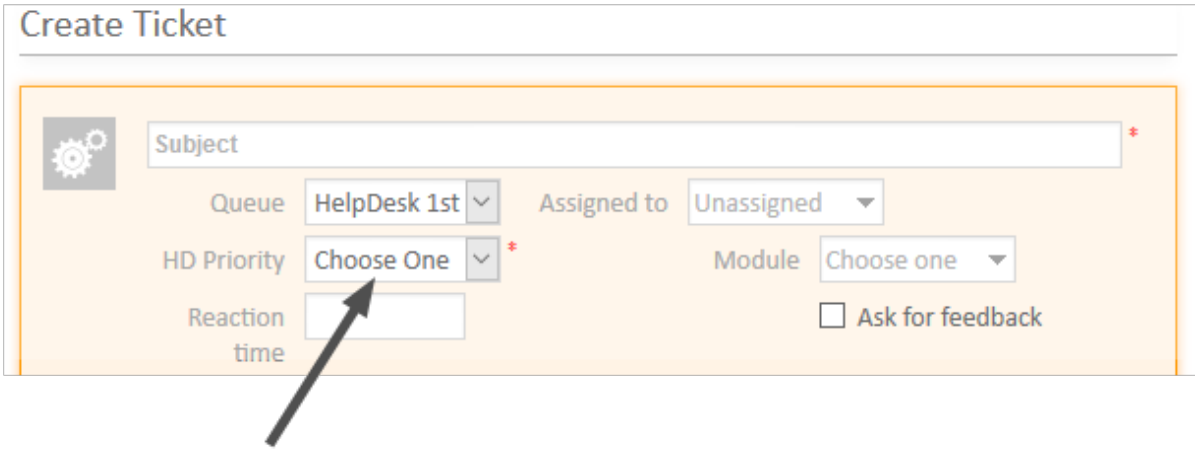


Figure 466: *ConSol CM Web Client - New ticket without default values*

Without a *Default values* script, no value is set for the priority when an engineer creates a new ticket in the Web Client.

To define a default value, a script of type *Default values* has to be created. First, we have to look up where the respective ticket field is to be found (in which ticket field group and under which name) in the ticket field administration. See section Ticket Field Administration (Setting Up the Ticket Data Model) for details.

> Please keep in mind that in a Default values script, you do not work in the workflow context! That means the `workflowApi` object (implementation of `WorkflowContextService`) is not available!

The following figure shows the ticket field *priority*. You reach this screen by opening the navigation item *Ticket Fields* in the navigation group *Tickets*.

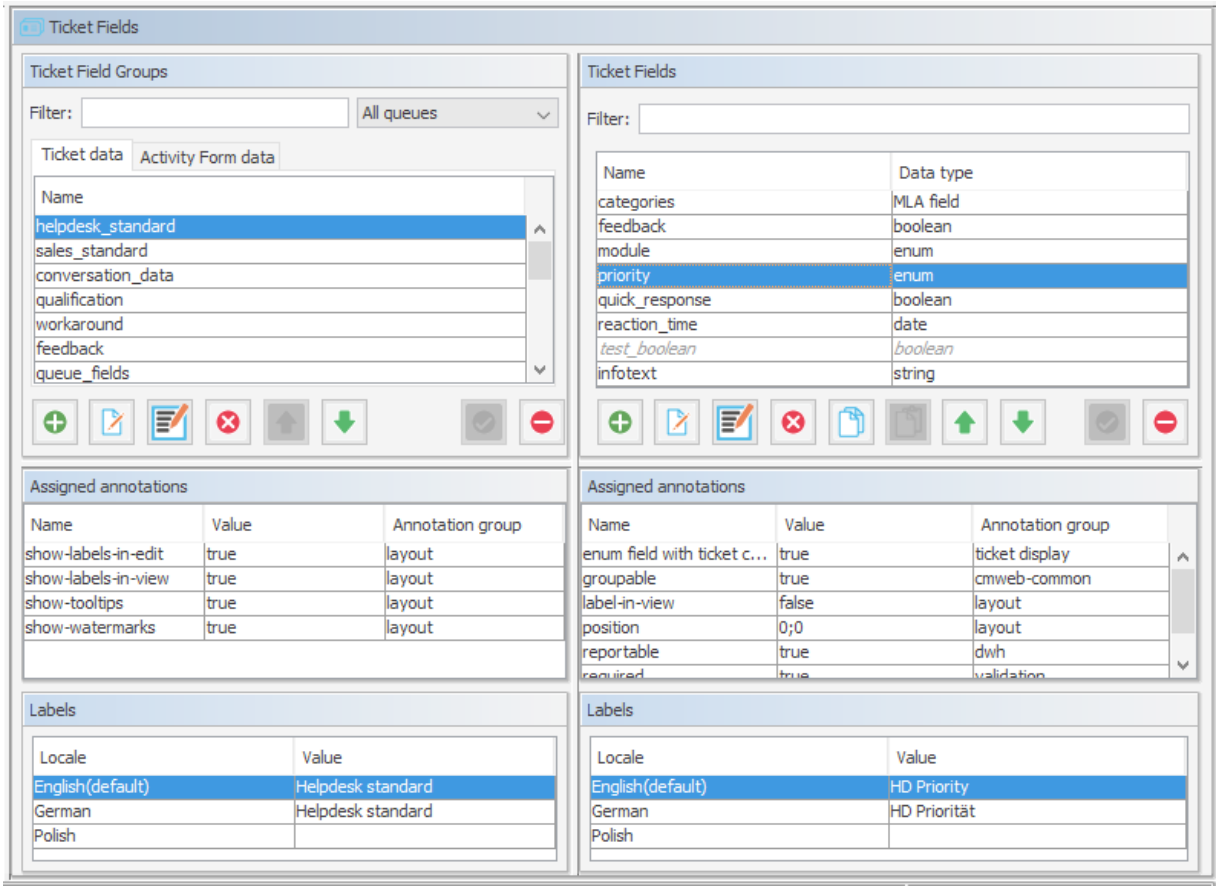


Figure 467: *ConSol CM Admin Tool - Tickets, Ticket Fields: priority*

Since *priority* is an ENUM field (i.e., an ordered list), we have to check the possible (technical) values which can appear in the list and we have to look for the required default value in the *Enum Administration* (navigation group *Lists*, navigation item *Enums*).

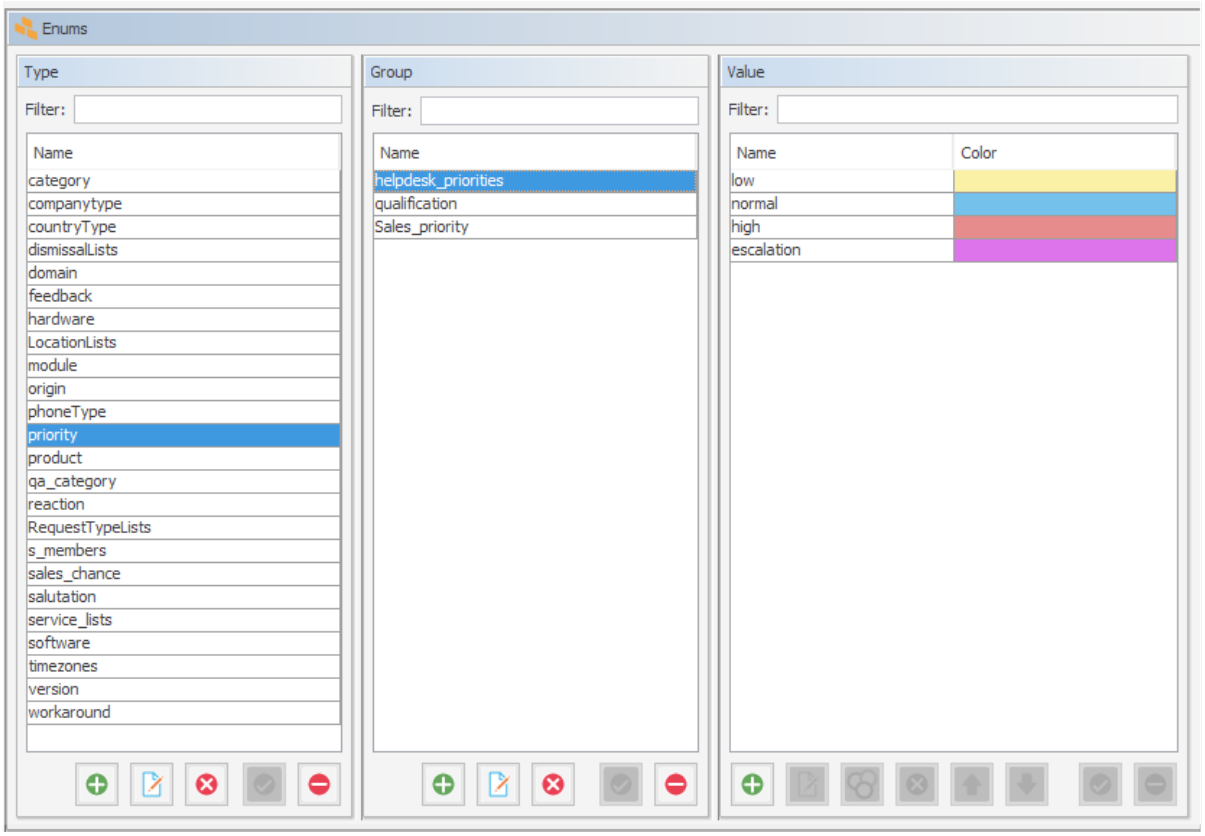


Figure 468: *ConSol CM Admin Tool - Lists, Enums: Priority values*

The group, the field, and the correct value can then be used in the respective Groovy method in the new script, located in the navigation item *Scripts and Templates*, navigation group *System*.

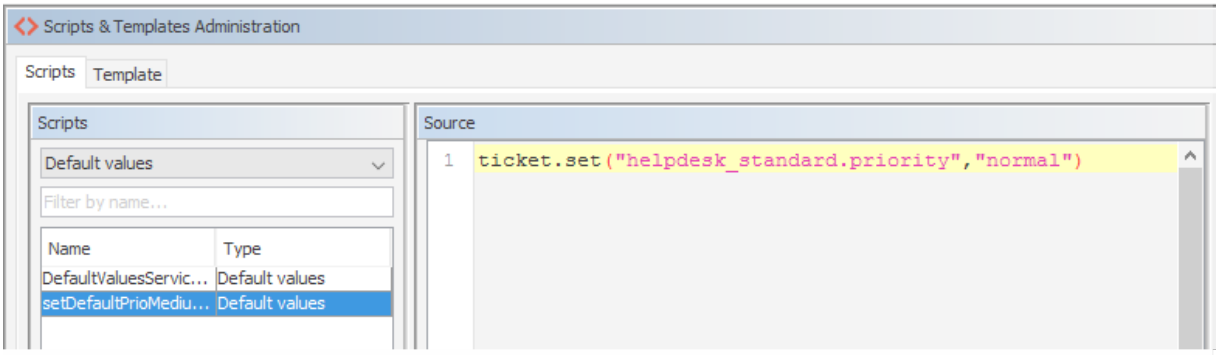


Figure 469: *ConSol CM Admin Tool - System, Scripts and Templates: Include group, field, and value in script*

In the queue administration (navigation group *Global Configuration*, navigation item *Queues*) we have to assign the script to the queue where the default value should be used.

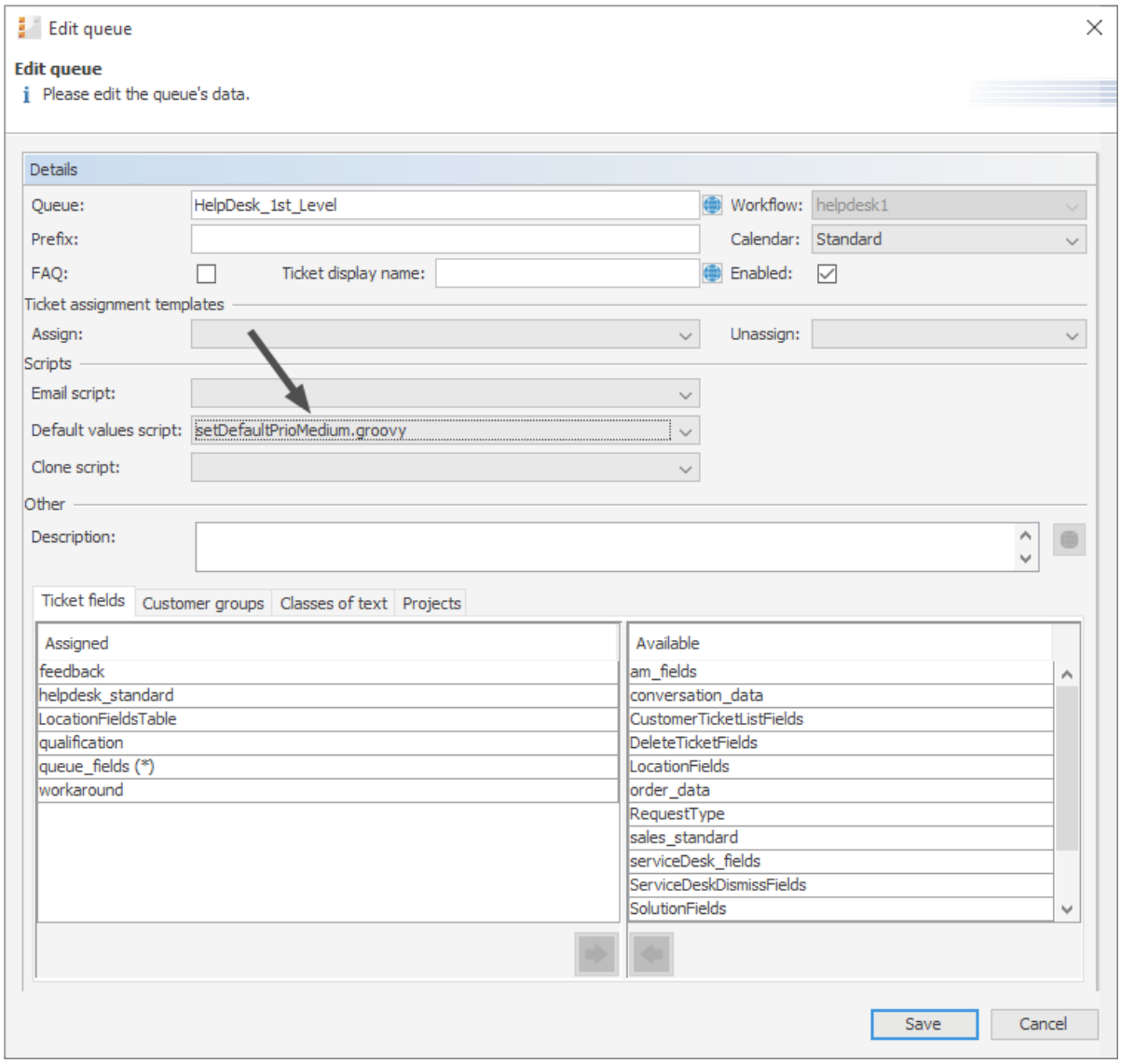


Figure 470: *ConSol CM Admin Tool - Global Configuration, Queues: Assign Default values script to queue*

When the engineer subsequently starts the *create ticket* operation in the Web Client, the default value *normal* will be set in the *Priority* field.

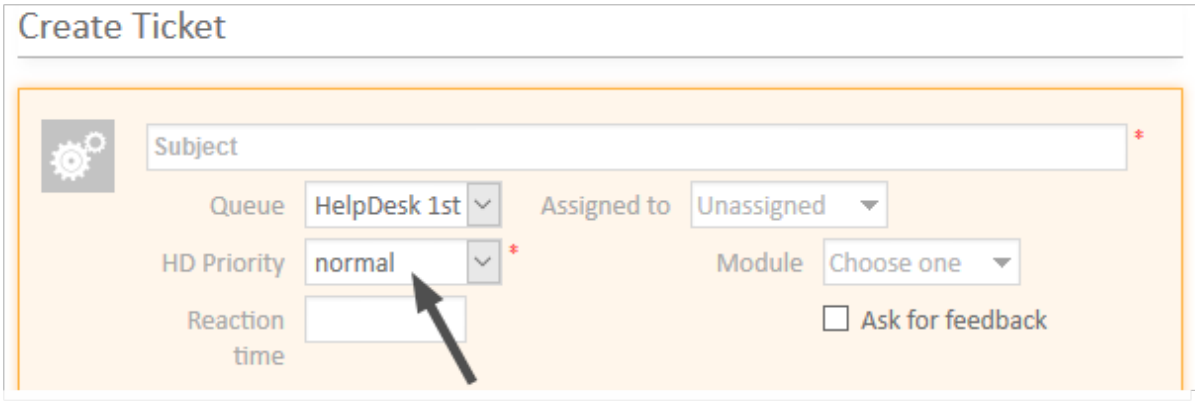


Figure 471: *ConSol CM Web Client - New ticket with default value*

Please keep in mind that for every queue there can be only one *Default values* script. If you have to define various default values, they have to be defined in **one** script. You might want to adapt the script name in this case, to better convey its intent.

If the same default value has to be set in different queues and is set together with other default values, this also has to be coded in one script for each queue.

Overwrite Mode for Default Values Scripts

By default, the fields of a ticket that are pre-filled by a *Default values* script are not overwritten, i.e., when the ticket is sent to another queue that has a different *Default values* script assigned, this second script will try to set fields that were already filled by the first script. Since this is not possible, the default value(s) from the first script are left intact.

If a *Default values* script should overwrite existing values, *overwrite* mode has to be activated. To activate this mode insert the following code at the beginning of your *Default values* script:

```
import com.consol.cmas.common.model.ticket.TicketPrototypeContext
TicketPrototypeContext.enableOverwriteMode()
```

F.8.1.7 Scripts of Type *Dependent Enum*

Dependent enums are hierarchical lists which provide a data structure similar to the one provided by *MLAs* (see section MLA Administration). In contrast to MLAs, with dependent enums only one level at a time is displayed. Depending on the value the user has selected in the list on this level, another list, the one in the sub-level, is opened. There do not have to be sub-lists for every list entry, so in graph terminology, the list might be a combination of nodes and leaves. A dependent enum script is assigned to the ticket field group, customer field group, or resource field group where it is required.

Please see the following example:

In *Help Desk* tickets a location can be selected. First, the user selects the *continent*. Depending on the selected continent, a sub-list with sub-continents or a sub-list with countries is displayed. All ticket fields have to be defined as regular ENUM fields first. In the script, the value of the first level list is checked and, depending on this value, another list is displayed in the second level. This can be used for as many levels as required, but please keep in mind that the editing of the script will become more complex with each level.

The *Dependent enum* script is placed in the Admin Tool. Please feel free to ask our ConSol CM consultants for support when creating and/or editing the script, as this is a rather complex task.

The following figure shows an dependent enum script. You reach this screen by opening the navigation item *Scripts and Templates* in the navigation group *System*.

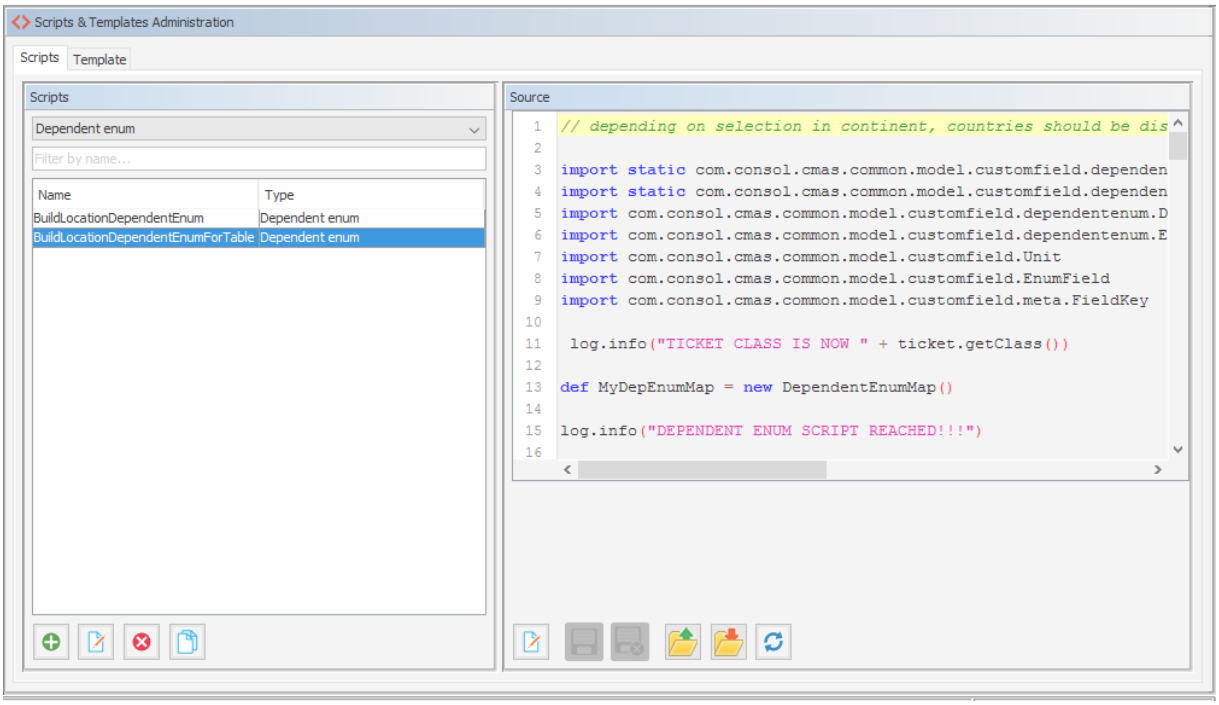


Figure 472: *ConSol CM Admin Tool - System, Scripts and Templates: Dependent Enum script*

The *Dependent enum* script is assigned to the ticket field group, customer field group, or resource field group where it is required.

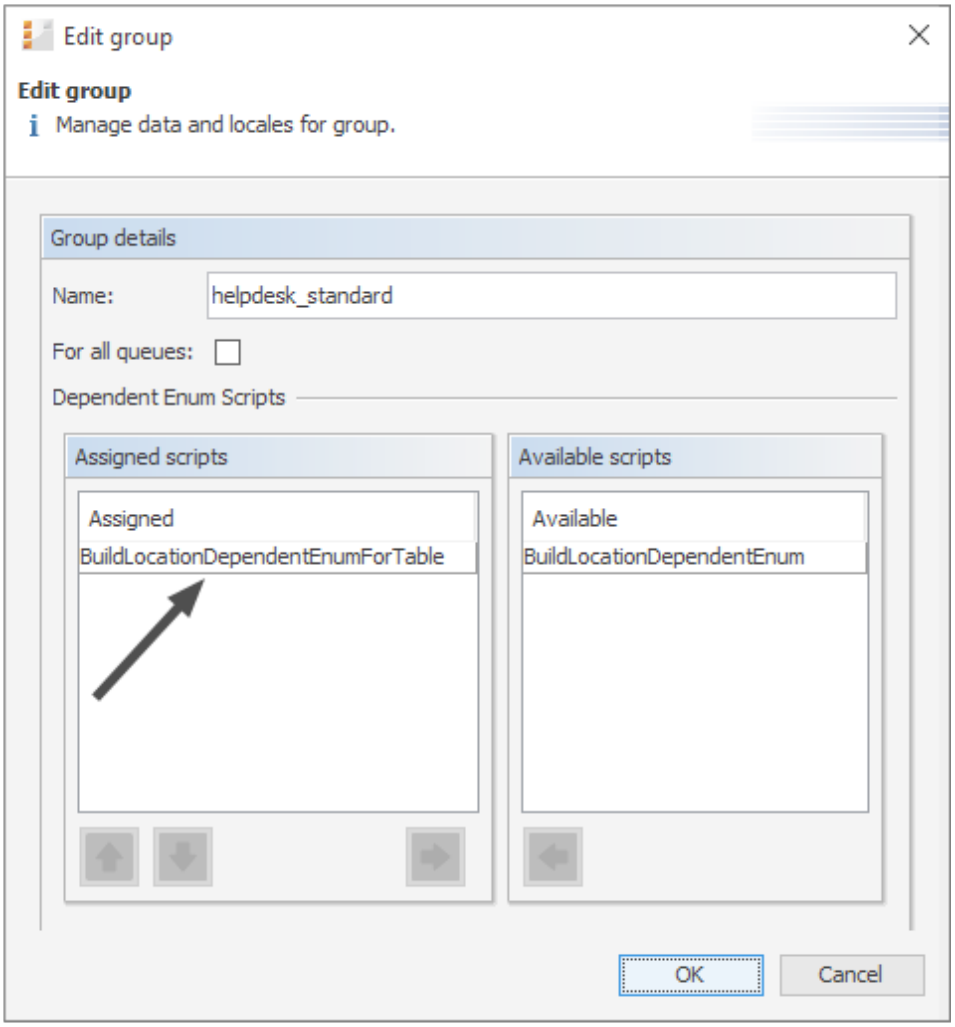


Figure 473: *ConSol CM Admin Tool - Tickets, Ticket Fields: Assign Dependent Enum script to ticket field group*

In the Web Client the engineer only sees the sub-list of the value selected in the first level list.

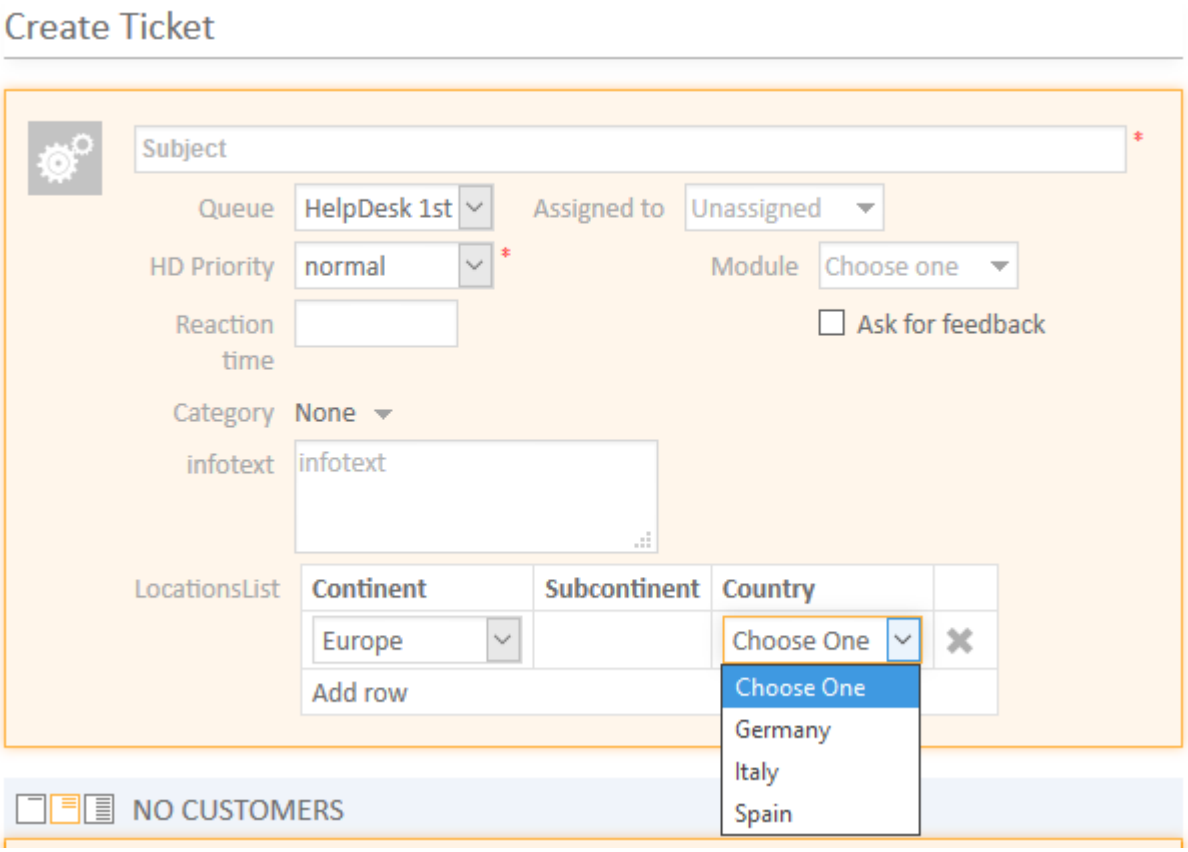


Figure 474: *ConSol CM Web Client - Sub-list of continent Europe*

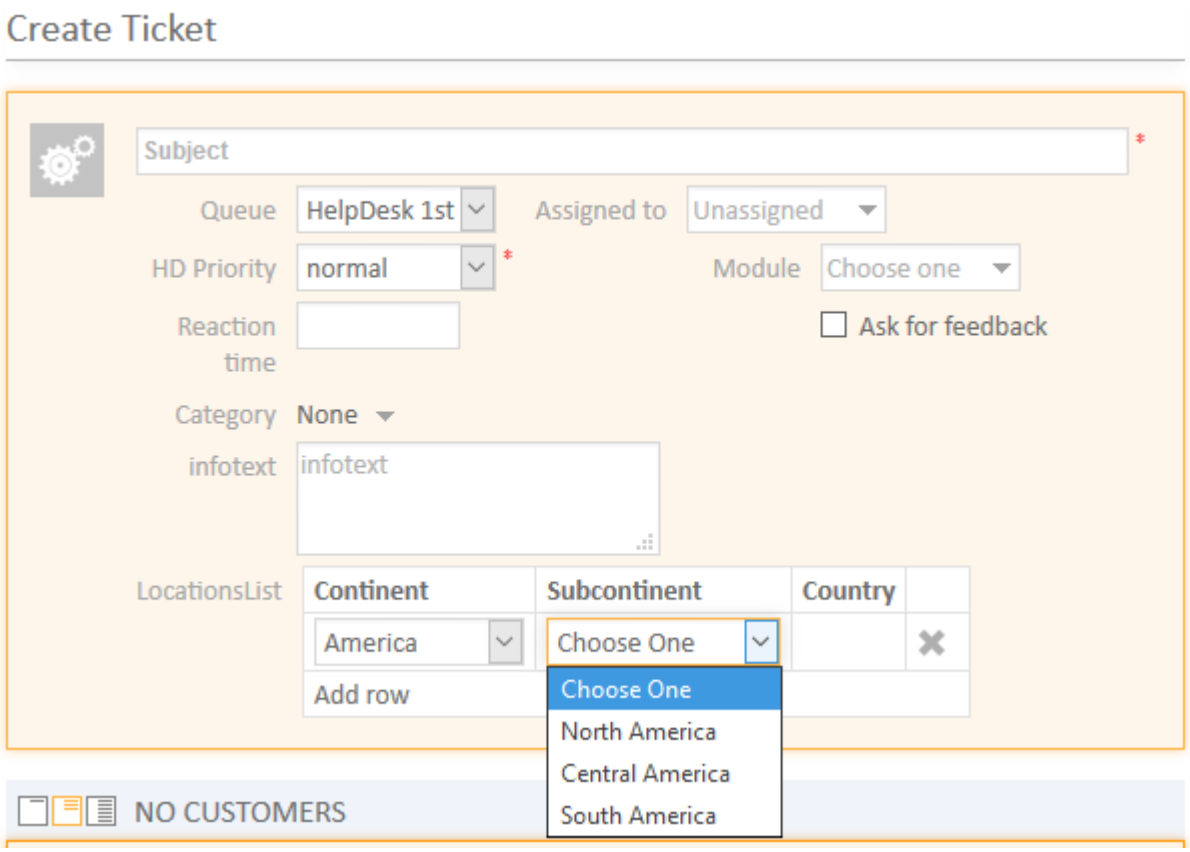


Figure 475: *ConSol CM Web Client - Sub-list of continent America*

F.8.1.8 Scripts of Type *Email*

Scripts of this type are used for several features. Some of the scripts are part of the default system configuration and have to be modified according to the customer-specific system configuration. You can also add your own scripts.

IMPORTANT INFORMATION

Since ConSol CM version 6.9.4, there are two modes to receive incoming emails:

- **Mule/ESB** - This mode has been available in ConSol CM since version 6. It is deprecated in CM versions 6.11 and up. Therefore, this mode is no longer treated in ConSol CM manuals. If you run an older CM system with Mule/ESB, please refer to older manuals. All ConSol CM 6.11 systems should run in NIMH mode. Please refer to older Administrator manuals to learn how to switch from Mule/ESB to NIMH.
- **NIMH** - *New Incoming Mail Handler*, available since version 6.9.4. This is the only available incoming email mode in ConSol CM version 6.11 and up.

The active email mode is set in the system property cmas-core-server, nimh.enabled:

- **true** - NIMH mode
- **false** - Mule/ESB mode

The sending of emails, i.e., the SMTP server configuration, is not influenced by the incoming email mode.

Email Scripts for the Processing of Incoming Emails in NIMH Mode

When an email is received by ConSol CM, it is processed by several scripts, see following figure.

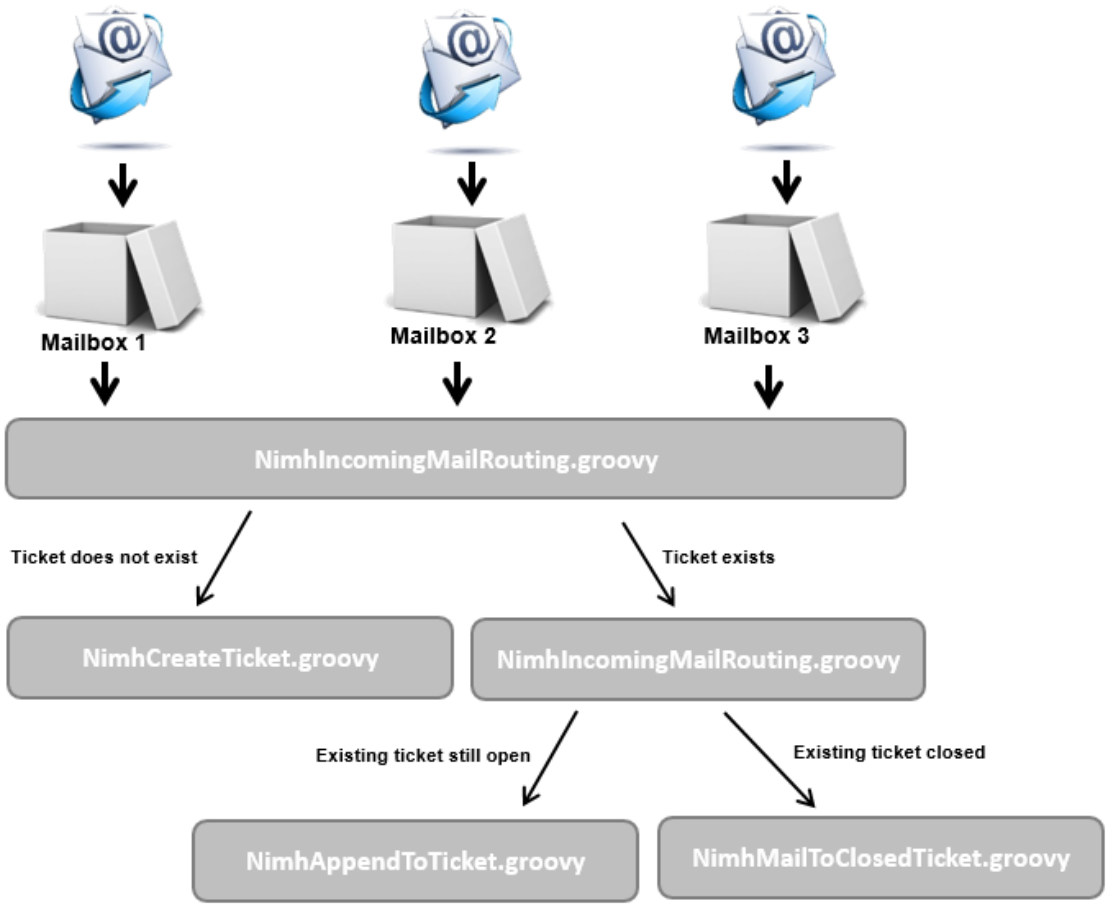


Figure 476: *ConSol CM Admin Tool - Email scripts (NIMH)*

- **NimhIncomingMailRouting.groovy**
  Standard script. This is the first script that is executed when an email comes in. Here, it is decided whether a new ticket has to be created, the email concerns an existing open ticket (then `NimhAppendToTicket.groovy` is executed), or if it pertains to a closed ticket (then `NimhMailToClosedTicket.groovy` is executed). This script does not require any changes to adapt it for a customer-specific environment.
- **NimhCreateTicket.groovy**
  Standard script which is responsible for the creation of a ticket when an email has been received in one of the ConSol CM-configured mailboxes (see section Email Configuration Using the Admin Tool for details). If the ticket subject does not match the regular expression for appending the email to an existing ticket, this script is executed. All emails which are received by ConSol CM (and have not been assigned to an existing ticket) are processed here, regardless of which mailbox they are collected from. In the script, the default queue for incoming emails has to be defined and more values of ticket fields can be defined (like, e.g., the default priority for email tickets). A default company is defined. In case the default company does not exist yet (in a new system), it will be created automatically. In the script, decisions can be made, e.g.,

determining which queue the new ticket should be created in, depending on the *To* address or other parameters. So usually, this script has to be adapted considerably. Please ask a ConSol CM consultant for support with this task.

- **NimhAppendToTicket.groovy**
  Standard script which is responsible for appending an email to an existing, open ticket. The assignment of the email to the ticket is performed using the comparison between the ticket subject and the required regular expression. Please see section Email Configuration Using the Admin Tool for a detailed explanation of this topic. Usually, no changes are required for this script.
- **NimhMailToClosedTicket.groovy**
  Standard script which is responsible for handling the email when it pertains to a closed ticket. The default system behavior is to create a new ticket for the customer (sender of the email) and to create a reference to the old/closed ticket. Usually, no changes are required in this script.

Email Scripts for Outgoing Emails

For every queue, an *Email* script can be configured. Please see section Queue Administration for an explanation how to configure this. An email which is written from a ticket in this queue (automatically by the workflow or manually by an engineer) passes through this script before it leaves the ConSol CM system. So in this *Email* script you can change or set queue-specific settings for the outgoing email. A common use case is the setting of a queue-specific Reply-To address in order to use team-specific Reply-To addresses.

An example of an outgoing *Email* script is the following script which is part of the ConSol CM default application:

- **ChangeOutgoingMail.groovy**
  Standard script that is not used, but serves as a template for outgoing *Email* scripts. You might want to use it to configure queue-specific *Email* scripts.

Within the outgoing *Email* script, the Java object `mailEntry` is implicitly available as the object `mail`. You have to set all required attributes for the outgoing email using the `mail.setAttribute()` or `mail.setAttributes()` methods.

```
def queueReplyAddress = "serviceteam@mycompany.com"
// you might also use system properties for the queue-specific email addresses and
 fetch an
// address using the configurationService!

mail.setAttribute('Reply-to', queueReplyAddress)
```

Code example 86: *Email script*

Common email attributes are:

- Bcc
- From
- Reply-to

- To
- Cc
- Subject

For a very detailed description of the email format, please refer to RFC 5322.

When you work with the configuration of the Reply-To email address, please note the following technical behavior of ConSol CM and adapt your system accordingly!

The technical background:

There are four potential Reply-To addresses which you deal with:

1. The **Reply-To address which is set with the system property `mail.reply.to`**. If it is set, it will be displayed in the Ticket Email Editor in the Web Client. If it is really the effective Reply-To address in an email depends on the configuration in the queue-specific outgoing email script. See next item. If the Page Customization attribute *showReplyTo* for the type *mailTemplate* is set to "false", no Reply-To address will be displayed in the Ticket-Email-Editor, but if the property `mail.reply.to` is set, this address will be used anyway - unless an outgoing mail script sets another address, see next item.
2. The **Reply-To address which is set in a queue-specific outgoing email script**. Since the outgoing email script is the last instance which processes an outgoing email, the Reply-To address set in this script will always be the effective Reply-To address which is used. In case the `mail.reply.to` property is set, this mail-reply.to-address will not really be used (but it will be displayed in the Ticket Email Editor which might cause some confusion! What that means for your system configuration is explained in the next section).
3. The **email address which is set in the system property `mail.from`.** If this is set and neither `mail.reply.to` nor a queue-specific Reply-To address is set, most email clients will set the From address as Reply-To address.
4. The **email address of the current engineer** (the engineer who is logged in to the Web Client). This personal email address is used as Reply-To address for emails from the Web Client if neither the `mail.reply.to` property is set nor a queue-specific outgoing email script is configured nor the `mail.from` property is set.

In the Web Client, in the ticket history, the Reply-To address which was really used is always displayed for outgoing emails. So even in case there should be a difference between the address which was displayed in the Ticket Email Editor (the `mail.reply.to` property) and the Reply-To address which was really used (the Reply-To in the queue-specific outgoing email script), the effective address is displayed. This would be the one from the script in this case.

What we recommend:

A system Reply-To address should always be set! You can decide if you

- work with the Reply-To address in the queue-specific outgoing email script

or

- use the `mail.reply.to` system property.

However, since the email communication should take place via ConSol CM and not using personal email addresses, one of the two system settings mentioned above should be used to prevent CM from using personal email addresses as Reply-To. The latter would automatically lead to customer emails being sent to an engineer's personal email account instead of CM.

What that means for your system configuration:

1. The simplest way to set a Reply-To address is by using the `mail.reply.to` system property. It will be displayed in the Ticket Email Editor and will be the effective Reply-To address.
2. If queue-specific Reply-To addresses are required, we recommend to write one outgoing mail script where queue names are mapped to specific Reply-To addresses. This can then be extended for Bcc, Cc or other addresses.
   You can combine the `mail.reply.to` property and queue-specific Reply-To addresses: for all queues without a specific outgoing mail script, the `mail.reply.to` address will be used, for all queues which have a queue-specific outgoing mail script that contains a Reply-To address, this will be used.

What that means when you work with workflow scripts which send emails:

(A detailed explanation is provided in the *ConSol CM Process Designer Manual*!)

- Use the object and method `configurationService.getValue("cmweb-server-adapter","mail.reply.to")` to retrieve the value of the system property and set it as Reply-To address in the outgoing email.
- Use the `Mail` object when the queue-specific script should be used: e.g. `mail.useDefaultScript()`. This will overwrite the `mail.reply.to` property!

If neither the system property nor the queue-specific outgoing email script is used, i.e. when the Reply-To address is not set, usually the From address will be used as Reply-To by the email client.

F.8.1.9 Scripts of Type Field Visualization

Introduction

Scripts of the type *Field Visualization* enable the customization of several GUI aspects of the Web Client and CM/Track V2. Possible use cases are:

- Organizing data fields in several groups with headlines
- Changing the background color of certain data fields to highlight important information
- Adding images as field content

- Adding maps as field content
- Adding Highcharts widgets as field content

This feature works for data fields in ticket, customer, and resource objects in the Web Client and for data fields in ticket objects in CM/Track V2.

Configuring a Picture Field with Scripted Visualization

The following example implements the display of an image on the contact page of the Web Client. The image border and the styling of the caption are done using a customized CSS file.

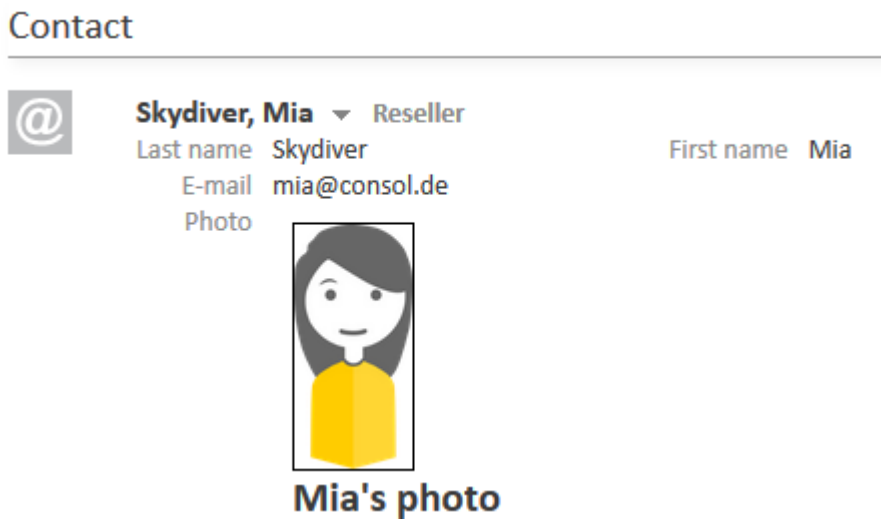


Figure 477: *ConSol CM Web Client - Customer field containing an image*

Carry out the following steps to configure such a field.

Step 1: Provide the Resources

First, you need to provide the required resources. Depending on the kind of visualization, this can be images, stylesheets, JavaScript files, or other files. The resources are stored in the file system. Alternatively, they can also be retrieved using URLs.

To store resources in the file system, create a new folder `resources` in the ConSol CM data directory and save the required resources in this folder. In this example, the resources folder contains a subfolder for each unit type (contact and company) . In each subfolder, there are subfolders which each carry the name of the ID of the respective unit. In the ID subfolder, there is an image with the name `person.jpg`. Additionally, the file `person.css` is located in the `resources` folder. This file contains layout information regarding the display of the image border and the caption.

In the current example, we work with the following directory structure (here: the pictures of two contacts are available):

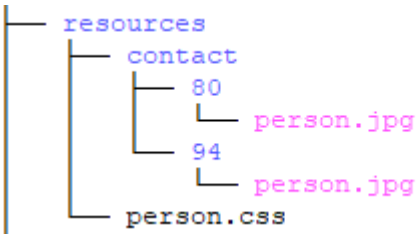


Figure 478: *Example directory structure of the resource directory for scripted field visualization*

Step 2: Write the Script

Now, you have to create a script of type *Field Visualization* which implements the logic of the visualization.

The script contains two methods, `resources()` and `render()`. The `resources()` method retrieves the resources from the file system or an URL and the `render()` method provides the logic for rendering the field value.

The methods provide several parameters:

- **pContext**: object from which the script is called
- **pFieldKey**: key of the field which calls the script
- **pFieldValue**: value of the field
- **pClient**: type of client, can be “web”, “track”, or “rest”

These parameters allow you, for example, to adapt the behavior to the field value or to configure a different behavior for the Web Client and CM/Track V2.

The following script implements the display of an image as shown in the above figure.

```
import com.consol.cmas.common.model.customfield.meta.FieldKey;
import
 com.consol.cmas.common.model.customfield.visualization.FieldVisualizationContext;
// ######### renderCustomerImages.groovy ##############
/**
* It returns simple html document which will be used to render custom field in view
 mode.
*
* All additional resources like css, js, or image files need to be enumerated in
 the resources() method.
* Js or css file will be automatically registered on html page.
* To use other files simply use their names, they will be replaced with correct
 html url.
*
* @param pFieldKey - the field key
* @param pFieldValue - the field value
```

```
* @param pClient - the client type for which render method is called(e.g.: track,
 web)
* @param pContext - it provides id of an object from which given field comes from
*/
def render(FieldKey pFieldKey, Object pFieldValue, String pClient,
 FieldVisualizationContext pContext) {
   def path = "${pContext.getUnit().getDefinition().getType().name().toLowerCase
    ()}/${pContext.getUnit().getId()}/person.jpg";
   return """
   <div class='photoBox'>
   <img src='${path}' class='person'>
   <h2>${pFieldValue}<h2>
   </div>
   """ as String
}
/**
* It returns list of resources which are required to visualize this custom field.
* Any files from {data.directory}/resources or external CDN server resource can be
 used here.
*
* All js or css files will be automatically registered on html page.
* To use other files simply use their names in render() method, they will be
 replaced with correct html url.
*
* data.directory - is the server shared directory configured in CM property
*
* @param pFieldKey - the field key
* @param pFieldValue - the field value
* @param pClient - the client type for which render method is called(e.g.: track,
 web)
* @param pContext - it provide id of object from which given field comes from
*/
def resources(FieldKey pFieldKey, Object pFieldValue, String pClient,
 FieldVisualizationContext pContext) {
   def path = "${pContext.getUnit().getDefinition().getType().name().toLowerCase
    ()}/${pContext.getUnit().getId()}/person.jpg";
   List<String> resources = [
      "person.css",
      path
   ] as String[];
   log.info resources;
   return resources;
}
```

Code example 87: *Field visualization script to show an image on the customer page*

Step 3: Configure the Data Field

To finish the configuration, create a data field (ticket, customer, or resource field), e.g., of type "String", and enter the name of the script as a value of the annotation `common, visualization`. Place the field in the Web Client GUI using the *move up/down* arrows and/or the `position` annota-

tion. Decide if the image should be shown even if the field is empty. If this is desired, assign the annotation `common, visualize-when-empty` to the data field. Otherwise, the image is only displayed if the field has some content.

> The annotation `common, visualize-when-empty` is applied for scripted field visualization both in the Web Client and CM/Track V2.

Configuring Context-Dependent Highlighting of Data Field Content

In the following example, the ticket field which contains the number of open tickets of the ticket's main customer is highlighted. If the number is less than 10, the field is displayed with a green background, if the number is greater than this limit, the background is red.

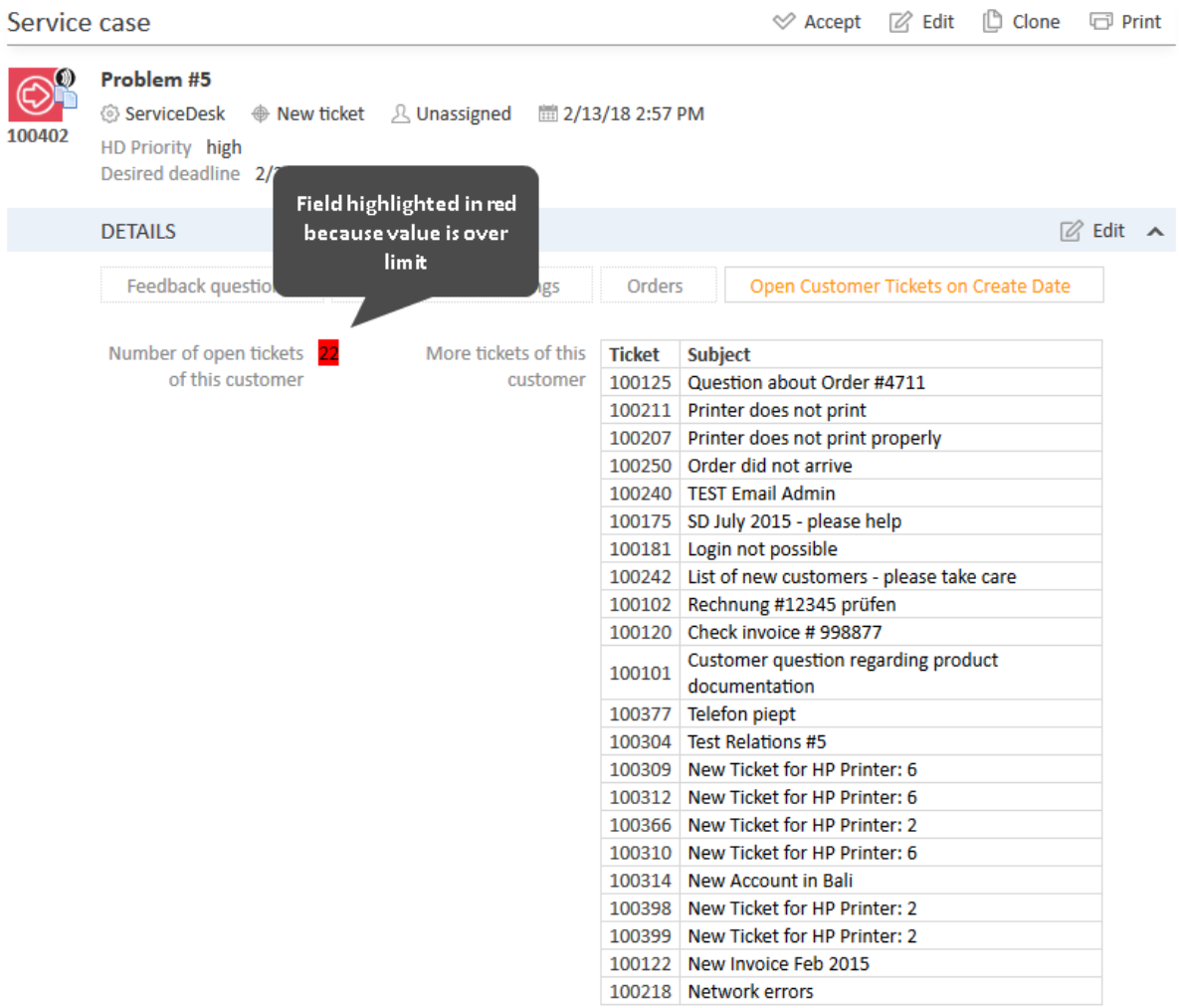


Figure 479: *ConSol CM Web Client - Number field highlighted because value is over limit*

Step 1: Provide the Resources

Only the .css file is required. In this example, it is called `highlight.css` and is a very simple .css file:

```
.red {
   background-color:red;
}
.green {
   background-color:green;
}
.default {
   background-color:inherit;
}
```

Code example 88: *Example hightlight.css file*

Step 2: Write the Script

```
import com.consol.cmas.common.model.customfield.meta.FieldKey;
import
 com.consol.cmas.common.model.customfield.visualization.FieldVisualizationContext;
// ################ highlightCriticalTicketNumbers.groovy ##############
/**
* It returns simple html document which will be used to render custom field in view
 mode.
*
* All additional resources like css, js, or image files need to be enumerated in
 the resources() method.
* Js or css file will be automatically registered on html page.
* To use other files simply use their names, they will be replaced with correct
 html url.
*
* @param pFieldKey - the field key
* @param pFieldValue - the field value
* @param pClient - the client type for which render method is called(e.g.: track,
 web)
* @param pContext - it provides id of an object from which given field comes from
*/

def render(FieldKey pFieldKey, Object pFieldValue, String pClient,
 FieldVisualizationContext pContext) {
   def highlight = "default";
   if(pClient.equalsIgnoreCase("track")) {
      return null
   } else {
      if (pFieldValue < 10) {
         highlight = "green";
      } else {
         highlight = "red";
      }
```

```
      return """
      <div>
         <span class="${highlight}">${pFieldValue}</span>
      </div>
      """ as String
   }
}
/**
* It returns list of resources which are required to visualize this custom field.
* Any files from {data.directory}/resources or external CDN server resource can be
 used here.
*
* All js or css files will be automatically registered on html page.
* To use other files simply use their names in render() method, they will be
 replaced with correct html url.
*
* data.directory - is the server shared directory configured in CM property
*
* @param pFieldKey - the field key
* @param pFieldValue - the field value
* @param pClient - the client type for which render method is called(e.g.: track,
 web)
* @param pContext - it provide id of object from which given field comes from
*/
def resources(FieldKey pFieldKey, Object pFieldValue, String pClient,
 FieldVisualizationContext pContext) {
   List<String> resources = [
      "highlight.css"
   ] as String[];
   return resources;
}
```

Code example 89: *Admin Tool script of type Field Visualization, used to highlight certain display values in a ticket field*

Step 3: Configure the Data Field

To finish the configuration, create a ticket field of type “number” and enter the name of the script as a value of the annotation `common`, `visualization`. Place the field in the Web Client GUI using the *move up/down* arrows and/or the `position` annotation.

Configuring Display of Graphics Based on the Highcharts Library

Graphics like statistics data or small reports can be displayed in data fields. The display is based on the ***Highcharts*** library. As an administrator, all you have to do is to create the data field (of type *visualization*) and write a script of type *Field Visualization* in which the ***Highcharts*** library is used to display the values which are retrieved in the script.

In the following example, a customer field, positioned in the groups section on the customer page, displays customer statistics. The example uses fixed values in the script, in real life you will work with values which are retrieved from real data in real time.

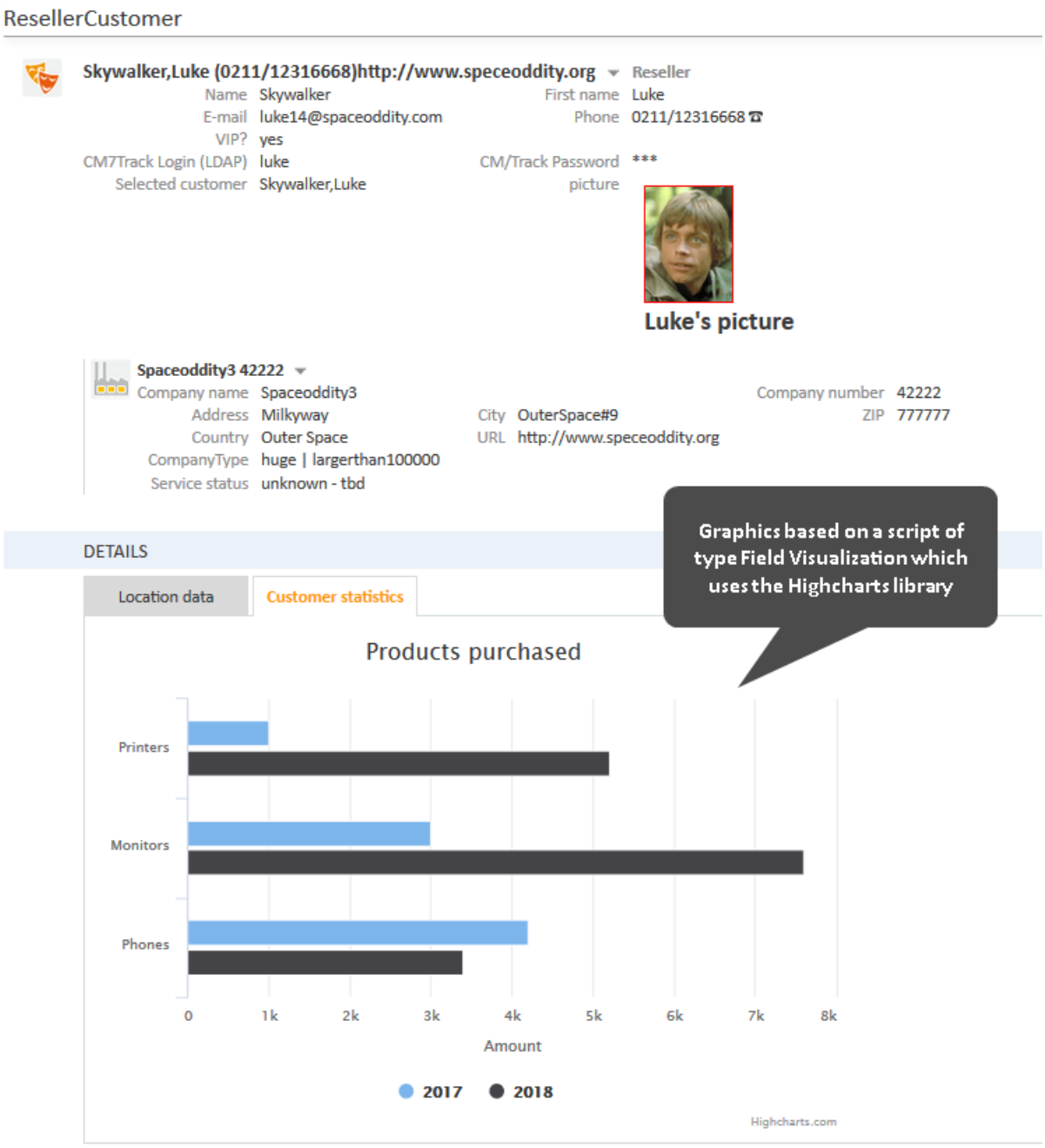


Figure 480: *Customer field annotated with visualization, in the groups section of customer data*

Step 1: Write the Script

```
import com.consol.cmas.common.model.customfield.meta.FieldKey;
import
 com.consol.cmas.common.model.customfield.visualization.FieldVisualizationContext;
/**
```

```
* It returns simple html document which will be used to render custom field in view
 mode.
*
* All additional resources like css, js, or image files need to be enumerated in
 the resources() method.
* Js or css file will be automatically registered on html page.
* To use other files simply use their names, they will be replaced with correct
 html url.
*
* @param pFieldKey - the field key
* @param pFieldValue - the field value
* @param pClient - the client type for which render method is called(e.g.: track,
 web)
* @param pContext - it provides id of an object from which given field comes from
*/
def render(FieldKey pFieldKey, Object pFieldValue, String pClient,
 FieldVisualizationContext pContext) {
  return """
  <div id="container" style="width:100%; height:400px;"></div>
  <script>
  \$(function () {
     var myChart = Highcharts.chart('container', {
        chart: {
           type: 'bar'
        },
        title: {
           text: 'Products purchased'
        },
        xAxis: {
           categories: ['Printers', 'Monitors', 'Phones']
        },
        yAxis: {
           title: {
              text: 'Amount'
           }
        },
        series: [{
           name: '2017',
           data: [1000, 3000, 4200]
        }, {
           name: '2018',
           data: [5200, 7600, 3400]
        }]
     });
  });
</script>
""" as String
}
/**
* It returns list of resources which are required to visualize this custom field.
* Any files from {data.directory}/resources or external CDN server resource can be
 used here.
*
```

```
* All js or css files will be automatically registered on html page.
* To use other files simply use their names in render() method, they will be
 replaced with correct html url.
*
* data.directory - is the server shared directory configured in CM property
*
* @param pFieldKey - the field key
* @param pFieldValue - the field value
* @param pClient - the client type for which render method is called(e.g.: track,
 web)
* @param pContext - it provide id of object from which given field comes from
*/
def resources(FieldKey pFieldKey, Object pFieldValue, String pClient,
 FieldVisualizationContext pContext) {
   List<String> resources = [
      "https://code.highcharts.com/highcharts.js"
   ] as String[];
   return resources;
}
```

Code example 90: *Admin Tool script of type Field Visualization, used to display statistics data based on the Highcharts library*

Step 2: Configure the Data Field

Create a customer field like the following. Indicate the name of the respective script of type *Field Visualization*. Assign the annotation `common, visualize-when-empty` to the data field. Otherwise, the chart is only displayed if the field has some content.

> The annotation `common, visualize-when-empty` is applied for scripted field visualization both in the Web Client and CM/Track V2.

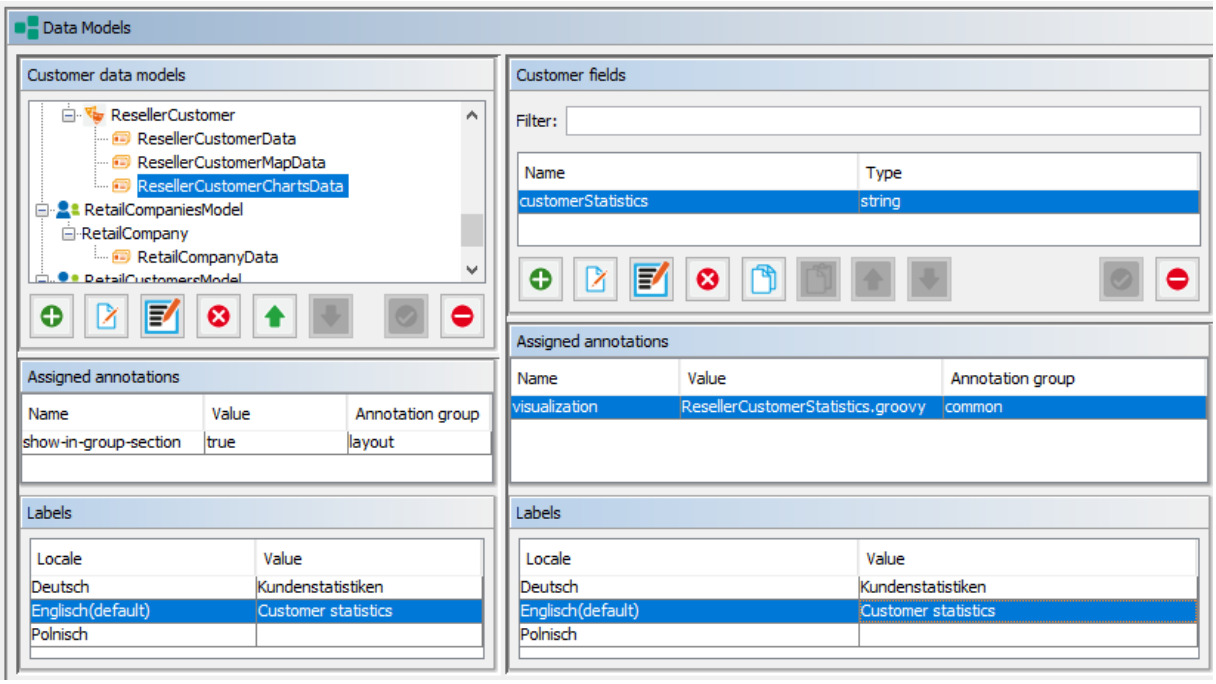


Figure 481: *ConSol CM Admin Tool - Customers, Data Models: Customer field for field visualization*

Configuring the Indication of a Location in a Map

You can integrate maps, e.g., of OpenStreetMap©, to display a location on an interactive map. This can be very helpful to display the location of customers' sites or machine locations (e.g., if machines are managed in your CM/Resource Pool).

In the following example, a company field of type *String* is used as hook for the display of an interactive map. The location of the customer has to be stored in respective company fields: *address*, *house number*, *zip*, and *city*.

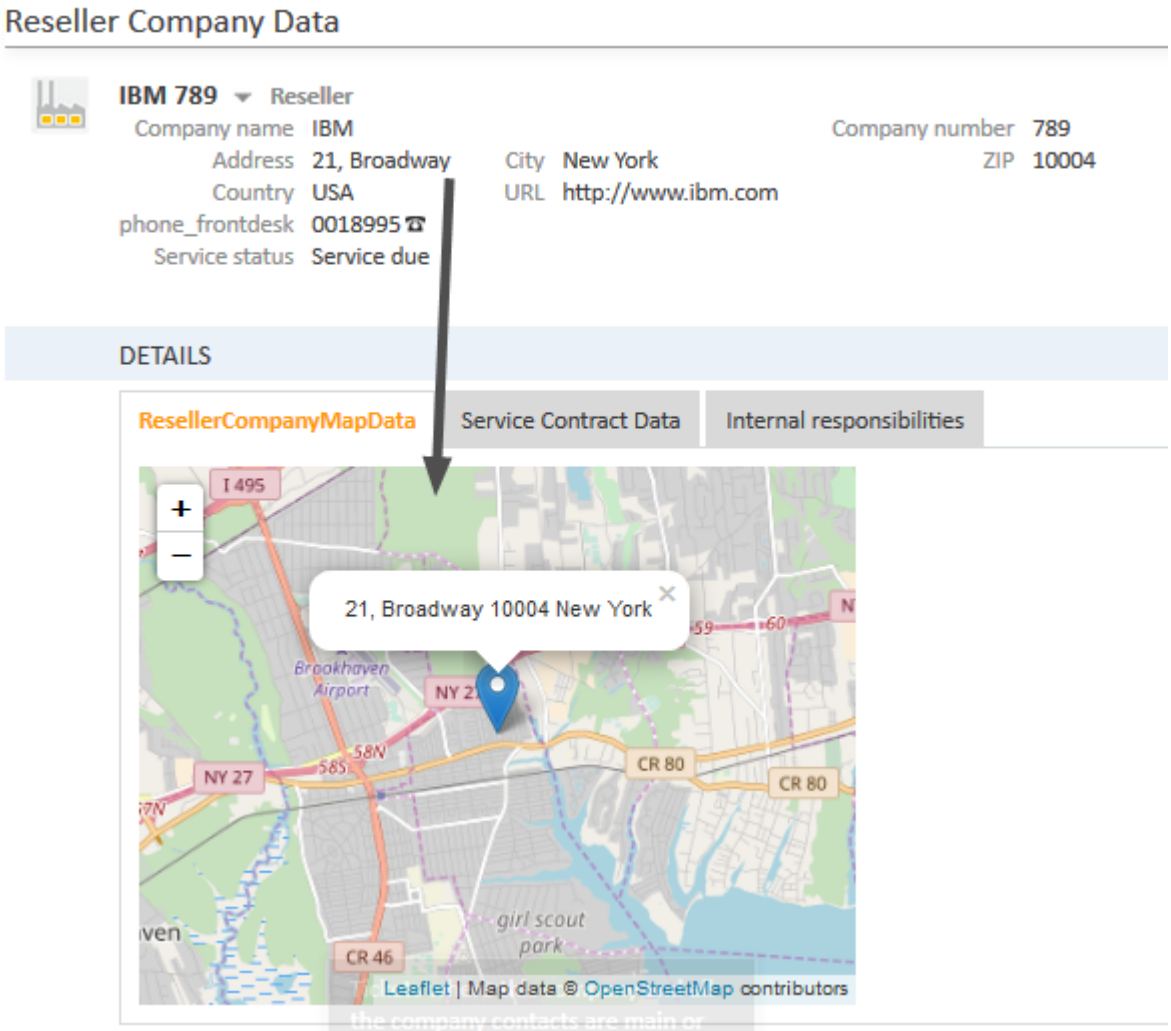


Figure 482: *ConSol CM Web Client - Interactive map (OpenStreetMap) which displays the location of the customer's site*

Step 1: Provide the Resources

The following files are required. If you need help with the implementation, please ask your ConSol CM consultant.

```
└── resources
    ├── maps
    │   ├── images
    │   │   ├── marker-icon.png
    │   │   └── marker-shadow.png
    │   ├── leaflet.css
    │   └── leaflet.js
```

Figure 483: *Directory structure and files required for the maps implementation in scripted field visualization*

Step 2: Write the Script

```
import com.consol.cmas.common.model.customfield.meta.FieldKey;
import
 com.consol.cmas.common.model.customfield.visualization.FieldVisualizationContext;

log.info ' Maps visualization script started ...'

// Maps configuration
mapsDivHeight = 300
mapsDivWidth = 400
initialZoom = 12 //a value between 1 and 20


//String msg shown when we do not have enough address data to construct the map,
 customize at own will
notEnoughAddressDataMsg = messageProviderService.getMessage
 ("web.map.not.enough.address.data")


/**
* Fill this method with your system's logic how the address string is collected.
 You can collect and construct it from
* single fields or maybe your visualization field already contains the proper
 string. The implementation here is
* suited for showroom where we use this script for several unit definition models.
 Adapt as needed.
*
* Expected output: address string in the form Franziskanerstraße 38, 81669 München
 or null if no address data is given
*/
String buildAddress(FieldKey pFieldKey, Object pFieldValue,
 FieldVisualizationContext pContext) {
   def unit = pContext.getUnit() //this script is suited for units only, however
    ticket and resource are possible as well
   if(!unit) return null
   def unitDefName = unit.getDefinitionName()
   def street = unit.get("ResellerCompanyData.address")
   def nr = unit.get("ResellerCompanyData.house_nr")
   def zip = unit.get("ResellerCompanyData.zip")
   def city = unit.get("ResellerCompanyData.city")

   //check if we have enough data to construct an address, we need at least city
   if(!city) return null

   return "${street?street:''} ${nr?nr:''} ${zip?zip:''} $city"
}


/**
* Returns rendered map, should work ootb when you adapted buildAddress() properly
*
* @param pFieldKey - the field key
* @param pFieldValue - the field value
```

```
* @param pClient - the client type for which render method is called(e.g.: track,
 web)
* @param pContext - it provides id of an object from which given field comes from
*/
def render(FieldKey pFieldKey, Object pFieldValue, String pClient,
 FieldVisualizationContext pContext) {
  def address = buildAddress(pFieldKey, pFieldValue, pContext)
  if(!address) return "$notEnoughAddressDataMsg" as String
  return """
  <div id="mapid" style="height: ${mapsDivHeight}px; width:
   ${mapsDivWidth}px"></div>
  <script>
  var marker = L.icon({
     iconUrl: 'maps/images/marker-icon.png',
     shadowUrl: 'maps/images/marker-shadow.png',
     iconSize: [25, 41], // size of the icon
     shadowSize: [41, 41], // size of the shadow
     iconAnchor: [12, 41], // point of the icon which will correspond to marker's
      location
     popupAnchor: [1, -34] // point from which the popup should open relative to
      the iconAnchor
  });
  function geoCode(address, callback) {
     var xmlHttp = new XMLHttpRequest();
     xmlHttp.onreadystatechange = function() {
        if (xmlHttp.readyState == 4 && xmlHttp.status == 200) callback
         (xmlHttp.responseText);
     }
     xmlHttp.open("GET", "https://nominatim.openstreetmap.org/search?q=" + address
      + "&format=json", true);
     xmlHttp.send(null);
  }
  geoCode("${address}", function(geocodeStr) {
     var geocodeJson = JSON.parse(geocodeStr)[0];
     var lat = geocodeJson.lat;
     var lon = geocodeJson.lon;
     var map = L.map(mapid);
     var osmUrl="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png";
     var osmAttrib='Map data © <a
      href=\"http://openstreetmap.org\">OpenStreetMap</a> contributors';
     var osm = new L.TileLayer(osmUrl, {attribution: osmAttrib});
     map.addLayer(osm);
     map.setView(new L.LatLng(lat, lon), ${initialZoom});
     L.marker([lat, lon], {icon: marker}).addTo(map).bindPopup
      ("${address}").openPopup();
  });
  </script>
  """ as String
}


/**
* It returns list of resources which are required to visualize this custom field.
```

```
* Any files from {data.directory}/resources or external CDN server resource can be
 used here.
*
* All js or css files will be automatically registered on html page.
* To use other files simply use their names in render() method, they will be
 replaced with correct html url.
*
* data.directory - is the server shared directory configured in CM property
*
* @param pFieldKey - the field key
* @param pFieldValue - the field value
* @param pClient - the client type for which render method is called(e.g.: track,
 web)
* @param pContext - it provide id of object from which given field comes from
*/
def resources(FieldKey pFieldKey, Object pFieldValue, String pClient,
 FieldVisualizationContext pContext) {
   List<String> resources = ["maps/leaflet.js", "maps/leaflet.css",
    "maps/images/marker-icon.png", "maps/images/marker-shadow.png"] as String[];
   return resources;
}
```

Code example 91: *Admin Tool script of type Field visualization, used to indicate a location in an OpenStreetMap*

Remember to create a label (used in variable `notEnoughAddressDataMsg` in the example above) which codes the display message to inform the engineer if a map cannot be displayed, because some address data is missing. If you need more information about the work with labels, please read the section Labels.

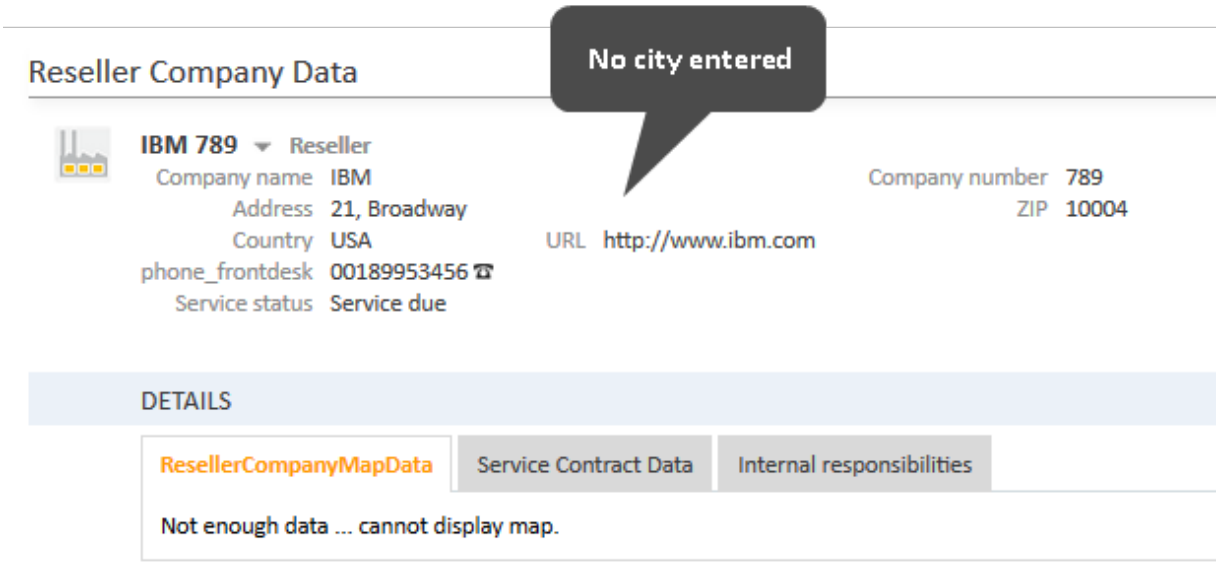


Figure 484: *ConSol CM Web Client - GUI message (based on a label) which is displayed if address data is missing*

Step 3: Configure the Data Field

Create a company field of type *String* and annotate it with *visualization*, thereby indicating the name of the script as value of the annotation. The script has to be of type *Field Visualization*. Assign the annotation `common, visualize-when-empty` to the data field. Otherwise, the map is only displayed if the field has some content.

> The annotation `common, visualize-when-empty` is applied for scripted field visualization both in the Web Client and CM/Track V2.

F.8.1.10 Scripts of Type *Workflow*

Scripts of this type are stored in the Admin Tool because they are used in numerous workflow scripts, i.e., the code in the Admin Tool script is needed more than once in one or more workflows. It is easier, less error-prone, and less time-consuming to store the scripts in one central location (Admin Tool) and just reference them in the workflows than to edit the same code in different locations in every workflow where it is used. Furthermore, during workflow development the Admin Tool script can be modified easily and the change is propagated immediately whereas when editing a workflow changes have to be deployed first.

Please see the *ConSol CM Process Designer Manual* for a detailed introduction to workflow programming. A short example will be provided here.

This code in a workflow activity will only reference the script, e.g.:

```
scriptExecutionService.execute(scriptProviderService.createDatabaseProvider
  ("DisplayCustomerData.groovy"))
```

In the Admin Tool, the respective script is stored in the navigation item *Scripts and Templates*, navigation group *System*.

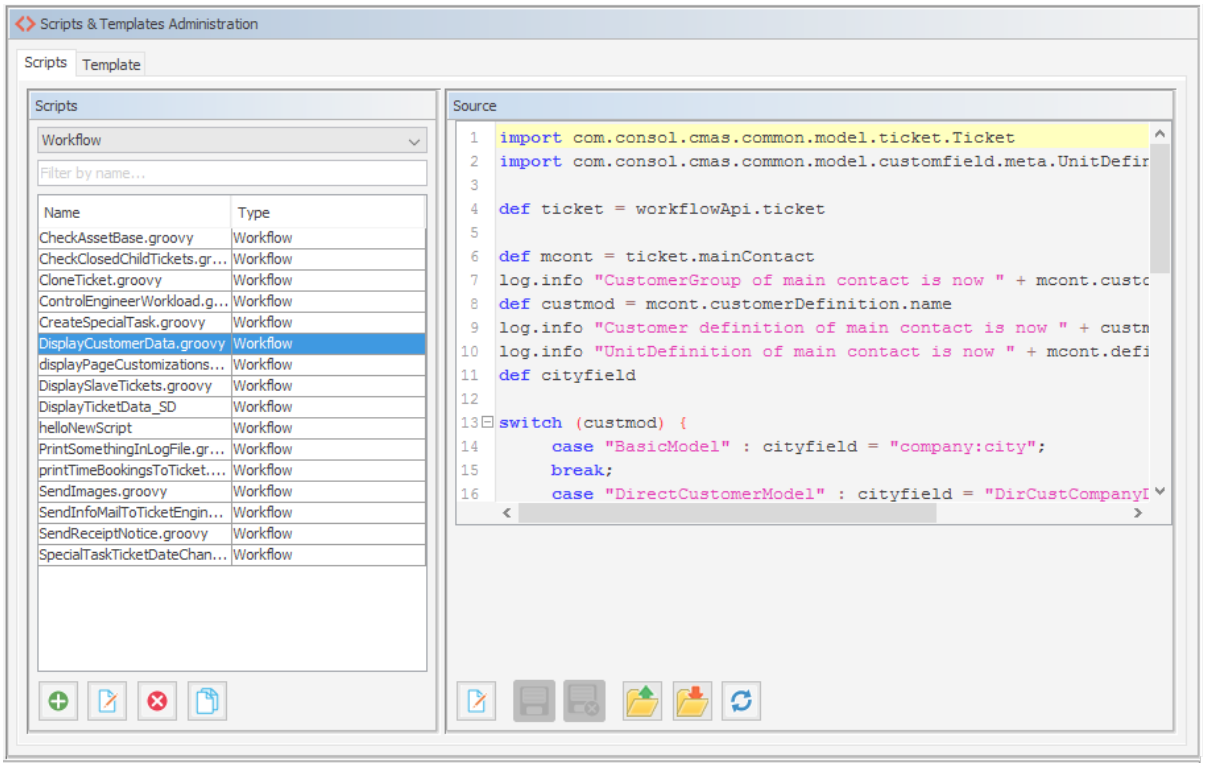


Figure 485: *ConSol CM Admin Tool - System, Scripts and Templates: Workflow script*

It is also possible to pass parameters (key/value pairs) to the Admin Tool script. This is explained in detail in the *ConSol CM Process Designer Manual*.

F.8.1.11 PostActivityExecutionScript

For certain use cases it might be required to execute a script when a ticket has run through a workflow activity. You might want to use this, for example, to display another ticket in the Web Client after the workflow activity has been executed. From a user's (engineer's) point of view, the Web Client **jumps** to the next ticket. The latter can be a child ticket or the next ticket in a list, depending on the use case.

The system behavior is defined in an Admin Tool script, the *PostActivityExecutionScript* (sometimes also called *Default Workflow Activity Script*). The name of the script has to be set in the system property cmweb-server-adapter, postActivityExecutionScriptName.

This script is executed after every **manual** workflow activity. That means you have to insert all control mechanisms and intelligence into the script:

- After which activity should the script do something?
  (for all other activities, nothing should happen)
- What should happen?

Starting with version 6.10.2, ConSol CM can open one of four different page types (in read mode) coming from the PostActivityExecutionScript:

- a ticket page
- a company detail page

- a contact detail page
- a resource page

The script just has to provide the respective object as return value. The following code shows an example for each of the four types.

```
switch(activity.name){
   case 'defaultScope/Goto_ticket':
      return ticketService.getByName("SUP-11")
   case 'defaultScope/Goto_contact':
      return unitService.getById(123)
   case 'defaultScope/Goto_company':
      return unitService.getById(456)
   case 'defaultScope/Goto_resource':
      return resourceService.getById(890)
}
```

Example: Jump to the next ticket in a list.

cmweb-server-adapter	pagemapLockDurationInSeconds	60
cmweb-server-adapter	postActivityExecutionScriptName	postActivityExecutionHandler
cmweb-server-adapter	queuesExcludedFromGS	

Figure 486: *ConSol CM Admin Tool - System, Properties: Property for definition of postActivityExecutionScriptName*

The PostActivityExecutionScript can also jump to a unit page (i.e., a company page or a contact page) or to a resource page simply by returning the unit or resource within the script.

Please see the following example script:

```
switch(activity.name){
   case 'defaultScope/Goto_the_ticket':
      return ticketService.getByName("SUP-11")
   case 'defaultScope/Goto_the_contact':
      return unitService.getById(123)
   case 'defaultScope/Goto_the_company':
      return unitService.getById(456)
   case 'defaultScope/Goto_the_resource':
      return resourceService.getById(890)
}
```

Code example 92: *PostActivityExecutionScript*

Another example, which also shows the Web Client behavior:

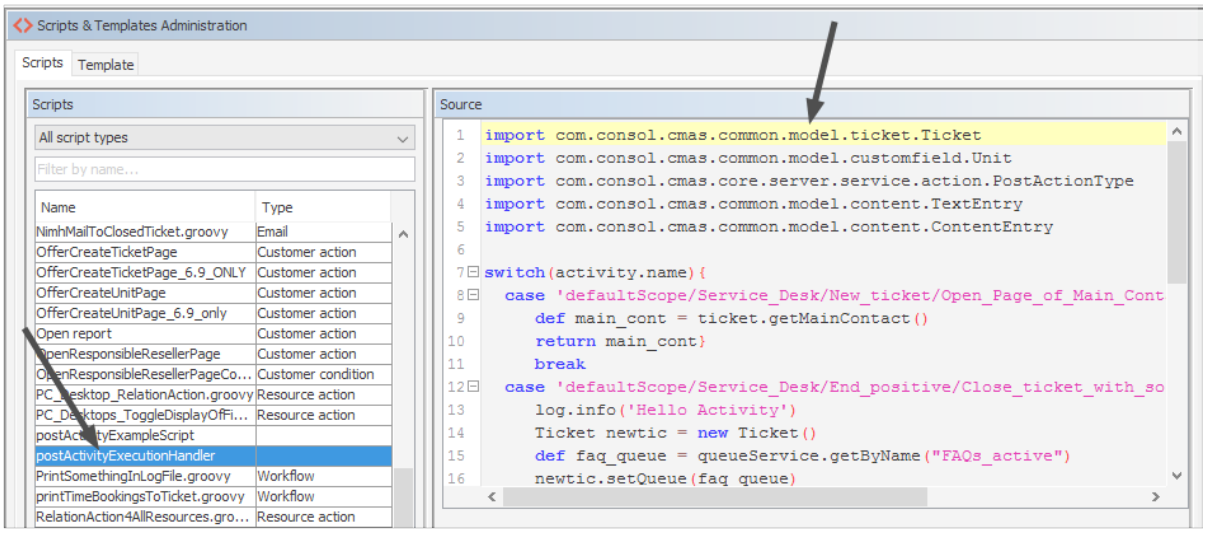


Figure 487: *ConSol CM Admin Tool - System, Scripts and Templates: PostActivityExecutionHandler script*

```
switch(activity.name){
   case 'defaultScope/Service_Desk/New_ticket/Open_Page_of_Main_Contact':
      def main_cont = ticket.getMainContact()
         return main_cont
   // ( ... )
}
```

Code example 93: *PostActivityExecutionHandler*

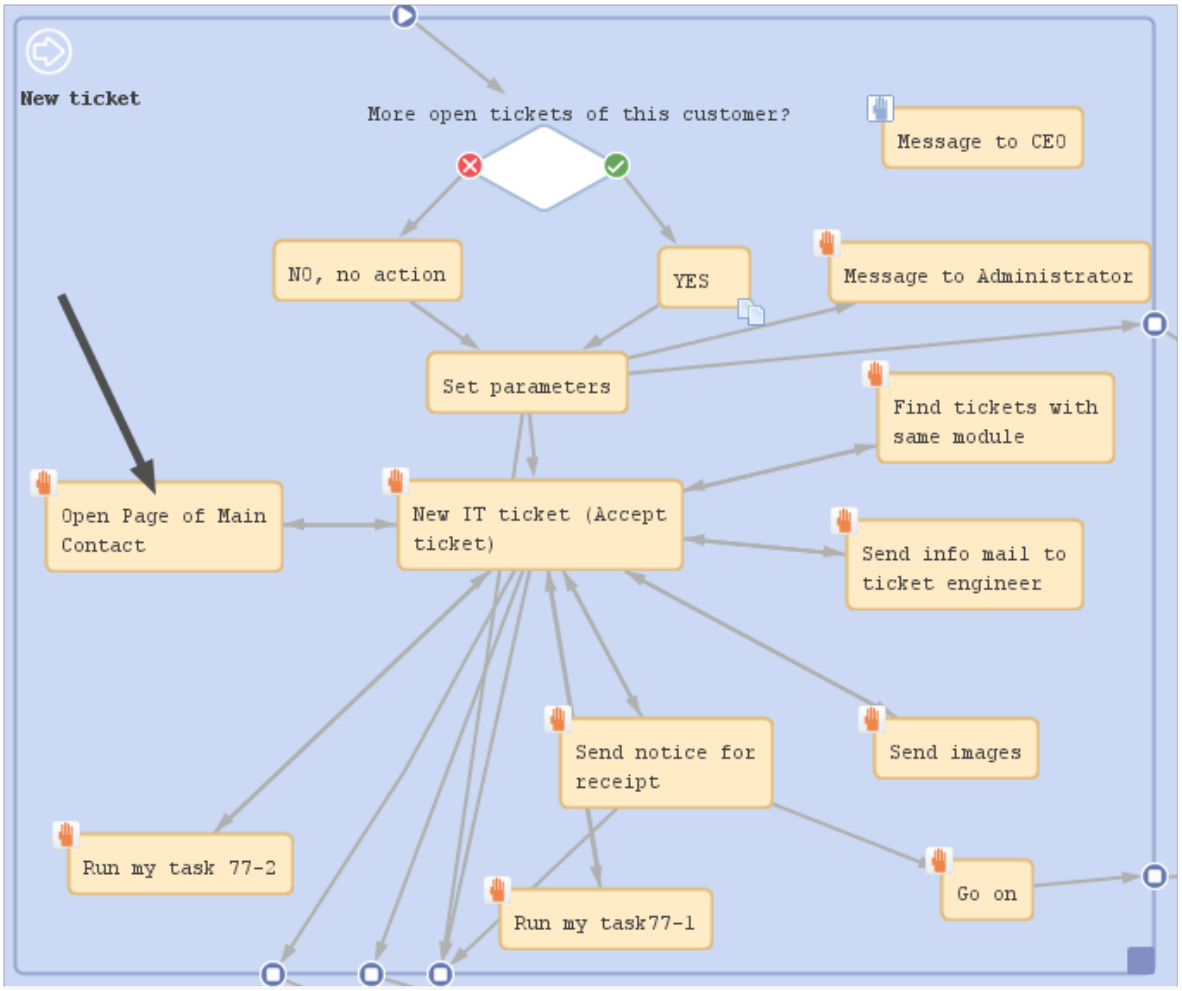


Figure 488: *ConSol CM Process Designer - Activity which will be controlled using the PostActivityExecutionScript*

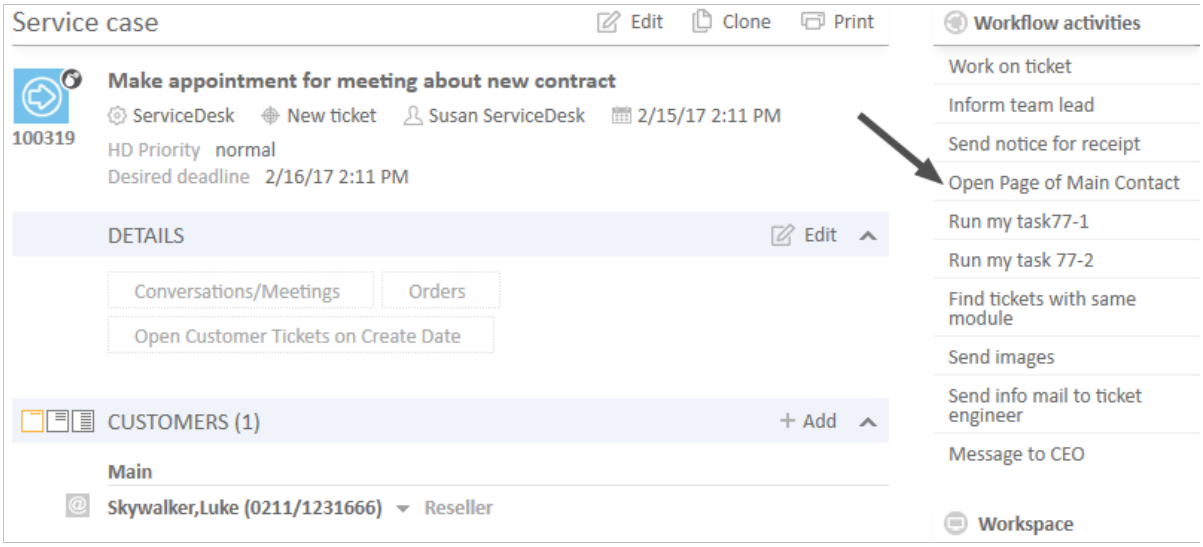


Figure 489: *ConSol CM Web Client - Ticket with workflow activity which will jump to contact page*

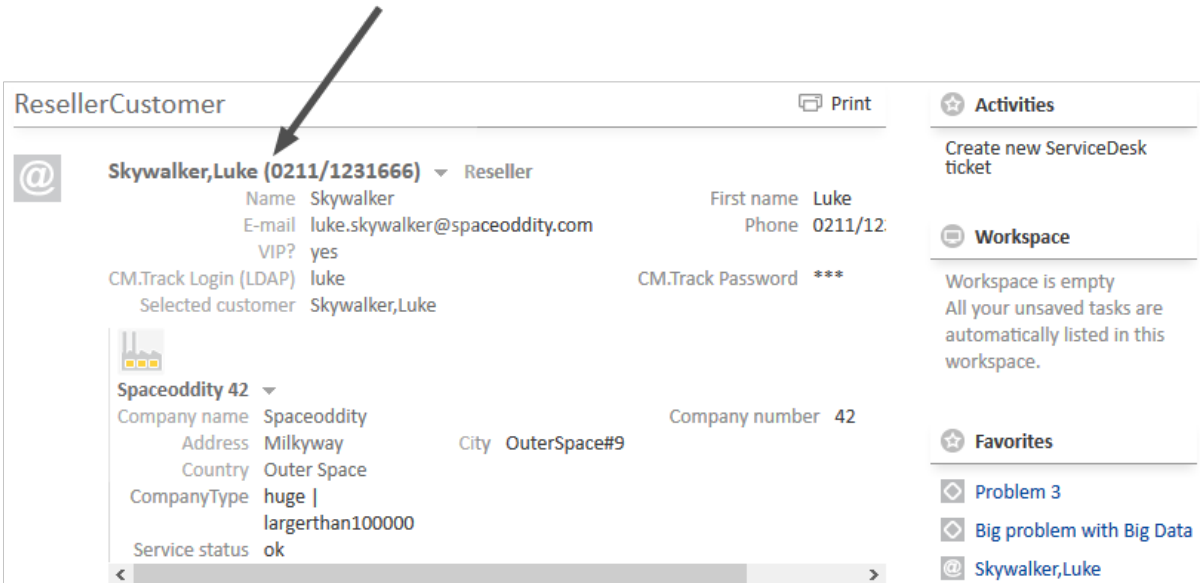


Figure 490: *ConSol CM Web Client - Contact page which has been opened directly from ticket*

F.8.2 Admin Tool Templates

F.8.2.1 Introduction to Templates in the Admin Tool

In ConSol CM, several types of templates are used:

- **Email templates** are stored in one of the following places:
  - the *Text Template Manager*, see section The ConSol CM Text Template Manager.
  - the *Templates* section of the Admin Tool. Details on that topic are provided in this chapter.
- **Document templates** are stored in
  - the Document Template Manager (part of CM/Doc), see section CM/Doc
- **Data representation templates** are stored in
  - the *Scripts & Templates Administration* of the Admin Tool. They are used for:
    - the definition of customer templates, see section Templates for Customer Data
    - the definition of resource templates, see section CM/Resource Pool - Templates for Resource Data
- **General templates** are stored in
  - the *Scripts & Templates Administration* of the Admin Tool. They are explained in this chapter.

In this chapter, the general templates in the *Scripts & Templates Administration* of the Admin Tool are explained.

Admin Tool templates are written using ***FreeMarker*** notation (see FreeMarker web site) and should only be edited by experienced ConSol CM consultants and administrators. A ConSol CM standard installation already contains system templates and some example templates which might help you, as an administrator, to define new templates for your special use cases.

F.8.2.2 The Admin Tool Template Editor

To work with templates, open the navigation item *Scripts and Templates* in the navigation group *System* and switch to the *Template* tab.

In the templates list, all templates are listed with:

- **Name**
  Mandatory. A template is referenced by its name when it is referenced by other objects.
- **Group**
  Optional. Groups help you temporarily sort the templates in the templates list. This setting does not have any technical implications and will be lost once you reload the data in the Admin Tool.

To open a template in the editor panel, mark it in the list and open it by clicking the *Edit* button. Each template must have a name, whereas the group name is optional.

If your system works with various languages, you can define each template for each language. Use the drop-down menu *Language* above the editor panel. The Web Client will display the template for the configured locale of the web browser. If there is no template for this language, the default language will be used. Every template has to be defined for the default language.

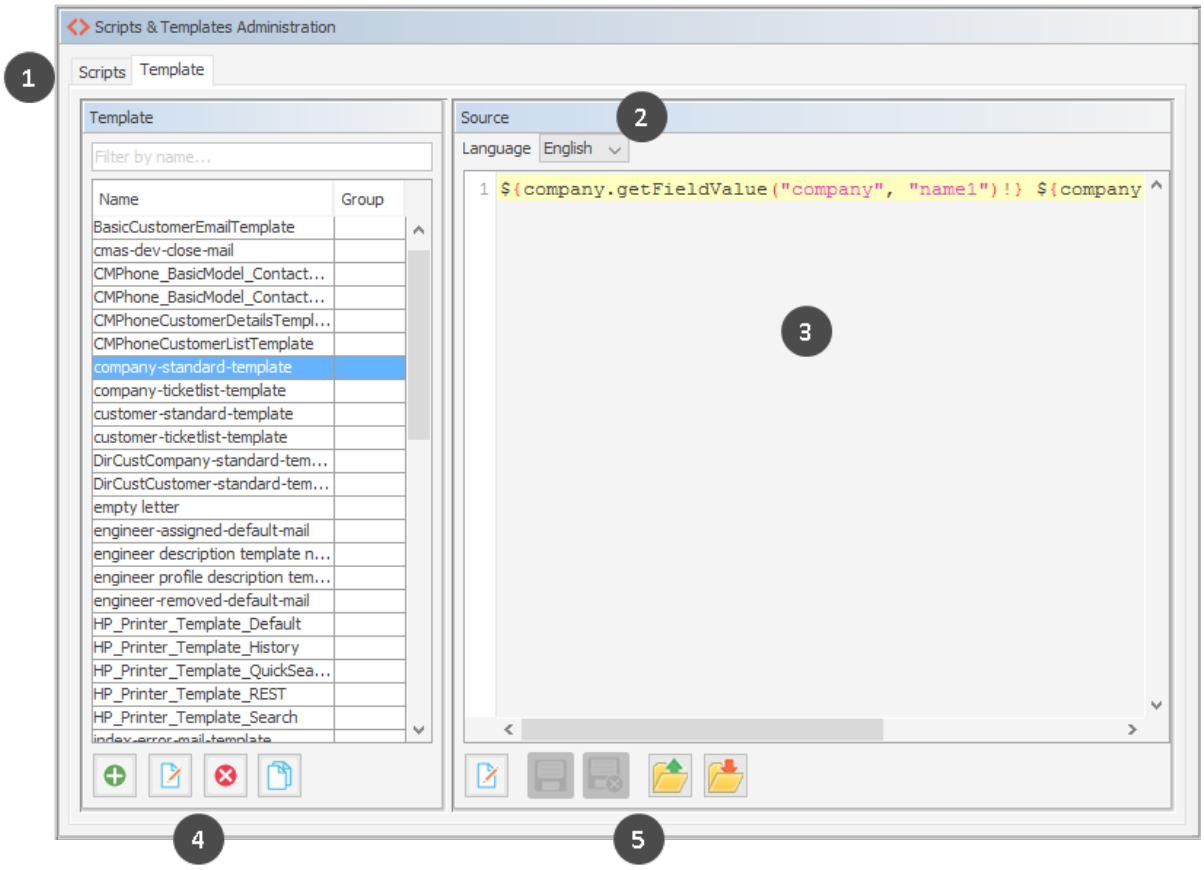


Figure 491: *ConSol CM Admin Tool - System, Scripts and Templates: Template editor*

The *Templates* screen shows the following items:

- Tabs to switch between scripts and templates (1)
- Language selector (2)
- Editor panel (3). Displays the template which is selected in the list.
- General buttons to add, delete or copy a template and to edit the template name and group (4)
- Buttons referring to the open template (5)
    - Edit the selected template
    - Save the template
    - Cancel (quit without changing the changes)
    - Upload a text file (as template) from the file system
    - Save the template as a file in the file system

F.8.2.3 Working with Admin Tool Templates

The Admin Tool templates represent a template pool. Each template can be referenced from different modules of the system and is always referenced by its name. In the following paragraphs, all modules where templates can be used are explained. Within a template, the customer fields, resource fields, and ticket fields are referenced by group name and field name, e.g., the company name within the customer field group *ResellerCompanyData* is referenced as shown in the following example.

```
${ResellerCompany.getFieldValue("ResellerCompanyData","company_name")!}
```

For a detailed explanation of working with ticket fields, please see section Ticket Field Administration (Setting Up the Ticket Data Model). Customer fields are explained in section Setting Up the Customer Data Model. Resource fields are explained in section CM/Resource Pool - Setting Up the Basic Resource Model.

⚠ Do not use line breaks in template statements!

System Templates

A default ConSol CM installation comes with several system templates. They are used in standard situations like error messages sent to an administrator. Please see the following list for an overview of the system templates:

- **attachment-type-error-mail-template**
  An email with this template is sent to the email administrator (email address given in system property cmas-nimh-extension, mail.process.error) when the attachment type of an incoming or outgoing email is not supported and thus the email cannot be processed.
- **cmas-dev-close-mail**
  Not used. Removed from the standard installation in ConSol CM version 6.10.1.
- **engineer description template name**
  Template used to render the engineer label, e.g., ticket owner. The template name “engineer description template name” is the default value. You can define another name for the template which should be used to render engineer names by using the CM system property cmas-core-server, engineer.description.template.name.
- **engineer profile description template name**
  Template used to render the label on a header of the page, next to the logout button.
- **index-error-mail-template**
  Not used. Removed from the standard installation in ConSol CM version 6.10.1.
- **password-reset-template**
  Template for the body of the email which is sent when a user requests a password reset (on login page).

- **representation_info_email_html**
  All emails sent by CM to the represented engineer are also sent to the representing engineer (see *Global Permissions: Representation Permissions* in section Role Administration). The template is used to configure the text which is added to the forwarded email.
- **representation_info_email_plain_text**
  Same as above, as plain text.
- **representation-create-email**
  The template for the email which is sent when a new representation configuration has been set up, e.g., when an engineer has selected a colleague of his as representing engineer, because he will be on vacation for some time. The email is sent to both, the new representing engineer and the engineer who is represented. All engineer-specific properties are available as variables in the template, for the representing engineer, e.g.:
  - Last name: ${representing.lastname}
  - First name: ${representing.firstname}
  - Login: ${representing.name}
  - Email: ${representing.email}
  - Fax: ${representing.fax}
  - Mobile: ${representing.mobile}
  - Phone: ${representing.phone}

  The variables for the engineer who is represented are built according to the same pattern, e.g., Last name: ${represented.name}.
- **representation-delete-email**
  The template for the email which is sent when a representation configuration has been stopped, e.g., when an engineer has deleted the representation configuration for a colleague of his as representing engineer, because he has returned from vacation. The email is sent to both, the old representing engineer and the engineer who was represented. For variables which are available, please see the previous paragraph.

Templates for Definition of Customer Format in the Web Client

The appearance of **customer data** (e.g., name, phone number, and room number or surname and forename only) in different sections of the Web Client can be formatted using templates. The definition has to be made for each customer object (i.e., for each company definition and for each contact definition) so that specific templates can be used within each customer group. The configuration of templates for customer objects is explained in section Templates for Customer Data.

The same principle applies to the representation of **resource data** (if CM/Resource Pool is active in your ConSol CM system). For a detailed explanation, please refer to section CM/Resource Pool - Templates for Resource Data.

In the following example, the customer data within the *ResellerCustomer* customer object should be represented in the standard template with forename and surname.

The following figure shows the template in the template manager. You reach this screen by opening the navigation item *Scripts and Templates* in the navigation group *System*, clicking the *Template* tab and selecting the desired template in the list.

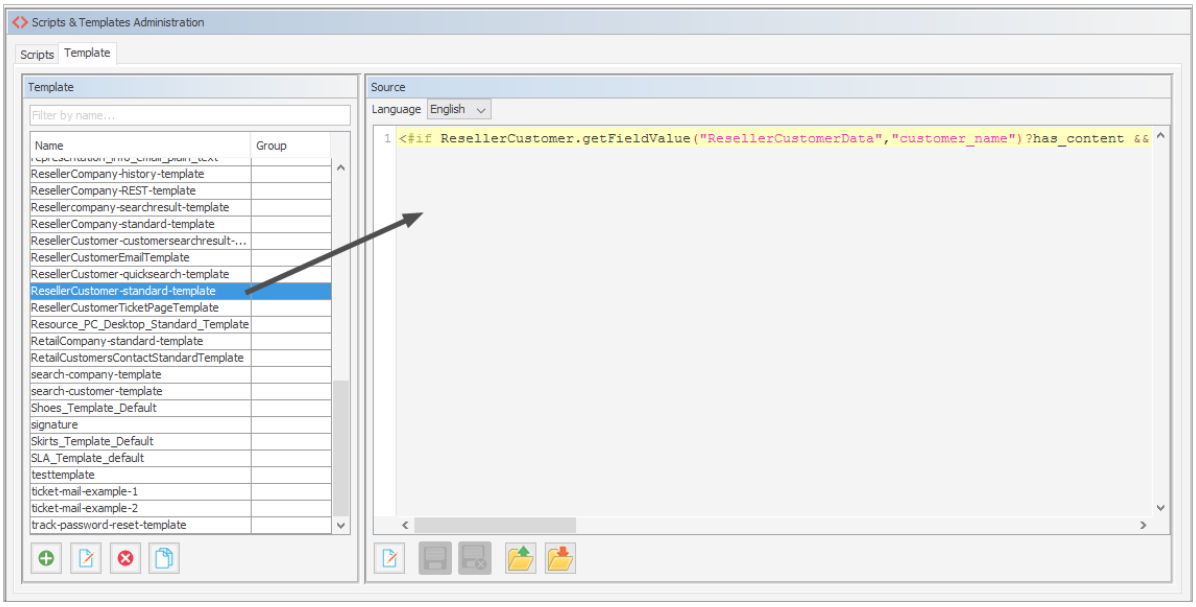


Figure 492: *ConSol CM Admin Tool - System, Scripts and Templates: Example of a customer format definition template*

```
<#if ResellerCustomer.getFieldValue("ResellerCustomerData","customer_name")?has_
 content && ResellerCustomer.getFieldValue("ResellerCustomerData","forename")?has_
 content>${ResellerCustomer.getFieldValue("ResellerCustomerData","customer_
 name")!},${ResellerCustomer.getFieldValue
 ("ResellerCustomerData","forename")!}<#else> ${ResellerCustomer.getFieldValue
 ("ResellerCustomerData","customer_name")!}</#if>Search for contacts of a certain
 company
```

Code example 94: *Customer format definition template*

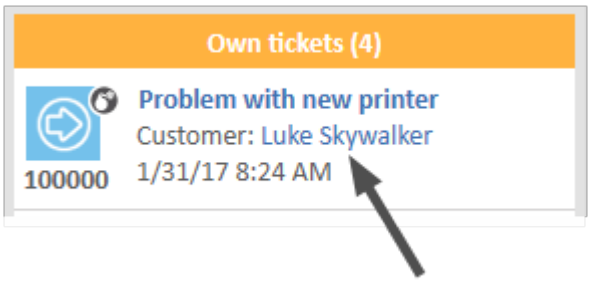


Figure 493: *ConSol CM Web Client - Example of a customer format definition template*

Ticket Assignment Templates

In the queue administration (see section Queue Administration), ticket-engineer-assignment templates can be selected. There are templates for the use cases *assign* and *remove*.

- The *assign* template (*Assign*) is used as text template for an automatic email which is sent by the system to the (new) engineer when a ticket is assigned to the engineer.
- The *remove* template (*Unassign*) is used as text template for an automatic email which is sent by the system to the (old) engineer when a ticket has been taken from the engineer.

You have to write and save the templates here in the *Template* section first. Then they will be available in the drop-down menu in the *Ticket assignment templates* section of the queue administration (see section Queue Administration). You can define a name of your choice - we recommend using a name which describes the use of the template, as shown in the following example.

The following figure shows the template in the template manager. You reach this screen by opening the navigation item *Scripts and Templates* in the navigation group *System*, clicking the *Template* tab and selecting the desired template in the list.

As default `From` address, the email address of the CM administrator (CM system property admin.e-mail) is used. You can also set the from address explicitly within the template (see following example).

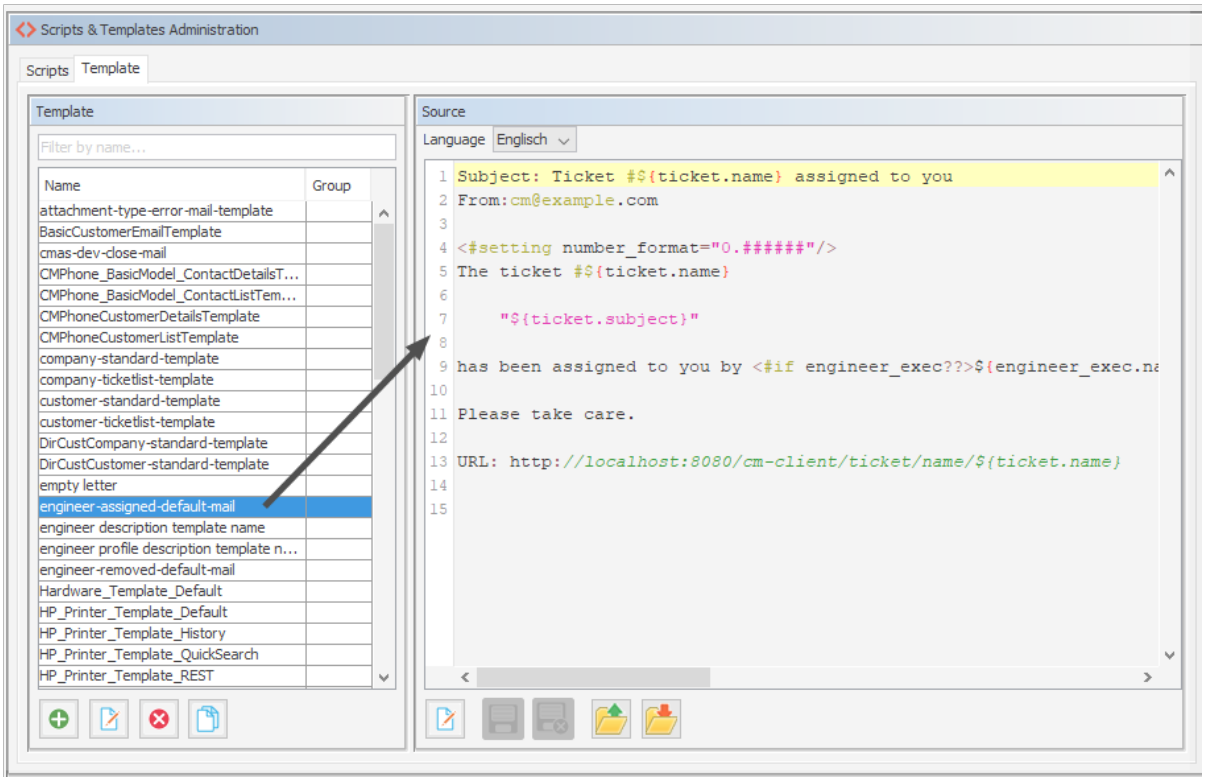


Figure 494: *ConSol CM Admin Tool - System, Scripts and Templates: Example of an email template for ticket assignment to an engineer*

```
Subject: Ticket #${ticket.name} assigned to you
From: cm@example.com

<#setting number_format="0.######"/>
The ticket #${ticket.name}

   "${ticket.subject}"

has been assigned to you by <#if engineer_exec??>${engineer_exec.name}<#else>the
 workflow</#if> <#if engineer_old??>(former engineer: ${engineer_old.name})<#else>
 (no former engineer)</#if>

Please take care.

URL: http://localhost:8080/cm-client/ticket/name/${ticket.name}
```

Code example 95: *Text of the example template engineer-assigned-default-mail*

The following figure shows the queue details. You reach this screen by opening the navigation item *Queues* in the navigation group *Global Configuration* and selecting the desired queue in the list.

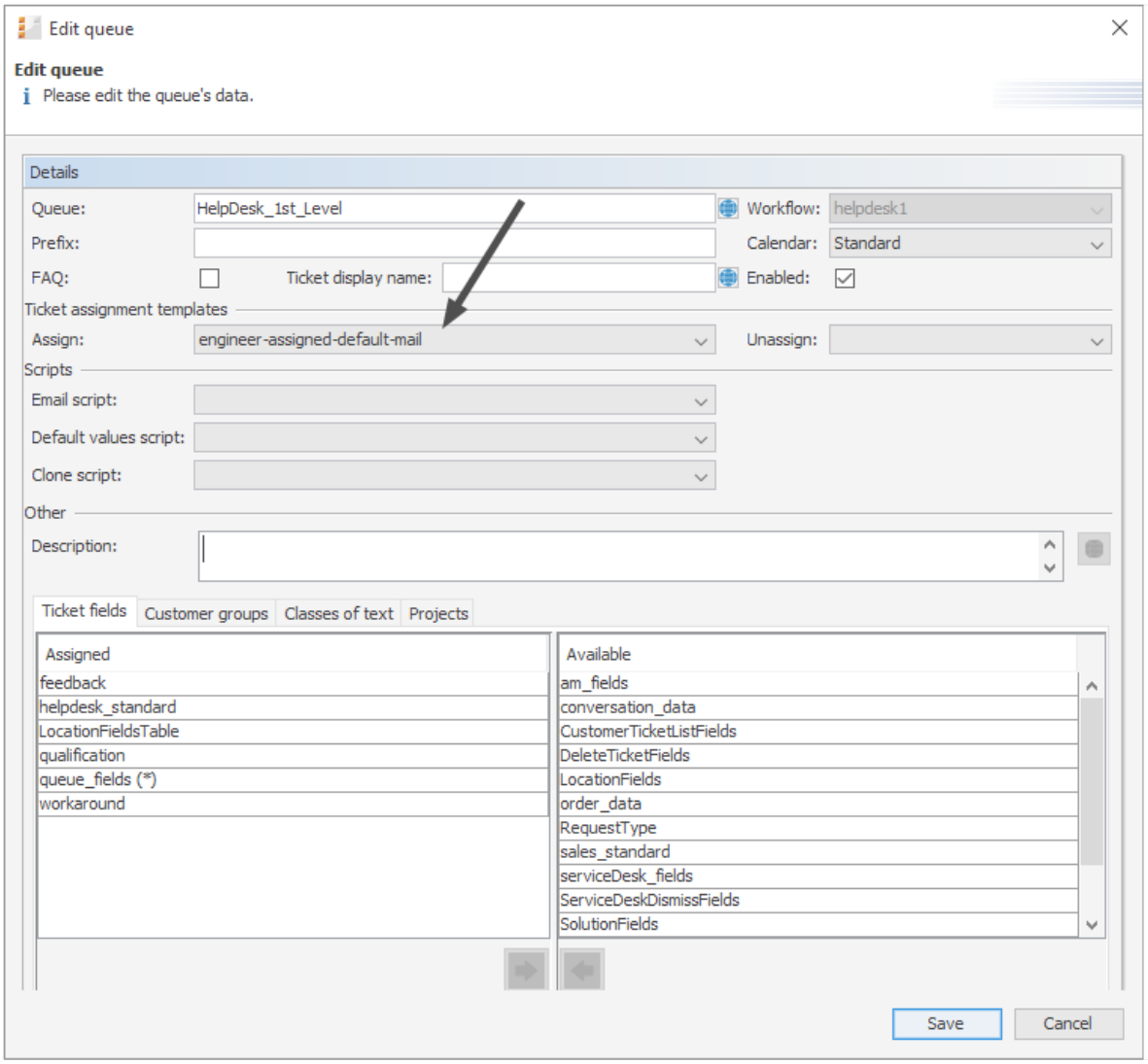


Figure 495: *ConSol CM Admin Tool - Global Configuration, Queues: Edit the queue to specify a ticket assignment template*

Password Reset Templates

When users forgot their passwords, they can request a new password using ConSol CM standard functionality. The user might be an engineer who works with the Web Client or a customer who has a login to CM/Track. For both cases, the administrator has to configure ConSol CM in a way that an email can be sent to the user who needs a new password. This email is based on a template which is stored in the Admin Tool, in the *Template* section. The email contains a hyperlink to a page where the password reset can be performed.

Password Reset Template for Engineers in the Web Client

When an engineer has forgotten his Web Client password, he can request a new password by using the link *Forgot your password?* on the initial login page. An email with a link to an URL where the engineer can set the new password is sent to the engineer.

> ⚠ Please note that this can only work if a valid email account is available for this engineer and if the respective value has been entered as email address for the engineer in the engineer data!

The email which is sent to the engineer is based on the template `password-reset-template`. This template name is mandatory, required for the password to work. The template could look like the following example:

```
Subject: Password reset procedure

<#setting number_format="0.######"/>
To reset your password please click the following link:

http://localhost:8888/cm-client/passwordChange?resetCode=${resetCode}

This link expires at ${expirationDate?string("yyyy.MM.dd HH:mm:ss")}.
```

Code example 96: *Template to reset the engineer's password*

The `From` address of the email which is sent to the engineer is defined by the CM system property cmas-core-security, password.reset.mail.from.

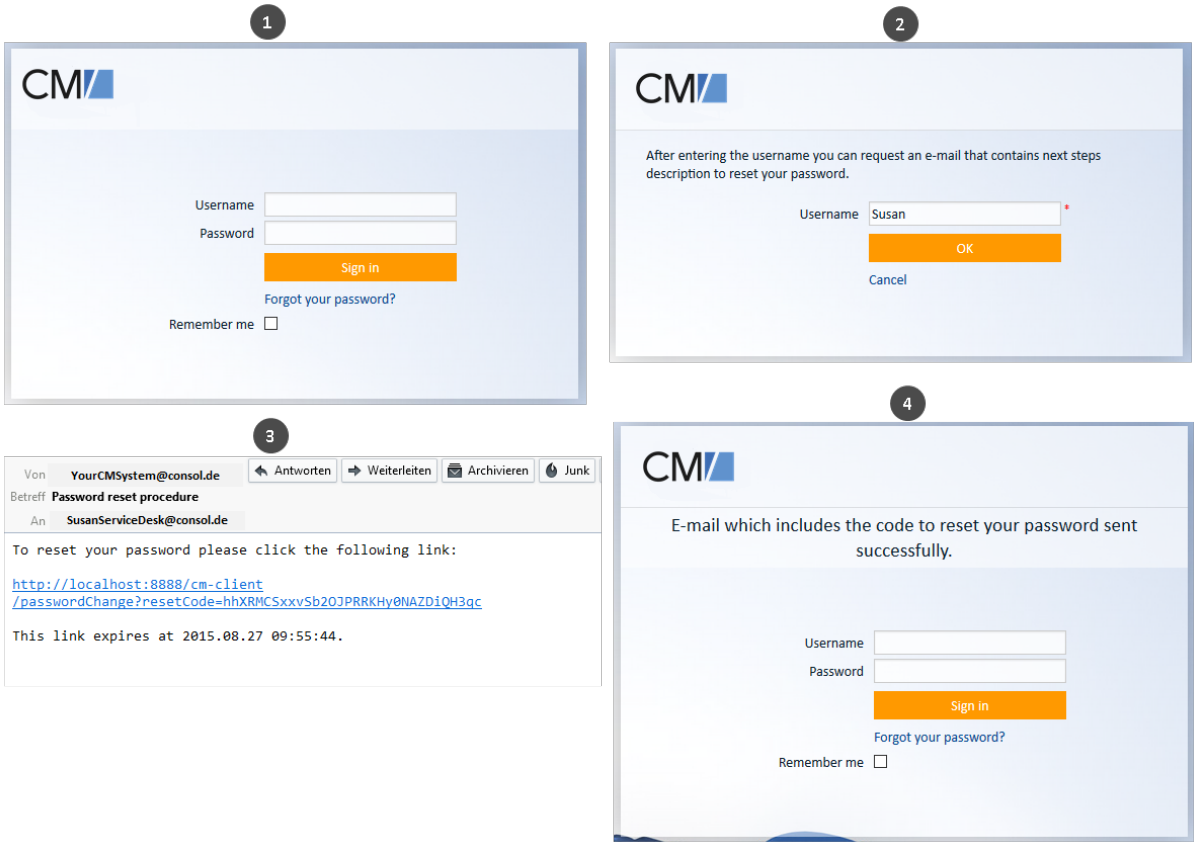


Figure 496: *ConSol CM Web Client - Password reset by an engineer*

Please note that the password reset in the Web Client is only possible if the standard mode is used. It is not possible if LDAP or Kerberos authentication is in operation. See section Authentication Methods in ConSol CM for an explanation of all possible authentication modes.

Password Reset Template for Customers Using CM/Track V1

When a customer has forgotten his CM/Track password, he can request a new password by using the link *Forgot your password?* on the initial login page. An email with a link to an URL where the customer can set the new password is sent to the customer.

Please note that this can only work if a valid email account is available for this customer and if the respective value has been entered as email address for the customer in the respective customer field.

The email which is sent to the customer is based on the template `track-password-reset-template`. This template name is mandatory, required for the password to work. The template has to be created/added manually, it will not be present in the system by default.

The template should be formatted like the following example (you can add any text you would like to send to your customers, but please note the `resetCode` variables and the URL!):

```
Your Password Reset Link:
http://mycm6system/cm-track/#track=set_new_password/resetCode-${resetCode}(Reset
 Code: ${resetCode})
Valid 24 hours only, please visit before expiry!
```

Code example 97: *Reset the password of a customer in CM/Track*

As From address for the email to the customer, the administrator email address is used.

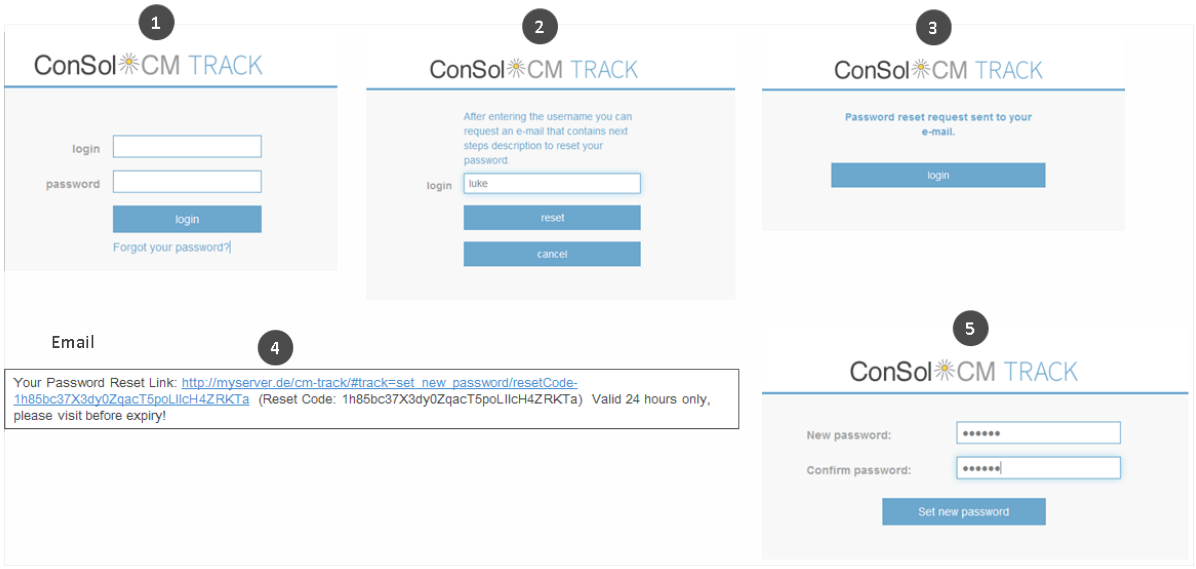


Figure 497: *CM/Track - Password reset by a customer*

Please note that the password reset in CM/Track V1 is only possible when the DATABASE mode is used. It is not possible when LDAP authentication is in operation. See section Authentication Methods for Customers in CM/Track for an explanation of all possible authentication modes.

Password Reset Template for Customers Using CM/Track V2

When a customer has forgotten his CM/Track password, he can request a new password by using the link *Forgot your password?* on the initial login page. An email with a link to an URL where the customer can set the new password is sent to the customer.

Please note that this can only work if a valid email account is available for this customer and if the respective value has been entered as email address for the customer in the respective customer field.

The email which is sent to the customer is based on the template `track-password-reset-template`. This template name is mandatory, required for the password to work. The template has to be created/added manually, it will not be present in the system by default.

The template should be formatted like the following example (you can add any text you would like to send to your customers, but please note the `resetCode` variables and the URL!):

```
Subject: Your Password Reset Link
Your Password Reset Link:
<#setting number_format="0.######"/>
To reset your password please click the following link:
http://myserver:myport/track/#/password-reset/resetCode-${resetCode}
This link expires at ${expirationDate?string("yyyy.MM.dd HH:mm:ss")}.
```

Figure 498: *Admin Tool template for customer password reset in CM/Track V2 (replace myserver and myport with your system parameters)*

Please note that the variable `expirationDate` is a system variable which is set to a date 24 hrs later than the time of the password-reset request by default. You can change the variable using the CM system property cmas-core-security, resetCode.expirationPeriod (Integer, milliseconds). The system property is not present by default but has to be created if it is required.

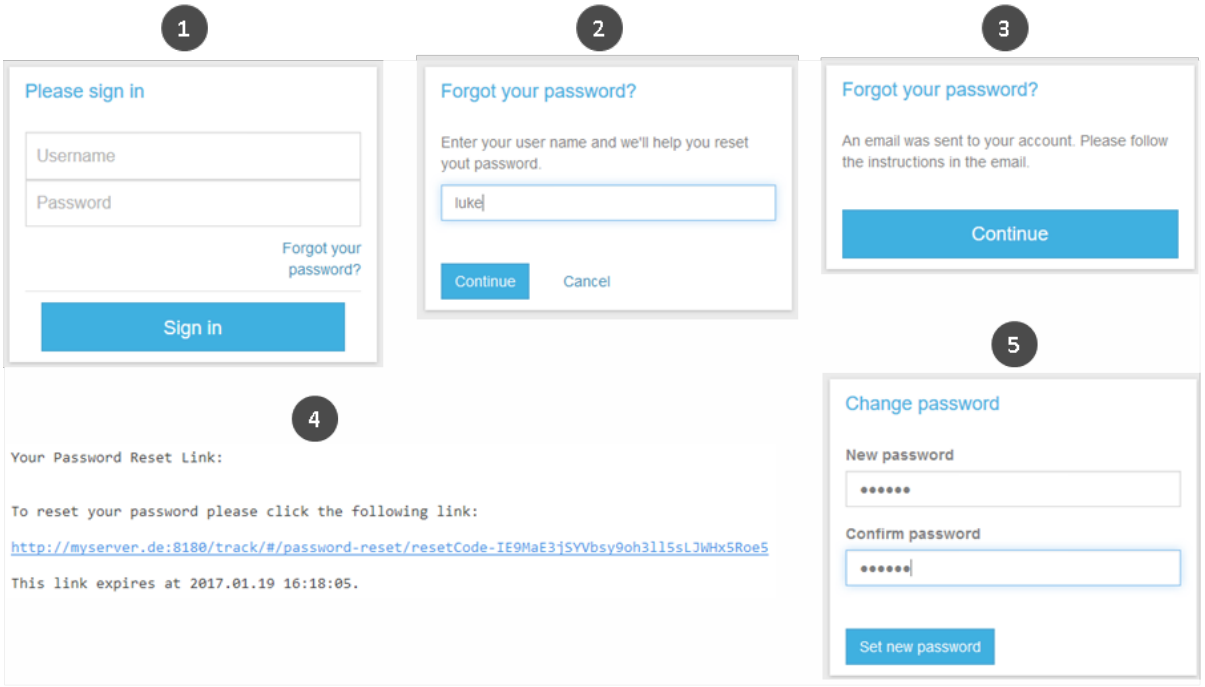


Figure 499: *CM/Track - Password reset by a customer*

Please note that the password reset in CM/Track V2 is only possible when the DATABASE mode is used. It is not possible when LDAP authentication is in operation. See section Authentication Methods for Customers in CM/Track for an explanation of all possible authentication modes.

Custom Defined Templates

A ConSol CM administrator or workflow developer can define any template that is required and store it in the *Scripts & Templates Administration*. When used in automatic emails which are sent by a workflow activity, you can always use the workflow API `renderTemplate()` method to reference a template. However, most email templates should be managed using the *Text Template Manager* (see section The ConSol CM Text Template Manager). There are only very few use cases which might require that email templates, or parts of email templates, have to be stored in the *Scripts & Templates Administration* of the Admin Tool.

F.9 Deployment (Import/Export)

This chapter discusses the following:

F.9.1 Introduction

ConSol CM offers the option to export the system configuration with or without run-time data into a file and to import this file into another ConSol CM system. The transfer file is called a *scenario* (sometimes also called a *scene*). It can contain various data - details are explained in the following sections.

Usually a scenario with configuration data is used to transfer data from a test to a staging or production system.

Since ConSol CM works with transfer keys for all objects it is highly recommended to transfer test to staging or production environments by using a scenario, and **not** to implement the same functionality in both systems. In case a transfer (export/import) is performed later on, there will be duplicate objects! Please read the detailed explanations below.

F.9.2 Scenarios

A scenario is a file in a proprietary ConSol CM format (similar to `.zip`, `.jar`, and `.tar`) that contains the data of a ConSol CM installation. It can be exported from one ConSol CM system and imported into the same or another system. This can be very helpful, e.g., when a test scenario is built on a test system which can then be transferred to a production server.

When an export file is created (see detailed explanation in the sections below) the administrator can decide which data should be included.

A scenario will **always** contain:

- all customer-specific system properties,
  i.e., system properties where the module name starts with *custom-*, please see section CM system properties for details!
- all page customizations
- all REST client GUI configurations

A scenario **can** contain, depending on the selection of the administrator (see figure below):

- run-time data
- configuration data

A scenario will **never** contain:

- general (not customer-specific) system properties (e.g., mail server, LDAP directory, etc.)

F.9.3 Deployment (Import/Export) Using the Admin Tool

In the Admin Tool, the deployments are managed using the navigation item *Import/Export* in the navigation group *System*. In this navigation item you can import or export scenarios (i.e., the whole configuration or part of it) in an application-specific format. You usually do this to transfer data between different ConSol CM installations. A typical example is transfer of the configuration from a test system to a production system.

All export/import actions are logged in the file `transfer.log`. A detailed description of this file is provided in the section Logging and Log Files.

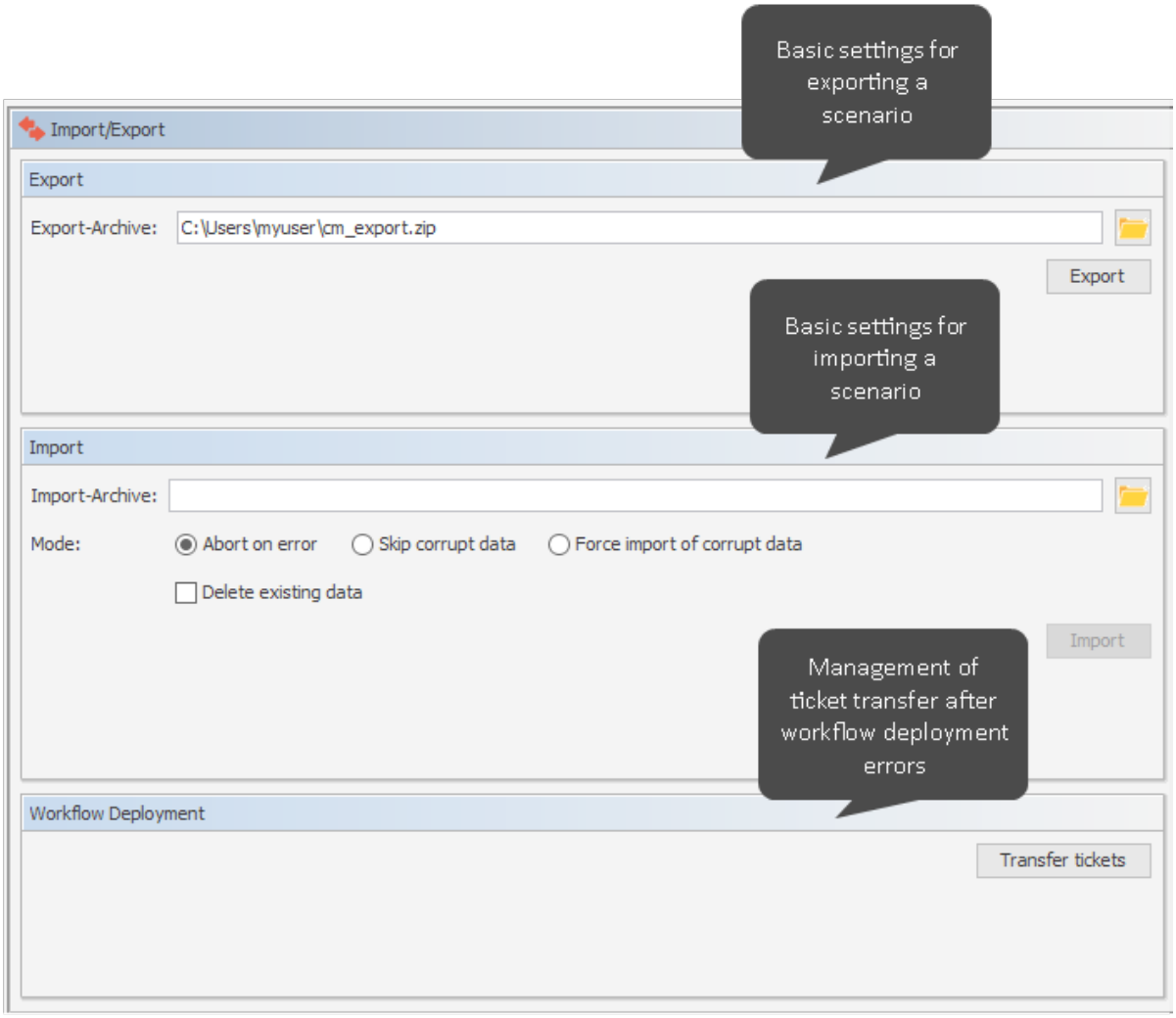


Figure 500: *ConSol CM Admin Tool - System, Import/Export*

The import of external data can modify, overwrite, or delete existing data irrecoverably. Although the user is prompted for confirmation at critical points during deployment, this cannot prevent erroneous handling. Use this function **only** if you are very sure what you are doing. In case of doubt please ask the ConSol CM support team or a ConSol CM consultant for assistance.

F.9.3.1 Export

- **Export-Archive:**
  Path to the export archive. When you open the navigation item *Import/Export*, the default path for the export (`<home directory of the admin user>/cm_export.zip`) will be set. Use this value or enter the path and name of the file you want to create. Alternatively, you can click the *Folder* button to open the file browser.

Click on *Export* afterwards to start the data export.

You will have to select the data that should be included in the export file (scenario):

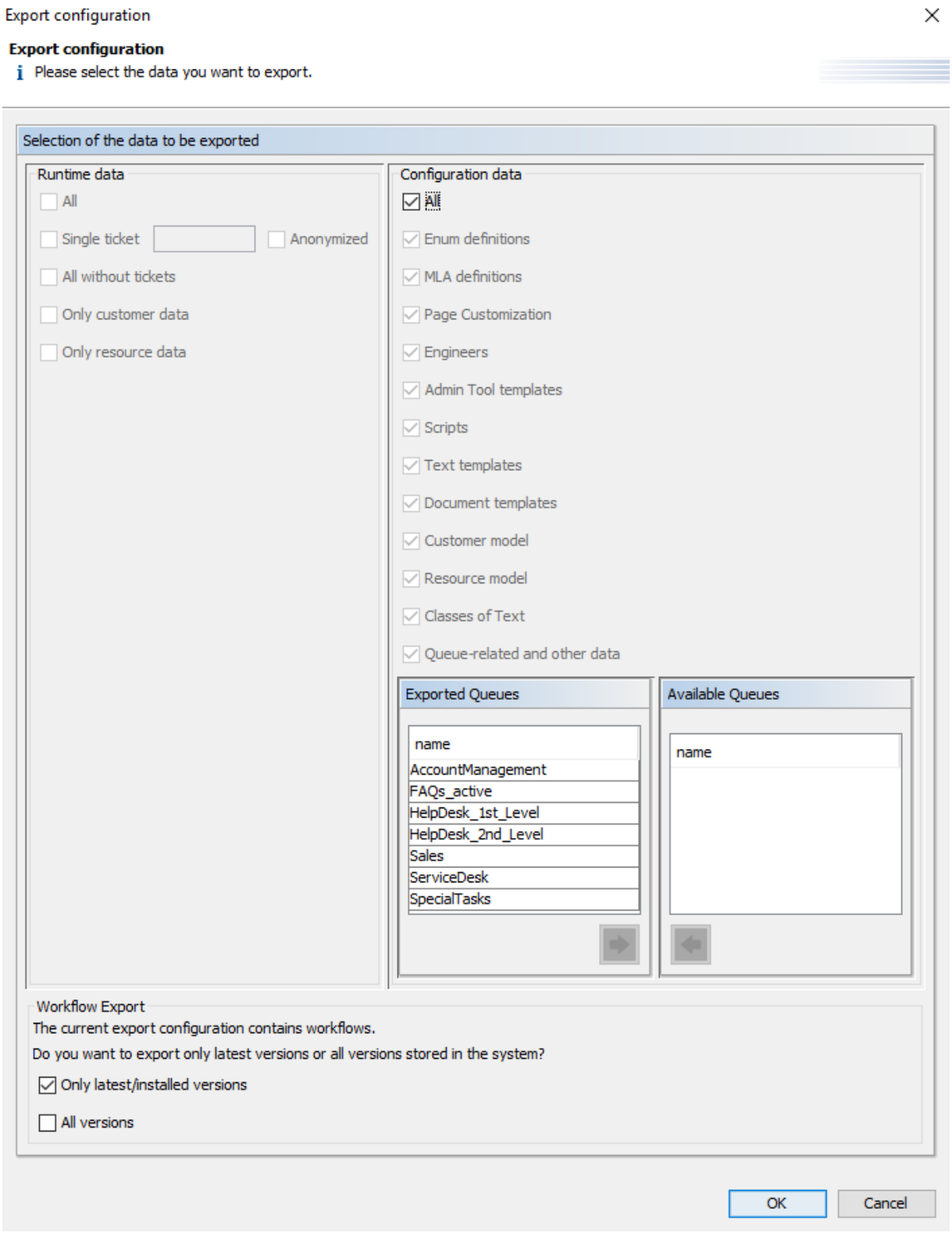


Figure 501: *ConSol CM Admin Tool - System, Import/Export: Export configuration*

- **Runtime data**
  This refers to data that is stored as operating data, e.g., tickets, resource data, and customer data.
  - **All**
    Ticket data, customer data, is exported completely **and** the complete configuration is exported. When you select the checkbox *All*, all other checkboxes are selected automatically. If CM/Resource Pool is active, the resource data is also exported.
  - **Single ticket**
    If the checkbox *Single ticket* is ticked, you have to insert the ticket name in the input field. (The ticket name is indicated below the ticket icon in the Web Client.)
    Tick the checkbox *Anonymized* to anonymize (hash) the exported data. In this way, no real customer, engineer, or resource information will be transferred to the importing system. The following data will be anonymized:
    - Content of ticket fields (some values as enum aren't anonymized)
    - Ticket attachments
    - Ticket history
    - Content of customer fields
    - Unit history

    
    The *Single ticket* option is not supposed to be used for regular system operation, but it is a functionality which is offered to facilitate bug-tracking. The export produces a .zip/.jar file containing the ticket data in XML files. All data required for the ticket (i.e., all related objects) are exported with the ticket, so be careful when the ticket is imported into another system - data might be changed here!
  - **All without tickets**
    The complete installation beside the tickets is exported, i.e., the customer data **and** the complete configuration. When you select the checkbox *All without tickets*, all other checkboxes except for *All* are selected automatically. If CM/Resource Pool is active, the resource data is also exported.
  - **Only customer data**
    Only customer data (i.e., the customer data model and the actual customer data) is exported. Nothing else. (The checkbox *Customer model* is checked automatically.)
  - **Only resource data**
    Only resource data (i.e., the resource data model and the actual resource data) is exported. Nothing else. (The checkbox *Resource model* is checked automatically.)
- **Configuration data**
  This refers only to the configuration in the Admin Tool, no run-time data is exported. Depending on the selection you have made, other check boxes might be checked automatically in order to cover all dependencies within the exported data.

- **All**
  The complete configuration is exported. When you select the checkbox *All*, all other checkboxes under *Configuration data* are selected automatically.
- **Enum definitions**
  The definitions and localizations of sorted lists (enums) are exported. This represents the data which is defined in the navigation item *Enums* (navigation group *Lists*). For details about enums, please refer to section Managing Sorted Lists: Enum Administration.
- **MLA definitions**
  The definitions and localizations of MLAs (multi level attributes) are exported. This represents the data which is defined in the navigation item *MLAs* (navigation group *Lists*). For details about MLAs, please refer to section MLA Administration.
- **Page Customization**
  Only the page customizations which have been defined specifically for the system (i.e. not the default values) are exported. These are the values which are stored in the database tables `cmas_web_customization` and `cmas_web_customization_values`. For a details overview of this topic, please refer to section Page Customization.
- **Engineers**
  Only the engineers with their data are exported. This also includes the roles the engineers have been assigned.
- **Admin Tool templates**
  Only the Admin Tool templates (see section Admin Tool Templates for details) are exported.
- **Scripts**
  Only the Admin Tool scripts are exported (see section Admin Tool Scripts for details).
- **Text templates**
  Only the templates that are stored in the Text Template Manager (see section The ConSol CM Text Template Manager for details) are exported.
- **Document templates**
  Only the Microsoft Word or OpenOffice templates are exported, this is only relevant if CM/Doc is in operation (see section CM/Doc for details).
- **Customer model**
  Only the customer fields that are used to define the customer model are exported. No run-time customer data is included.
- **Resource model**
  Only the resource fields that are used to define the resource model are exported. No run-time resource data is included.

- **Classes of Text**
  - If set to “true”, all classes of text which are available in the CM system will be exported.
  - If set to “false”, only the classes of text which have been assigned to a queue which is included in the export will be exported. As a consequence classes of text without queue association will not be exported at all in this case.
- **Queue-related and other data**
  Only queue configuration and general configuration settings are exported (workflows, queues, ticket fields, enum values, MLAs, roles, views, system properties, ...), in short: everything which is not included above. Transfer the names of all queues you would like to include in the export file (scenario) into the list with *Exported Queues*.

- **Workflow Export**
  This panel is activated when you have selected *All* or *Queue-related and other data* in the *Configuration data* panel.
  The selection in the *Workflow Export* panel might influence the size of the exported scenario considerably! You might want to delete some workflows before you export a scenario which contains workflows.
  - **Only latest/installed versions**
    Only the installed versions of the workflows will be exported, i.e. the versions which are currently deployed and active in the system. All older versions will not be exported .
  - **All versions**
    All versions, of all workflows will be exported, currently deployed and older versions. Please be aware that checking this check box might cause the export to take very long and to a huge scenario, depending on the number of workflows which are stored in the system.

If you would like to export the complete configuration, select *All* in the *Configuration data* section. The export/import of subsets (e.g., templates only) is usually applied when selected data (e.g., from a test environment) has to be transferred to another (e.g., live) system.

F.9.3.2 Import

General

> Please note that during the import, the DWH “LIVE” mode must be switched off!

The *general principles* of ConSol CM scenario import are:

1. If the checkbox *Delete existing data* has **not** been **selected**, the scenarios are **merged**, as described here:
   - Data are only **added**, nothing is deleted.

- If the imported scenario contains the **same field/parameter** as the original scenario, the value from the imported one **overwrites** that of the original scenario.
  Example: For the field *priority*, the imported scenario has the annotation `position` = "0;2". The original scenario contains the value `position = 2;2` for the field *priority*. Thus, in the resulting scenario after the import, the value for `position` is "0;2".
- If the imported scenario contains **more parameters** than the original scenario, the parameters are **added** to the original one.
  Example: The imported scenario has the annotation `visibility` = "none" for the field *PersonID*. In the original scenario, the field *PersonID* is present, but does not have the annotation. Thus, in the resulting scenario after the import, the field *PersonID* will have the annotation `visibility` = "none" and will thus be invisible.
- If the imported scenario contains **less data/fewer parameters** than the original one, the original data will be present in the resulting scenario. **Nothing is deleted**.
  Example: If the field *PersonID* in the imported scenario no longer contains the annotation `visibility` = "none", but the original scenario does contain the annotation, it will remain. Thus, in the resulting scenario the field *PersonID* is still invisible.
- For **scripts and templates**, the **latest version** (according to the time stamp) is used, no matter from which scenario.
- **Objects** are identified by an **internal key** (*transfer key*).
  When an imported scenario contains an object with the same name but another transfer key, technically, these are two objects, and the new object will be added from the import to the original scenario (e.g., when a user *Mr. Miller* exists in both scenarios, there will be one user *Mr. Miller* and one user *Mr. Miller (1)* in the resulting scenario after the import.
  To make sure you can transfer another import scenario from the same source (test system), you can delete the original *Mr. Miller* user and transfer the tickets to *Mr. Miller (1)*, an operation that is supported by the ConSol CM Web Client. Then rename *Mr. Miller (1)* to *Mr. Miller*. Now, the *Mr. Miller* user has the transfer key that originated in the import scenario and during the next import there will be no problem.
  The general use case is: The transfer key is created by the ConSol CM system and allows the re-import and/or the update of the configuration data.
- **CM System properties**which start with *custom-* are added (i.e. created) in an installation during import if they did not exist before. In case the custom-property is already present in the importing system, the value is updated, i.e. the imported value is set. This behavior is implemented in CM versions 6.11.0.5 and up. In versions 6.11.0.4 and below, the value was not overwritten.

2. If the checkbox *Delete existing data* has been **checked**, the entire system is deleted, i.e., **all** existing data are **deleted**. All data means:
   - Configuration data
   - Run-time data

> That means if *Delete existing data* has been selected, it is not possible to preserve anything from the original scenario. **Everything is deleted! Only** system properties are **not** deleted.

3. Import of runtime data:
   - Export and import of runtime data is designed for the use with test systems or development systems. For data export and import on a large scale, ETL tools have to be used and **not** the export/import mechanism!
   - If an object already exists (identified by its transfer key) in the target database, the object will by no means be modified by the import, i.e. no fields of the object are added or deleted, and existing fields are not modified. In this way, existing objects (tickets, customers, resources) are protected from being modified accidentally with test data.

The following parameters have to be set for an import operation:

- **Import-Archive:**
  Enter path and file name of the archive from which the data shall be imported. Alternatively, you can click the *Folder* button to open the file browser.
- **Mode:**
  Here you can choose what the import should do if an error occurs:
  - **Abort on error**
    This mode is recommended for production systems.
  - **Skip corrupt data**
    This mode is recommended for imports into test systems. It might even be reasonably applied to production systems, because an unexpected error can lead to a corrupt system, but the import continues even if an error occurs. The problem can be probably handled quickly afterwards, whereas a new import might take longer to perform.
    Example: A referenced object is not found during the import of a view which references a queue which cannot be found.
  - **Force import of corrupt data**
    Choose this mode only if you want to clone a system with corrupt data, e.g., on a development server or if the support team is performing error analysis.

Click on *Import* afterwards to start the data import.

> PLEASE NOTE:
> Runtime data are only imported into the target fields if these fields are empty. If a field contains an entry, this entry will not be overwritten during an import of a scene. Please use ETL (extract - transform - load) operations for export/import of runtime /operating data! This is not a use case for ConSol CM scene export / import.

Configuring System Behavior During Scene Import

The import of the configuration data in a scene is separated into several database transactions. This helps limit memory usage on the database server. However, in case of fatal errors during the import this could result in an inconsistent system state.

A ConSol CM system property can be used to configure the system in a way that all configuration items of a scene import are encapsulated in one large database transaction. Therefore, when enabling this option, a fatal error during import of a configuration item cannot leave the system in an inconsistent state. Rather the whole encapsulating transaction will be rolled back putting the system configuration-wise into the state before the import. The transaction encapsulation does not include localization strings or runtime data which are handled independently of this setting.

Use the ConSol CM system property cmas-core-server, config.import.global.transaction.enabled.

> PLEASE NOTE:
> Enabling this transaction encapsulation may result in longer import execution times and higher memory requirements on the database server!

F.9.3.3 Transformation of User Name and Password Fields During Import into CM 6.11

In ConSol CM version 6.11, the behavior of two customer fields (former data object group fields) has changed:

- The customer field which is annotated with `username`.
  The user name is now controlled to be unique.
- The customer field which is annotated with `password`.
  The passwords are only stored as encrypted strings.

Thus, when you import a scenario from a ConSol CM system with a version lower than 6.11 to a system running version 6.11 (or higher), two transformations will be performed automatically. The checkbox *Delete existing data* for the import operation does not have any influence on this behavior.

- **username:**
  The customer field which is annotated as `username`will be used. If several identical user names are found in the customer data, the later ones will be appended by a number like *Huber_1* for the second original username *Huber* and so on. The dialog after the import will inform about these user name changes.
  Please note that user names can be defined as case sensitive in CM versions 6.11 and up. Use the CM system property cmas-core-security, policy.track.username.case.sensitive.
- **password:**
  The customer field which is annotated as `password` will be used. The passwords will be stored as encrypted strings using the new mechanism.

F.9.3.4 Workflow Deployment (for Deployment Error Recovery Only)

Usually, all operations concerning workflow design and deployment are performed using the ***Process Designer***. However, if an error occurs during workflow deployment, you can transfer the tickets that could not be transferred into the new workflow using the following options.

First select the queue(s), then choose the transfer mode:

- **Remain at last activity**
  The ticket will try to stay at its position in the process:
  - If the activity and scope have not been changed, i.e., no change in position for the ticket.
  - If the activity is no longer present, i.e., the ticket goes as far back in the process as necessary to find the last consistent position in the process.
- **Restart process**
  The ticket goes back to the *START* node of the process/workflow.

Please also read the detailed explanation of the workflow deployment process in the *ConSol CM Process Designer Manual*.

F.10 License Management

This chapter discusses the following:

F.10.1 General Information about Licenses in ConSol CM

A ConSol CM license file is a text file which contains entries for several modules. For each module, the number of valid licenses is indicated. For example, the following excerpt of a license file shows the ConSol CM Web Client, CONCURRENT_USERS section. Ten licenses have been purchased.

```
[CONCURRENT_USERS]
contractParty = Demo-Licence ConSol
products = WEB_CLIENT,REST
version = 6.10
expirationDate = 31.12.2016
licenses = 10
signature = XXX
```

ConSol CM works with **concurrent users** (sometimes also called floating licenses), i.e., the number of users who are logged in simultaneously is registered, no user names are checked. That means the number of engineers who are managed in the Admin Tool (see section Engineer Administration) does not have to be identical to the number of Web Client licenses.

A license is consumed when the user logs in. The license is handed back to the server when the user session is terminated, i.e., when the user logs out or when the user session is terminated automatically by the server because the session timeout has been reached (see system property cmas-core-server, server.session.timeout).

F.10.2 Sections of a License File

A ConSol CM license file can contain the following sections. All licenses are **concurrent** licenses, see explanation above.

- [ADMINTOOL_USERS]
  The number of users who can log in to the CM Admin Tool.
- [CONCURRENT_USERS]
  The number of CM engineers who can log in to the Web Client
- [PROCESS_DESIGNER]
  The number of users who can log in to the CM Process Designer
- [TRACK]
  The number of customers who can log in to the portal CM/Track
- [TRACK_USERS]
  The number of user profiles for the portal CM/Track. This is the number of engineers who are marked as *Track* in the Engineer Administration.
- [REST_USERS]
  The number of users who can access the REST API. The number of TRACK_USERS is not included in this number (CM/Track also uses the REST API.)
  CM/Phone will also consume REST licenses, one license per client (PC/laptop) where CM/Phone is installed and active.

Since several (or even a great number of) customers can use the same user profile in CM/Track, the license numbers of [TRACK] and [TRACK_USERS] might differ considerably. For example, there might be two user profiles and 1000 customers should be allowed to log in to CM/Track at the same time. This would mean 2 [TRACK_USERS] licenses and 1000 [TRACK] licenses. Please read section CM/Track V1: System Access for CM/Track Users (Customers) or CM/Track V2: System Access for CM/Track Users (Customers) for a detailed explanation.

F.10.3 Managing the ConSol CM License Using the Admin Tool

You have to import a valid license for your ConSol CM system in the navigation group *System*, navigation item *License*. You will receive a license for a test and/or a productive system when you have signed the respective software contracts with ConSol. If the license has expired or will expire soon, you can import a new license file. Of course the Admin Tool will always start, even if the license has expired. In the Web Client, in CM/Track, and in the Process Designer, the login is not possible when the license has expired.

Please ask your consultant for details. The license is a plain text file. The license can be modified during ConSol CM operation, no system downtime is required.

There is no *Back* button to undo changes with one click when you enter or delete text in the *License* field. If you accidentally change parts of the license, close the Admin Tool **without** clicking *Save*. This will discard all changes you made to the license text. When you restart the Admin Tool afterwards, the license will be in the same condition as it was before you made the changes.

Figure 502: *ConSol CM Admin Tool - System, License*

Choose one of these two ways to import your ConSol CM license file:

- Load the license using the file browser next to the field *License file*. Click on *Save* (1).
- Insert the entire text of the license file via copy and paste. In case an old license is present, just replace the entire text. Click on *Save* (2).

You should receive a message that the license has been imported into the system successfully. It goes into effect immediately, without further action.

F.10.4 Expert Information about Accessing Content of CM Licenses

The content of CM licenses can also be queried using the MBean `licenseDeployer`. This MBean offers three methods:

- getRemainingDays
- deployLicence
- getLicenseInfo

If you are interested in using this feature and would like to have support implementing a system which uses the MBean access, e.g., to set up monitoring for your CM system, please contact your CM consultant.

The following figure shows an example using the JConsole.

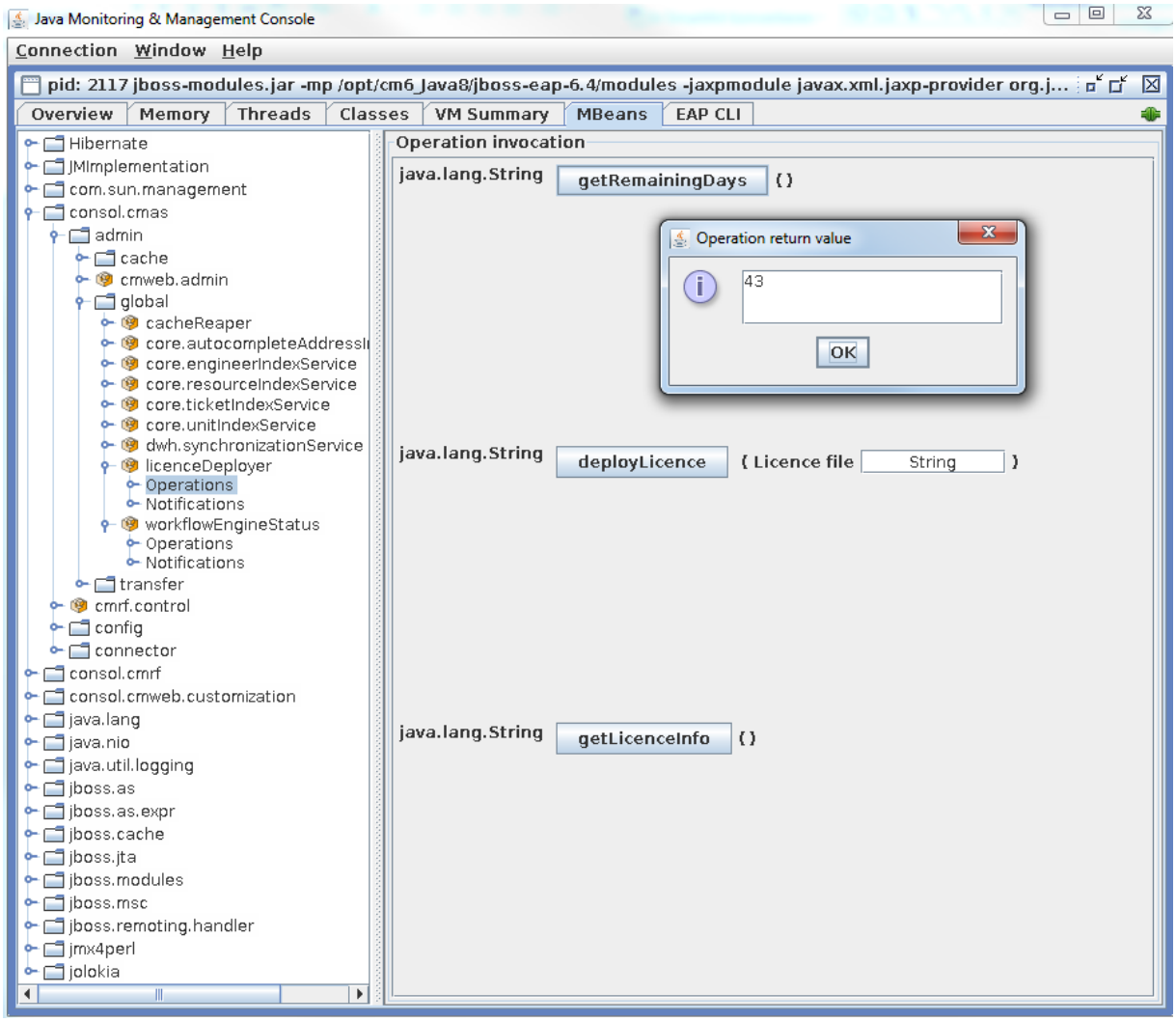


Figure 503: *Retrieving the remaining days of a CM license using the JConsole*

In case you have purchased a CM license with a limited period of validity, we recommend to set up a monitoring for the license which sends a notification a certain time before the license expires.

F.11 System Properties

This chapter discusses the following:

F.11.1 Introduction

In the navigation item *System Properties* in the navigation group *System*, you can add, modify, or delete system settings, so called *system properties*, for the ConSol CM application. System properties are used to store, for example, values for the number of seconds for a session timeout, for admin email addresses, or for the configuration of the search page size.

> ⚠ **DO NOT WORK ON THIS NAVIGATION ITEM UNLESS YOU KNOW EXACTLY WHAT YOU ARE DOING !!!**
>
> On this page, you have access to basic system settings called *system properties*.
>
> Do not modify, edit, or delete any values of system properties unless you know exactly what the impact will be.

F.11.2 System Property Overview

A system property has the following parameters:

- **Module**
  Mandatory. This indicates in which ConSol CM module the system property will be used.
- **Property** (name)
  Mandatory. The name of the system property. The system property is referenced by this name throughout the system.
- **Type**
  Mandatory. Data type of the system property, i.e., of the value.
  - **String**
    A regular string field
  - **Password**
    A field which will contain a password and will therefore not be displayed in plain text.
  - **Email**
    A field which contains an email address, i.e., it has to be formatted according to the standard email format (<address>@<domain>).
  - **Boolean**
    A boolean (true/false) field.
  - **Integer**
    A whole number field (no fractional part). Used, for example, for time intervals or number of restarts.
- **Value field**
  The value of the system property. Must be set according to the given data type.
- **Description**
  Optional. A text description for the system property.
  In case you add company-specific system properties which contain integers for time intervals, we strongly recommend that you use the description to explain the unit of measurement, since it makes a big difference whether an escalation goes off within five minutes, five hours, or five days!
- **Restart required**
  Boolean field (checkbox) to indicate whether a change of the system property will only become active after a system restart.
  - **Optional**
    Boolean field (checkbox) to indicate if the system property has to be present or is optional. The following behavior will be configured:
    - optional = “true”:
      - The value of the system property can be NULL.
      - The system property can be deleted, e.g., using the Admin Tool.

  - optional = “false”:
    - The value of the system property cannot be NULL.
    - The system property cannot be deleted, i.e., there must be a value in the *Value* field.

- **REST accessible**
  Boolean. Starting with CM version 6.11, the value of a system property can be retrieved via REST API if this checkbox is set to “true”. For detailed information about the REST API and the required commands, please refer to the *ConSol CM REST API Documentation*.

  > ⚠ Please be very careful with the decision to set this value to “true”! Using REST, a regular system account is sufficient for retrieving data!

The system properties are managed in a table where three of the system property parameters are displayed as columns:

- **Module**
  Module name
- **Property**
  The name of the system property.
- **Value**
  The value of the system property.

You can sort the table according to a column by clicking on the column header. Another click reverses the order.

The following example shows the pop-up menu which is used to set a system property, here `cmas-core-server, server.session.timeout`. You reach this screen by opening the navigation group *System*, the navigation item *System Properties* and by selecting the property in edit mode.

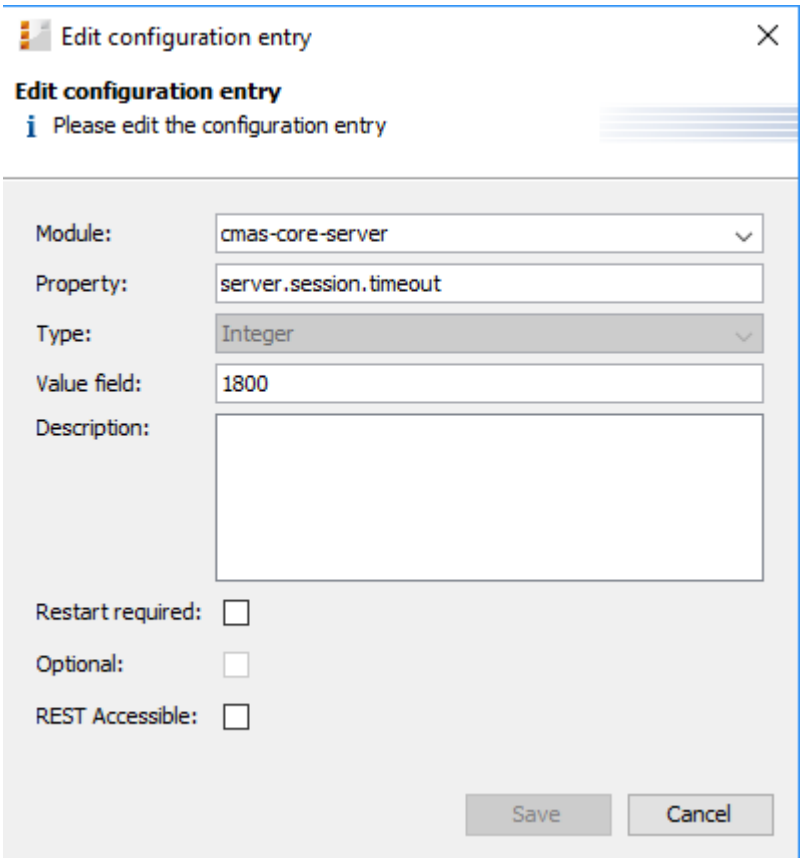


Figure 504: *ConSol CM Admin Tool - System, System Properties: Editing one system property*

F.11.3 Setting System Properties

The values for system properties are set in different ways. System properties can be ...

- filled automatically with default values by the ConSol CM system, e.g., `cmweb-server-adapter, ticketListRefreshIntervalInSeconds` is set to "180" seconds. You can modify the values if necessary.
- prepared by the system, i.e., the system properties are in the list but they are not filled with the required values. For example, LDAP-related system properties will only be filled (manually by an administrator) when you configure the LDAP authentication.
- filled by Admin Tool configurations which you perform by using the graphical user interface of the Admin Tool. For example, when you enter mailbox names in the *Email* configuration, the values will be entered into the respective NIMH system properties, e.g., `cmas-nimh, mailbox.0.connection.host`. In those cases, you should always use the graphical configuration and not edit the system property directly!
- not set at all in a default installation. Then they have to be added manually, e.g., the activation of the TEF (Task Execution Framework) by adding the system property `cmas-app-admin-tool, start.groovy.task.enabled` and setting its value to "true".
- added to a system manually as customer-specific system properties. If the module name starts with *custom-*, those system properties are exported in a scenario (see section Deployment (Import/Export)). For example, you can define and add your own system properties to manage escalation times. In this way, you can store the values here in the system properties administration and an administrator can change the values without requiring any programming knowledge. In the script code where the system properties are referenced, only the system property name is used. This is explained in the following section.

F.11.4 Programming with System Properties

To use a system property in a Groovy script, i.e., to retrieve the system property's value, use the following class and method:

- `configurationService.getValue(String pModule, String pProperty).`

For example, to retrieve a specific escalation time, use:

```
def mytime = configurationService.getValue("custom-
 servicedesk","escalation.time.medium2")
```

This will retrieve the value “10”.

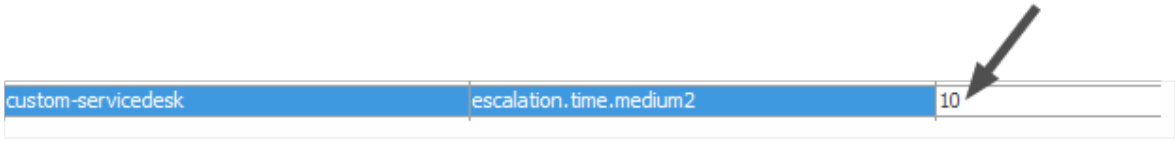


Figure 505: *ConSol CM Admin Tool - System, System Properties: Customer-specific system property*

F.12 Working with Text Templates

Text templates are pre-defined texts which an engineer can open and either use as-is or modify. Text templates may be used for emails or ticket comments anywhere text, headers, and footers can be specified. Another example is documents which have to be edited using Microsoft Word or OpenOffice.

In both cases, the templates offer not only text, but certain data fields can also be pre-filled with data from the ticket, e.g., customer name or ticket subject.

ConSol CM includes two modules which provide text templates:

- The Text Template Manager for editing and managing email and ticket comment templates (see section The ConSol CM Text Template Manager)

and

- CM/Doc with the Document Template Manager for editing and managing Microsoft Word and OpenOffice templates (see section CM/Doc).

F.12.1 The ConSol CM Text Template Manager

F.12.1.1 Introduction to Working with Email and Ticket Text Templates

Using the Text Template Manager, two types of templates can be defined:

- **Email templates** for emails written from the ConSol CM system
  - manual emails (written by an engineer using the Ticket Email Editor)
  - automatic emails initialized by the system (e.g., sent by a workflow script when a certain workflow activity is executed)
- **Text templates** for ticket texts (comments)
  - during ticket creation
  - during ticket editing

Email Templates

Why Email Templates?

When a system works with emails, several criteria have to be considered. If all those requirements are met, email templates are a very helpful tool in everyday working life.

- The emails have to have a strictly defined layout, usually following the company's CD (corporate design).
- The texts have to follow the company's letter/text guidelines.
- Texts that are used very frequently have to be provided by templates in order to save time and to avoid typos and other errors while typing the text.
- Customer-, system-, and engineer-specific data have to be integrated into the text.
- The template management should be performed by an administrator and/or power user. No system configuration by the company should be required.

ConSol CM provides a function set which takes all those criteria into consideration.

Emails in ConSol CM

Emails are used for core features in ConSol CM. Those features are described in detail in section Email Functionalities in ConSol CM, so here, only a brief review is given.

ConSol CM can receive and send emails. Sending emails can serve various purposes:

- **An engineer writes an email directly from the ticket, using the Ticket Email Editor.**
  This can be an email to the customer, to a co-worker, or to any other person with a valid email address. Often, there are standard texts which are used every day for several recipients. To avoid typing the same text over and over again, ConSol CM offers email templates. These are text templates where parameters like customer name, ticket number, or engineer name and phone number can be integrated. When the template is used, the system fills in the parameters automatically with the valid data from the current ticket. The engineer can add more text or modify the text as required, so email templates are not static, but dynamic.
  Emails which are sent manually either do not use a template or are based on a template from

the Text Template Manager. Templates from the Script and Template Administration of the Admin Tool are not available here.

- **The system sends an email automatically.**
  This can be an internal email like a reminder for an engineer when the ticket has entered the escalation status or an internal email to a supervisor when a ticket needs approval before continuing on through the process. Or it can be an external email to the customer, like a confirmation of receipt or a notice that a ticket has been solved. The email is generated automatically based on the respective email template.This can be an email template from the Text Template Manager or from the Script and Template Administration of the Admin Tool.

Ticket Text Templates

Why Ticket Text Templates?

Using ticket text templates, i.e., predefined text segments you can access when you create or edit a ticket, serves several purposes:

- You, as a ticket engineer, save a lot of time by not typing the same text over and over again.
- You do not risk forgetting important points (e.g., in questions for a pre-qualification when you talk to your customer on the phone).
- You do not have to worry about typos.
- You do not have to look up ticket and/or customer data, because all data is integrated into the text automatically.

Ticket Text Templates in ConSol CM

Ticket text templates are defined very similar to email templates. Only the *Used within* parameter is set in a different way when the template is created using the Text Template Manager.

Technically, there is no difference between email and ticket text templates! So for each template you, working with the Text Template Manager, can decide if the template should be used as ticket text template, as email template, or both.

Email and Ticket Text Templates in ConSol CM

Email and Ticket Text Template Components

In email and ticket text templates in ConSol CM you can use free-form text and all data that is available for a customer, an engineer, and/or the ticket. All available components are explained in section The Library of Markers.

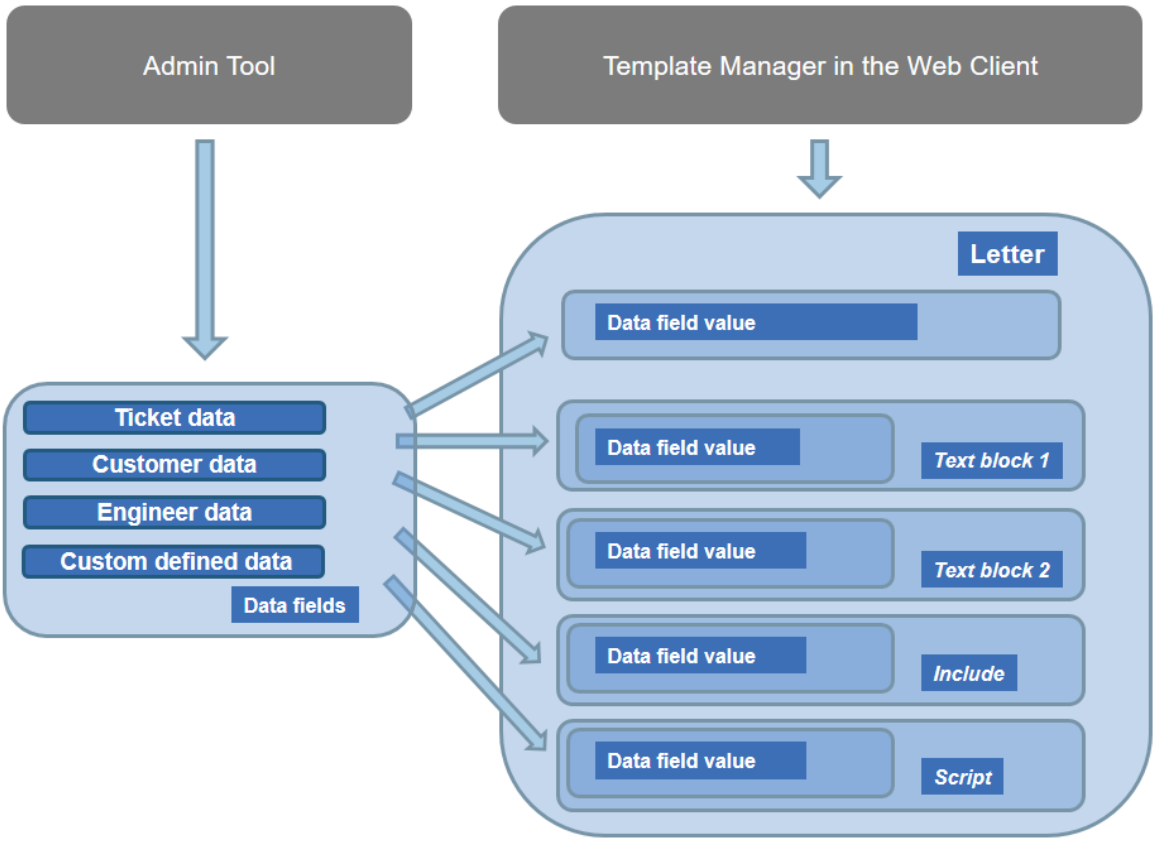


Figure 506: *ConSol CM - Availabe components / data for email templates*

Please refer to the *ConSol CM User Manual* section *Creating a New Ticket*, *Updating Tickets*, and *Sending Emails* for a detailed description how to use the ticket editing functionalities and the Ticket Email Editor.

Storage and Management of Email and Ticket Text Templates

Email Templates

Email templates are stored and managed at two different locations in ConSol CM:

1. In the Text Template Manager
2. In the Script and Template Administration of the Admin Tool (this is not covered here, but in the respective section of this manual; see Admin Tool Templates)

Ticket Text Templates

Ticket text templates are stored and managed at two locations in ConSol CM:

1. In the Text Template Manager (here, ticket text templates are managed)
2. In ***CM/Doc*** (here, Microsoft Word and OpenOffice documents can be stored for use with CM/Doc, see section CM/Doc)

F.12.1.2 Introduction to the Text Template Manager

The ConSol CM Text Template Manager is a Web Client-based tool for the creation and management of email and ticket text templates. See section Working with the Text Template Manager.

The following figure shows the Text Template Manager. You reach this screen by clicking the *Text templates* link in the Web Client.

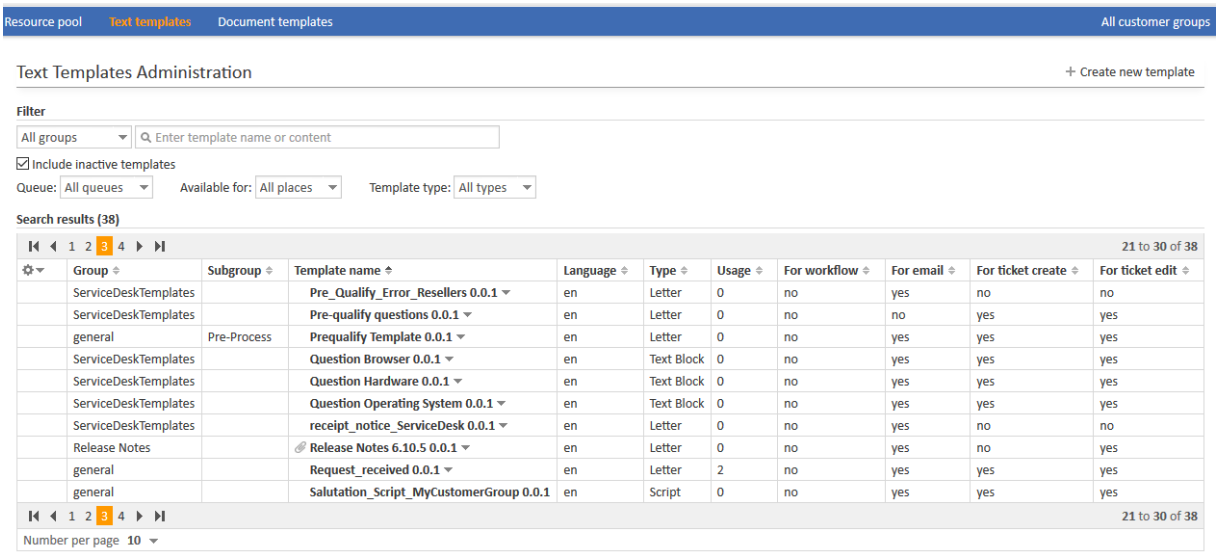


Figure 507: *ConSol CM Web Client - Text Template Manager*

Every user who has been assigned a role with the permission Write template can access the item *Text templates* in the main menu (which opens the Text Template Manager).

The following figure shows the role management screen. You reach this screen by opening the navigation item *Roles* in the navigation group *Access and Roles* and selecting a role.

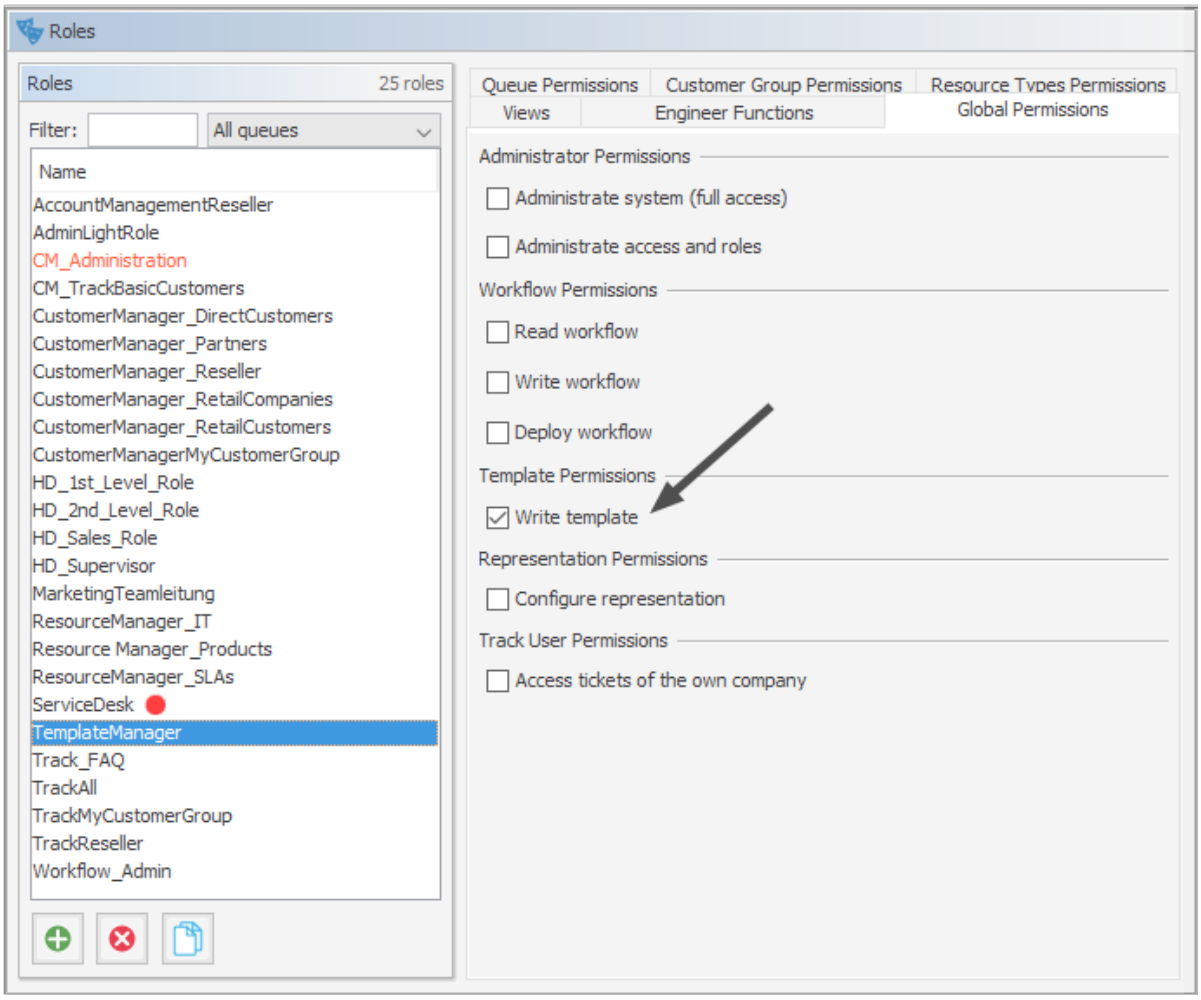


Figure 508: *ConSol CM Admin Tool - Access and Roles, Roles: Permissions for role TemplateManager*

> We recommend to create a role (e.g., TemplateManager) that has only the permission Write template, with no queue permissions or other permissions. Every user who should have access to the Text Template Manager can be given this role. In this way, regular permissions (e.g., queue and customer group permissions) are not merged with Text Template Manager permissions and you can grant and remove the Text Template Manager permission in a very flexible way.

When the permission has been granted, the user has access to the main menu item *Text templates*.

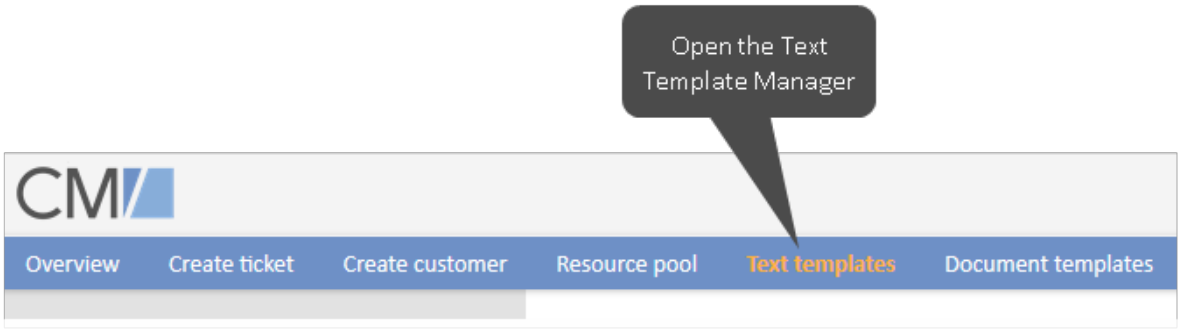


Figure 509: *ConSol CM Web Client - Main menu with Text Template Manager access*

F.12.1.3 Working with the Text Template Manager

The Text Templates Administration Panel

When you open the Text Template Manager, the *Text Templates Administration* panel is displayed:

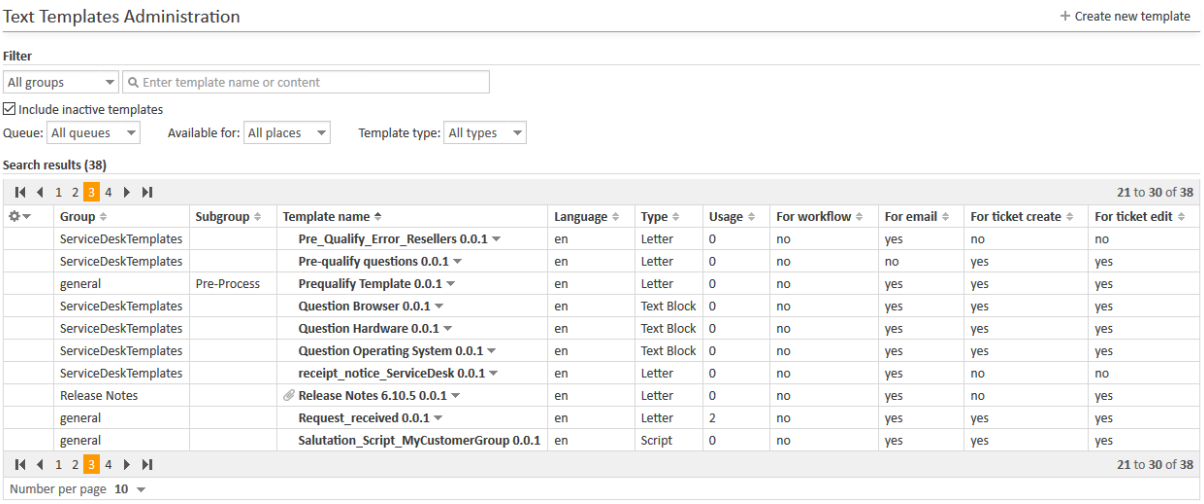


Figure 510: *ConSol CM Web Client - Text Template Manager*

A list of all existing templates is shown. If a template uses data fields (ticket fields or customer fields) which are no longer available, that line of the respective template is displayed in red. Templates which have an attachment are marked by the attachment (paper clip) icon. Inactive templates are displayed in gray.

Filter

Initially, all templates are displayed in the list of *Search results*. You can filter the displayed list entries by using the filters in the upper part of the page:

- **Groups/Subgroups**
  Select one group or subgroup, multiple selection is not possible in this drop-down menu. Only the templates belonging to this group/subgroup will be displayed in the list of templates

(*Search results*).

- **Text filter**
  Enter a template name or part of a template name or enter part of the template content. Only the templates matching the given criteria will be displayed in the list of *Search results*.
- **Include inactive templates**
  Mark this checkbox to have all templates displayed, active and inactive. If the checkbox is not selected, only active templates are displayed.
- **Queue**
  Select one queue from the list, multiple selection is not possible here. Only templates which are available for the selected queue will be displayed in the list of *Search results*.
- **Available for**
  Select one value from the list, multiple selection is not possible here. Only the templates matching the given criterion will be displayed in the list of *Search results*. Possible values are:
  - **All places** (default)
    All templates are displayed.
  - **Email**
    Only templates which are available for emails will be displayed in the list of *Search results*.
  - **Ticket create**
    Only templates which are available as text templates during ticket creation will be displayed in the list of *Search results*.
  - **Ticket edit**
    Only templates which are available as text templates during ticket editing will be displayed in the list of *Search results*.
  - **Workflow**
    Only templates which have been marked as workflow templates will be displayed in the list of *Search results*.

  Please see the following sections for details about which implications it will have to define the availability of a template.
  The availability of each template is also indicated in the respective column of the *Search results* list (*For email*, *For ticket create*, *For ticket edit*, *For workflow*), see explanation below.
- **Template type**
  Select one value from the list, multiple selection is not possible here. Only the templates matching the given criterion will be displayed in the list of *Search results*. Possible values are
  - **All** (default)
  - **Letter**
  - **Include**
  - **Text Block**
  - **Script**

  All types are explained in detail in the following sections.

List of Templates / Search Results

The list contains the following columns. It can be sorted according to a column by clicking on the column header. Another click will reverse the display order.

- **Group**
  Mandatory. The group of a template does not have any technical or functional implications within the Text Template Manager. It is used to order the list in a certain way, in the Text Template Manager as well as in the Web Client. Thus, when a great number of templates is defined in your CM system, it makes sense to use the Groups feature in order to facilitate the engineers' work with the templates using the Ticket Email or Comment Editor.
- **Subgroup**
  Optional. The subgroup of a template does not have any technical or functional implications within the Text Template Manager. It is used to order the list in a certain way, in the Text Template Manager as well as in the Web Client. Thus, when a great number of templates is defined in your CM system, it makes sense to use the Subgroups feature (in addition to the Groups feature) in order to facilitate the engineers' work with the templates using the Ticket Email or Comment Editor.
- **Template name**
  The template name. This is also used in workflows to indicate the required template and is displayed in the Web Client (Ticket Email Editor or Ticket Comment Editor) in the template selection.
- **Language**
  The language that has been selected during creation of the template (can be modified). The web browser of an engineer will display the template according to the browser locale. So when you need a template in different languages, make sure to set this value correctly.
- **Type**
  There are five different types of templates which will be explained in detail in the subsequent sections:
  - **Letter**
    This is the basic form of a template. Letter templates are offered in the Ticket Email Editor or Ticket Comment Editor and can be used as workflow email templates. All other template types are only sub-components of a letter.
  - **Include**
    This is a sub-component of a letter which can be used in letters. In this way, you can use the same text in several templates. A typical example is the signature of a company which is used in all other templates. The signature should be defined as include and then be integrated in all other (letter) templates where the signature is required. Thus, the template administrator has to maintain the signature in only one location and can be sure that it is used in every other template correctly.
  - **Workflow Include**
    This is the same as an include but used only for workflows.
  - **Text Block**
    This is also a sub-component of a letter. It can be toggled on or off when writing emails, i.e., the text will be displayed or not. A good example is the first analysis in a help desk

team where the same questions are sent to every customer. One text block can contain hardware questions, one software questions. Depending on the intent of the email, the engineer uses either one.

- **Script**
  This template type is only available for administrators (i.e., a user who logs in to the Web Client using an administrator account). Here, "intelligent" templates can be constructed, like a template that sets *Dear Sir* for a male and *Dear Madam* for a female customer, depending on the value of the field *salutation*.

- **Usage**
  Indicates how often the template is used.
- **For workflow**
  Boolean. A template can be marked as workflow template. Then it is not available in the Ticket Email Editor or Ticket Comment Editor but can only be used by the workflow for automatic emails.
- **For email**
  Boolean. All templates which have been marked as *Available for Email* will be marked *yes*.
- **For ticket create**
  Boolean. All templates which have been marked as *Available for Ticket create* will be marked *yes*.
- **For ticket edit**
  Boolean. All templates which have been marked as *Available for Ticket edit* will be marked *yes*.

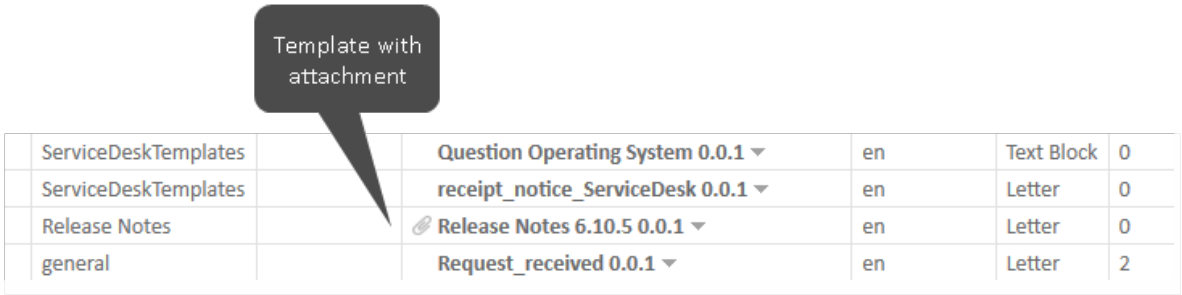


Figure 511: *ConSol CM Web Client - Text Template Manager: Template with attachment*

For every template you can select an operation by using the context menu:

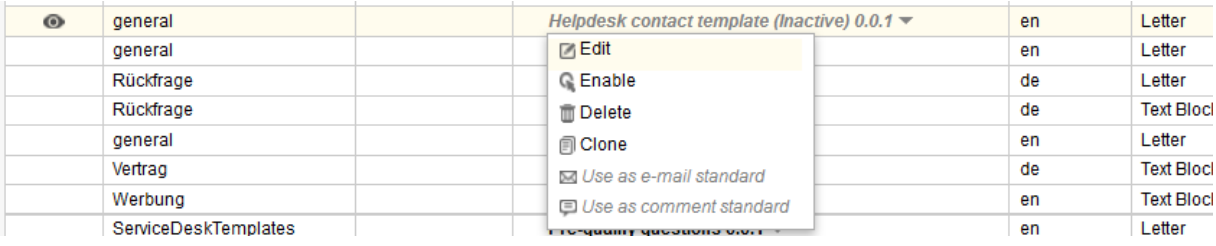


Figure 512: *ConSol CM Web Client - Text Template Manager: Context menu of a template*

- **Edit**
  Edit the template. The same functionality as described for creating a new template is available.
- **Disable**
  (or **Enable** for disabled templates)
  Only enabled (= active) templates are active and available in the system.
- **Delete**
  Delete the template. This is not possible when the template is used by a workflow or when an include or text block is used in other templates (letters).
- **Clone**
  Create a copy of the template. A new name is required in this case.
- **Use as email standard**
  (or **Unset standard** for the current standard template)
  Only one template can be marked as the standard email template. This will be automatically inserted into any email that is opened in the Ticket Email Editor. It can then be removed by the engineer or used in the email. Usually a signature or footer is defined as standard template.
- **Use as comment standard**
  (or **Unset standard** for the current standard template)
  Only one template can be marked as the comment standard template. This will be automatically inserted into a comment that is opened in the Ticket Comment Editor. It can then be removed by the engineer or used for the comment.

A standard template must not contain text blocks or variables.

The template is displayed with a red font if it contains references to fields which are not known to the system. Please edit the template to make sure that all markers refer to available fields. You can find more information about markers in the section The Library of Markers.

Preview Functionality within the List of Search Results

In order to get a quick overview of the templates, e.g., when you search the list for a certain topic and you scroll through the list, you can use the preview functionality. Click on the eye icon in the first column of a line to open the preview for the respective template.

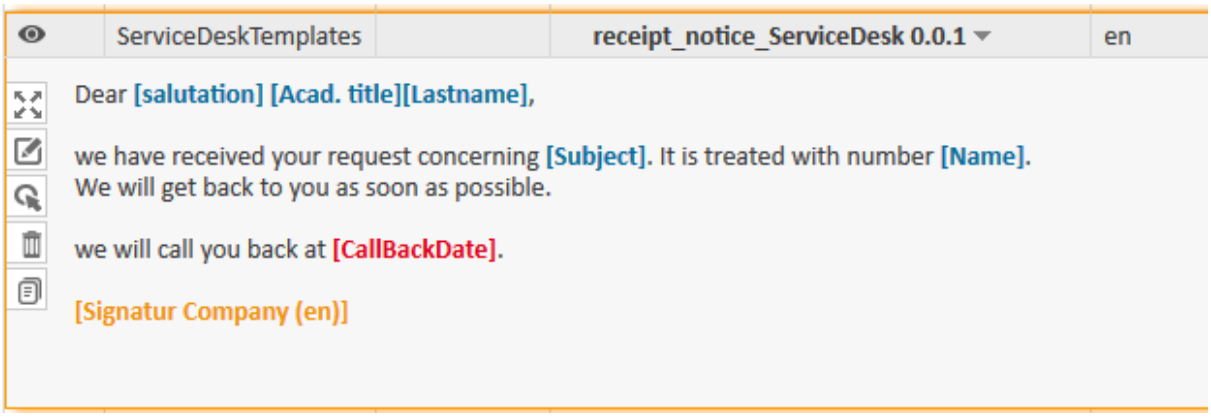


Figure 513: *ConSol CM Web Client - Text Template Manager: Preview functionality*

The following icons/functionalities are available here (the mouse-over will help you identify the icons):

- **Expand view**
  Displays the entire template.
- **Edit**
  Open the edit mode. For details, please refer to the explanations in the following sections.
- **Disable**
  (or **Enable** for disabled templates)
  Offers a quick way to disable/enable a template.
- **Delete**
  Delete the template. This is not possible when the template is used by a workflow or when an include or text block is used in other templates (letters).
- **Clone**
  Create a copy of the template. A new name is required in this case.
- **Attachment**
  In case the template contains one or more pdf attachments (not shown in the figure above), the two attachment buttons allow you to scroll through these attachments. A preview of each attachment document is displayed.

Create a New Template

Here, an example for an email template is shown. The same principle can be applied to ticket text templates.

Create a New Letter

To create a new template, click on the *Create new template* link in the Text Template Manager. On the *New Template* page, you can enter all parameters for the new template. In our first example, a letter is created which serves as confirmation of receipt for the customer. It can be automatically sent from the workflow or be used in the Web Client.

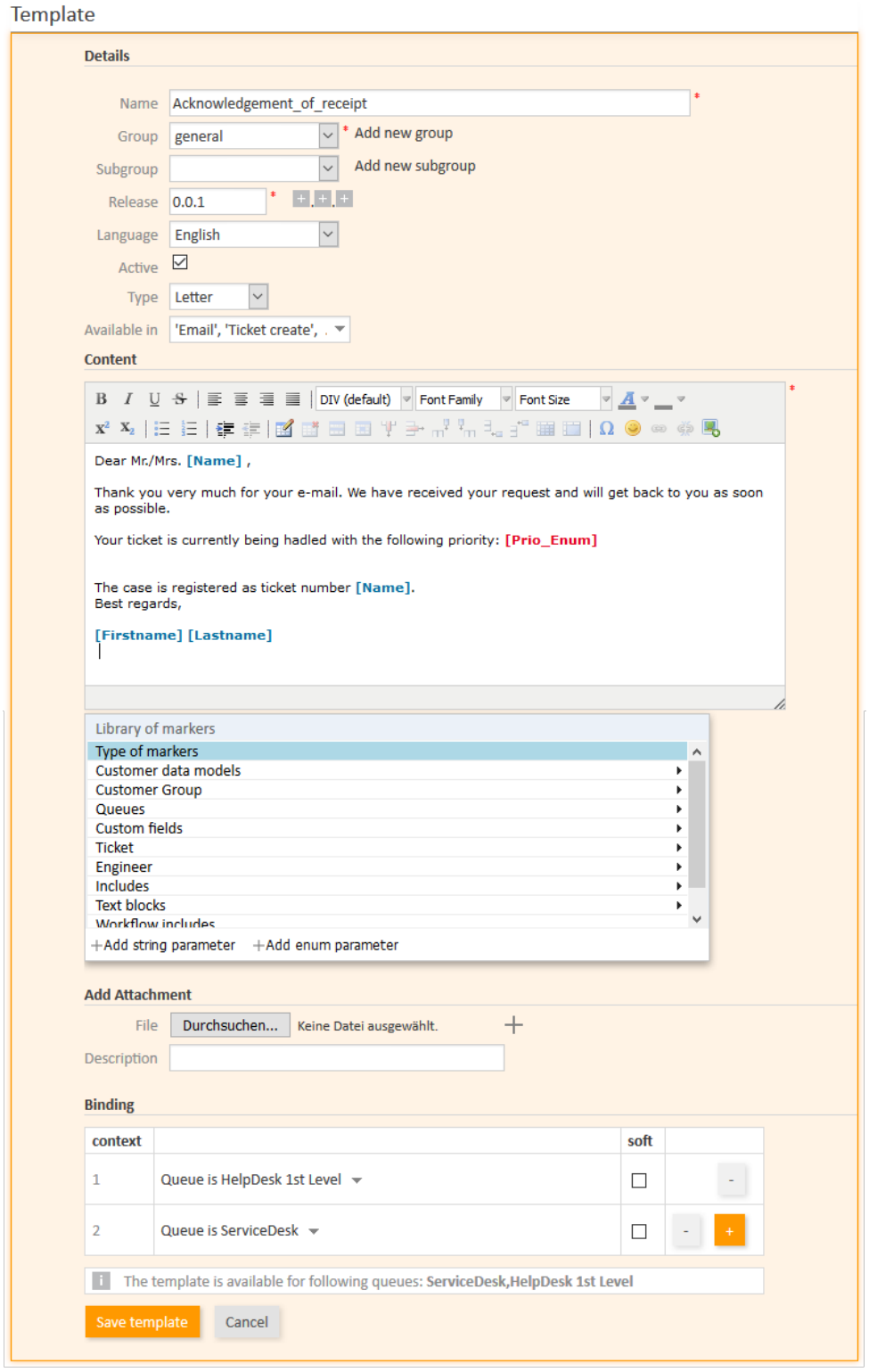


Figure 514: *ConSol CM Web Client - Create a new template*

- **Name**
  The name of the template. This name will be displayed in the Ticket Email and Comment Editor in the Web Client, and it will be used as technical reference when a template is used within a script (Admin Tool or workflow).
- **Group**
  The group (see previous section). You can either use an existing group or create a new one. The group is used to order the templates within the list of *Search results* in the Text Template Manager and the group is displayed in the Ticket Email and Comment Editor in the Web Client to facilitate work for the engineers.
- **Subgroup**
  The subgroup (see previous section). You can either use an existing subgroup or create a new one. The subgroup is used to order the templates within the list of *Search results* in the Text Template Manager and the subgroup is displayed in the Ticket Email and Comment Editor in the Web Client to facilitate work for the engineers.
- **Release**
  If you want to set up a versioning system for the email templates, you can set the release, i.e., version, here. You can type the numbers (three digits) or you can use the "+" buttons as help.
- **Language**
  Choose the language of the template. This can be important if you work in an international team. ConSol CM can be used in as many languages as required, and this can be configured using the Admin Tool and in the Process Designer. To make sure the emails are sent in the correct language, the corresponding locale has to be set here.
- **Active**
  Select if the template should be active (= enabled) or inactive (disabled). This can be changed later, so you can design a template and work on it and set it active when you are finished.
- **Type**
  Select the type (letter, include, text block, script) of the template. See previous section for explanation.
- **Available in**
  - **Workflow**
    Select if the template should be available in workflows (i.e., not available in the Ticket Email Editor or in the Ticket Comment Editor).
  - **Email**
    Select if the template should be available in emails.
  - **Ticket create**
    Select if the template should be available during ticket creation.
  - **Ticket edit**
    Select if the template should be available when the ticket is edited.
- **Content**
  Here you define the content of the template/letter. You can combine any free text and

components of the Library of markers (below the content section, see section The Library of Markers for details). Write the text and select the desired element from the library by double-clicking on it.
All functionalities of the Rich Text Editor are available for the design of the template content. Please note that the availability of additional formatting features is influenced by the Page Customization attribute *cmRichTextEditor*! See section cmRichTextEditor of the Page Customization chapter for a detailed explanation of the features.

> Please note that <script> tags will be escaped automatically by the system. In this way, it is not possible to insert executable Java Script code into a text.

- **Add Attachment**
  Here you can add one or more attachment(s) to a template. Use the file browser to select a file from the file system of your computer. All common file types are possible. The system behavior concerning an attachment depends on the availability which is defined.
  - **Attachment in email templates**
    An attachment defined in a template which is available in emails will be attached to the tickets like a regular ticket attachment. Additionally, it will be pre-selected as email attachment when the engineer writes an email using the Ticket Email Editor and selects the respective email template.

    > Attachments in email templates are ignored when the template is used in the workflow, i.e. the attachments are only added to emails which are sent using the Ticket Email Editor.

  - **Attachment in ticket comments (ticket edit, ticket create)**
    An attachment defined in a template which is available for ticket edit and/or ticket create will be attached to the ticket like a regular ticket attachment. Subsequently, it can also be used as email attachment, but it will not be pre-selected in the Ticket Email Editor.
- **Binding/Context**
  Here you can define a certain context for the template. In case nothing is defined here, the template will be available in all queues and the availability will not depend on any parameters. In case limiting queues and/or ticket fields are defined, the availability of the template will be restricted respectively. You can select
  - Ticket fields for queues
  - Queues

  where the template should be available, see section Binding Templates to Queues or to Specific Parameters for details.

Please note that the availability of templates in the Web Client also depends on the customer group of the ticket! If parameters are used which reference customer fields from a certain customer data model, the template will only be available if the ticket is assigned to a customer from a customer group which uses this model!

In the Web Client, i.e., in the Ticket Email Editor, the template *Acknowledgement_of_receipt* would have the following layout:

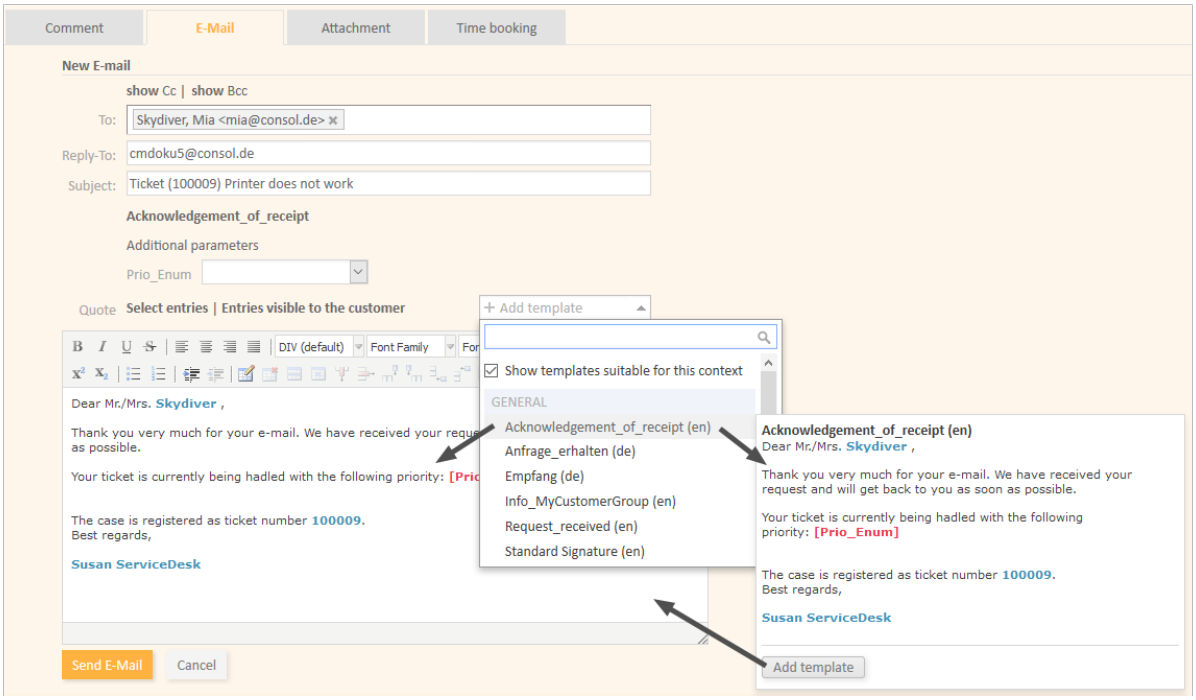


Figure 515: *ConSol CM Web Client - Email template in Ticket Email Editor*

New Template Created from an Outbound Email

You can also open the Text Template Manager via the context menu of an outbound email in the history section of a ticket. The Text Template Manager will be opened in the mode Create new template with two settings preselected:

- Type: Letter
- Available in: Email

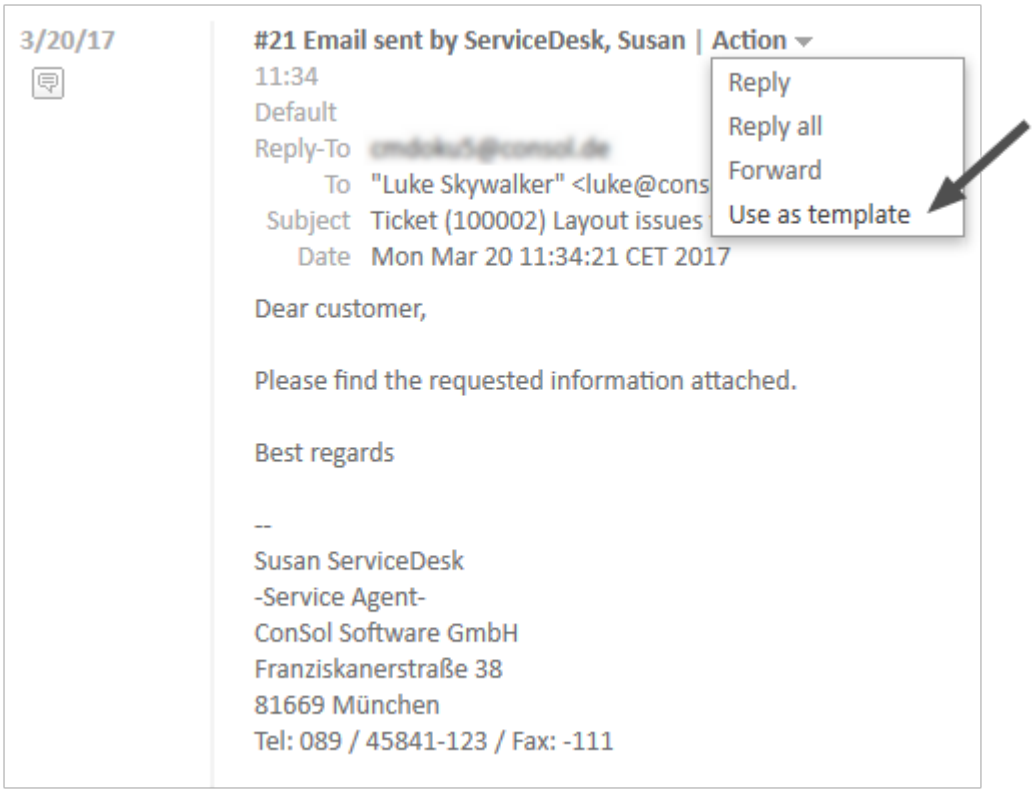


Figure 516: *ConSol CM Web Client - Creating a new email template from an outbound email*

The Library of Markers

The Library of markers provides a collection of all data fields that are available in the system. These are:

- **Default fields**
  Like *queue* or *engineer* with all corresponding data like *queue name* or *engineer firstname* or *engineer lastname*.
- **Ticket fields and/or customer fields**
  That have been designed specifically for the system like, e.g., *customer service number*. Please note that the localized values of the ticket fields and customer fields are displayed in the Library of markers!
- **Components of the Text Template Manager**
  That are used in other components, e.g., includes or workflow includes.
- **Scripts**
  That have been defined by an administrator and can help provide content in a dynamic way.

The following table provides examples for fields that could be found in a CM system. The names displayed in the Library of markers are the localized names of the ticket fields resp. of the customer fields. If no localization is provided, the (technical) field name is displayed. If you would like to read the

information about ticket fields, please refer to chapter Ticket Field Administration (Setting Up the Ticket Data Model). For customer fields (i.e., customer data), see section Setting Up the Customer Data Model.

Field Group or Main Component	Ticket Field Resp. Customer Field (Example)	Explanation
Customer data models		<entry point for all customer-specific fields>
Customer data models	Customer groups	<entry point for all customer-specific fields of the selected customer group>
<Contact or company>	Salutation	
	Academic title	
	Forename	
	Lastname	
	Phone	
	Email	
	<more fields depending on FlexCDM definition>	
Customer Group	Name of the customer group	
Queue	Name	The name of the queue where the ticket is being processed at the moment
Ticket fields for queue	All ticket fields of ticket field groups that have been assigned to the queue	
Ticket	ID	The internal ticket ID, not displayed in the Web Client
	Name	The ticket name, the ID in the Web Client
	Subject	

Field Group or Main Component	Ticket Field Resp. Customer Field (Example)	Explanation
	Engineer	The current engineer who is assigned to the ticket. Can be “NULL” (empty) if no engineer is set.
	Creation Date	Opening date of the ticket
Engineer	Login	The login name of the engineer who is currently logged in to the system
	Firstname	First name, last name, email of the engineer, be sure that the respective field has been filled in the engineer data. See chapter Engineer Administration for details about engineer management.
	Lastname	
	Email	
Includes	<all available includes>	
Text Blocks	<all available text blocks>	
Workflow Includes	<all available workflow includes>	
Scripts	<all available scripts>	

If customer-specific fields are used in a template, this template will only be offered when the ticket is assigned to a main customer from the respective customer group!

Since customer fields differ between customer data models, it might be necessary to define a similar template for several customer groups.

By clicking on *Add string parameter* or *Add enum parameter* you can define fields where the engineer has to fill in data at run-time. In the following example, the responsible consultant is added as string parameter and the ticket priority is added as enum parameter. An ENUM (sorted list, see section Managing Sorted Lists: Enum Administration) will only be offered as enum parameter if the option *Template enabled* has been set for the ENUM.

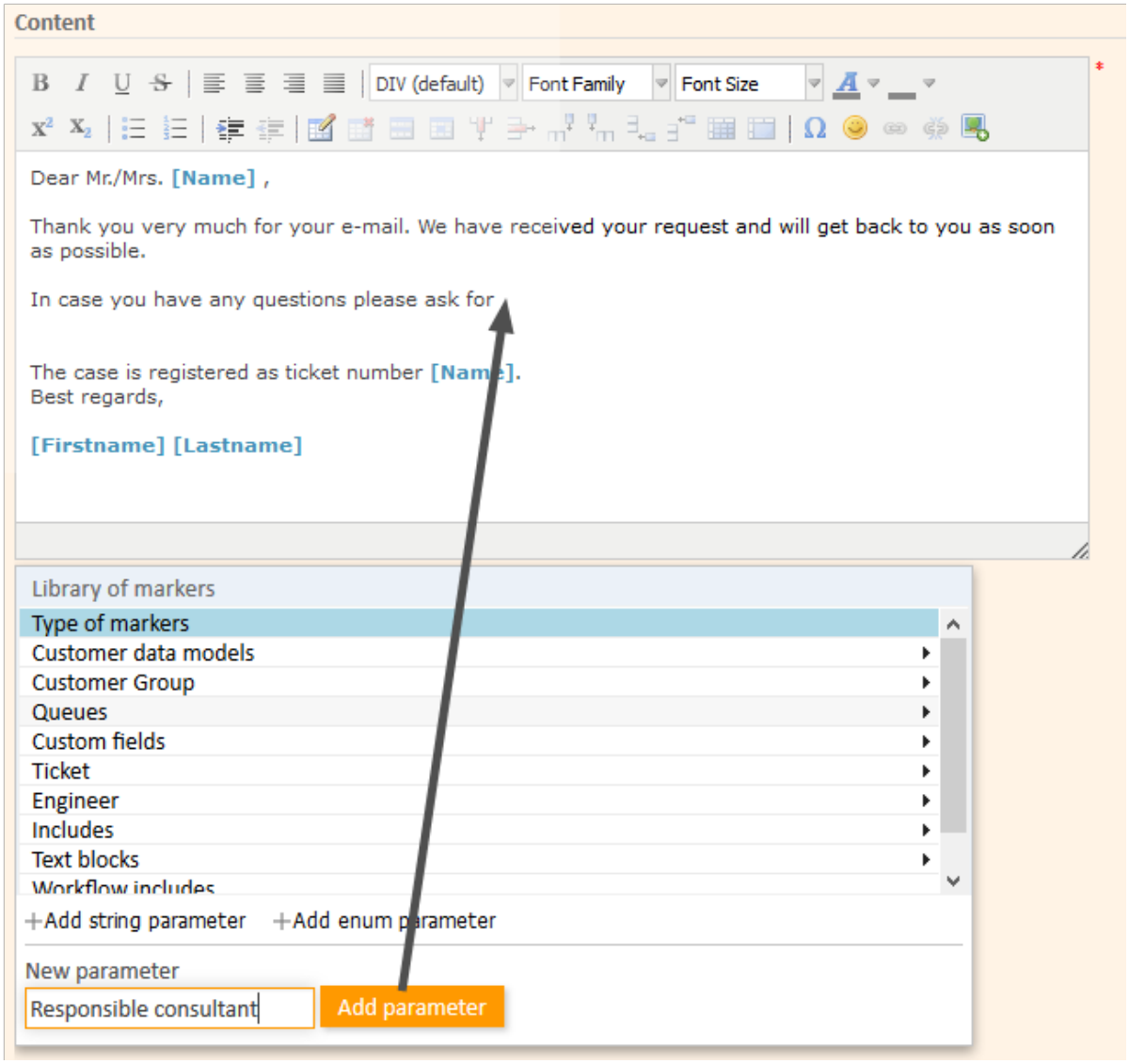


Figure 517: *ConSol CM Web Client - Text Template Manager: Add string parameter in Library of markers*

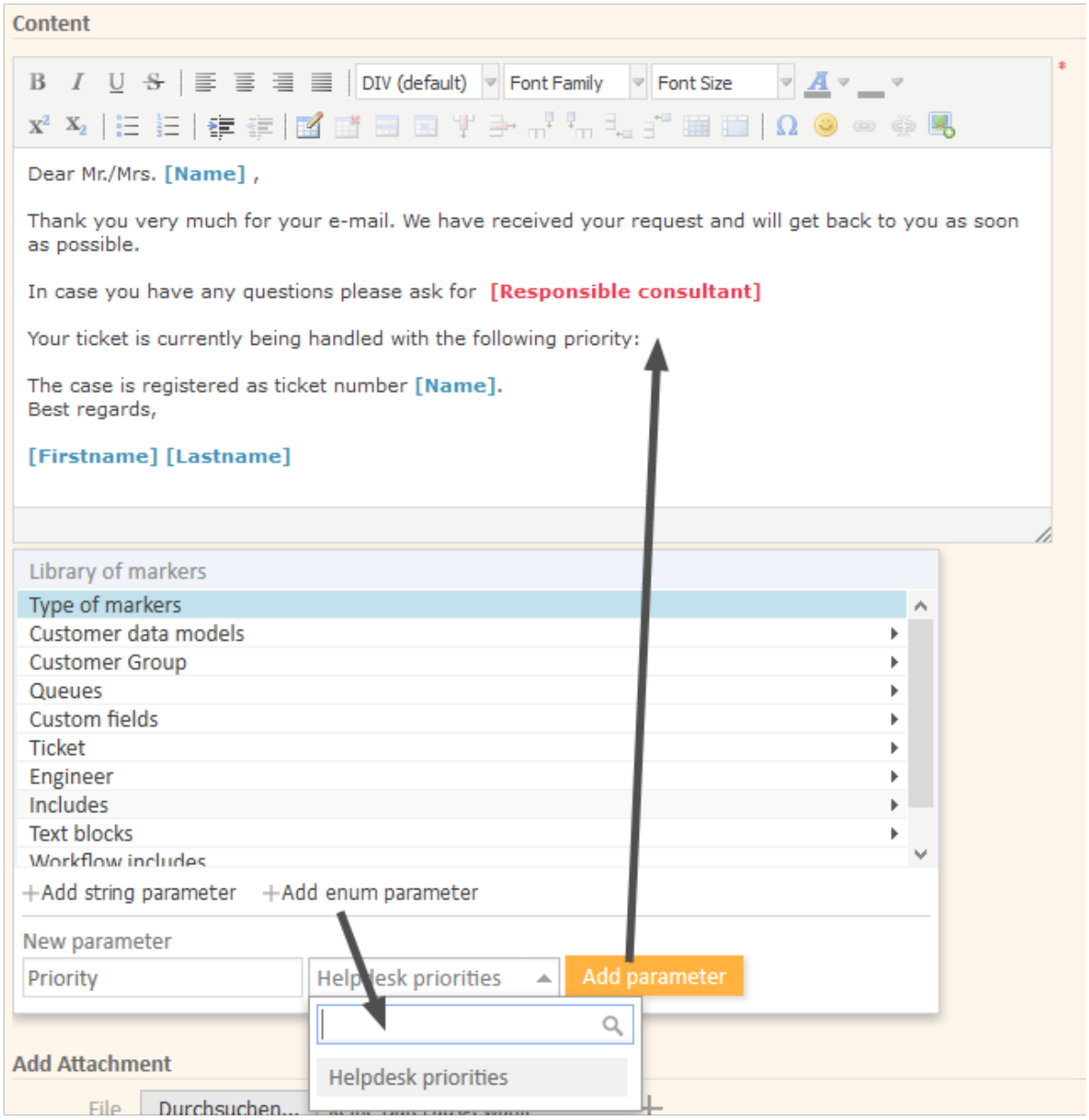


Figure 518: *ConSol CM Web Client - Text Template Manager: Add enum parameter in Library of markers*

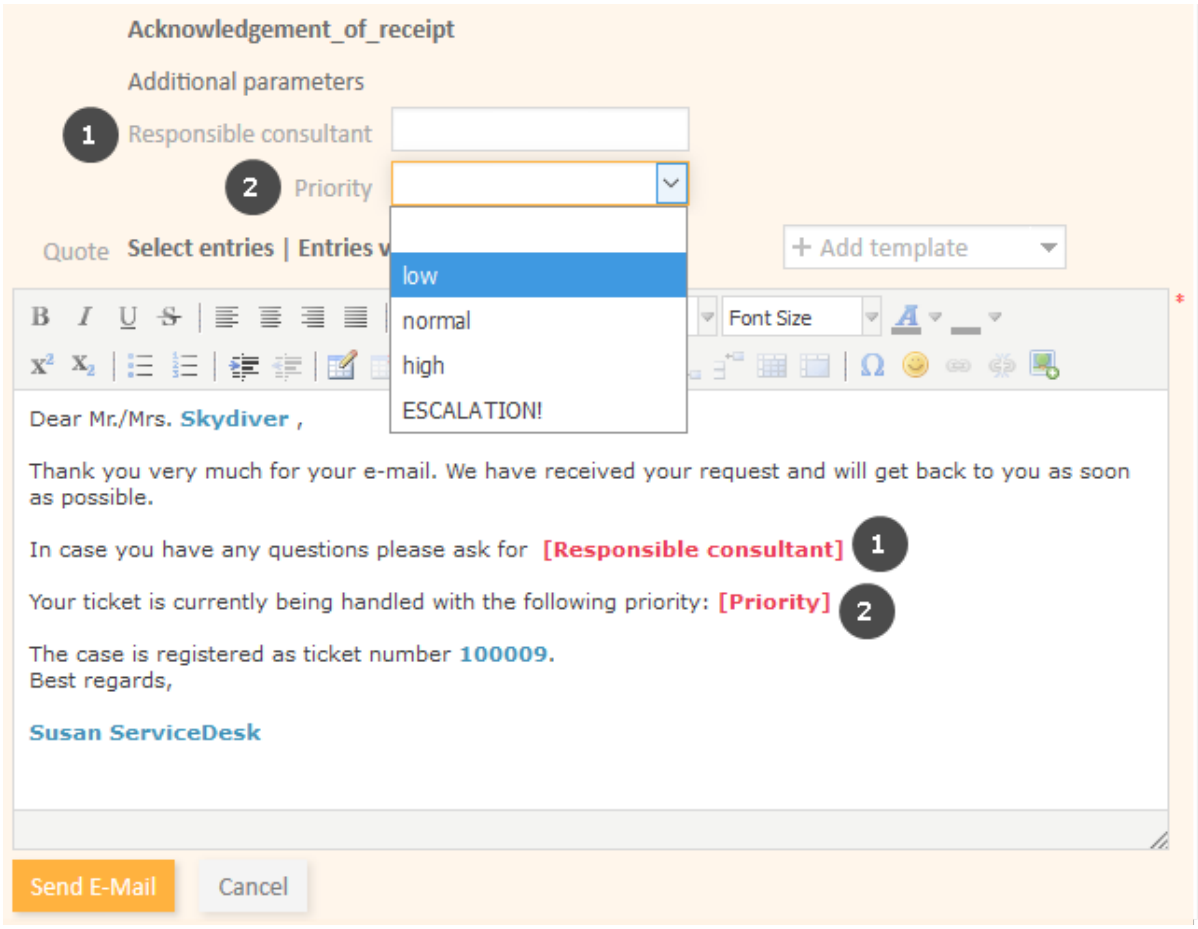


Figure 519: *ConSol CM Web Client - Ticket Email Editor: Filling in parameters, 1*

In the above example, *Responsible consultant* (1) is a string parameter, where the engineer can add the desired value in a text field. *Priority* (2) is an enum parameter, where the engineer can select the desired value from a drop-down list.

When the engineer opens the Ticket Email Editor in the ticket and enters data in the fields (here *Mike* as *Responsible consultant* and *low* as *Priority*), the (new) data is automatically written into the field in the template.

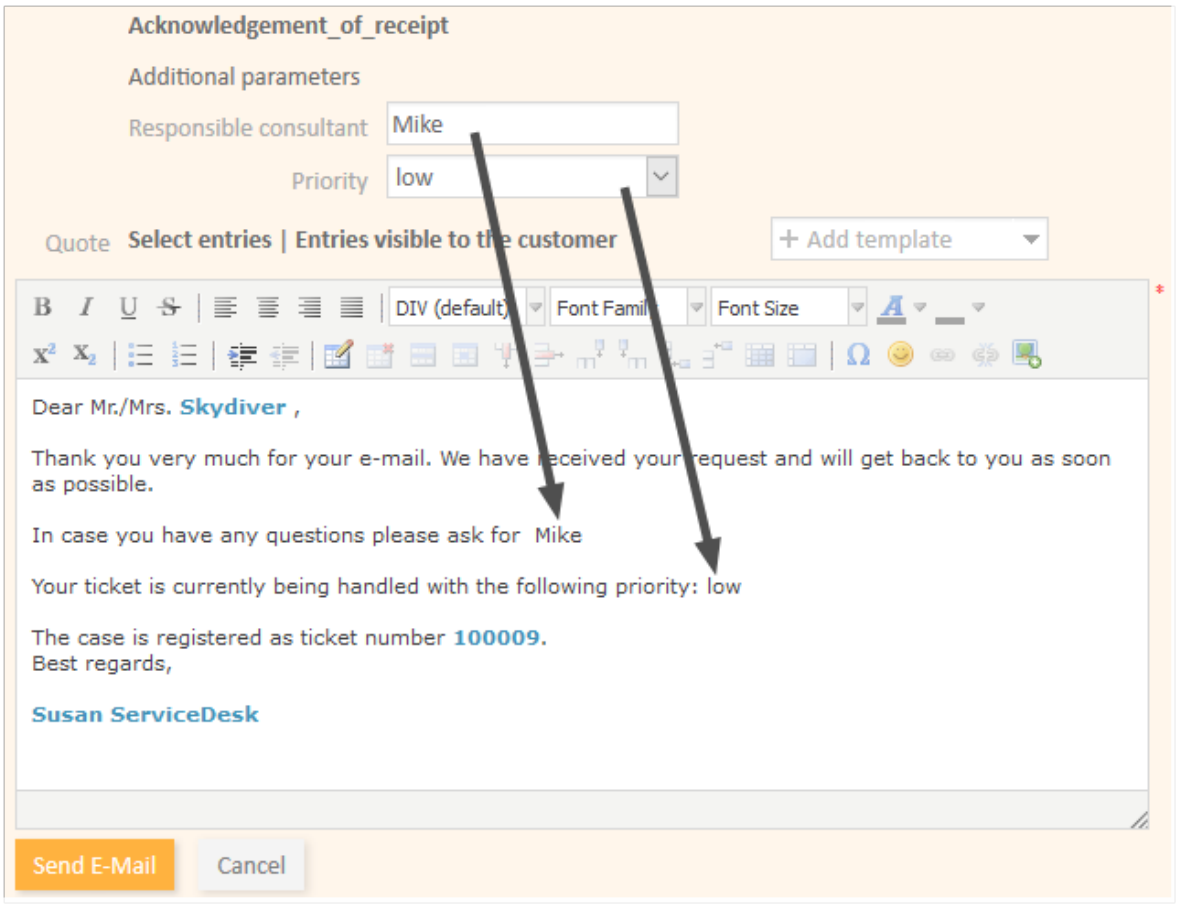


Figure 520: *ConSol CM Web Client - Ticket Email Editor: Filling in parameters, 2*

Create a New Include or Workflow Include

An include is a template that cannot be selected by the engineer directly (in the Ticket Email or Comment Editor), but is a component which is integrated in other email or ticket text templates, namely in letters.

A standard use case for an include is the signature, so we will show you the corresponding example. In order to define the standard company signature, define a signature as an include and integrate it into the standard company signature which is a letter.

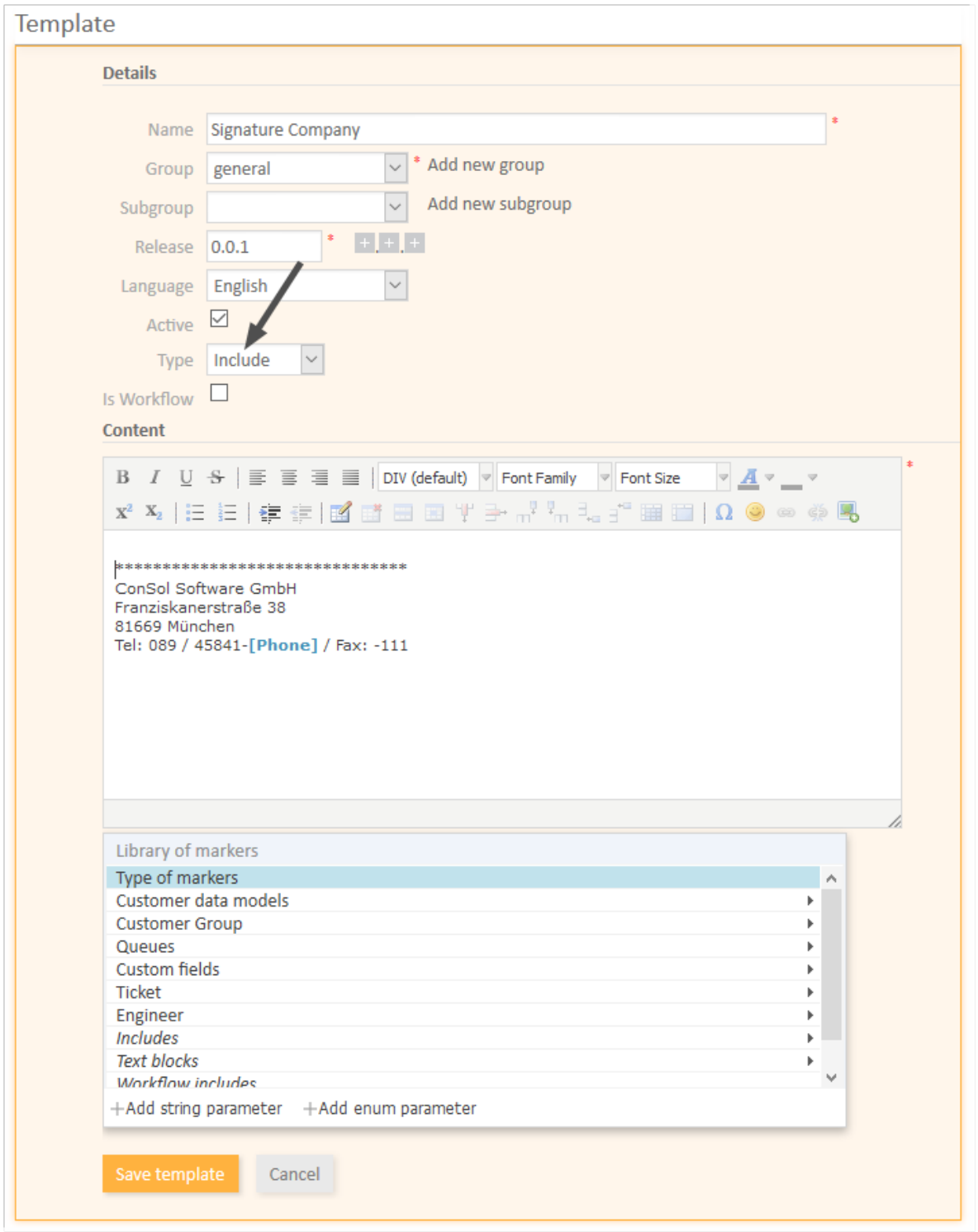


Figure 521: *ConSol CM Web Client - Text Template Manager: Signature Include*

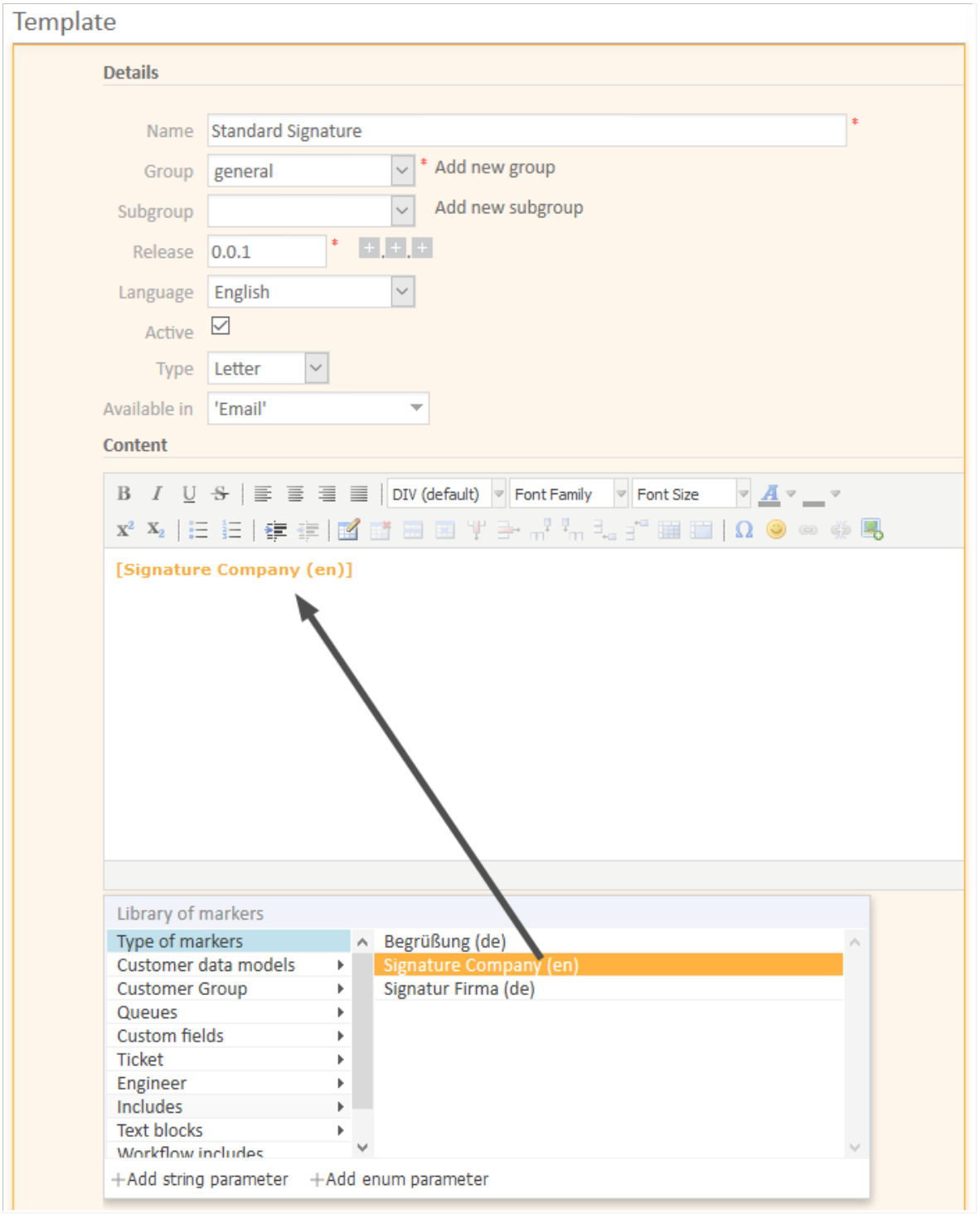


Figure 522: *ConSol CM Web Client - Text Template Manager: Standard signature letter*

The implications of this example:

- The *Standard_Signature* can be defined as email standard. This will cause it to be automatically displayed for every new email. Of course, the engineer can change the template.
- The *Signature_Company* can be used in any other template if required (compare image of the new template).

Create New Text Blocks

A text block is a template that cannot be selected by the engineer directly (in the Ticket Email or Comment Editor), but is a component which is integrated in other email or ticket text templates, mostly in letters. Usually, several text blocks are offered in one letter so that the engineer can select which one (s) to use.

The following example shows how to use three text blocks to ask the customer to answer preliminary analysis questions.

First, the text blocks are created:

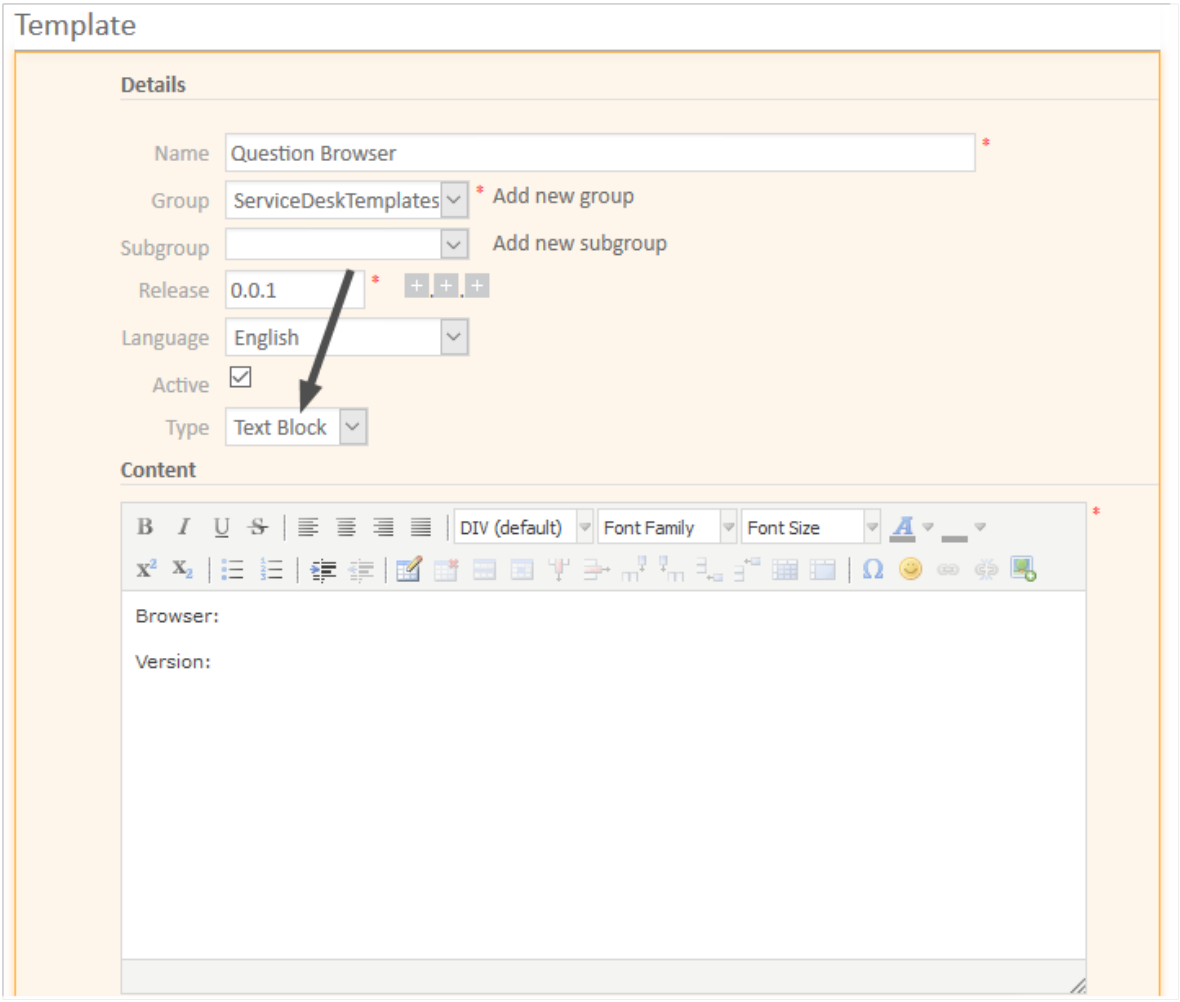


Figure 523: *ConSol CM Web Client - Text Template Manager: Create first text block*

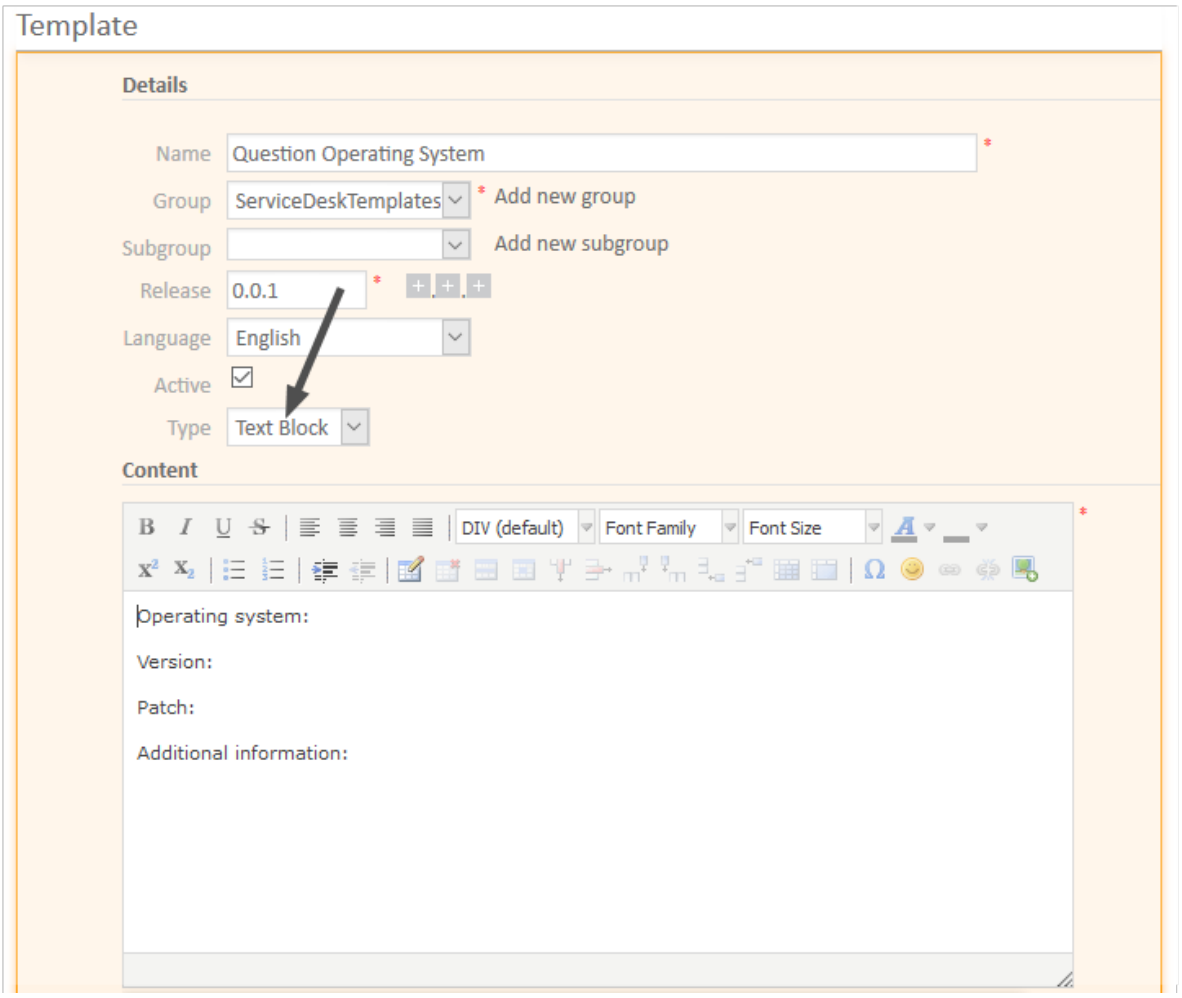


Figure 524: *ConSol CM Web Client - Text Template Manager: Create second text block*

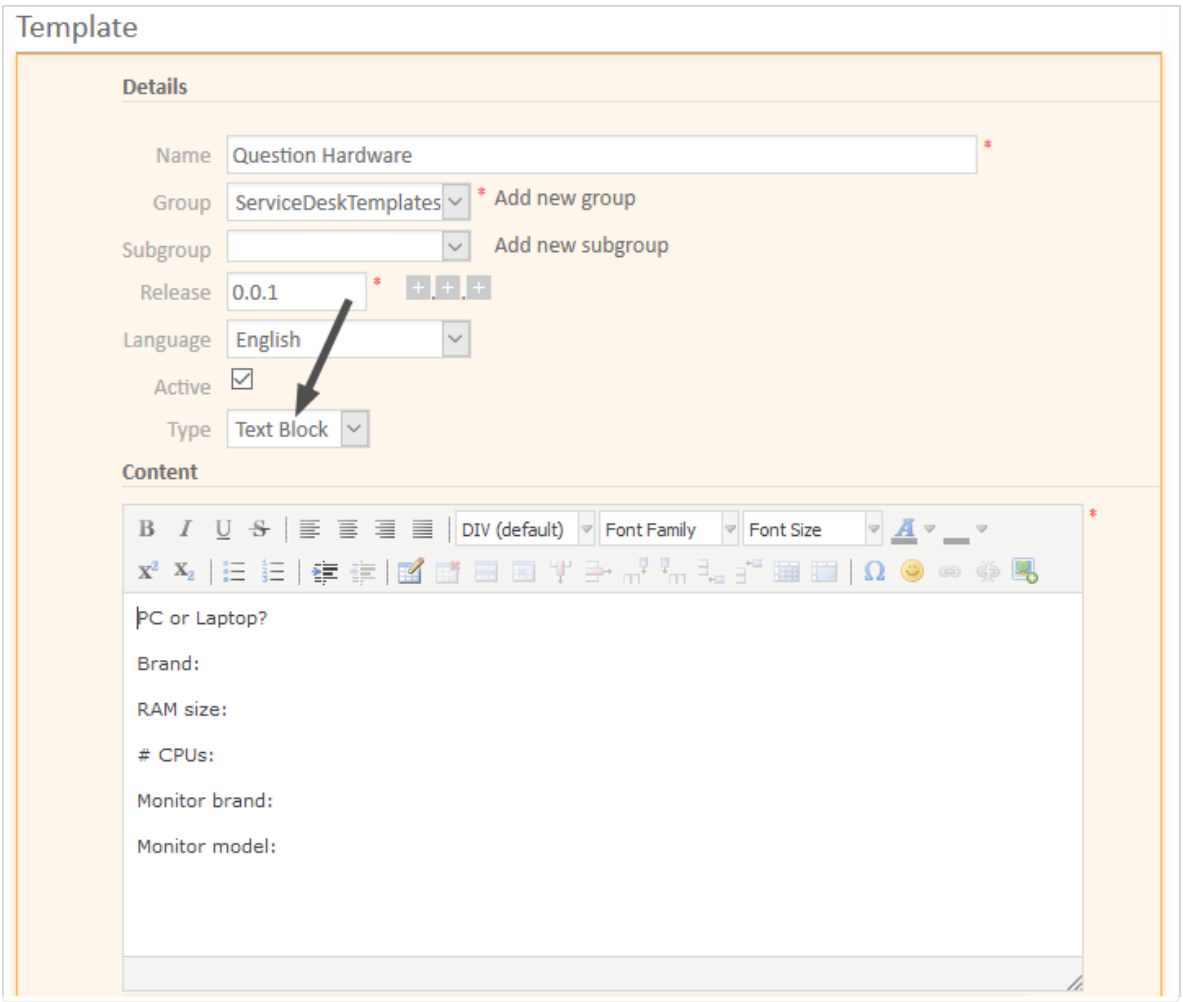


Figure 525: *ConSol CM Web Client - Text Template Manager: Create third text block*

Then the letter is created where the text blocks should be used:

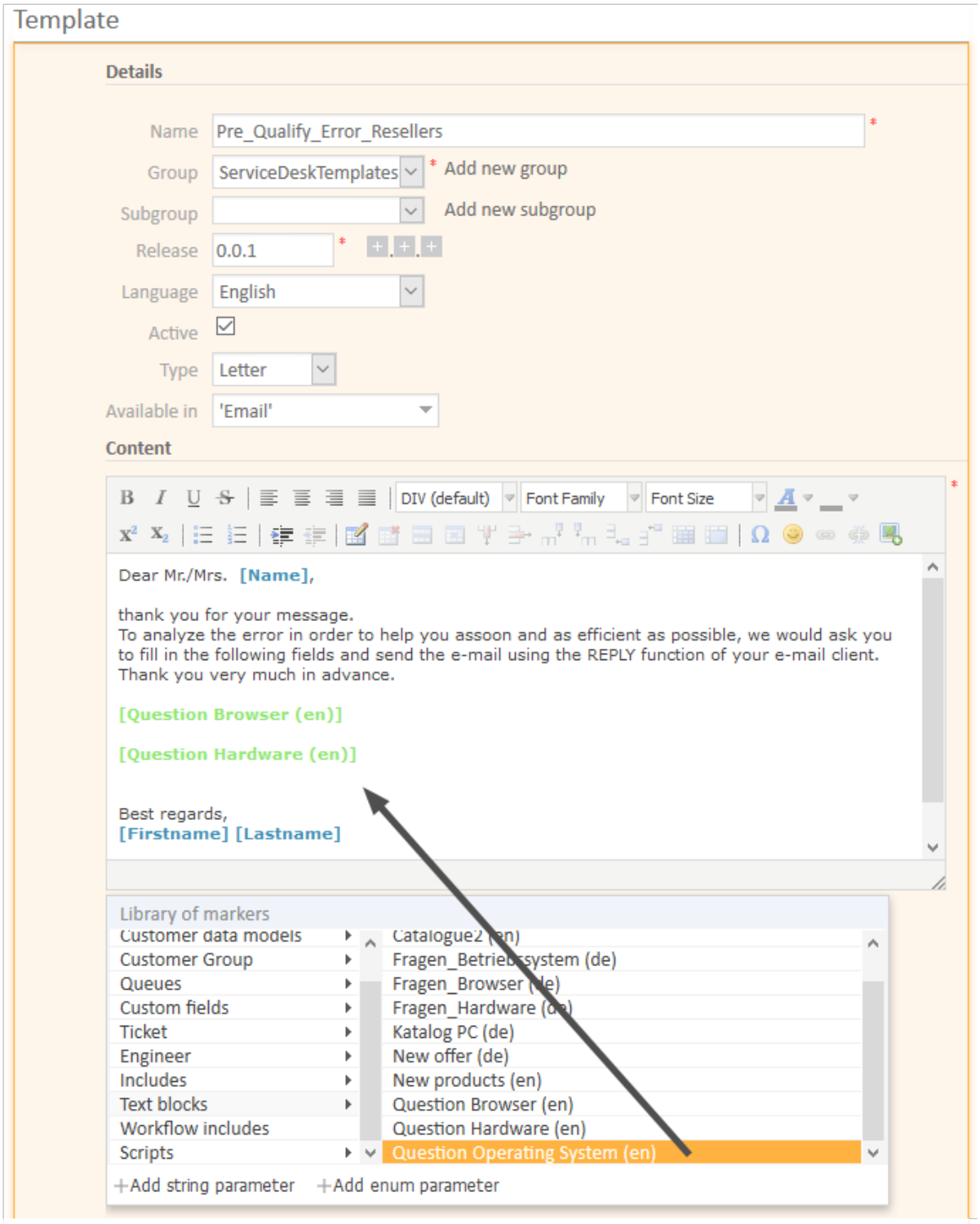


Figure 526: *ConSol CM Web Client - Text Template Manager: Create letter for text blocks*

In the Web Client, the engineer can then decide which text blocks to use and which ones to deactivate:

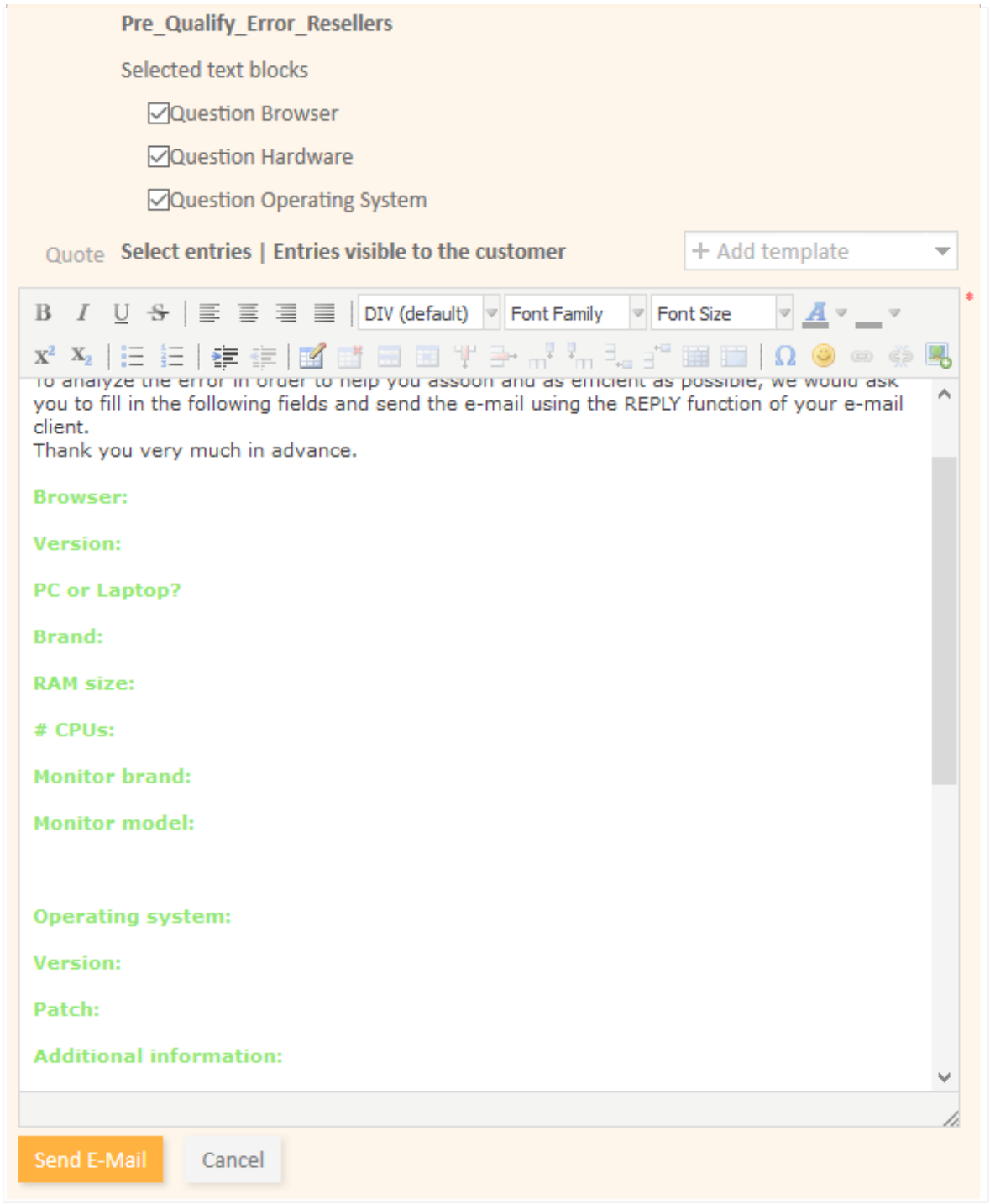


Figure 527: *ConSol CM Web Client - Ticket Email Editor: All text blocks selected*

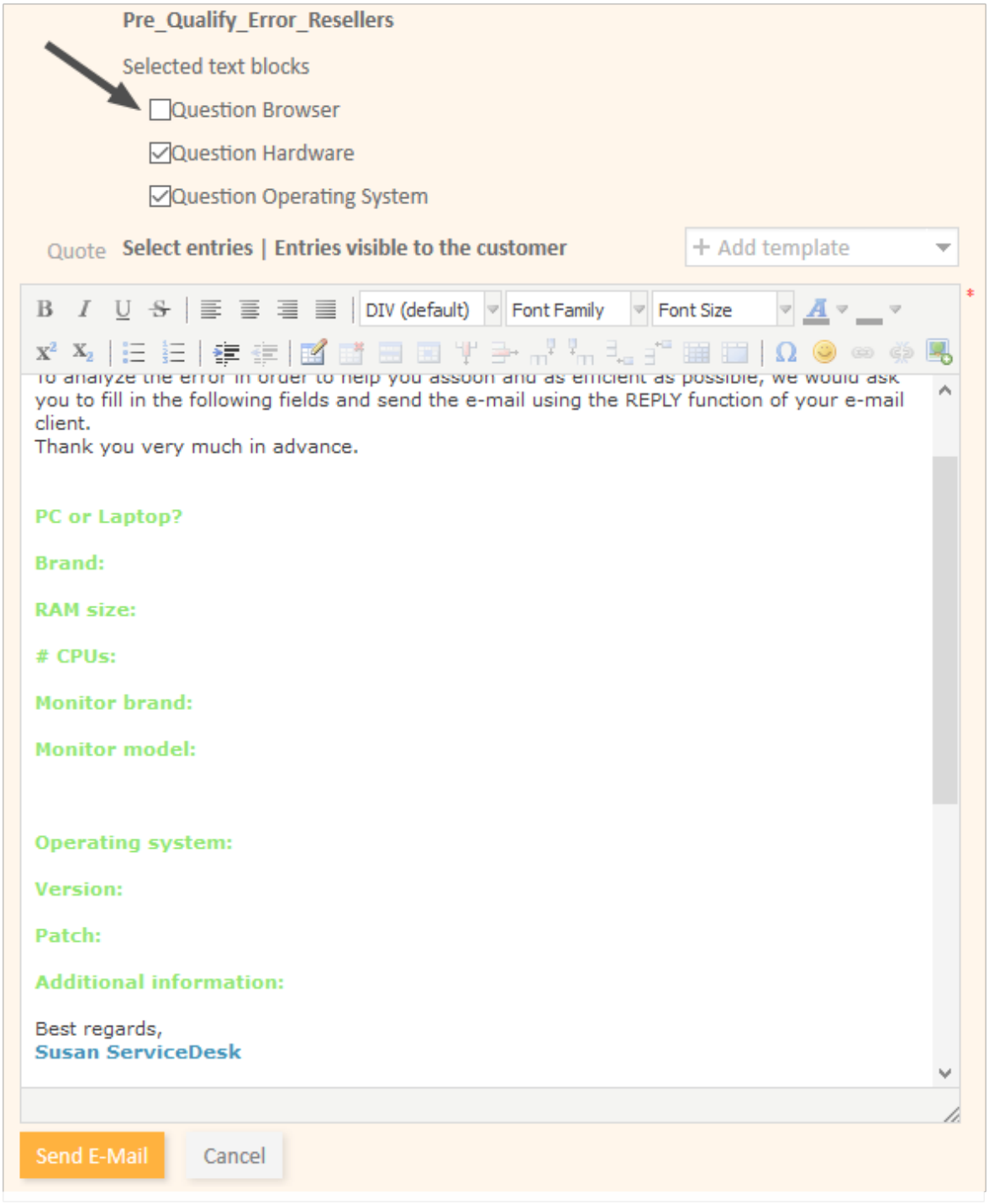


Figure 528: *ConSol CM Web Client - Ticket Email Editor: One text block deactivated*

Create and Use a Script

Sometimes a system has to have a certain "intelligence" regarding the words and phrases used in email templates, because they are not static but have to be adapted in a dynamic way. A standard example is the use of *Dear Sir* for male customers (salutation = “Mr”) and *Dear Madam* for female customers (salutation = “Mrs”).

Use cases like this can be covered using scripts in the Text Template Manager.

This can only be achieved by a ConSol CM administrator. When you log in to the Web Client as a regular user with template managing permissions, you can define all template types but no scripts. In order to be able to select *Script* as template type, you have to log in with an administrator account.

For information about the syntax that is used, please see the following web links:

- FreeMarker
- FreeMarker directives

Please note that in the standard Freemarker notation, angle brackets (<>) are used for statements. Here, in an HTML environment, this cannot work, and therefore, instead of angle brackets, square brackets ([ ]) have to be used.

An example script is the following:

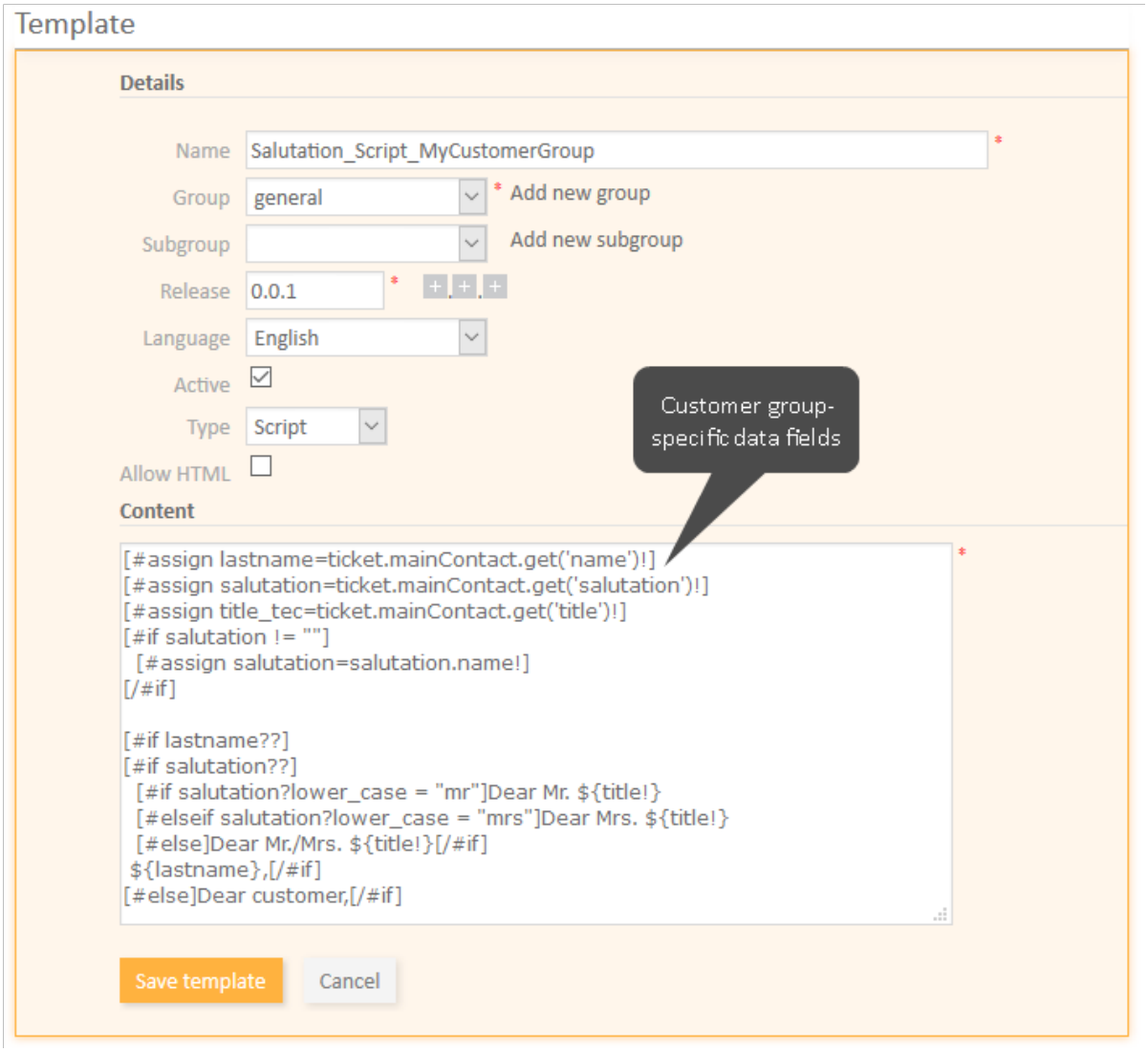


Figure 529: *ConSol CM Web Client - Text Template Manager: Example script for salutation*

You can also use HTML syntax in a script as shown in the following example.

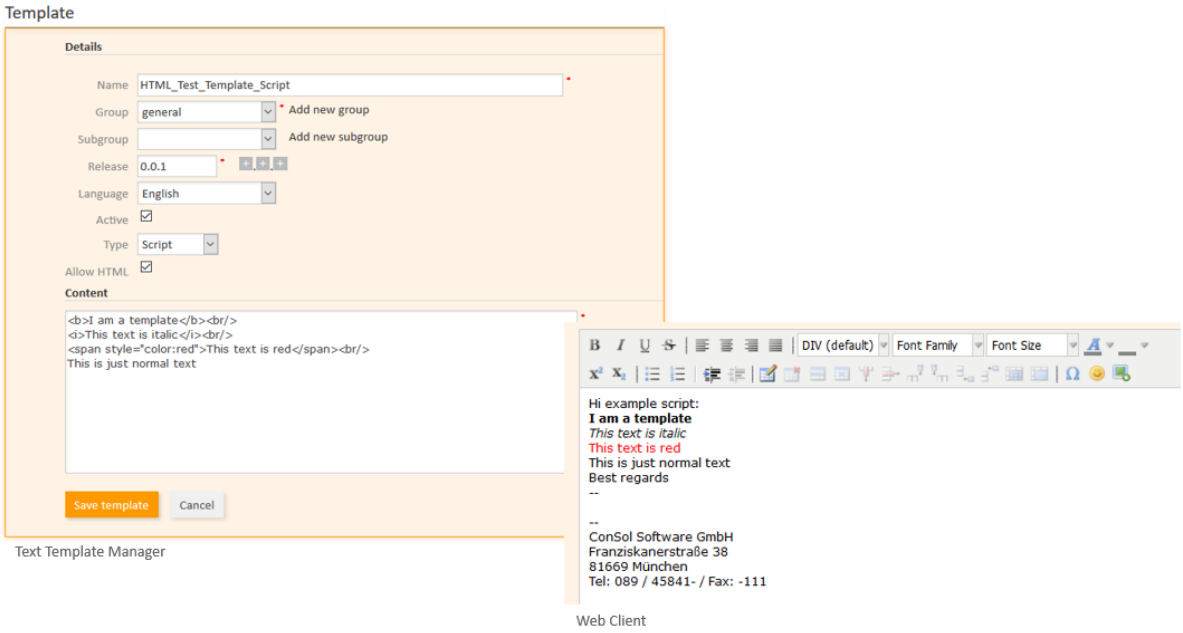


Figure 530: *ConSol CM Web Client - Editing a script with HTML syntax in the Text Template Manager and using the template in a ticket*

Please keep in mind that ...

- the values of the fields are the technical values (in the example, the technical value of the field *salutation* is “mr” / “mrs”, the localized value for EN would be “Mr” / “Mrs”). Always use the technical values!
- the fields are the ticket fields and customer fields managed in the Admin Tool. Please refer to sections Ticket Field Administration (Setting Up the Ticket Data Model) and Setting Up the Customer Data Model for details.

The script is then integrated into a letter template:

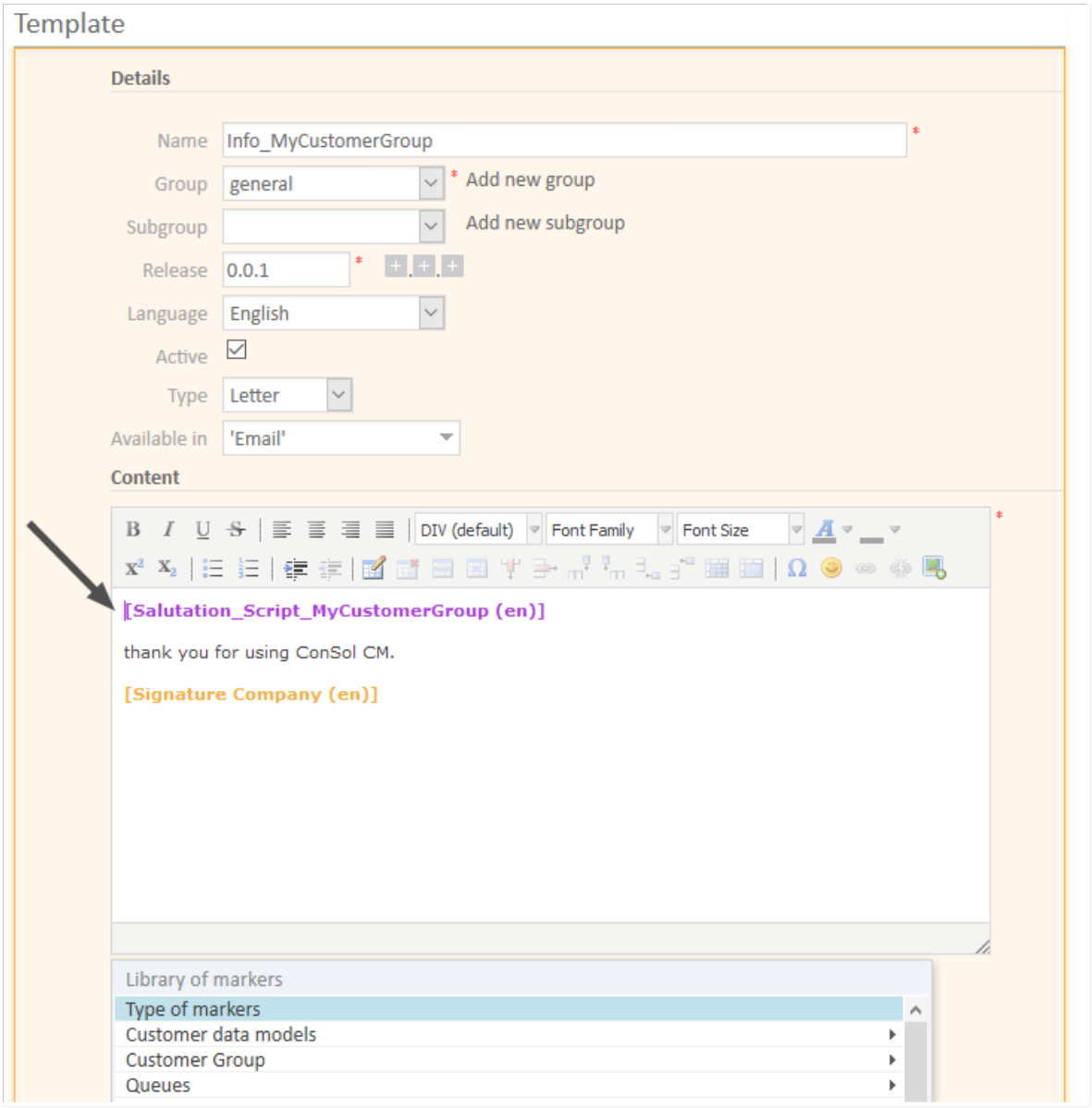


Figure 531: *ConSol CM Web Client - Text Template Manager: Insert script into letter*

In the Web Client, the emails are formatted as requested.

Example 1 (for *Mrs*):

Figure 532: *ConSol CM Web Client - Emails formatted by script (customer data, example 1)*

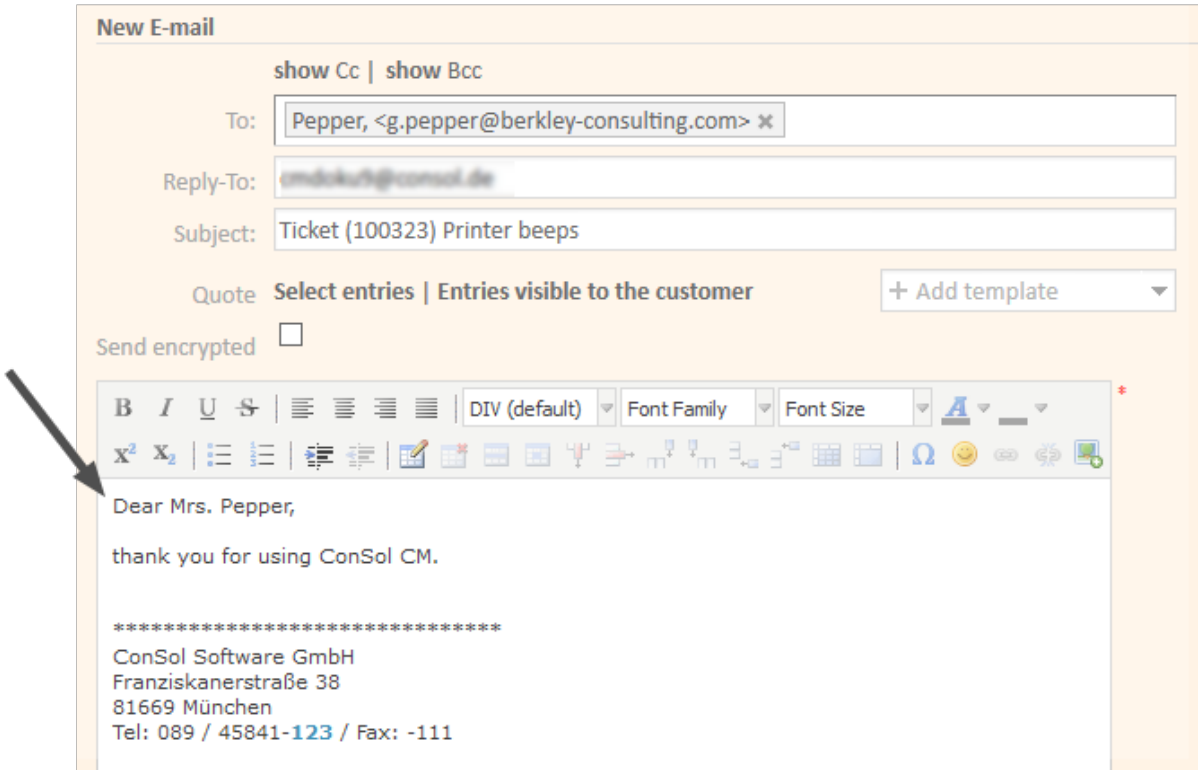


Figure 533: *ConSol CM Web Client - Emails formatted by script (email, example 2)*

Example 2 (for **Mr**):

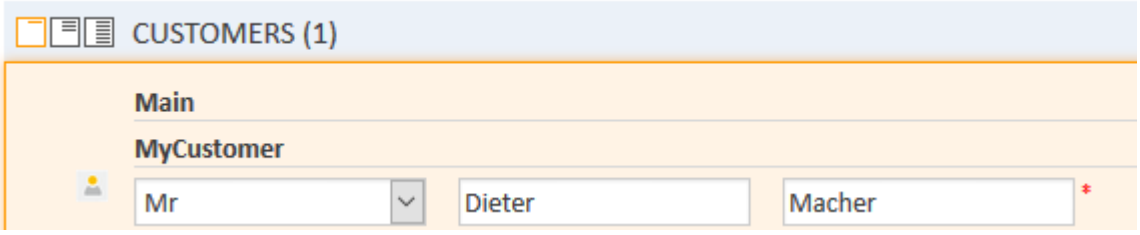


Figure 534: *ConSol CM Web Client - Emails formatted by script (customer data, example 1)*

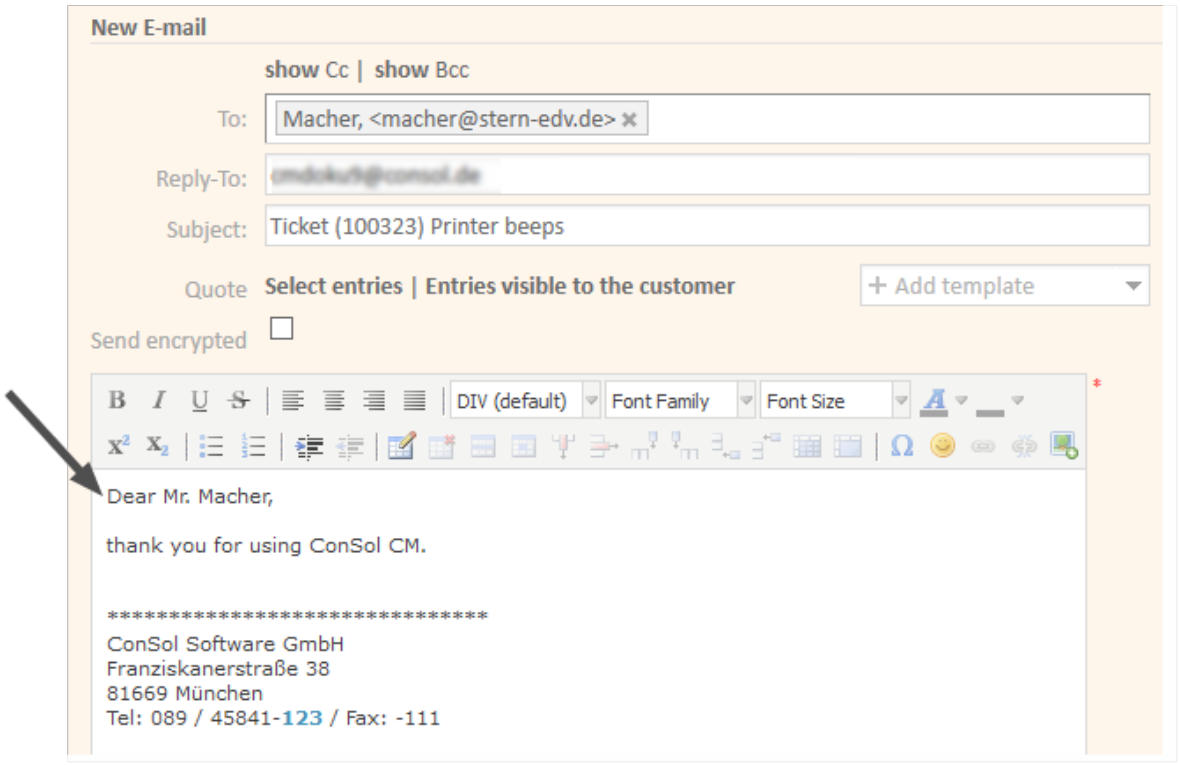


Figure 535: *ConSol CM Web Client - Emails formatted by script (email, example 2)*

Use a Script to Set Queue-Specific Signatures

In case your company uses more than one standard signature, e.g., in emails, you can cover this use case also by using a template script. The following example demonstrates how to use a specific signature for each queue. This might be useful if each department works with a specific queue. So by defining queue-specific signatures in emails you can provide a department-specific signature.

Step 1:
Create the signatures for each department. These are regular text templates of type *Letter*. For example:

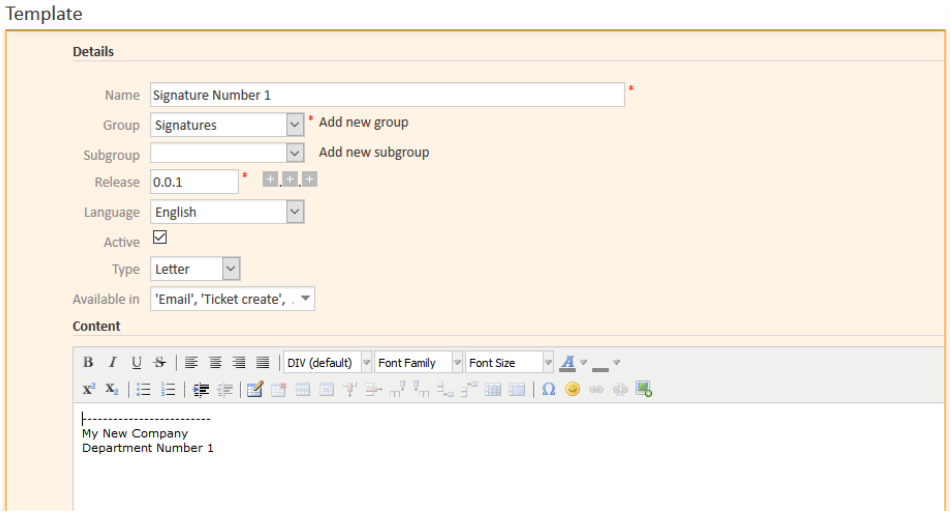


Figure 536: *ConSol CM Web Client - A simple signature*

Step 2:
Create a template of type *Script* which manages the queue-specific template selection, for example:

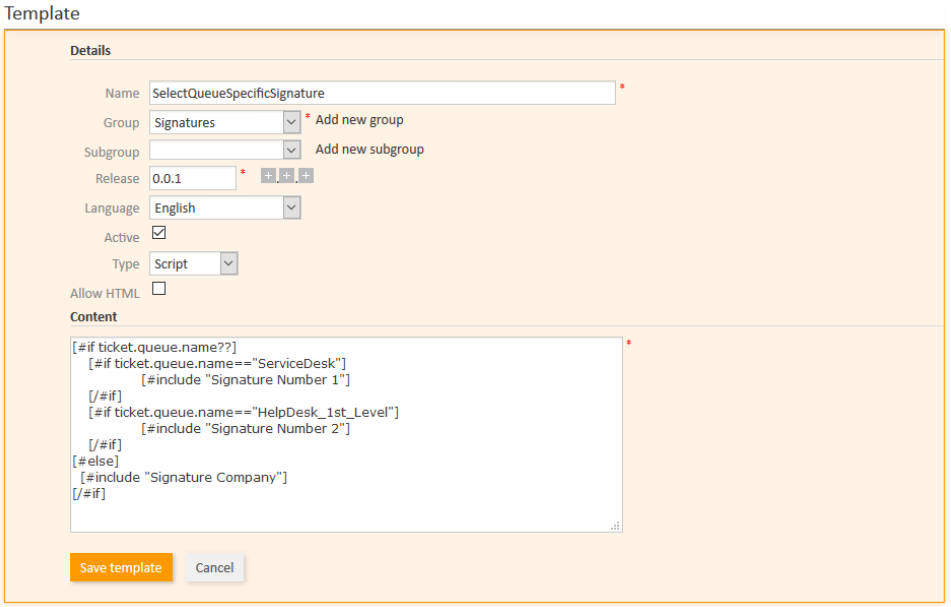


Figure 537: *ConSol CM Web Client - A template of type Script which manages queue-specific selection of signatures*

Step 3:
Create a simple template of type *Letter* which is used as email standard template and include the script.

That's it!

An engineer can now use the Ticket Email Editor and will see a different signature depending on the queue of the ticket. See the following two figures as examples.

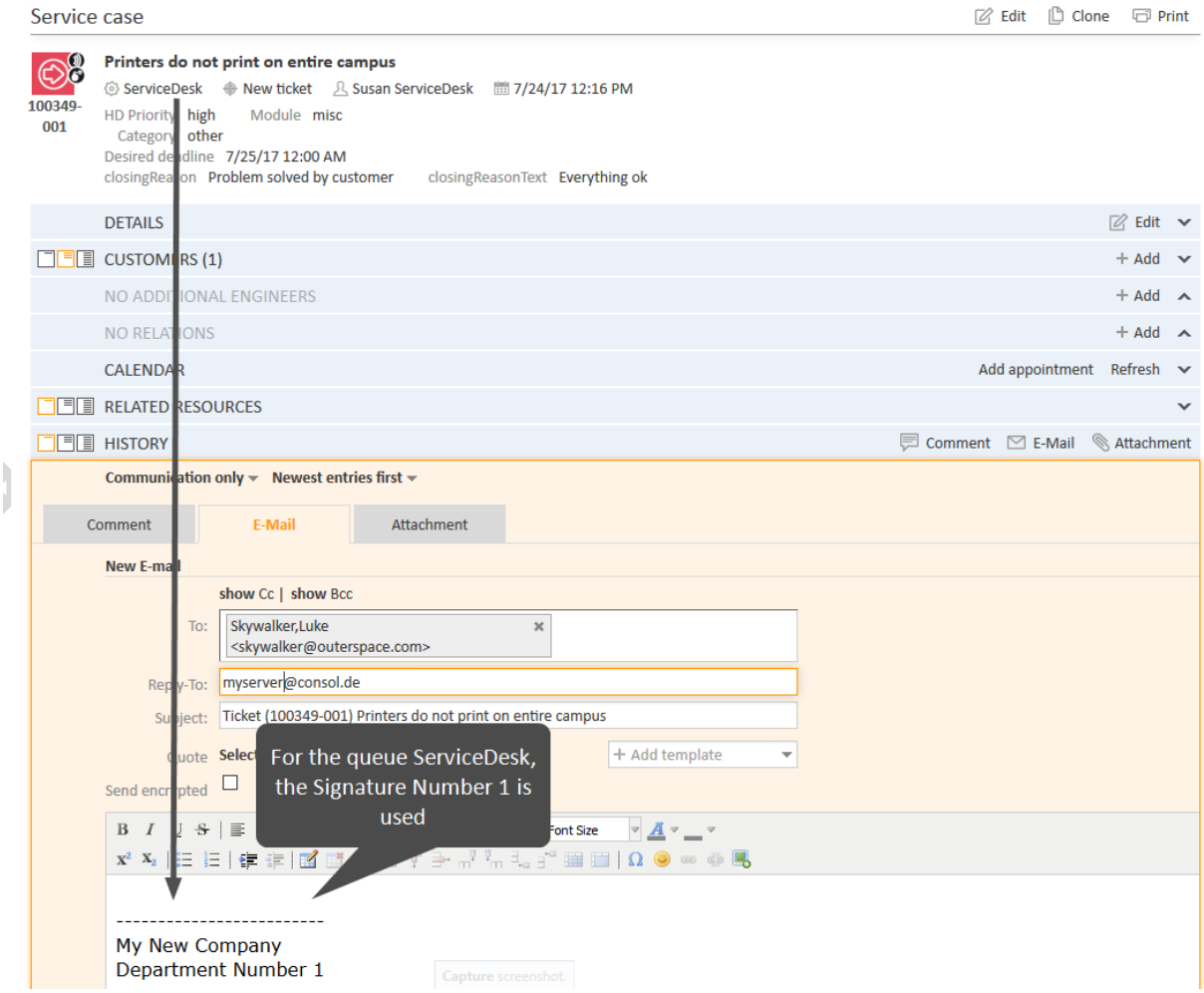


Figure 538: *ConSol CM Web Client - Queue-specific signature, managed by template script, 1*

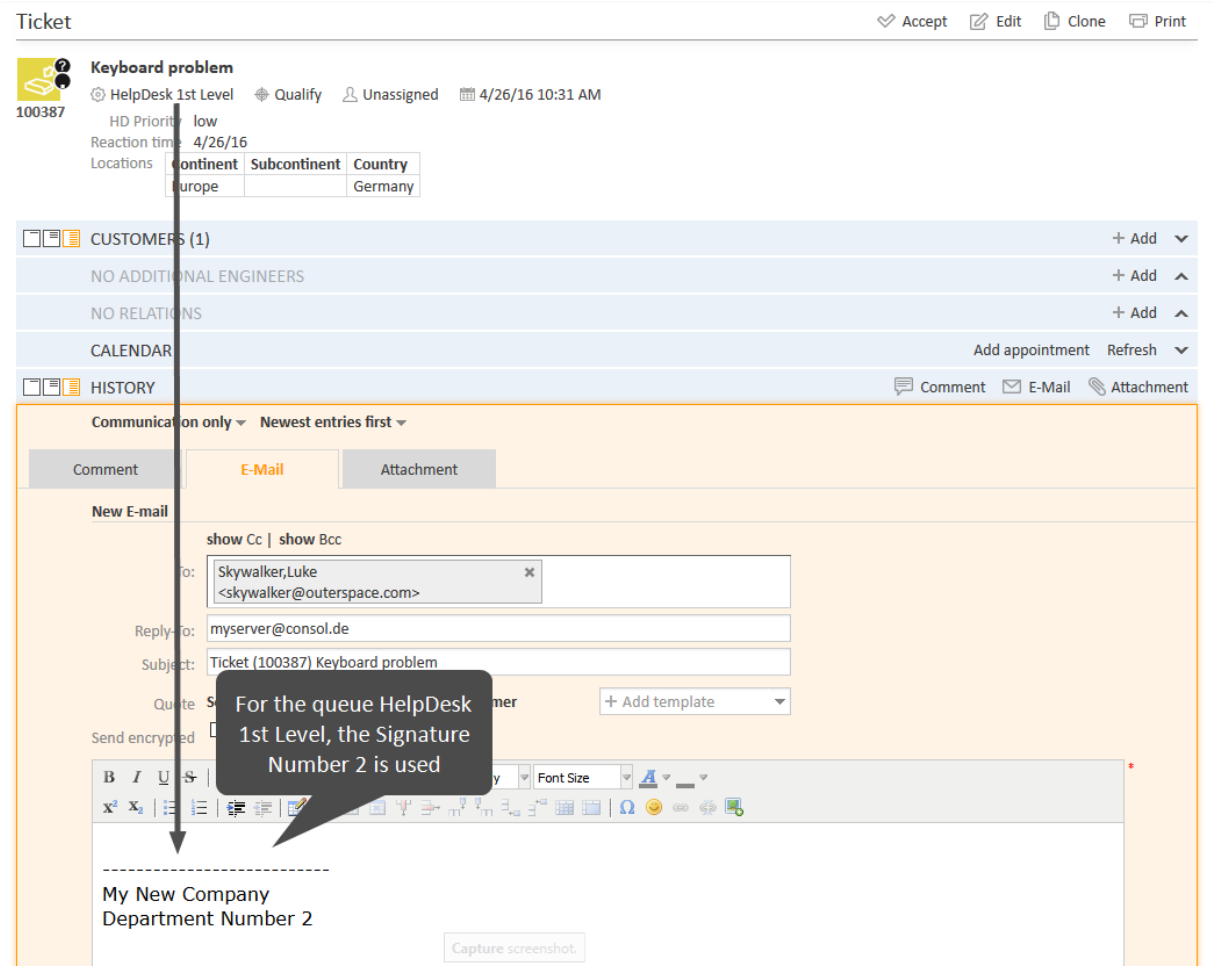


Figure 539: *ConSol CM Web Client - Queue-specific signature, managed by template script, 2*

Binding Templates to Queues or to Specific Parameters

The section *Binding* is the last section of the *New Template* page of the Text Template Manager.

For every template you can decide if it should be displayed everywhere without any restrictions (i.e., in every queue and without regarding any parameters) or if it should be bound (= restricted) to specific criteria. This can be:

- queues
- queue-specific parameters (e.g., display the template only if the priority *high* has been set for the ticket)

You can select queues and/or parameters by selecting a *context*, as shown in the following pictures. Use the "+" button to add more binding parameters or the "-" button to remove existing parameters.

Example 1: The template should only be displayed in the *Helpdesk 1st Level* queue:

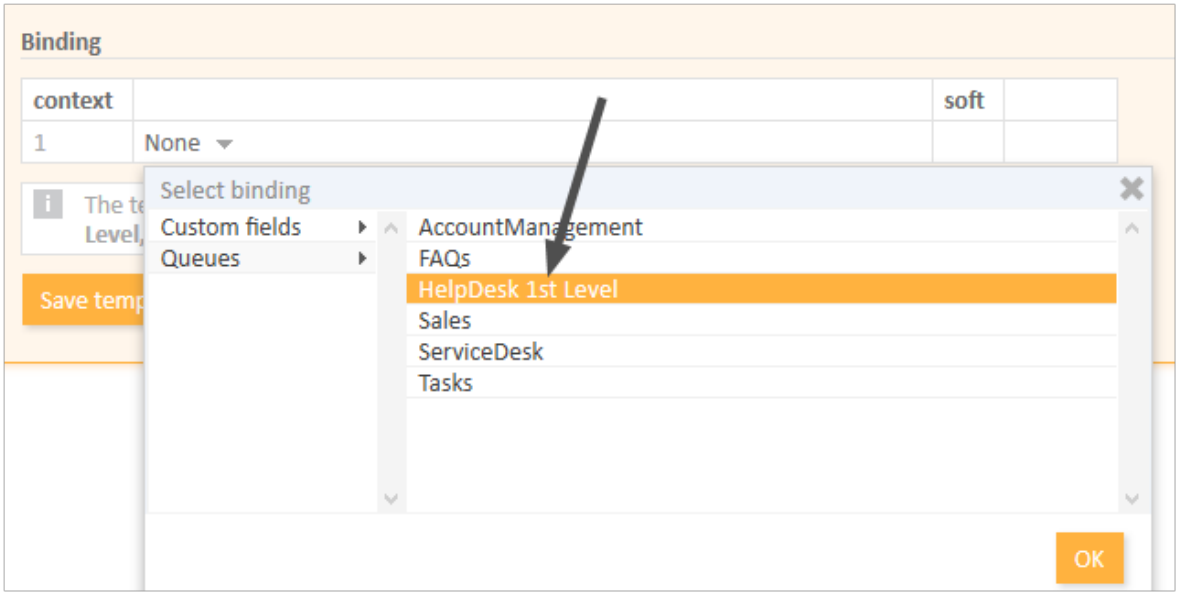


Figure 540: *ConSol CM Web Client - Show template for specific queue*

Example 2: The template should only be displayed in the *Helpdesk 1st Level* queue and only if the priority is *high*:

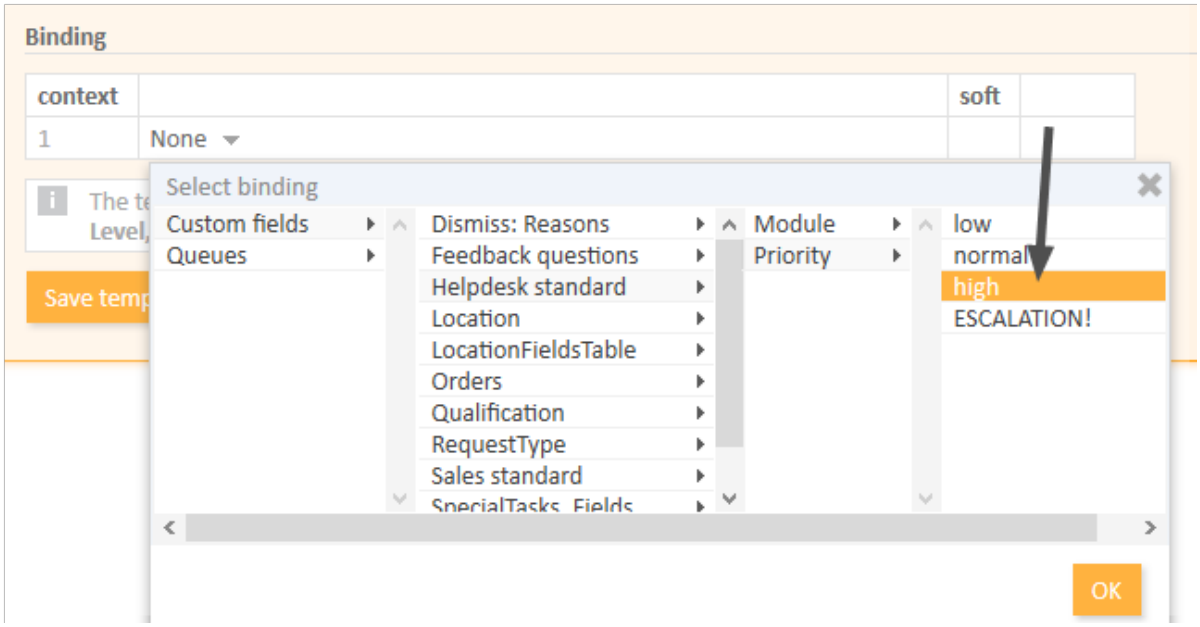


Figure 541: *ConSol CM Web Client - Display template for specific queue and priority*

Please note the following concerning bindings of a template to specific parameters:

1. As specific parameters only ticket fields of type ENUM are available.
2. One or more specific parameters (i.e., ticket fields of type ENUM) can be excluded. This means they will no longer be offered in the *Select binding* menu. If a really great number of ticket fields is loaded each time the text template editor is opened in edit mode of a template, this might decrease the performance. Loading of the edit page might take quite long. In such cases, it can help to exclude ticket fields from the list as long as it is really clear that they will not be required for template design in any case.
   You can exclude specific ticket fields using the page customization of the markersLibrary (Type).

Hard and Soft Binding

When the template is only displayed in one (or more) selected queue(s), as shown in the example above, the template is bound to those queues or to any other selected (restricting) parameter. There are two types of bindings:

- **Hard binding**
  The checkbox *soft* is **not** checked:
  The template is only displayed (offered in the Ticket Email Editor or Ticket Comment Editor) in the selected queues or for the selected parameters. The engineer who works with the template cannot change this configuration.
- **Soft binding**
  The checkbox *soft* is checked:
  As default setting, the template is only displayed (offered in the Ticket Email Editor or Ticket Comment Editor) in the selected queues or for the selected parameters. However, the engineer can change the display by clicking on the button *More templates*. All softly bound templates are displayed as well.

F.12.1.4 Migrating Templates from CM Version 6.8 and Less to CM Version 6.9 and Up

When a ConSol CM system is updated from a version 6.8 and less to a version 6.9 and up, the scripts in the Text Template Manager have to be checked and modified manually.

Parameters which refer to tickets and queues do not have to be changed. However, due to the new customer data model, FlexCDM, the syntax for references to customer fields has to be adapted.

F.12.1.5 Page Customization for Email Template Functionalities

Please refer to section Page Customization to learn some details about how to adapt email template parameters.

F.12.2 CM/Doc

F.12.2.1 Introduction to CM/Doc

Even in companies where most processes are managed by IT applications, a great number of documents still have to be printed out or are required in `.doc` or `.pdf` format for other reasons. These can be, for example:

- invoices
- contracts
- documents concerning the acceptance of IT systems
- orders

ConSol CM offers the add-on CM/Doc to print documents directly from the business management process. CM/Doc supports Microsoft Word documents and (in CM versions 6.10.1 and up) OpenOffice documents.

Templates guarantee that ...

- all documents of one type are identical (text and layout).
- all documents match the company's CD (corporate design).
- no engineer has to type the same text over and over again.

Data from the ticket can be integrated into the template automatically, these can be:

- ticket data (e.g., amount in an invoice, service level in a contract)
- customer data (name and address of the main customer (and of the company, if the main customer is a contact belonging to a company), additional customers are **not** relevant in CM/Doc!)
- engineer data (name, phone number, email address of the engineer who works on the case)

When CM/Doc is active in ConSol CM, the engineer can select the required Microsoft Word template (or OpenOffice template) in the ticket. The document is opened in Microsoft Word (OpenOffice) automatically with all required data fields already filled in. The engineer can then work on the document and save it. It is automatically attached (as a regular attachment) to the ticket and can be opened by users who have **read** access to the ticket and who have installed the required software (Microsoft Word, OpenOffice) on their PCs.

With special adaptations implemented by the ConSol CM consulting team, the ConSol CM system can be extended in a way that `.docx` documents can also be converted to `.pdf` files in order to make sure no further changes can be made to the document.

F.12.2.2 Requirements for Using CM/Doc

On the client PC or laptop, the following requirements have to be met to use CM/Doc:

- A JRE (Java Runtime Environment) for the web browser has to be installed, because CM/Doc is based on Java applets. For the supported Java versions, please refer to the current *System Requirements*.
- A web browser which supports Java Applets. Please refer to the current *System Requirements*.

- Microsoft Word / OpenOffice has to be installed. For the supported Microsoft Word and OpenOffice versions, please refer to the current *System Requirements*.

F.12.2.3 Availability of CM/Doc

CM/Doc is available in ConSol CM version 6.7 and higher and is part of the standard function set of the application.

F.12.2.4 Configuring the ConSol CM System for CM/Doc

If you want to activate CM/Doc in your ConSol CM system, the first step is to set the system property cmweb-server-adapter, cmoffice.enabled to “true”.

You reach this screen by opening the navigation item *System Properties* in the navigation group *System* and clicking the respective entry.

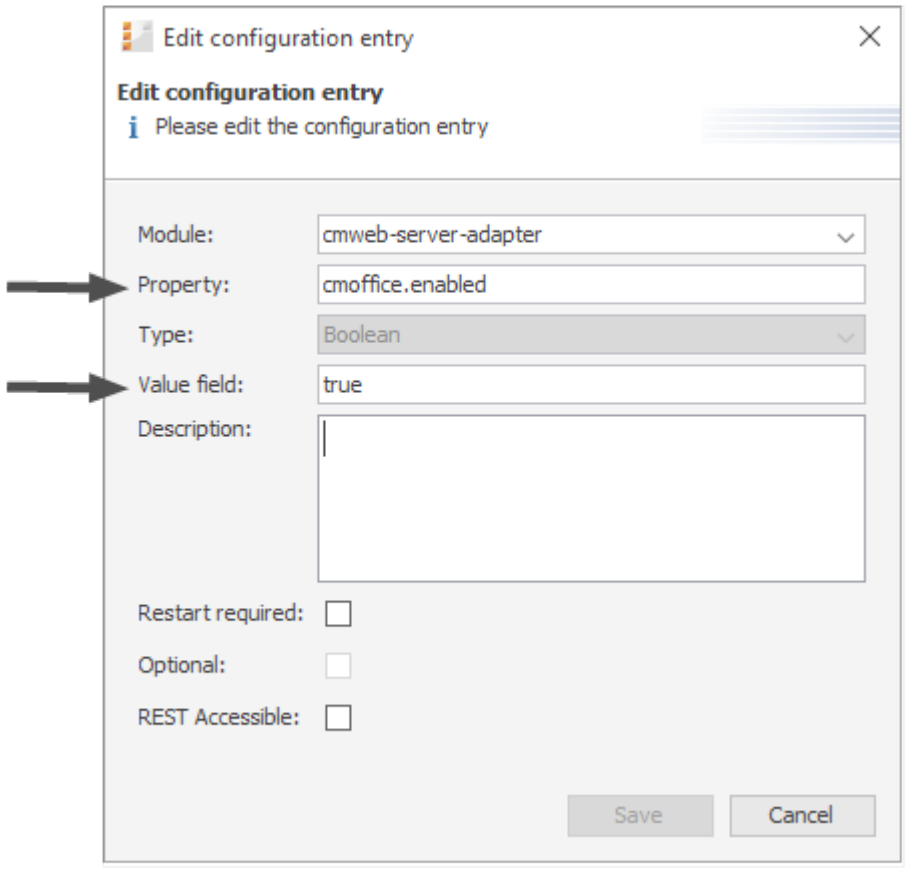


Figure 542: *ConSol CM Admin Tool - System, System Properties: Configuration of system property for CM/Doc*

F.12.2.5 Creating an Engineer Role with Permissions for the Document Template Manager

Only an engineer who has the permission *Write template* (see the following figure) can start the ***Document Template Manager*** in the Web Client. So, as a second step, you have to create one or more role(s) with the respective permissions. For a detailed explanation about setting role permissions, please refer to section Role Administration.

You reach this screen by opening the navigation item *Roles* in the navigation group *Access and Roles* and selecting a role.

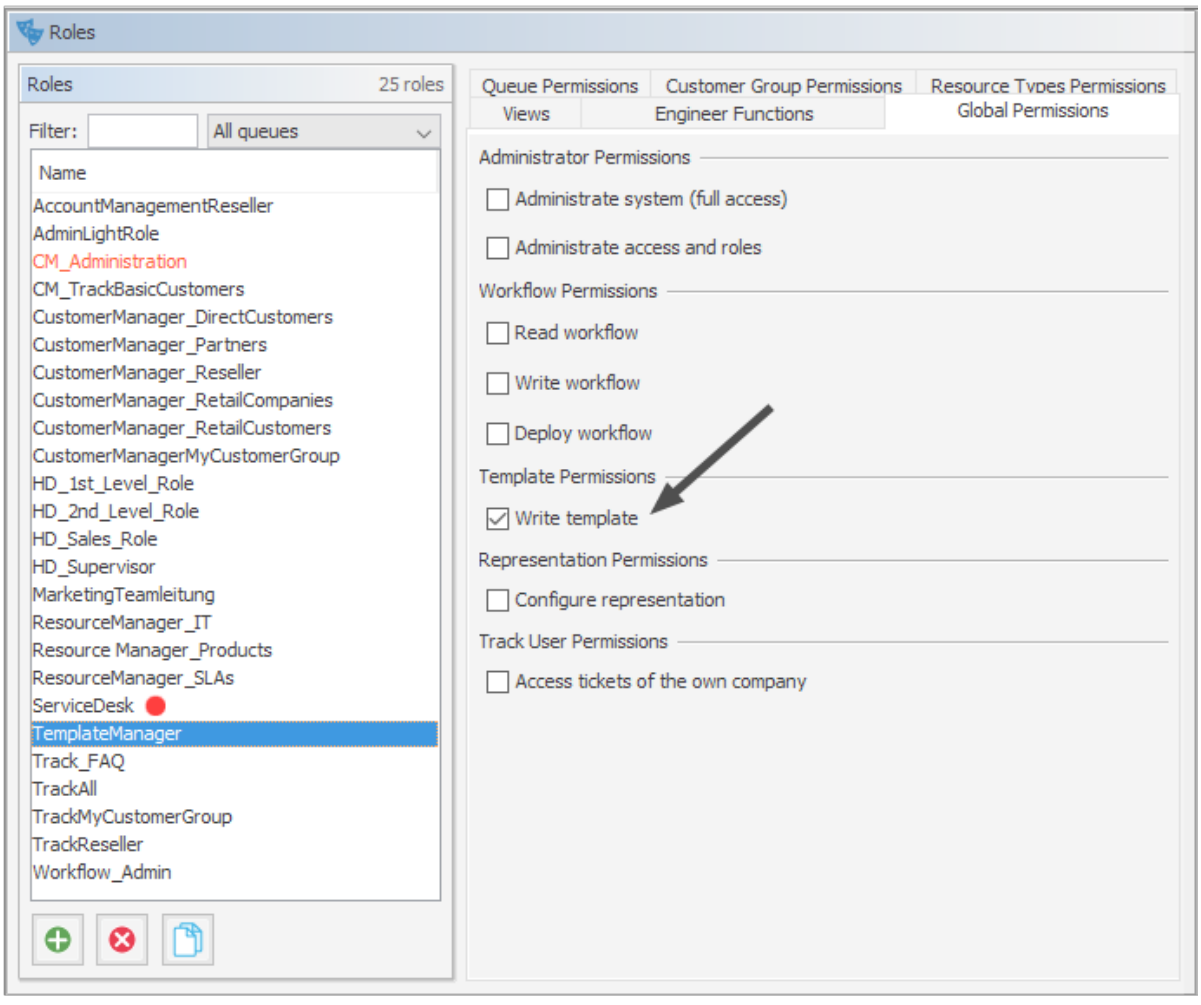


Figure 543: *ConSol CM Admin Tool - Access and Roles, Roles: Necessary template permissions*

We recommend to create a role (e.g., *TemplateManager*) that has only the permission Write template, with no queue permissions or other permissions. Every user who should have access to the Document Template Manager can be given this role. In this way, there is no merge between regular user permissions and document template permissions and you can grant and retrieve the document template permission in a very flexible way.

However, the permission *Write template* also grants access to the Text Template Manager (for text templates, see section The ConSol CM Text Template Manager).

F.12.2.6 Creating Microsoft Word Templates and Making Them Available

Creating Microsoft Word Templates

As a third step, you have to create the Microsoft Word templates. This is done using Microsoft Word. Please create `.doc` or `.docx` files as templates, **not** `.dot` files!

The OpenOffice-specific configuration is explained in section Using CM/Doc with OpenOffice. However, most of the configuration steps are identical or very similar for Microsoft Word and OpenOffice templates.

Creating the Template Object in the Template Library

As a fourth step, you have to fill in the requested data fields as *merge* fields in the Microsoft Word template, i.e., you create a CM/Doc template from your regular Microsoft Word document. This is done using the ConSol CM Web Client.

To perform steps three and four, log in to the Web Client and click on *Document templates* in the main menu to open the Document Template Manager.

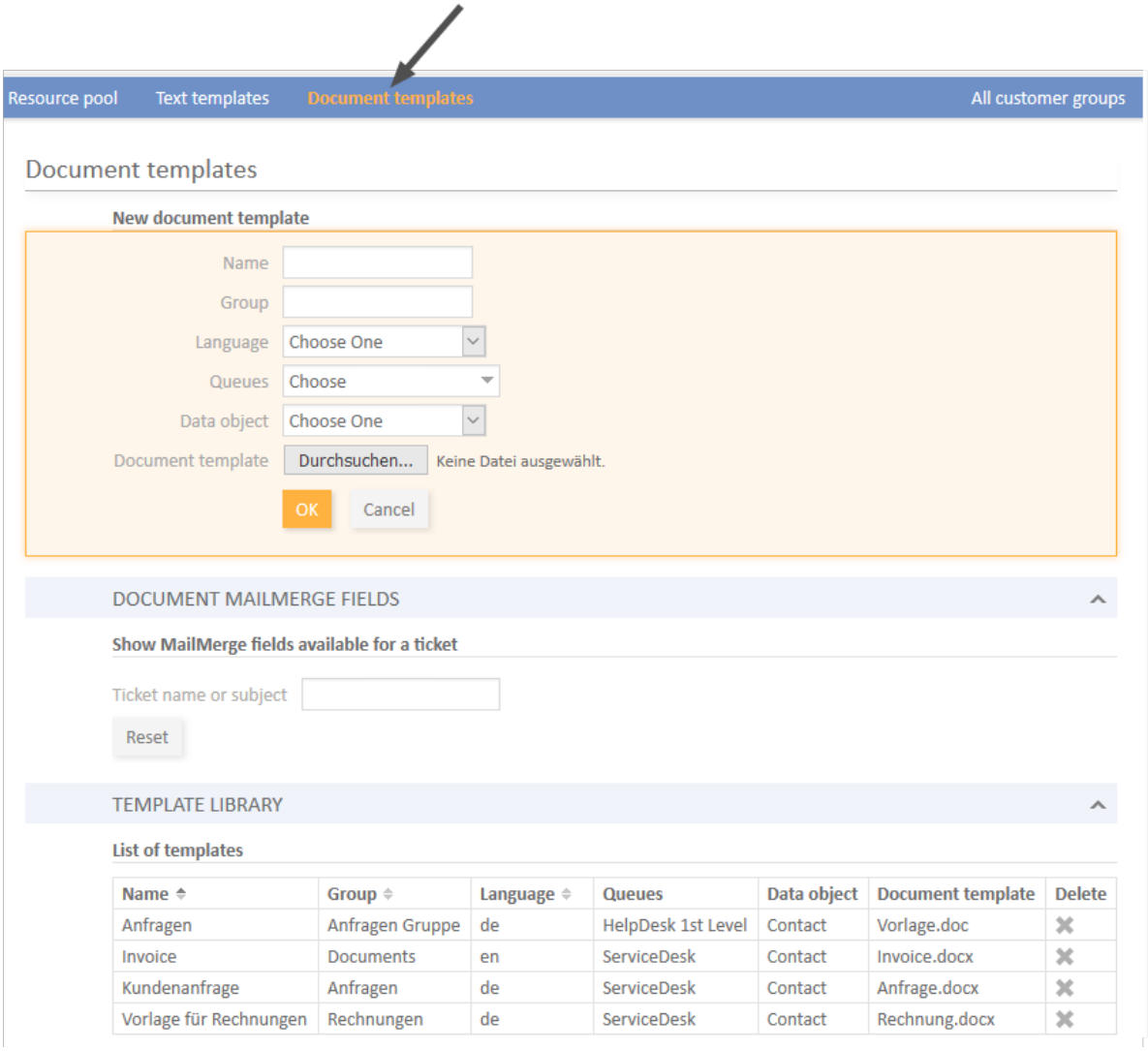


Figure 544: *ConSol CM Web Client - Document Template Manager*

The Document Template Manager is opened.

Enter the following data for each new template, then click on *OK*:

- **Name**
  The name of the (new) template. This will be displayed for engineers in the Web Client.
- **Group**
  The name of the template group. This does not have any technical implications but serves as an easy-to-use parameter to sort the templates in the template list in the Document Template Manager.
- **Language**
  Select the required language. All languages which have been activated in the Admin Tool are listed.

- **Queues**
  Select the queues for which the template should be available.
- **Data object**
  For the selection of the data object which should be used as reference object for customer data within the template. All data objects are listed, please see section Selection of the Data Object and Its Consequences for a detailed explanation about which data object you have to select.
- **Document template**
  Use the file browser to select the `.doc` or `.docs` file that should serve as a template in the file system.

Selection of the Data Object and Its Consequences

In the form where you have to fill in all fields for an Microsoft Word template, you also have to select a data object, i.e., a customer object (contact or company object) from your customer data model. If you are not familiar with the ConSol CM customer data model concept, please read the Customer Data Model Section first.

In the drop-down menu *Data object*, the data objects are listed in the following format:

- [localized name of the data object (localized name of the customer data model)]

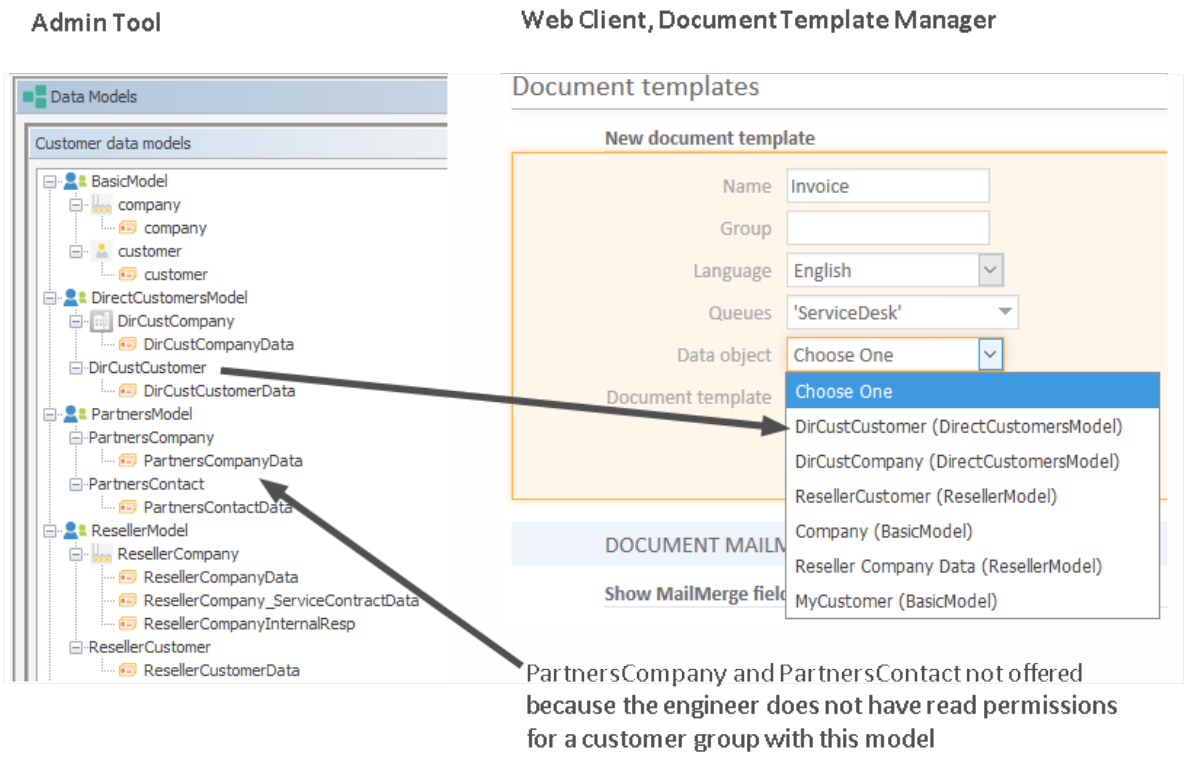
For a data object to be listed in the drop-down menu, the following conditions have to be met:

1. The customer data model with the respective data object has to be created in the Admin Tool.
2. The customer data model has to be assigned to at least one customer group.
3. The engineer who works on the document templates has to have at least **read** access to at least one customer group with the respective customer data model.

You have to select the data object which represents the main customer of the tickets. The Microsoft Word template will only be available in tickets which have main customers from a customer group with this customer data model. This means if you want to use the same Microsoft Word template text for two customer groups with two different customer data models, you have to create two ConSol CM Microsoft Word templates, one for each customer data model. You can, of course, use the same Microsoft Word template for two or more customer groups which all have the same customer data model.

If you try to assign a data object to an Microsoft Word template which is not compatible with the selected queue (because the selected data model is not assigned to the selected queue), you will get an error message.

Please be aware that it makes only sense to select an object at company level if the parameter *Company as customer* has been set for this customer data model.



Please be aware that if you leave the *Data object* field empty, the Microsoft Word template will theoretically be available for all customer groups, but there might be run-time errors if the required fields are not found when the template is loaded. ConSol CM will display a message that field content is missing and the fields will be filled with the string 'n/a'. However, we recommend that you always select a data object to keep the system in good working order.

How to Control in Which Tickets the Microsoft Word Template Will Be Offered

If the Microsoft Word template is offered in a certain ticket (see section Using Microsoft Word Templates from within the Web Client below) depends on two parameters:

- **Queue**
  The template is only listed for tickets in the selected queue(s).
- **Customer data model**
  The template is only listed in tickets which have, as main customer, a customer object which is based on the selected data object.

Example (we anticipate the use of Microsoft Word templates in the Web Client, for details see section Using Microsoft Word Templates from within the Web Client):

- First, the Microsoft Word template is defined for the queue *ServiceDesk* and for the data object *ResellerCustomer* (first figure).
- In the Web Client, the template **can** be used in a ticket which is in the queue *ServiceDesk* **and** which has a main customer of the *ResellerCustomer* data object type (second figure).

- In the Web Client, the template **cannot** be used in a ticket which is in the queue *ServiceDesk* and which has a main customer of a data object which is **not** of the *ResellerCustomer* data object type (third figure).
- In the Web Client, the template cannot be used in all tickets which are **not** in the queue *ServiceDesk* (not shown).

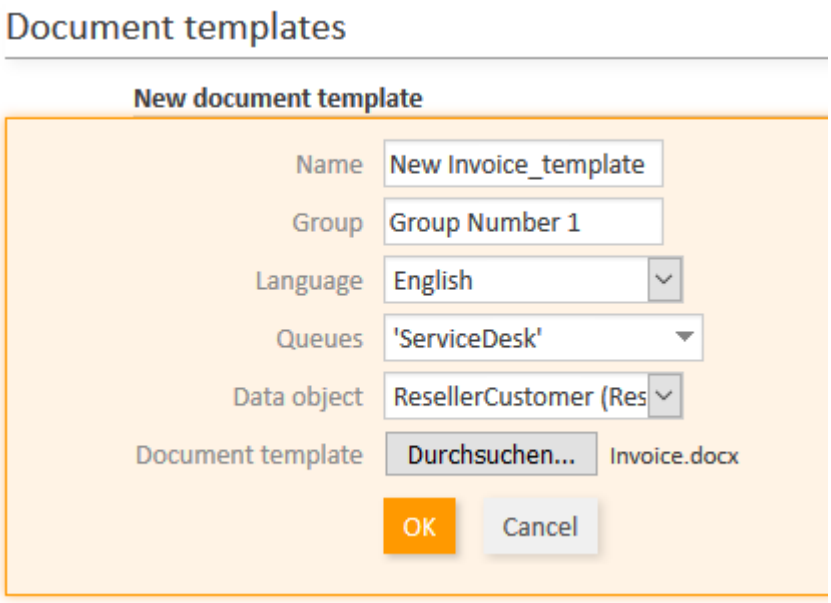


Figure 545: *ConSol CM Web Client - Definition of a Microsoft Word template for one queue and for a certain data object*

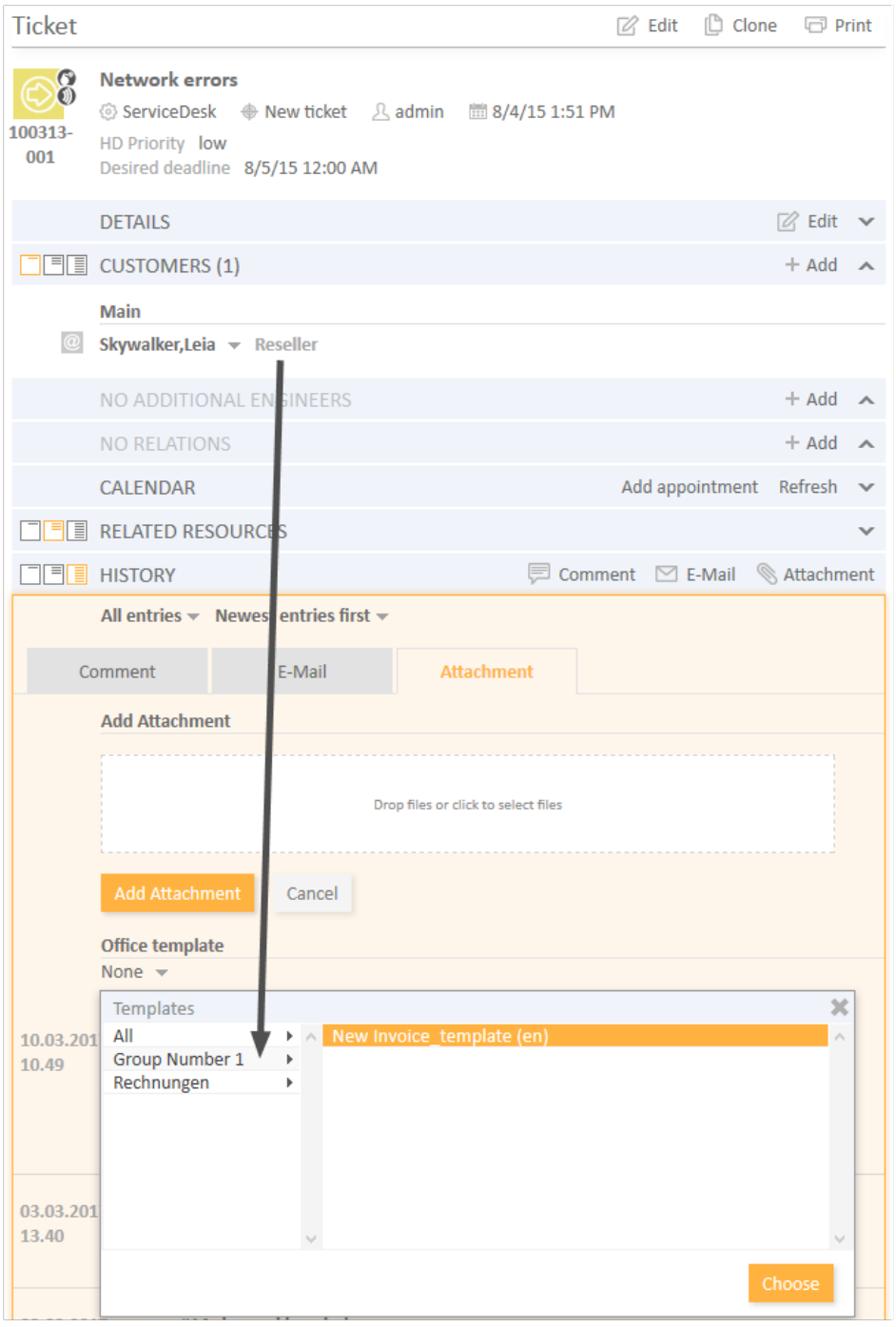


Figure 546: *ConSol CM Web Client - Microsoft Word template offered because of queue and customer group*

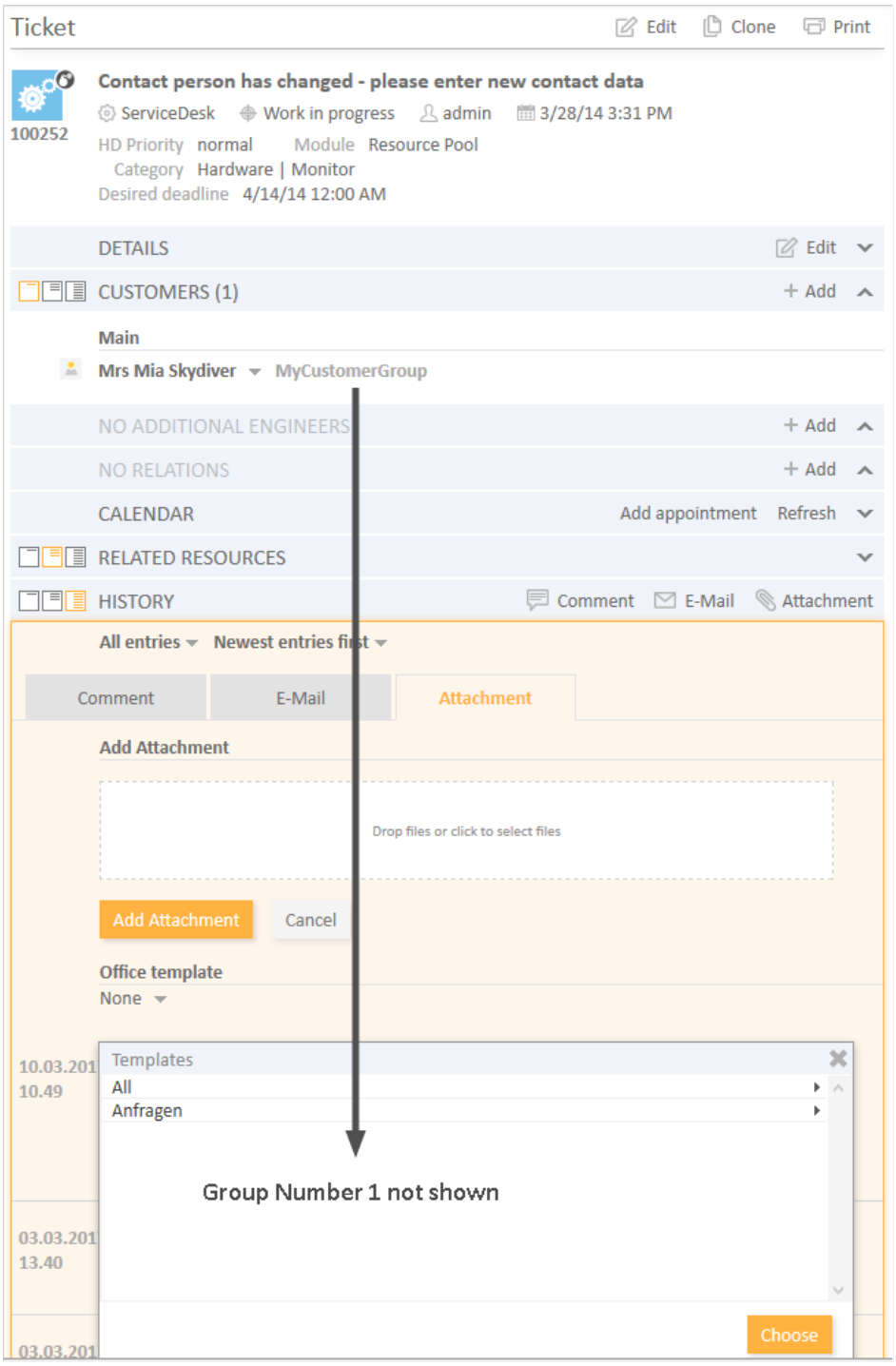


Figure 547: *ConSol CM Web Client - Microsoft Word template NOT offered because of queue and customer group*

Creating and Editing the Required Microsoft Word Document

When you have filled in all fields and clicked *OK*, the new template appears in the *Template library*, *List of templates*. After entering the new template data, you can perform the following step directly or you can click on the name of the template file (in the *Document template* column) to download and edit the template.

In the next step, *MailMerge* fields, which represent the fields of ticket and customer data, can be added to the template. Select a ticket which has all the requested fields by using the *Document MailMerge fields* section. Enter the ticket name or subject in the field under *Show MailMerge fields available for a ticket* and select the correct one from the search result list.

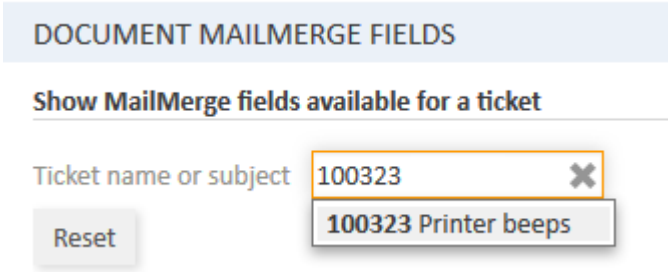


Figure 548: *ConSol CM Web Client - Selection of a ticket to see all required data fields*

All available *MailMerge* fields are displayed.

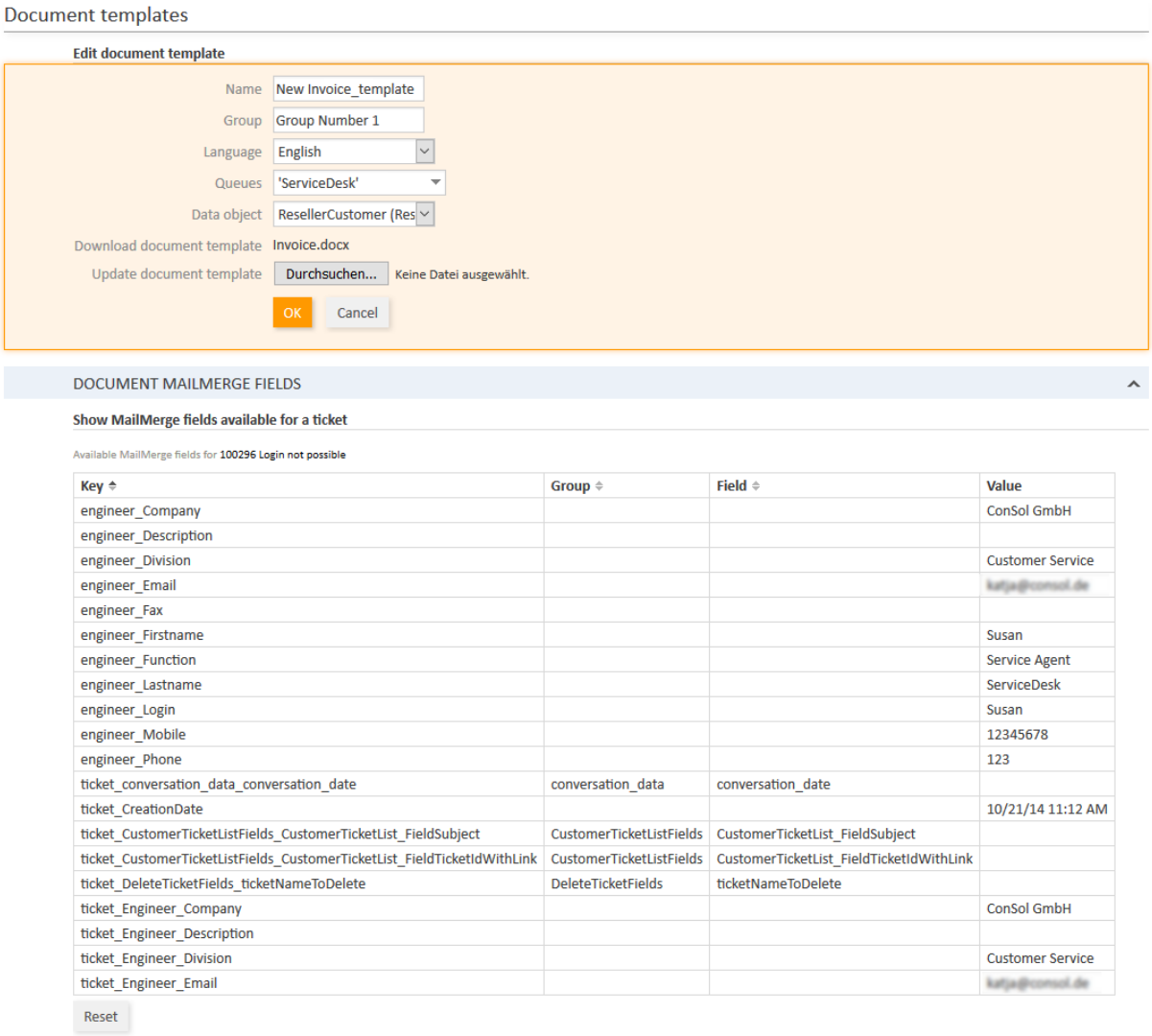


Figure 549: *ConSol CM Web Client - Show available MailMerge fields*

The list contains the following columns:

- **Key**
  The name of the field (this is the one you will have to use for the MergeField later on)
- **Group**
  This is set for customer data and for ticket data. For customer data, the customer field group is shown (here, only the customer field groups from the selected data object are offered). For ticket data, the ticket field group is shown.
- **Field**
  This is set for customer data and for ticket data. For customer data, the customer field name is shown (here, only the customer fields from the selected data object are offered). For ticket data, the ticket field name is shown.

- **Value**
  The value of this field in the selected ticket. You do not have to use this in the document template.

Download and open the Microsoft Word template from one of the following locations. In both cases, the technical file name of the template is shown:

- from the *Edit document template* section (at *Download document template*)
- from the list in the *Template Library* (*Document template* column)

In this document, go to the position where you want to insert the first field (in our example this will be the customer name). Use *Insert -> Quick parts -> Field* to insert the *MergeField*. Copy and paste the key of the MailMerge field you need into the respective field (*Field properties*, *Field name*) in Microsoft Word:

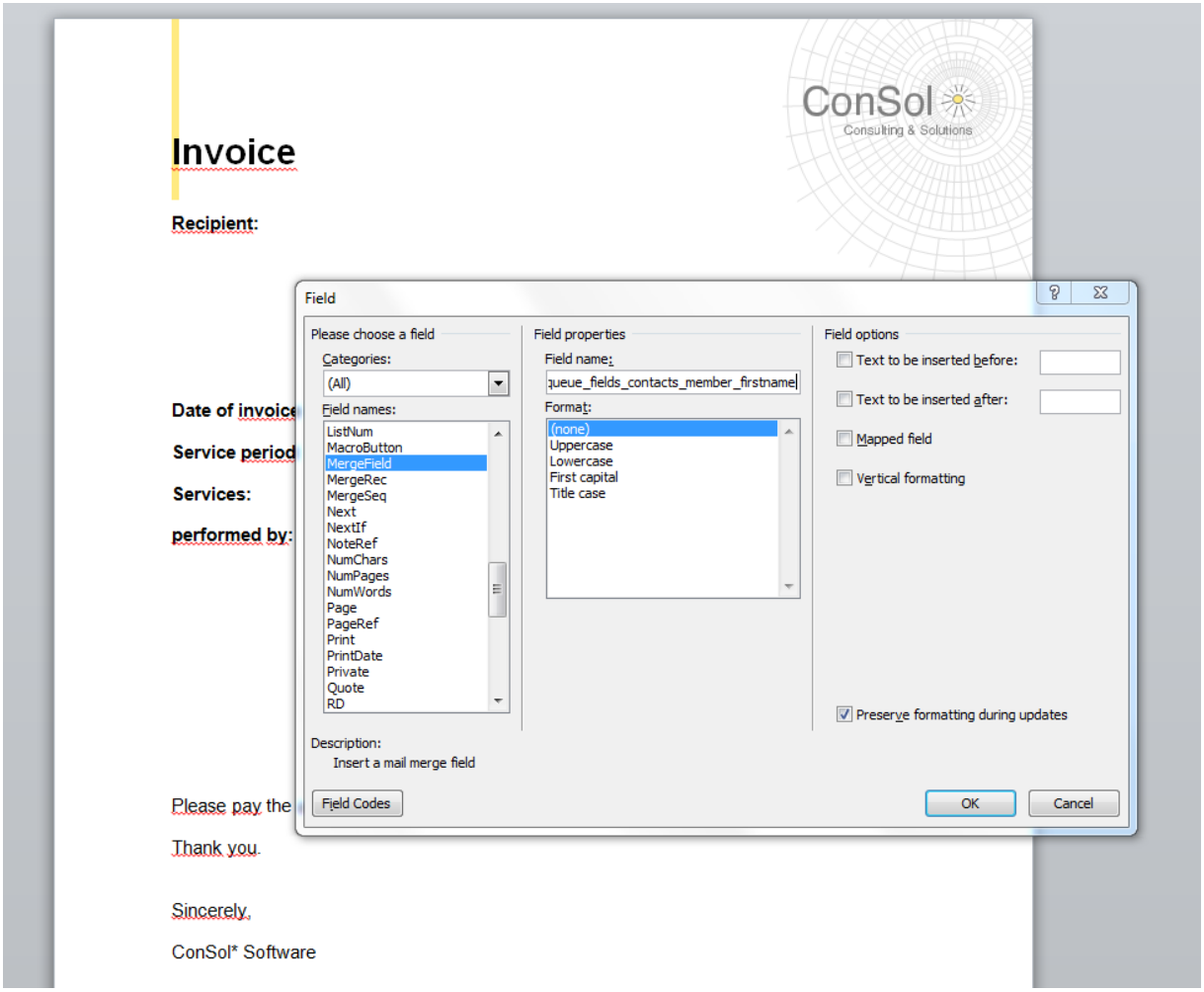


Figure 550: *Microsoft Word - Insert MergeField into document*

Do this with all fields you would like to have pre-filled when the Microsoft Word template is opened. In the end, your template might look like the example in the following figure.

You can use all fields which are offered in the ticket, i.e.:

- **Company fields**
  Customer fields from the selected data object if this is a company, or from the company of the selected data object if the selected object is a contact. The keys start with *ticket_queue_fields_* and have a customer field group name as *Group*.
- **Contact fields**
  Customer fields from the selected data object if it is a contact, or from the contact of the selected data object if the selected data object is a company. The keys start with *ticket_queue_fields_* and have a customer field group name as *Group*.
- **Ticket data fields**
  Ticket fields from the ticket. The respective keys start with *ticket_* and have a ticket field group name as *Group*.
- **Engineer data**
  - Data concerning the current engineer of the ticket. The respective keys start with *ticket_Engineer_* and do not have *Group* and *Field* names.
  - Data from the engineer who is currently logged in. This must **not** be confused with the engineer of the ticket! The respective fields start with *engineer_* and do not have *Group* and *Field* names.

Figure 551: *Microsoft Word example template*

Save the Word document in the local file system and then upload it using the field *Update document template*. This will store the template in the Document Template Manager.

Details about ConSol CM Keys for MailMerge Fields

For customer data, there are always two keys for the same value, i.e., two keys which in the end retrieve the value from the same customer field. You can always use either one - there is no difference as far as the behavior in the templates is concerned.

The two keys are:

1. A generic field, e.g., ticket_queue_fields_contacts_member_companyReferenceField_**company_name**
2. A field which comes from the specific customer data model according to the following syntax: xxx_[data object]_[customer field group]_[customer field], e.g., ticket_queue_fields_contacts_member_companyReferenceField_ResellerCompany_ResellerCompanyData**_company_name**

F.12.2.7 Using Microsoft Word Templates from within the Web Client

Creating a New Microsoft Word Attachment Using a Document Template

When Microsoft Word templates are available for a queue, an engineer can use them by clicking on *Attachment* in the *History* section of a ticket and by selecting the requested template. Microsoft Word is started (if it is not already open) and a document based on the selected template is created. The document is opened, with all values/parameters filled-in at the correct positions. This might look like the example in the following figures.

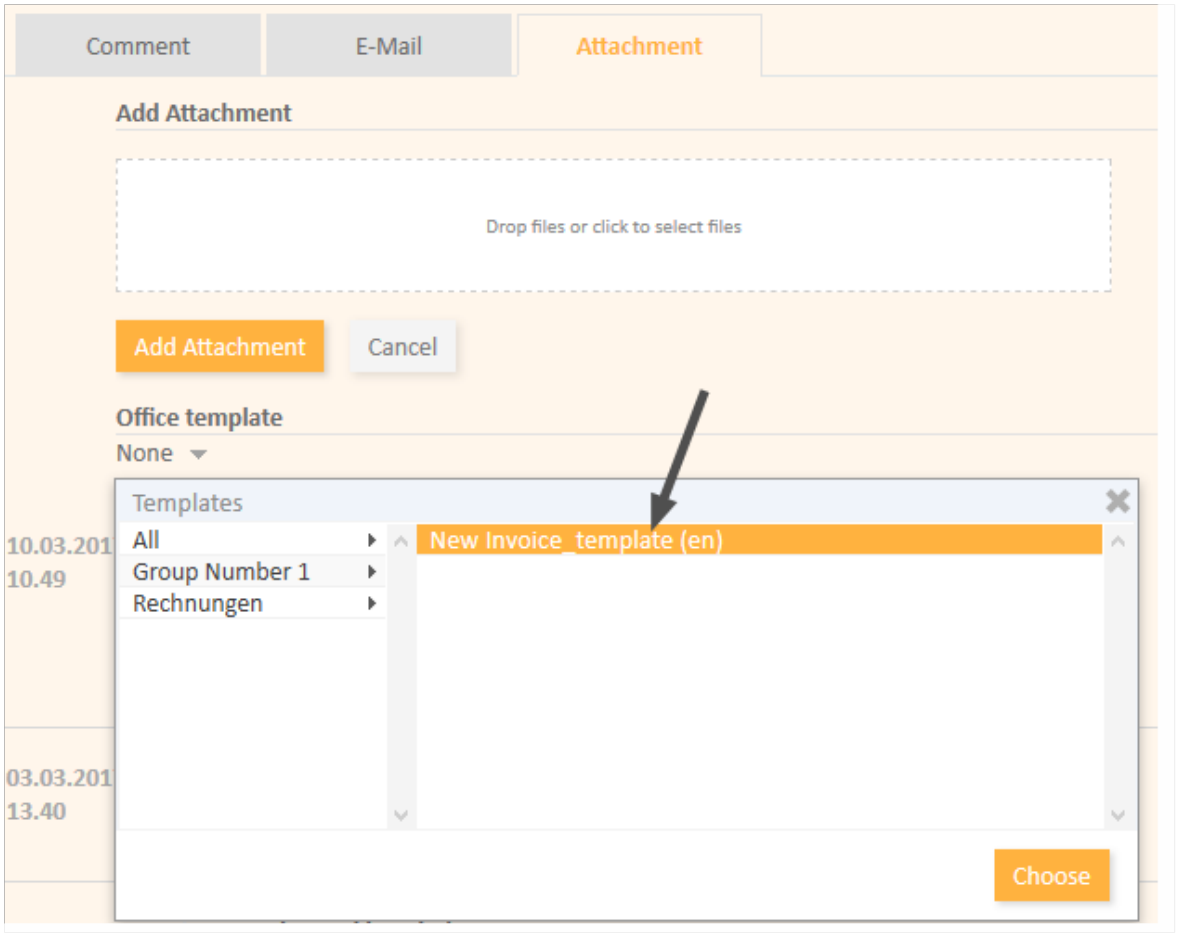
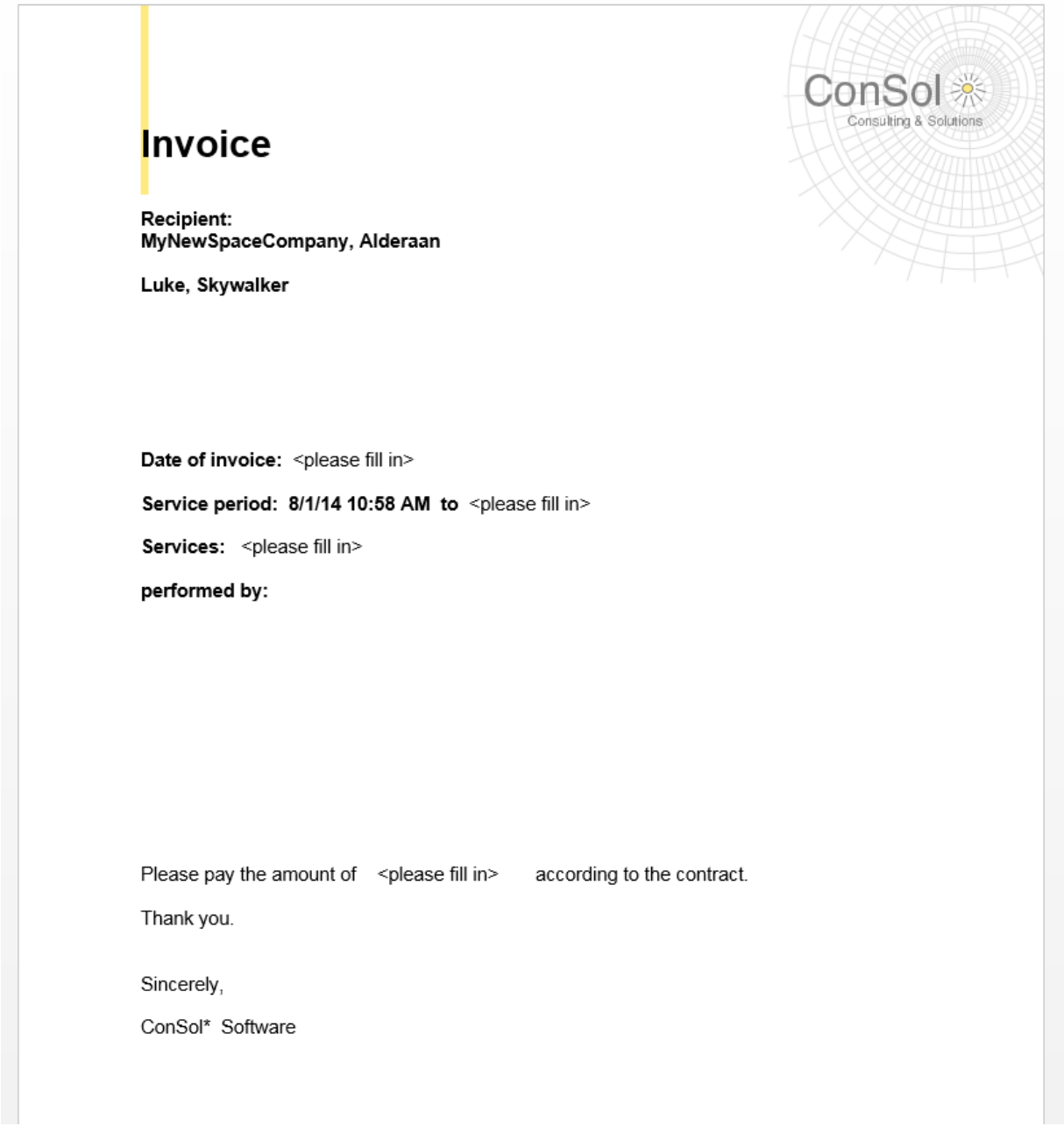


Figure 552: *ConSol CM Web Client - Microsoft Word template available as attachment*



Invoice

Recipient:
MyNewSpaceCompany, Alderaan

Luke, Skywalker

Date of invoice: <please fill in>

Service period: 8/1/14 10:58 AM to <please fill in>

Services: <please fill in>

performed by:

Please pay the amount of <please fill in> according to the contract.

Thank you.

Sincerely,

ConSol* Software

Figure 553: *Microsoft Word example document*

The engineer can then edit the document, if required, and save it. This will automatically attach the new version of the document to the ticket.

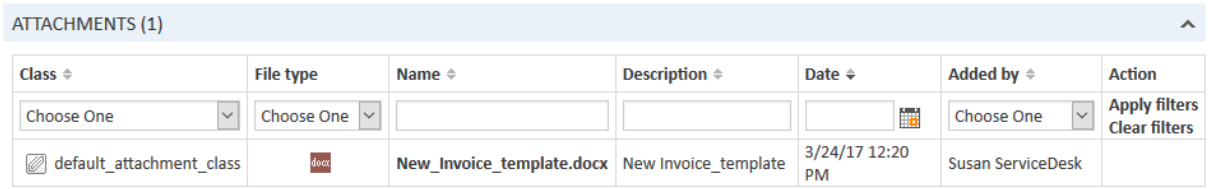


Figure 554: *ConSol CM Web Client - Newly edited Word document as attachment at the ticket*

From here on, working with the Word document is exactly the same as working with regular Word (.doc, .docx) attachments. See next section.

Working with Existing Microsoft Word Attachments

An engineer can also open a Microsoft Word document which has been attached to the ticket. As an engineer, click on the attachment name. This will open the file in Microsoft Word. Edit the document and save it. A new version of the document will be attached to the ticket automatically.

F.12.2.8 Using CM/Doc with OpenOffice

To work with CM/Doc using OpenOffice, perform the following steps:

Step 1: Activate CM/Doc in Your CM System

See section Configuring the ConSol CM System for CM/Doc.

Step 2: Create the Role with the CM/Doc Access Permissions

See section Creating an Engineer Role with Permissions for the Document Template Manager.

Step 3: Create / Set System Properties

Set the following system property / properties in section *System Properties*, navigation group *System*:

- The path to the OpenOffice main program directory: cmweb-server-adapter-.cmoffice.oo.path.<NUMBER>
  These properties are numbered (starting with 0) so that different paths can be used to accommodate different OpenOffice installations on varying operating systems and different system configurations. So a possible list of properties and values for the path configuration would be:
  - `cmoffice.oo.path.0`: C:\Program Files (x86)\openoffice\program
  - `cmoffice.oo.path.1`: /usr/lib/openoffice/program
  - `cmoffice.oo.path.2`: /usr/lib64/openoffice/program

The handling of OpenOffice documents in the ConSol CM Web Client is identical to the handling of Microsoft Word documents. When preparing a document template with ConSol CM data in OpenOffice the basic handling also mirrors the procedure in Microsoft Office. So, generally, the detailed explanation in the sections above (starting in section Creating Microsoft Word Templates and Making Them Available) applies here, too. When you want to add a field to the OpenOffice template, the dialog *Fields* can be opened by selecting the menu entry *Insert -> Fields -> Other* and selecting the tab *Variables*.

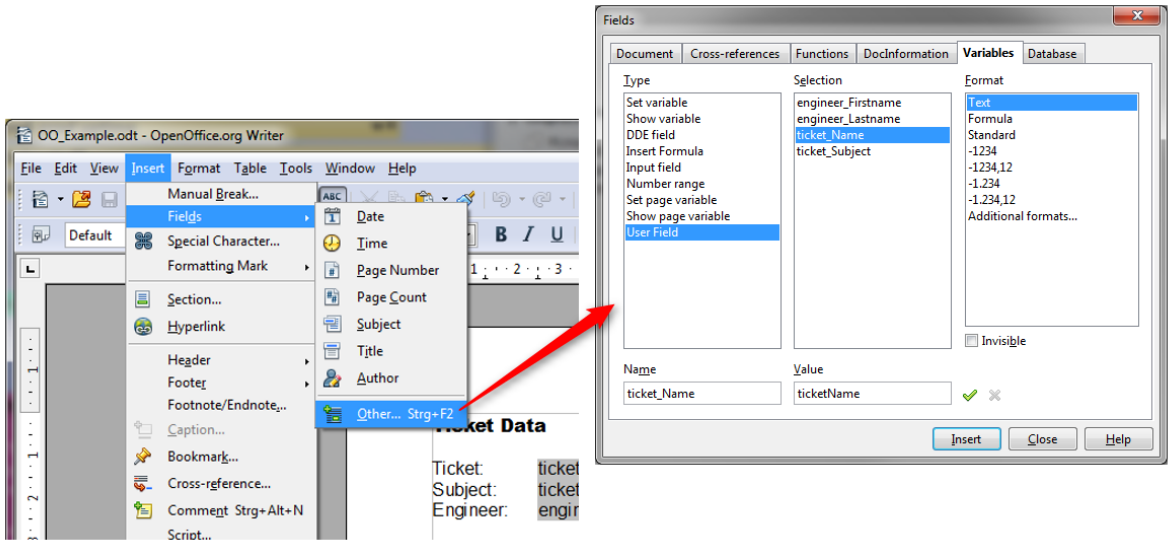


Figure 555: *Creating a document template using OpenOffice*

This tab allows you to add a ConSol CM field. To do so, select *User Field* as a *Type,* and *Text* as a *Format*. In the next step you enter the field name (from the *Key* column in the *Document MailMerge fields* section) in the *Name* field below the *Type*. Add a useful *Value* for recognizing the field in the document. Click the green checkmark to the right of the field *Value*, so that the field is shown under *Selection* in the middle, and make sure it is selected there. Click the *Insert* button on the bottom right side of the dialog to insert the field into the document at the current cursor position. This has to be repeated for every field that you want to use in the document.

F.12.2.9 Configuring the Saving Strategy for CM/Doc

Up to ConSol CM version 6.10.5.3, a new Word / OpenOffice attachment is saved and attached to the ticket with each SAVE operation from within the Word / OpenOffice document. This provides a well maintained history of all attached Word / OpenOffice documents but might - on the other hand - lead to a great number of attachments which might increase the ticket size considerably.

Thus, starting with CM version 6.10.5.4, the saving strategy can be configured using the CM system property cmweb-server-adapter, cmoffice.strict.versioning.enabled, a boolean value, default “true”. The two possible values mean:

- **true** (default, behavior up to 6.10.5.3)
  A new version of the Word/ OpenOffice document is saved and attached to the ticket with each SAVE operation from within the document.
- **false**
  A SAVE operation from within the Word / OpenOffice document overwrites the existing attachment with the same name as long as Word / OpenOffice has not been restarted in the meantime.

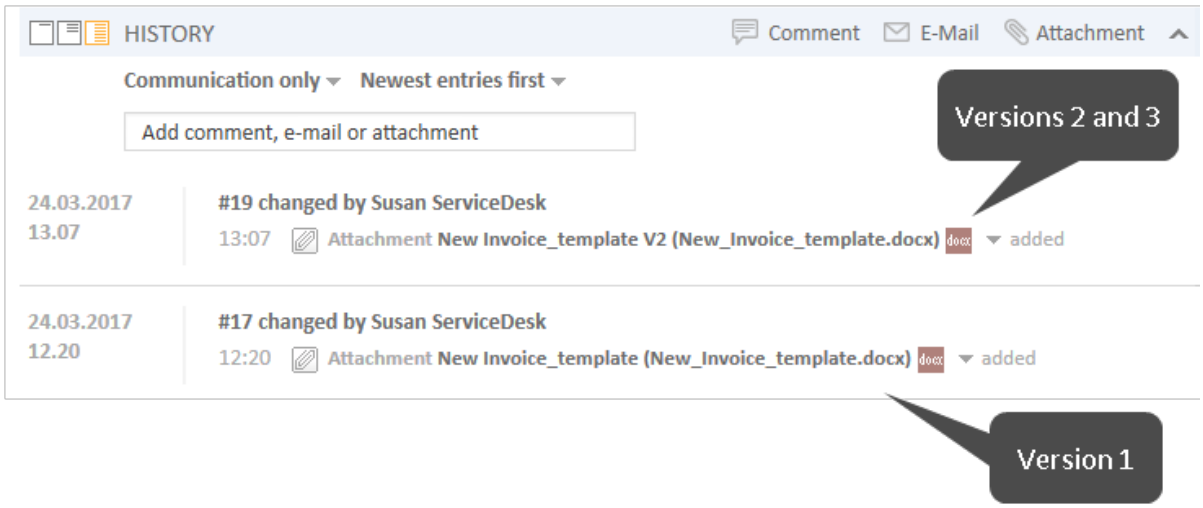


Figure 556: *ConSol CM Web Client, documents attached to a ticket, cmoffice.strict.versioning.enabled=false*

F.13 Time Booking Using ConSol CM

This chapter discusses the following:

F.13.1 General Introduction to Time Booking Using ConSol CM

In ConSol CM, working hours can be booked to tickets. There are two booking modes:

1. Manual bookings, see section Manual Time Bookings
2. Automatic bookings (available in ConSol CM versions 6.9.4.2 and up), see section Automatic Time Bookings (Available in CM Versions 6.9.4.2 and Up)

F.13.2 Manual Time Bookings

In ConSol CM, an engineer can book working hours to a ticket. Those working hours can then be reported.

With manual time booking, working hours are always booked on projects which have to be assigned to one or more queues. For example, if your company plans to perform a migration from Windows 8 to Windows 10 clients and all the working hours should be registered for this migration project, the ConSol CM administrator has to create a migration project and assign it to all queues where tasks for this project might be performed. Then engineers can book their times on the project and can see their own reports for the project. Additionally, a report over all time bookings, of all engineers, may be implemented using the DWH (Data Warehouse, see section Data Warehouse (DWH) Management).

F.13.2.1 Configuration of Manual Time Booking Using the Admin Tool

In order to enable engineers to book working hours on projects the ConSol CM administrator has to perform two steps using the Admin Tool:

1. Create the projects on the navigation item *Projects* (navigation group *Global Configuration*), see section Projects.
2. Assign one or more projects to the desired queues within the Queue Administration.

In the following example, three projects are created. Engineers in the *HelpDesk_1st_Level* queue should be able to book working hours on two of them. Thus, the two projects have to be assigned to the *HelpDesk_1st_Level* queue.

You reach the following screen by selecting the navigation group *Global Configuration*, navigation item *Projects*.

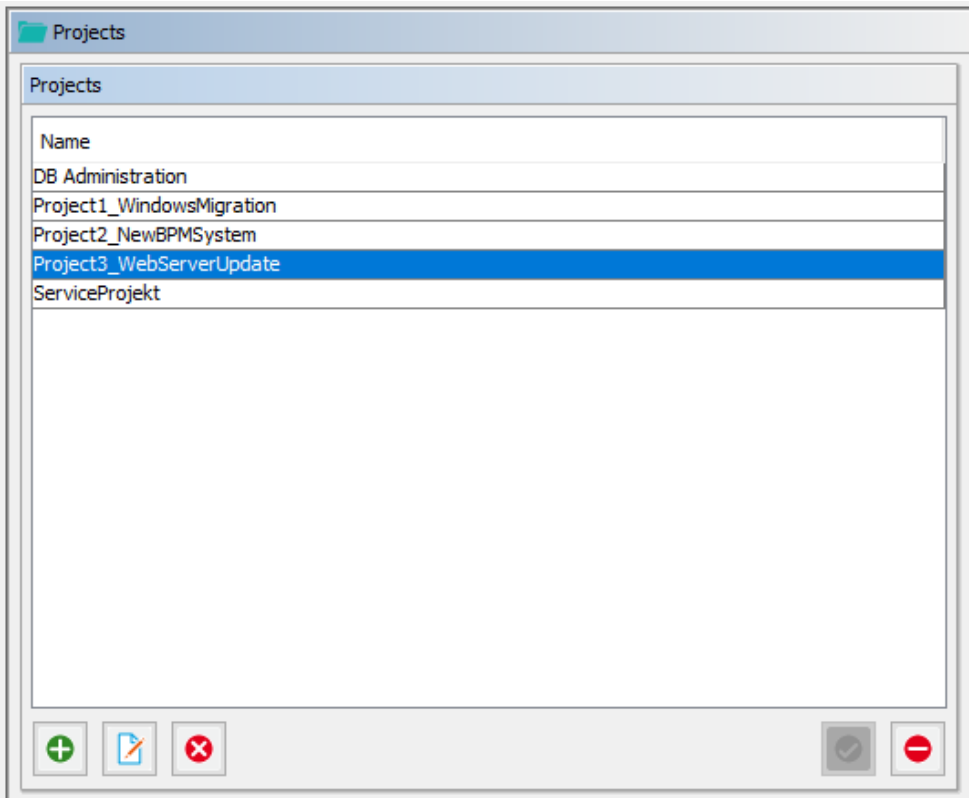


Figure 557: *ConSol CM Admin Tool - Global Configuration, Projects: Management of projects*

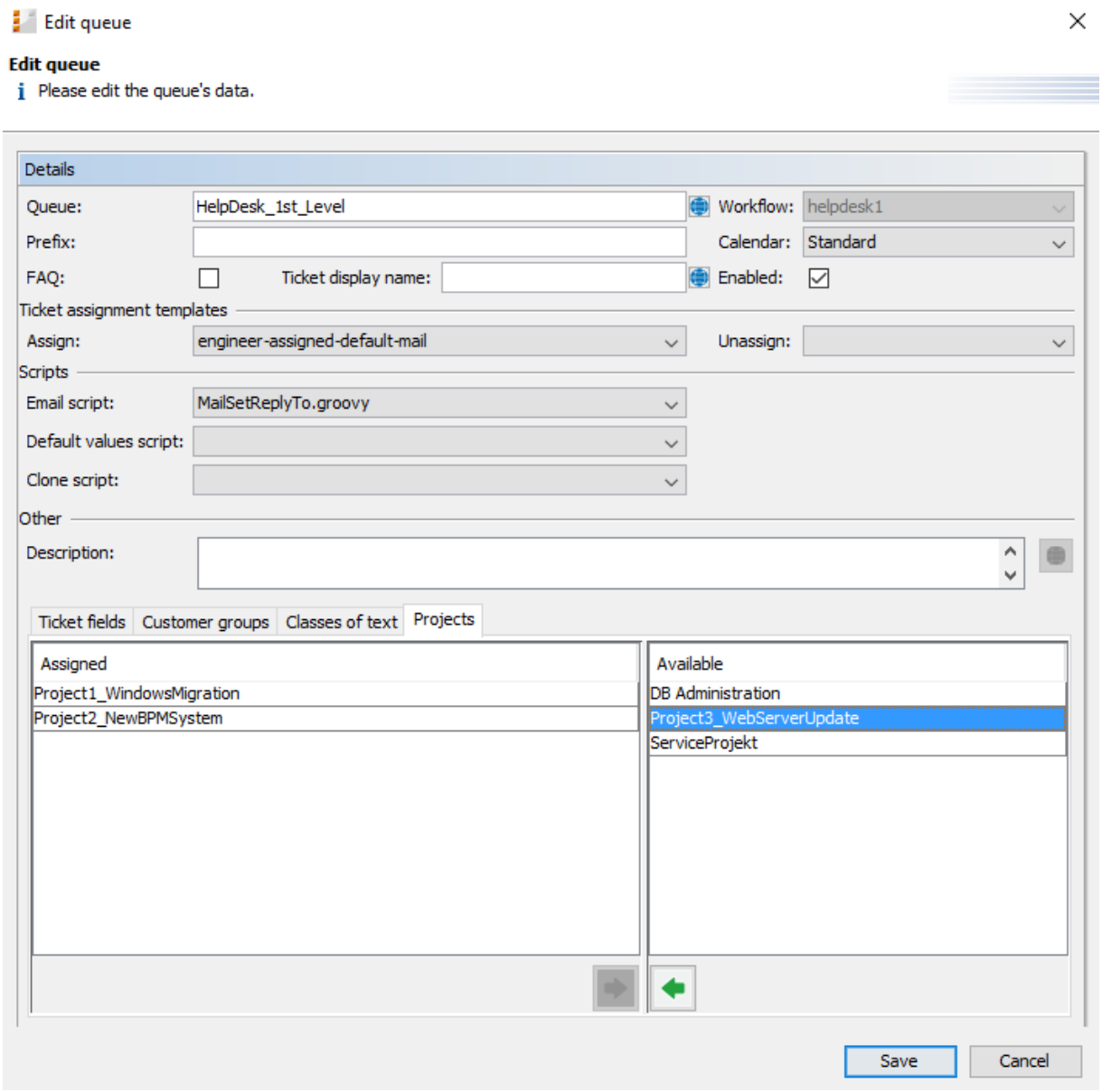


Figure 558: *ConSol CM Admin Tool - Global Configuration, Queues: Assigning projects to a queue*

F.13.2.2 Manual Time Booking from a User's Point of View (Web Client)

Please see the *ConSol CM User Manual* for a detailed explanation of the time booking feature. Here, only a brief overview is provided.

The user (engineer) can book working hours on a ticket using two different modes:

1. Using the *Time booking* section in a **ticket** to book working hours directly on this ticket.
2. Manual time bookings on the engineer profile page.

Using the *Time booking* Section in a Ticket to Book Working Hours Directly on This Ticket

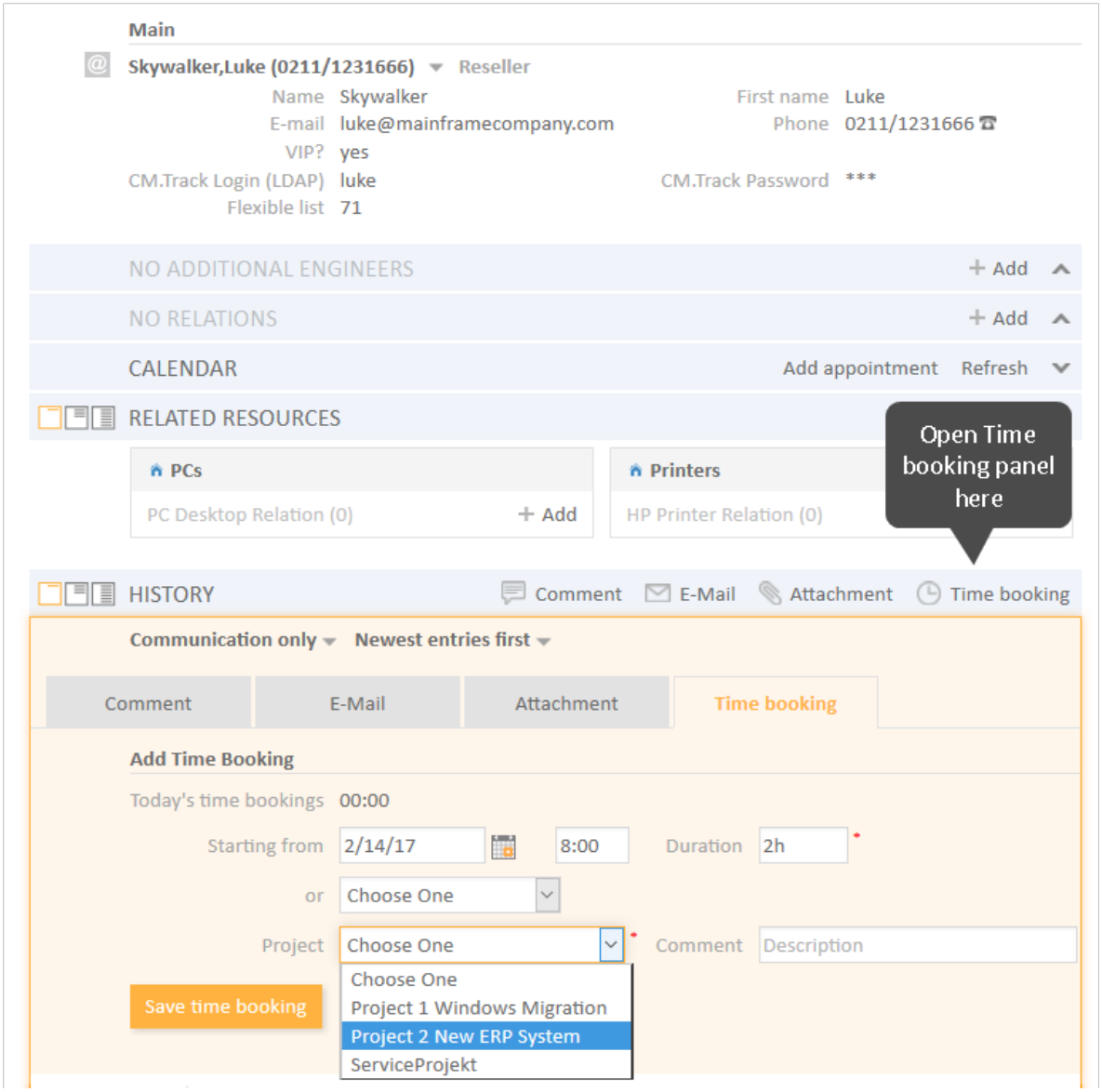


Figure 559: *ConSol CM Web Client - Time booking in a ticket*

Manual Time Bookings on the Engineer Profile Page

Using the *Time booking* section on the **engineer profile page** to book working hours on a ticket. Only tickets where the engineer has performed certain activities and tickets owned by the engineer can be selected. A project also has to be selected from the list.

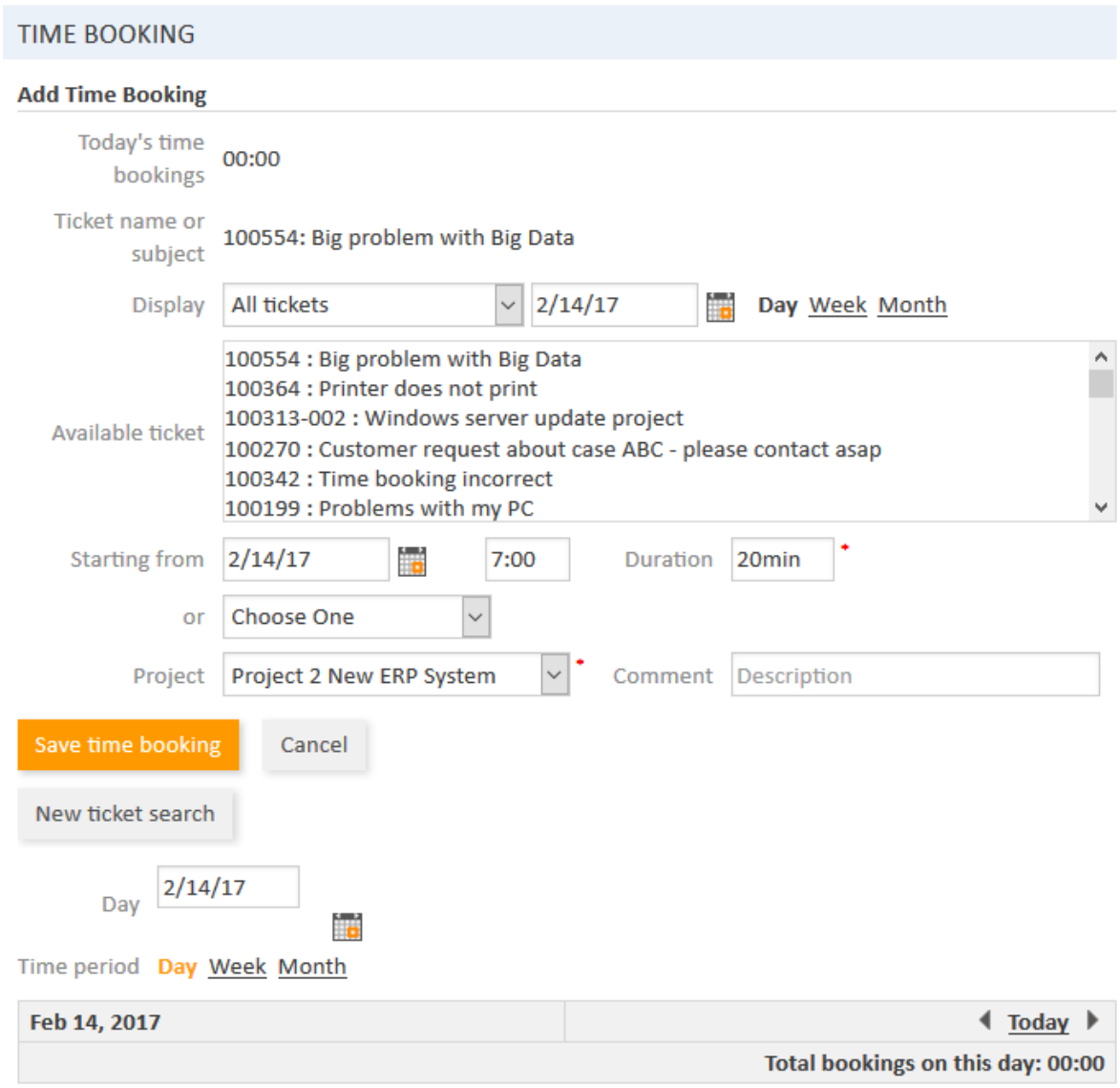


Figure 560: *ConSol CM Web Client - Time booking on the engineer profile*

Engineers can see a list of their time bookings on the engineer profile page. An example is shown in the following figure.

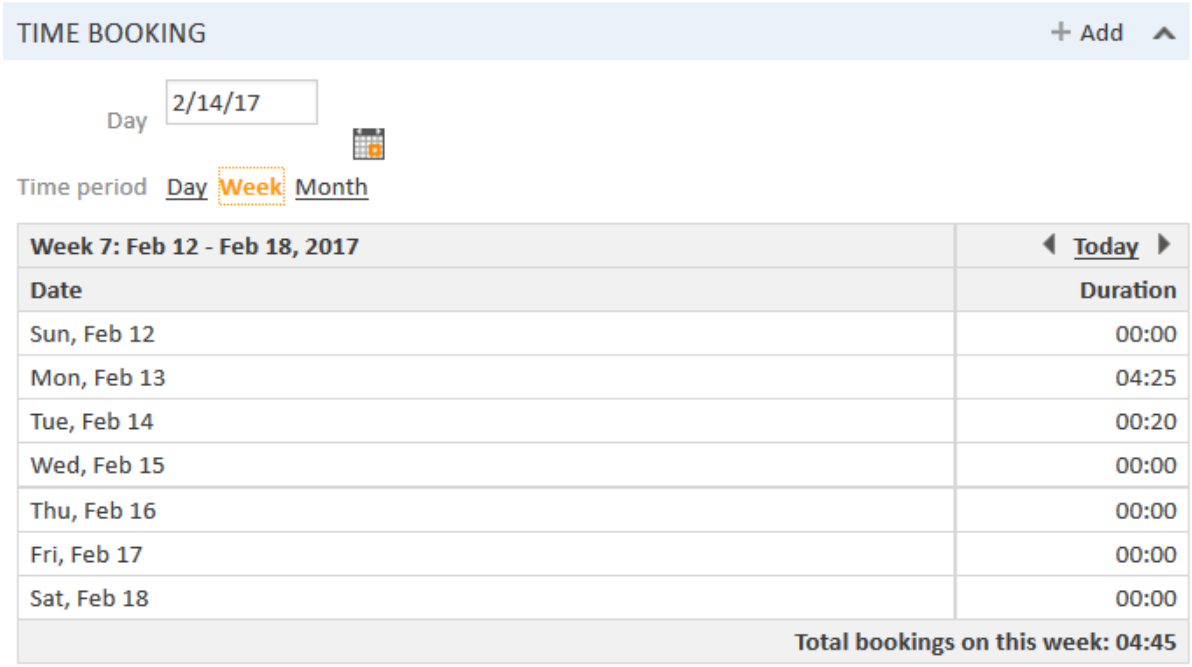


Figure 561: *ConSol CM Web Client - Time booking report on the engineer profile*

As an engineer, you can select whether you would like to see the bookings for the current day, week, or month. In the *Day* view, the projects are indicated, in the *Week* and *Month* view, only the sum of the booked times per day/week is indicated.

F.13.3 Automatic Time Bookings (Available in CM Versions 6.9.4.2 and Up)

F.13.3.1 Introduction to Automatic Time Booking

ConSol CM can be configured in a way that working hours are tracked and booked on tickets automatically. These bookings always refer to tickets and cannot be linked to projects.

The following times are registered:

- **Duration of work with the Rich Text Editor**
  Started when the Rich Text Editor is opened and stopped when the *Add* button is clicked.
- **Duration of creating a ticket**
  Started when *Create ticket* is selected and stopped when the *Create* button is clicked.

Time booking is suspended when a ticket is transferred to the workspace and resumed when the ticket is brought back to the active work.

No times are booked on the ticket when ...

- an operation is canceled
- the engineer logs out manually
- the engineer session is ended automatically

Times are always booked with minute-precision and are always rounded up to the next full minute.

F.13.3.2 Configuration of Automatic Time Booking

In order to enable this functionality in your ConSol CM system, set the system property cmweb-server-adapter, automatic.booking.enabled to “true”.

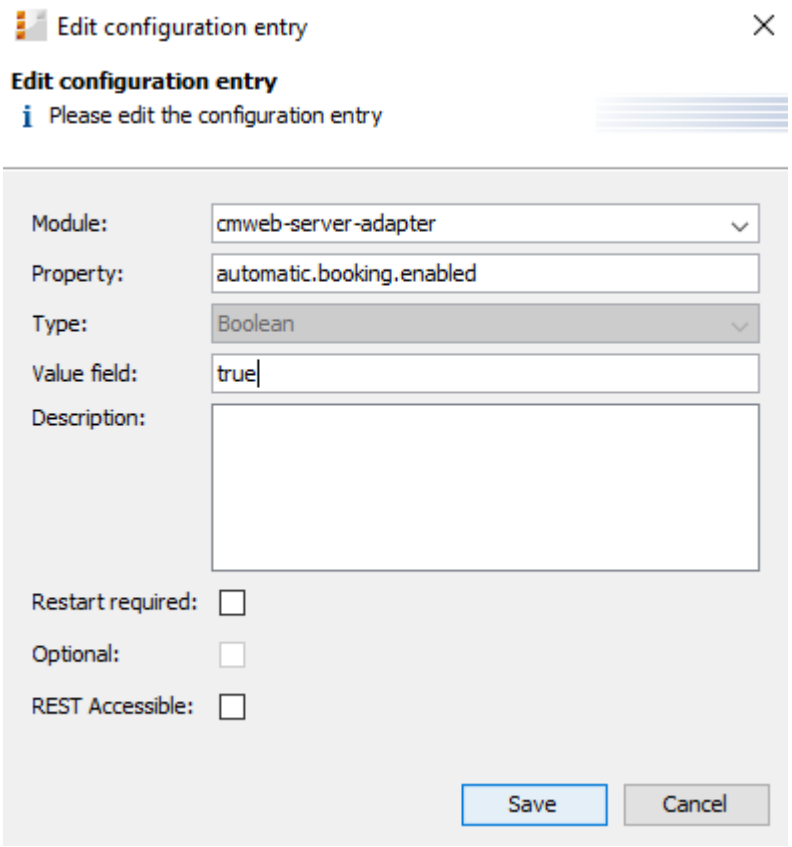


Figure 562: *ConSol CM Admin Tool - Set system property to switch on automatic time booking*

F.13.3.3 Automatic Time Bookings from a User's Point of View (Web Client)

The engineer does not have to do anything in particular to work with automatic time booking. When he enters a comment in a ticket or creates a ticket, the time is booked on this ticket automatically and can be seen in the **time booking report on the engineer profile page**. An example is shown in the following figure.

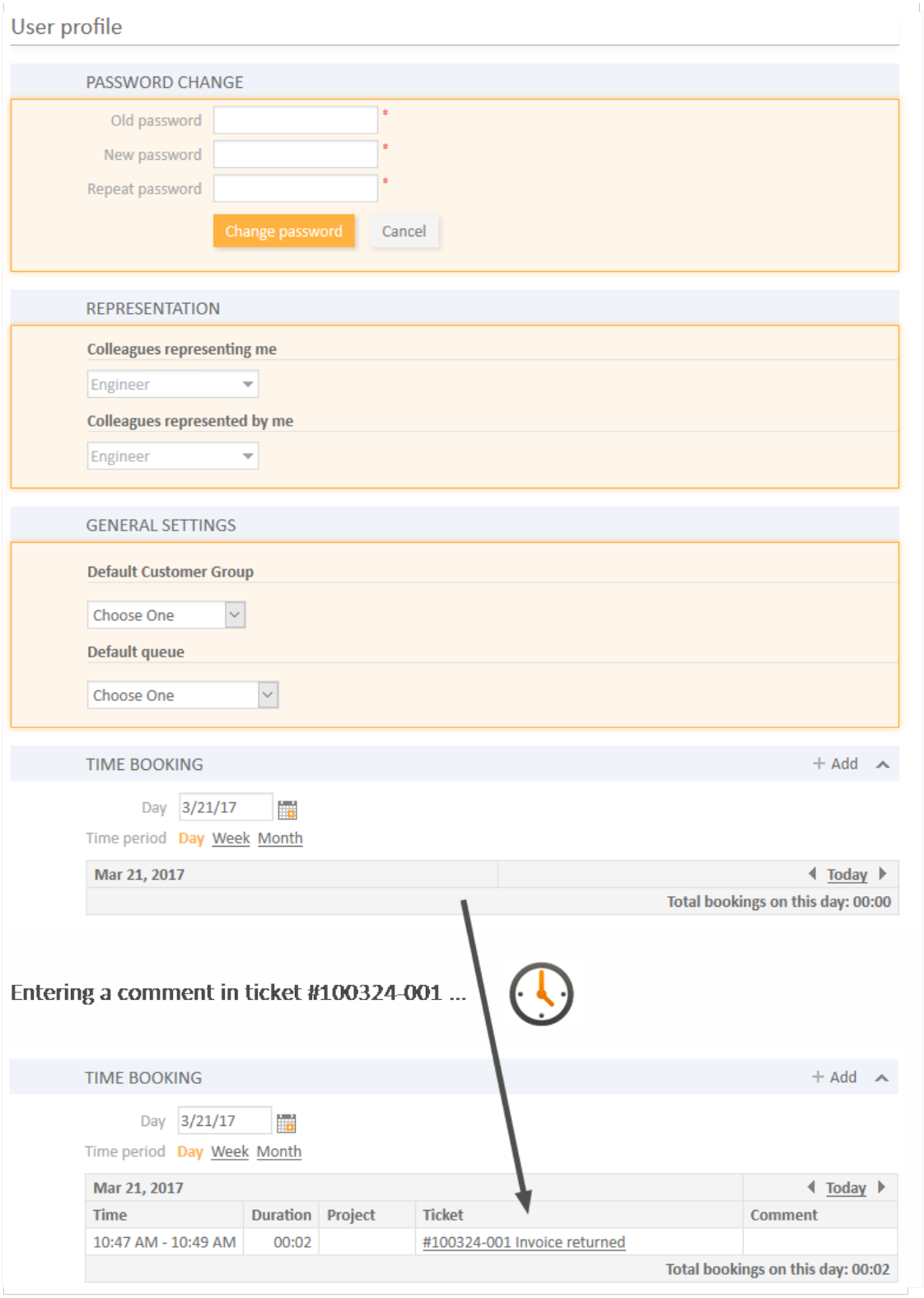


Figure 563: *ConSol CM Web Client - Automatically booked time in time booking report on engineer profile page*

F.13.4 DWH Reports

If your company would like to be able to report at a more detailed level, the DWH provides a good basis. Reports can be developed that use the DWH data and provide, e.g., the times booked on a certain project by all engineers.

F.13.5 Page Customization for Time Booking

If the time booking feature is not required, you can turn off the feature by using the *Page Customization*, see section Page Customization for details.

The following two parameters are relevant in this context:

- timeBookingSection
- timeBookingFeature

F.13.6 Using Time Booking Data in Scripts

Starting with ConSol CM version 6.11, the CM API contains a method to retrieve the time intervals booked on a ticket. Use the following method of the object `workflowApi`.

```
Set<TimeBooking> getBookings(Ticket pTicket)
```

You can use this method in Admin Tool scripts and in workflow scripts. Please note that in Admin Tool scripts, the workflow context might not be available!

The following example shows a script which can be called from a workflow activity. You might want to write the code directly into the script or place it in an Admin Tool script which is called from the workflow. For details about this principle, please refer to the *ConSol CM Process Designer Manual*.

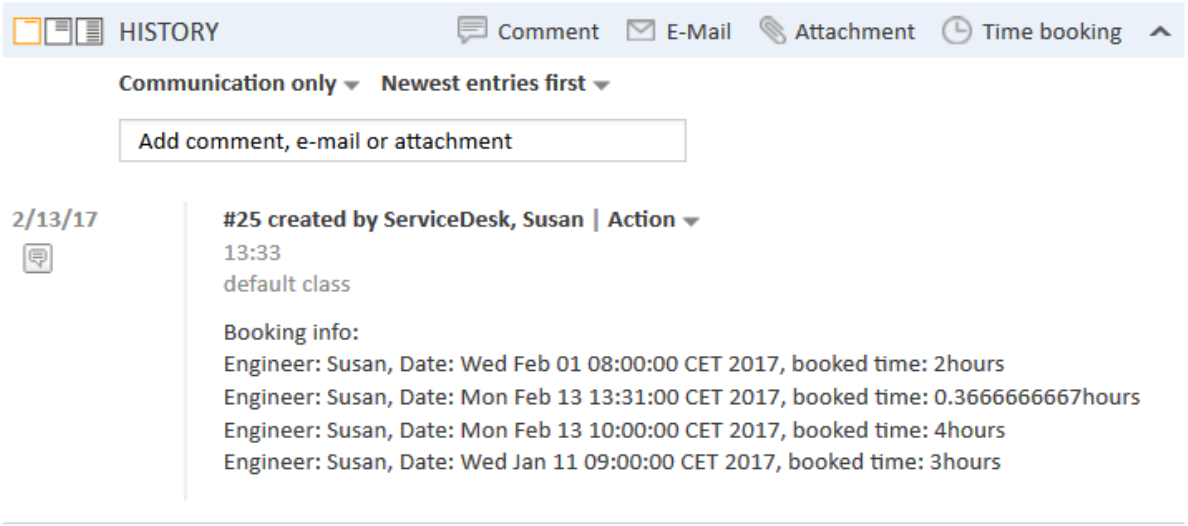
F.13.6.1 Example of a Time Booking Script

The script:

```
import com.consol.cmas.common.model.ticket.Ticket
import com.consol.cmas.common.model.time.TimeBooking

log.info 'Running script printTimeBookingsToTicket ...'

def ticket = workflowApi.ticket
def bookings = workflowApi.getBookings(ticket)
def eng
def per
def per_date
def per_time
def text = 'Booking info: \n'

if (bookings.size() > 0){
   log.info 'Time bookings found in ticket ' + ticket.name
   bookings.each() { bk ->
      eng = bk.engineer.name
      per = bk.timePeriod
      per_date = per.bookingDate
      per_time = per.bookedTime /1000 / 60 / 60
      def text1 = 'Engineer: ' + eng + ', Date: ' + per_date + ', booked time: ' +
       per_time + 'hours'
      text += text1 + '\n'
   }
   workflowApi.addTicketText(text, 'Time Booking report for this ticket', false)
} else {
   workflowApi.addValidationError('INFO', 'No booking on this ticket.')
}

return
```

Code example 98: *Admin Tool script, called from a workflow: all time intervals booked on the ticket are printed into the ticket history as comment*

The output in the respective ticket:



F.14 Authentication Methods in ConSol CM

For the authentication of users in ConSol CM, you can configure one of the following three methods:

- **Standard / Database**
  All user data is kept in the ConSol CM database. This authentication method is available for engineers (see Database Authentication for Engineers) and customers (see Database Authentication for Customers).
- **LDAP**
  The credentials are kept in an LDAP server. ConSol CM sends a request to this server to authenticate the users. This authentication method is available for engineers (see LDAP Authentication for Engineers in the Web Client) and customers (see LDAP Authentication for Customers in CM/Track).
- **Kerberos SSO**
  The credentials are taken from a valid session using Kerberos, see Single Sign-On with ConSol CM Using Kerberos (in a Windows Domain). This authentication method is only available for engineers in the Web Client.

F.14.1 Authentication Methods for Engineers in the Web Client

There are three methods for authenticating engineers in the Web Client:

- **Database authentication**:
  The login and the password are stored in the CM database, see Database Authentication for Engineers.
- **LDAP authentication**:
  The login is stored in the CM database, in the engineer administration, the password is stored on an LDAP server, see LDAP Authentication for Engineers in the Web Client.
- **Kerberos authentication**:
  The login is stored in the CM database, in the engineer administration, the password is stored in Kerberos, see Single Sign-On with ConSol CM Using Kerberos (in a Windows Domain).

F.14.1.1 Database Authentication for Engineers

Setting the User Name and Password

The user name and password are provided in the engineer data (navigation group *Access and Roles*, navigation item *Engineers*, see Engineer Administration). The following fields are relevant for database authentication:

- **Login:**
  Mandatory. This field contains the account name which has to be entered on the login page of the Web Client. Please use only international alphabetic and numeric characters, no blanks, punctuation marks, or special characters such as umlauts, hyphens, or the like.
- **Email:**
  Mandatory. The engineer's email address. Please use only international alphabetic and numeric characters, hyphens, underscores, periods, and the @ sign. The entry of multiple email addresses in one line is not allowed.
- **Password:**
  Mandatory. The engineer's password is mandatory. Please use only international alphabetic and numeric characters, and punctuation marks, do **not** use any special characters such as, e.g., umlauts. The password entered will be shown as a string of asterisks. Please see section Configuring the Password Policy for information about the optional password policy.

Configuring the Password Policy

This configuration is optional.

The following system properties can be used to implement a certain password policy. All following system properties are located in the module `cmas-core-security`. (For details about system properties, please refer to the section System Properties.)

- policy.password.pattern (String)
  RegEx pattern for the password, example and default value: “^.3,$”
- policy.password.age (Integer)
  Maximum validity period, in number of days, example “183” (6 months), default value: “5500” (= 15 years, i.e., no password change enforced). In case you would like to have the engineer

change his password asap, use one of the two following values:

- **0**
  The engineer will be forced to change his password on the next login.
- **1**
  The engineer will be forced to change his password the next day.

- policy.rotation.ratio (Integer)
  This defines the number of previous passwords which may not be identical, example and default value: “1”.
- policy.username.case.sensitive (Boolean)
  Defines whether the password is case-sensitive. Example and default value: “true”. Note that this setting is affected by the MySQL collation setting and needs the correct collation to work properly with MySQL.

Resetting an Engineer's Password

If an engineer has forgotten his password, he can request a new password by using the link *Forgot your password* on the initial login page. An email with a link to a URL where the engineer can set the new password is sent to the engineer.

Please note that this can only work if a valid email account is available for this engineer and if the respective value has been entered as email for the engineer in the engineer data!

The email which is sent to the engineer is based on the template `password-reset-template` which is stored in the *Templates* section of the Admin Tool. Please see section Admin Tool Templates for a detailed explanation of templates in general and section Password Reset Template for Engineers in the Web Client for details about the engineer password reset.

The password reset in the Web Client is only possible if the standard authentication mode is used. It is not possible if LDAP or Kerberos authentication is in operation. See section Authentication Methods in ConSol CM for an explanation of all possible authentication modes.

F.14.1.2 LDAP Authentication for Engineers in the Web Client

Introduction to ConSol CM LDAP Authentication

ConSol CM offers LDAP authentication for the Web Client as a standard feature, i.e., instead of managing the passwords for the ConSol CM engineers in the ConSol CM database, they can be retrieved from an LDAP server (like e.g., a ***Microsoft Active Directory*** server).

When engineers want to log in to the ConSol CM Web Client, they enter their user name and password and press *Enter*. Behind the scenes, the ConSol CM server sends a request with the engineer's user name and password and asks the LDAP server whether those credentials are correct.

If the credentials are correct, the approval is sent back to the ConSol CM server and the engineer is logged into the Web Client.

Please keep in mind that the LDAP connection is only used to authenticate the user (confirm the identity). The authorization (i.e., the assignment of access permissions in the system) is done via the engineer and role administration in the Admin Tool. For every user who should work with the system as an engineer, an engineer account has to be created in the engineer administration!

Please see also the following picture for an explanation of the CM authentication process using LDAP.

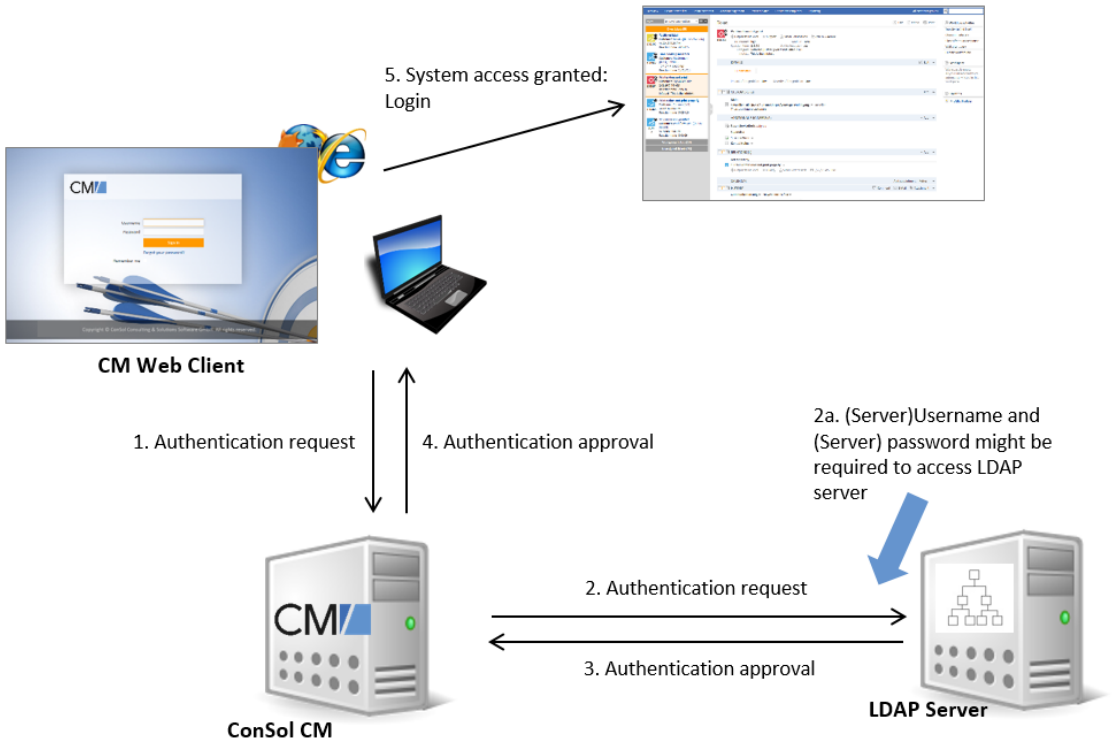


Figure 564: *ConSol CM - LDAP authentication process*

Configuring LDAP Authentication

There are two ways you can enable the ConSol CM system to use LDAP authentication for engineers in the Web Client:

- Select LDAP authentication during system setup and enter the requested parameters (system properties) after the setup.
- Set up the system with the regular authentication mechanism and switch to LDAP later on, i.e., enter all required system properties later on.

Configuring LDAP During Initial Setup

During system setup you can select *LDAP* as the authentication mode. This will set the system property cmas-core-security, authentication.method (see below) to “LDAP”. No further parameters are entered. You have to set the LDAP parameters manually. Please see the next section for an explanation.

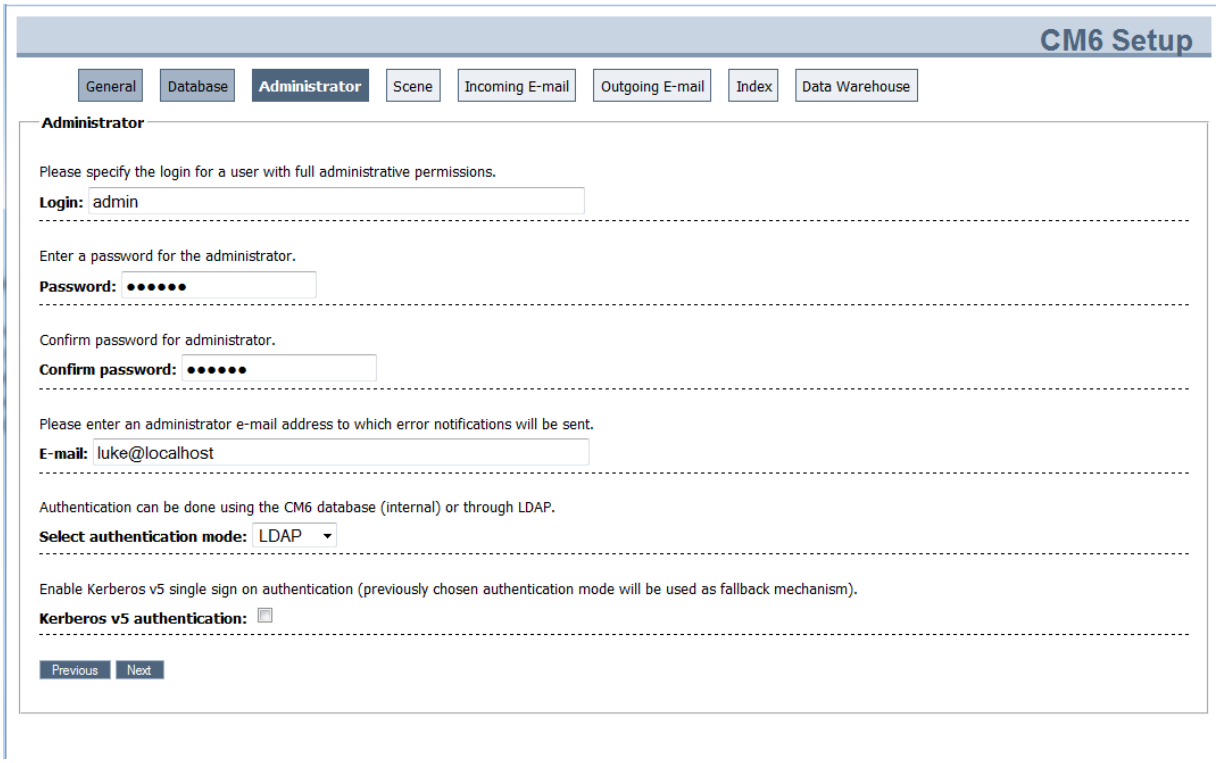


Figure 565: *ConSol CM system set-up - Authentication mode LDAP*

Switching the Authentication Mode to LDAP in a Running System

To switch the authentication method to LDAP, you have to set the required values in the system properties (navigation group *System*, navigation item *System Properties*, see System Properties):

- authentication.method
  LDAP
- ldap.authentication
  simple
- ldap.basedn
  The DN (distinguished name) of the LDAP (sub-)tree where the required attributes are located.
- ldap.initialcontextfactory
  The Java class name for the initial context factory of the LDAP implementation when using LDAP authentication. Should usually be `com.sun.jndi.ldap.LdapCtxFactory`.

- ldap.password
  Password for connecting to the LDAP server to look up users. Only needed if look-up cannot be done anonymously.
- ldap.userdn
  LDAP user for connecting to the LDAP server to look up users. Only needed if look-up cannot be done anonymously.

  ⚠ A server user name/password pair might be required to access the LDAP server. If you are not sure, you might want to use an LDAP browser to confirm.

- ldap.providerurl
  The complete URL for the LDAP server:

  ```
  ldap://<HOSTNAME>:<LDAP PORT>
  ```

- ldap.searchattr
  Search attribute for looking up the LDAP entry connected to the CM login, i.e., the attribute which is used as user name for the authentication.

Using LDAPS (LDAP over SSL)

Introduction

Per default, when an LDAP client accesses an LDAP server, the information is transferred in clear text. In case you want the user name and password to be transferred to the LDAP server in encrypted form, you have to set up the LDAP authentication using LDAPS.

Preparations

You have to configure the CM server machine (Java) in a way that can use certificates. One way to do this for a Linux environment is described in the following section.

1. Retrieve the certificate:
   `openssl s_client -connect dc2.mydomain.com:ldaps`
2. The answer will contain a section which starts with "---BEGIN CERTIFICATE " and ends with "END CERTIFICATE ---".
   Copy this section to a file, e.g., `/tmp/certificate2_dc2_mydomain_com.txt`
3. Import the certificate to the truststore of your machine, e.g., /home/mydirectory/mytruststore
   ```
   $JAVA_HOME/bin/keytool -import -alias <arbitrary> -trustcacerts -keystore /home/mydirectory/mytruststore -file/tmp/certificate2_dc2_mydomain_com.txt
   ```
   You have to enter (set) a password.
4. Enter the truststore in the ConSol CM config file in JAVA_OPTS:
   ```
   -Djavax.net.ssl.trustStore=/home/mydirectory/mytruststore -Djavax.net.ssl.trustStorePassword=<see above>
   ```

LDAPS Configuration in the ConSol CM Admin Tool (System Properties)

Configure the ConSol CM server as shown in the following example:

- cmas-core-security, ldap.authentication = simple
- cmas-core-security, ldap.basedn = OU=myOU,DC=myDC
- cmas-core-security, ldap.initialcontextfactory = com.sun.jndi.ldap.LdapCtxFactory
- cmas-core-security, ldap.password = myLDAPpw
- cmas-core-security, ldap.searchattr = sAMAccountName
- cmas-core-security, ldap.userdn = myLDAP_UserDN

Depending on the LDAP server configuration, use one of the following values for the server URL:

- **Standard LDAPs port**
  cmas-core-security, ldap.providerurl = ldaps://dc2.mydomain.com:636
- **LDAPs port Global Catalogue**
  cmas-core-security, ldap.providerurl = ldaps://dc2.mydomain.com:3269

Configuring Engineer Accounts for LDAP Authentication

Use the engineer administration (navigation group *Access and Roles*, navigation item *Engineers*) in the Admin Tool to configure the engineer accounts.

When LDAP is used as authentication method, it is not possible to set the ConSol CM password within the engineer administration. The pop-up window for engineer management provides the following fields which are relevant for LDAP authentication. Please refer to section Engineer Administration for details concerning the other (non LDAP-related) data fields.

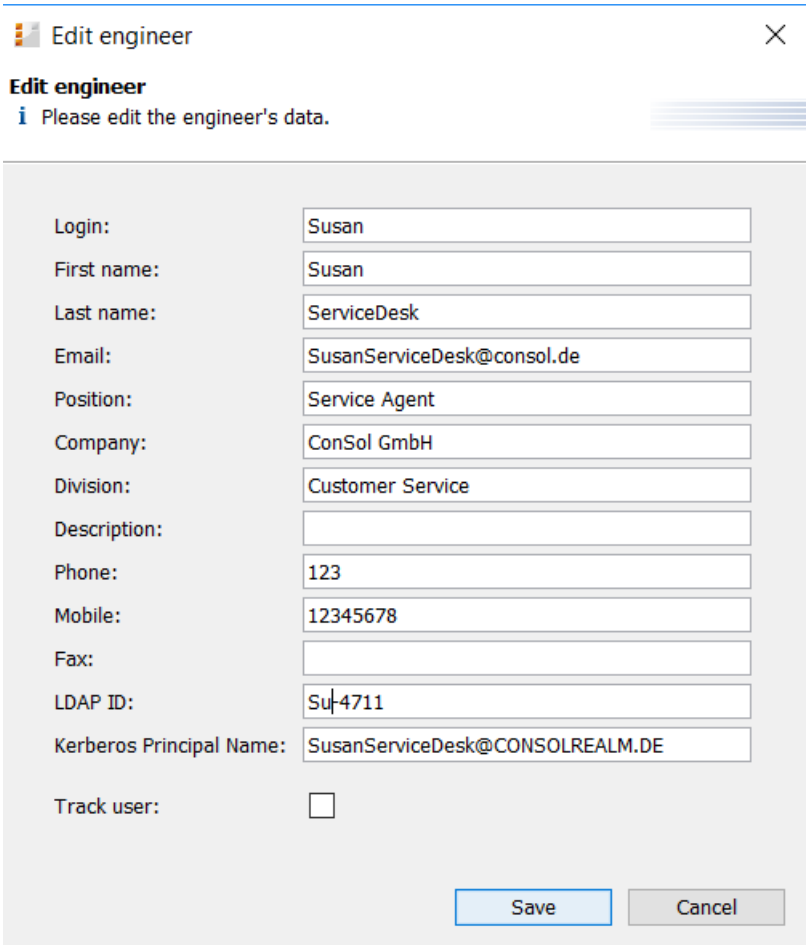


Figure 566: *ConSol CM Admin Tool - Access and Roles, Engineers: Editing an engineer*

- **Login**
  If no LDAP ID has been provided, this is used as the user name during the LDAP authentication process which is looked up in the LDAP directory in the `ldap.searchattr` node.
- **LDAP ID**
  If you would like to employ special user names in ConSol CM which are not identical to the values used in the LDAP directory, fill in this field. During the LDAP authentication process, this LDAP ID is used as the user name which gets looked up in the LDAP directory in the `ldap.searchattr` node.

F.14.1.3 Single Sign-On with ConSol CM Using Kerberos (in a Windows Domain)

Short Introduction to Kerberos

Kerberos is a network protocol which is used to authenticate a user or a system to verify if a user or system has the identity it claims to have.The authentication mechanism is based on so called *tickets*. When a user or system has been authenticated by a Kerberos system, other systems, e.g., a ConSol CM server, can rely on this authentication and grant access for the user or system to their own resources. One of the advantages of using Kerberos is that no passwords (encrypted or plain) are sent

over the network. Only information which has been encrypted using the password is sent. Once authenticated, a client has access to all resources within the Kerberos domain/realm without any further login (single sign-on, SSO).

The current version of Kerberos is V5, which was initially developed in 1993. If you want to really dig into this topic, you might want to read RFC 1510 (initial RFC) or RFC 4120 (V5).

The following components are important in a Kerberos infrastructure:

The **Key Distribution Center**, which contains two active components:

- **Authentication Service**
  Performs the authentication (e.g., using Microsoft Active Directory as in our example) and creates a **Ticket Granting Ticket (TGT)** if the user or system has been authenticated successfully.
- **Ticket Granting Service**
  Creates tickets which will grant access to other systems. The user or system who/which wants to receive a **service ticket** has to present a TGT in order to get this *service ticket*.

Short Introduction into ConSol CM with Kerberos in a Windows Domain

The *single sign-on* feature allows users to authenticate against ConSol CM automatically with, e.g., their ***Windows*** credentials.

This authentication mechanism ...

- works completely transparent, no user interaction (i.e., filling in login screen) is required,
- does not interfere with existing authentication mechanisms. If Kerberos authentication fails, whatever authentication mechanism was configured (e.g., LDAP or database authentication, see CM System Property cmas-core-security, authentication.method) is used.

The single sign-on feature is based on the ***Kerberos V5*** protocol, which is integrated in the ***Windows Active Directory*** (AD).

The ConSol CM server works as a *non-Windows Kerberos service* and can be installed on any operating system/application server.

The ConSol CM server is part of a Windows domain where the Domain Controller is the Key Distribution Center and also runs the Active Directory which stores the user authentication information.

The following graphic provides an (simplified) overview of the steps which are required in order to provide a valid client/server session for a ConSol CM client and the respective ConSol CM server. The explanations in this sections will guide you through the entire configuration process which is required to run ConSol CM with Kerberos authentication.

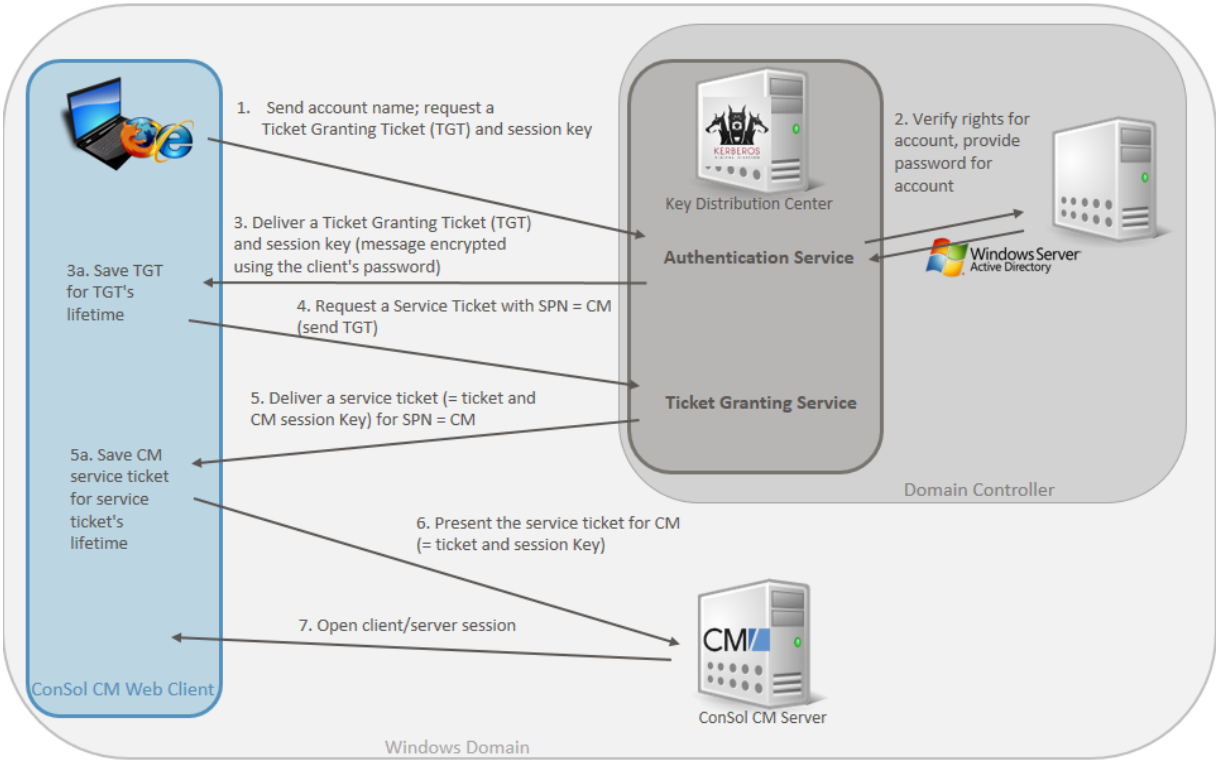


Figure 567: *Kerberos Authentication in a Windows domain for ConSol CM access*

Steps in the Authentication Process

Step 1: Send Account Name, Request a Ticket Granting Ticket (TGT) and Session Key

The user logs in to the PC/laptop with username and password. The PC/laptop sends a message to the Authorization Server requesting a ticket granting ticket (TGT).

In case pre-authentication has been enabled, a time stamp will be encrypted using the user's password hash as an encryption key. If the Authentication Server reads a valid time when using the user's password hash (stored in the Active Directory) to decrypt the time stamp, the Authentication Server knows that request is not a renewal of a previous request.

In the example, the pre-authentication is disabled.

Step 2: Verify Rights for Account, Provide Password for Account

The Authorization Server verifies the user's access rights in the Active Directory and the AD provides the user's password for step 3.

Step 3: Deliver a Ticket Granting Ticket (TGT) and Session Key (Message Encrypted Using the Client's Password)

The Authentication Service creates a Ticket Granting Ticket (TGT) and a session key. These are sent in a message to the client, encrypted with a key derived from the user's password.

The PC/laptop prompts the user for a password and uses the password to decrypt the incoming message. When decryption succeeds, the user will be able to use the TGT to request a service ticket.

Step 3a: The TGT is saved for the user for the TGT's lifetime and can be used for all service ticket requests during this period of time.

Step 4: Request a Service Ticket for CM

When the user wants access to a service, i.e., when an engineer wants lo log in to ConSol CM, the workstation client application (CM Web Client) sends a request to the Ticket Granting Service containing the SPN (Service Principal Name) of the requested service, the client name, the realm/domain name, and a timestamp (name and timestamp encrypted with the session key). The user proves his identity by sending an authenticator (TGT) received in step 3.

Step 5: Deliver a Service Ticket for CM (= Ticket and Session Key)

The Ticket Granting Service decrypts the ticket and authenticator, verifies the request, and creates a service ticket for the requested service (SPN = ConSol CM). The service ticket contains the user (engineer) name. It also contains the realm/domain name and service ticket lifespan. The Service Ticket is encrypted using the Services (CM's) secret. The Ticket Granting Service sends the service ticket to the user.

Step 5a: The Service Ticket is saved for the user for the Service Ticket's lifetime and can be used for all requests to the SPN during this period of time.

Step 6: Present the Service Ticket for CM (= Ticket and Session Key)

The client application (CM Web Client) now sends a service request to the CM server containing the ticket received in step 5 and an authenticator. The service (CM server) authenticates the request by decrypting the session key. The CM server verifies that the service ticket and authenticator match, and then grants access to the service (ConSol CM application).

Step 7: Open Client/Server Session

The client/server session is in operation.

Encryption in the Process

Usually, Kerberos is run with symmetric encryption (i.e., one key is used to encrypt and decrypt the message). In Kerberos, encryption is based on the user's password. Optionally, public-key cryptography might be used in a Kerberos environment.

Microsoft Windows uses Kerberos as default authentication method, using RC4 for encryption, together with HMAC (Hash-based Message Authentication Code).

Kerberos Terminology

Kerberos Principal

A ***Kerberos*** principal is a unique identity to which Kerberos can assign tickets. This might be a user's (person's) account or a system user account. Kerberos principals are composed of a number of components. Each component is separated by a component separator, usually "/". The last component is the realm/domain, separated from the rest of the principal by the realm/domain separator, usually "@". If there is no realm/domain component in the Kerberos principal, then it will be assumed that the principal is in the default realm/domain for the current environment.

Usually, a Kerberos principal is composed of three parts: the primary, the instance, and the realm/-domain. The format of a typical Kerberos V5 principal is `primary/instance@REALM`.

- The primary is the first part of the principal, this is the user name (in case of a person's account) or the name/account of a system user or a host name.
- The instance is an optional string that qualifies the primary. The instance is separated from the primary by a slash (/). In the case of a user, the instance is usually `null`, but a user might also have an additional principal, with an instance called `admin` , which he uses to administrate a database. The principal `mycmuser@CONSOL.DE` is completely different from the principal `mycmuser/administrator@CONSOL.DE`, with a separate password and separate permissions. In the case of a host, the instance is the fully qualified hostname, e.g., `mymachine.consol.com`.
- The realm is the Kerberos realm, usually the domain name, in upper-case letters. For example, the machine `myserver@consol.com` would be in the realm `CONSOL.COM`.

Keytab File

A `keytab` is a file containing pairs of Kerberos principals and the respective encrypted keys (which are derived from the Kerberos password). When a password is changed, you need to rebuild the keytab file. In the ConSol CM context this means, when the password of the (system) user (***tomcat*** in our example) has changed, **not** when a CM engineer has changed his password.

Configuration of Kerberos Single Sign-On for ConSol CM

Requirements

For Kerberos-based single sign-on in ConSol CM you need:

- Domain controller for the Windows domain
- ConSol CM server
- Windows clients

Since in a Kerberos environment, timestamps are used to calculate and check the time period of validity of the tickets, it is absolutely indispensable that all components work with the same system time! For example, a time server/service based on NTP can be used. (Some Win DCs tolerate a max. five-minute difference.)

Setting Up the System

Basic Principle and Required Steps

In order to work with SSO as CM engineer, the respective CM server has to be a member of a Windows domain. The domain controller acts as ***Kerberos Key Distribution Center*** (KDC) and the user-/account names and passwords are managed in the Active Directory. Therefore, you have to

- register the CM server as Windows domain member.

ConSol CM has to be registered as Kerberos service so that Kerberos service tickets can be created for users who want to work with CM. Therefore CM has to be registered with Kerberos. Because CM runs under a certain system user (*tomcat* in our example) this system user has to be registered for the respective machine. This is why you will have to

- build a keytab file for the system user *tomcat* for the CM machine.

Since it is not the system user (*tomcat)* who will then work with CM but the CM engineers,

- the option *Trust this computer for delegation to any service* has to be set in the KDC.
  In this way, the system user (*tomcat* in our example) can "impersonate" other users (the CM engineers).

On the CM machine, it is required to configure the application server in a way that Kerberos can be used and to let CM know where to find the keytab file with its system user and principal name to use in Kerberos encryption and communication. This is why you have to

- define the respective module in the application server
- store the keytab file on the CM server
- configure the `cm6-kerberos.properties` file

For users who want to use SSO with Kerberos and CM, a CM engineer name must be assigned to a Kerberos principal name. This is why you have to

- define Kerberos principal names for the CM engineers. This is done in the Admin Tool, in the Engineer Administration.

Operations to Be Performed on the Domain Controller

The first step is to configure the domain controller so that it knows the ConSol CM server, i.e., to integrate the CM Server into the Windows domain. In our example the domain controller is called `mywin2003srv`, the domain is `MYSSODOMAIN.COM`.

Registering the ConSol CM Server Machine

First, the ConSol CM server machine needs to be registered in the Active Directory of the domain controller. In our example, the domain controller is the computer *MyComputer*.

The radio button *Trust this computer for delegation to any service (Kerberos only)* must be activated!

Execute: `dsamsc`, machines

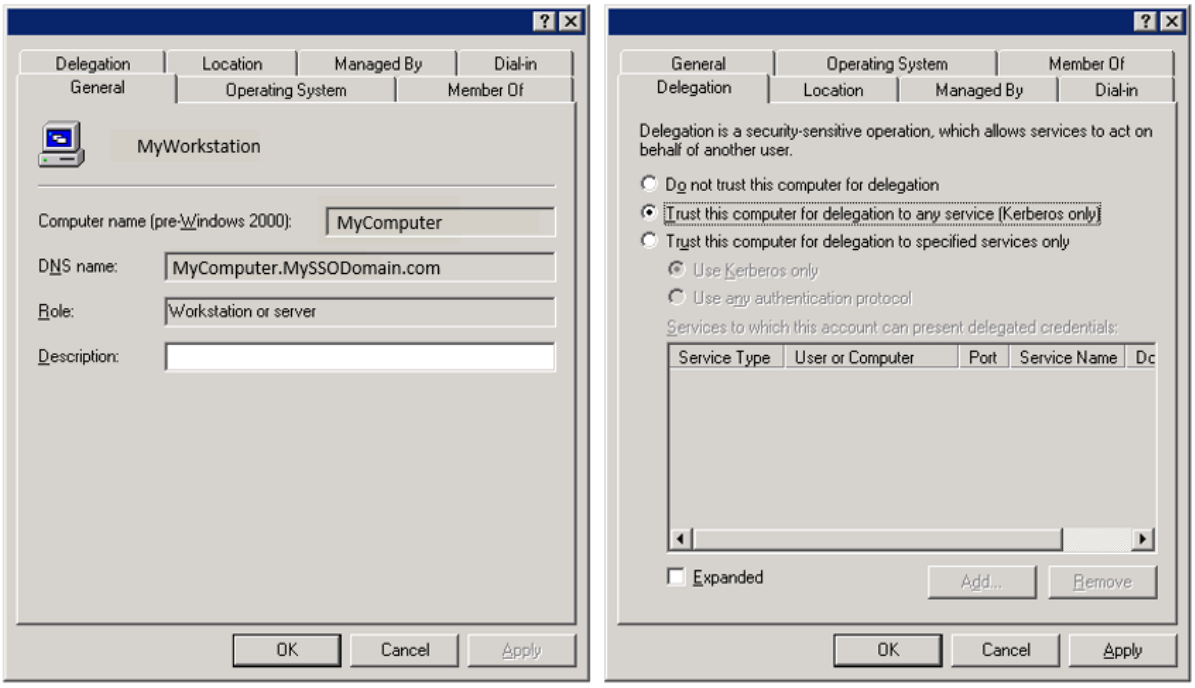


Figure 568: *Registering the ConSol CM server machine*

Registering the ConSol CM Server User

Second, the user under which the ConSol CM server process will run is created and registered in the Active Directory (which acts as Key Distribution Center), in our example the user *tomcat*.

The following account options must be enabled:

- Account is trusted for delegation
- No Kerberos pre-authentication needed

Execute: `dsamsc`, users

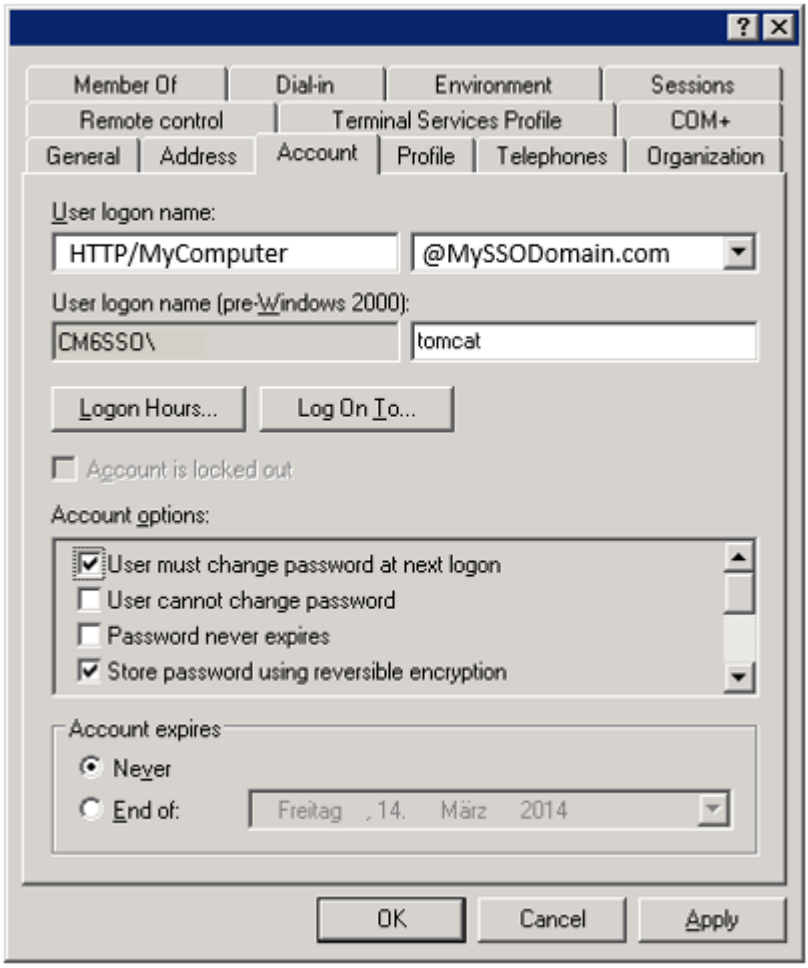


Figure 569: *Registering the ConSol CM server user*

Make sure that the CM application server process (e.g., JBoss) runs under the indicated user on the CM machine!

Generating the keytab File

On the domain controller the ConSol CM server is created as a new (non-Windows) Kerberos service and a Kerberos `keytab` file is generated. This file will be required later on the ConSol CM server machine. The `keytab` file contains the shared secret key of the service (SPN, Service Principal Name).

Use the following ktpass command:

```
C:\Program Files\Support Tools>ktpass /out tomcat.keytab /ptype KRB5_NT_PRINCIPAL
 /princ HTTP/MyComputer.MySSODomain.com@MYSSODOMAIN.COM /pass consol.123 /mapuser
 tomcat /crypto rc4-hmac-nt
```

⚠ If `ktpass` is not available, the ***Windows Server 2003 Support Tools*** must be installed, available here.

Explanation of the ktpass command

```
/out tomcat.keytab
```

=> Use `tomcat.keytab` as output file

```
/ptype KRB5_NT_PRINCIPAL
```

=> Specifies the principal type.

- **KRB5_NT_PRINCIPAL** is the general principal type (recommended).
- **KRB5_NT_SRV_INST** is the user service instance.
- **KRB5_NT_SRV_HST** is the host service instance.

```
/princ HTTP/MyComputer.MySSODomain.com@MYSSODOMAIN.COM
```

=> Specifies the Kerberos principal name in the form host/computer.<domain>@<realm> . See User logon name in configuration (figures above)

```
/pass consol.123
```

=> sets the password for the *princ* user indicated with `/princ`

```
/mapuser tomcat
```

=> Maps the name of the Kerberos principal (`/princ` parameter), to the specified domain account.

```
/crypto rc4-hmac-nt
```

=> Specifies the keys that are generated in the keytab file:

- **DES-CBC-CRC** is used for compatibility.
- **DES-CBC-MD5** adheres more closely to the MIT implementation and is used for compatibility.
- **RC4-HMAC-NT** employs 128-bit encryption.
- **AES256-SHA1** employs AES256-CTS-HMAC-SHA1-96 encryption.
- **AES128-SHA1** employs AES128-CTS-HMAC-SHA1-96 encryption.
- **All** states that all supported cryptographic types can be used.

Operations to Be Performed on the ConSol CM Server

Run the application server (e.g., JBoss) under the system user which is registered in the Windows domain (*tomcat* in our example).

Install ConSol CM as usual, then enable and configure Kerberos as described in the next steps.

Enabling Kerberos in ConSol CM

If you do an initial set-up, you can choose whether Kerberos should be enabled. Please note that this is only a hint and additional configuration is needed (see next steps).

If your ConSol CM is already configured without Kerberos enabled, you can enable it in the Admin Tool by setting the system property cmas-core-security, kerberos.v5.enabled to “true”. A server restart is required to activate the new setting.

Configuring Kerberos

A ConSol CM server reads configuration parameters from the file `cm6-kerberos.properties` from the classpath. (Each cluster node may need separate configurations so each node will read the `cm6-kerberos.properties` file from the classpath.)

- **Under JBoss 5:**
  `../jboss/server/{domain}/conf/cm6-kerberos.properties`
- **Under WebLogic:**
  `../{domain}/cm6-kerberos.properties`
- **Under JBoss 7 and WildFly 8.2:**
  1. mkdir -p ${jboss}/modules/system/layers/base/com/consol/cmas/main/
  2. Create and edit ${jboss}/-modules/system/layers/base/com/consol/cmas/main/module.xml

```
<module xmlns="urn:jboss:module:1.1" name="com.consol.cmas">
   <resources>
      <resource-root path="."/>
   </resources>
</module>
```

3. Put the properties file here: ${jboss}/-modules/system/layers/base/com/consol/cmas/main/cm6-kerberos.properties
4. Edit the configuration, e.g., ${jboss}/standalone/configuration/cm6.xml

```
<subsystem xmlns="urn:jboss:domain:ee:1.1">
   ...
   <global-modules>
      <module name="com.consol.cmas" slot="main" />
   </global-modules>
</subsystem>
```

Code example 99: *JBoss 7*

```
<subsystem xmlns="urn:jboss:domain:ee:2.0">
   ...
   <global-modules>
      <module name="com.consol.cmas" slot="main" />
   </global-modules>
</subsystem>
```

Code example 100: *WildFly 8.2*

In case you have a cluster of more than one ConSol CM servers in operation, each server has to have its own properties file.

The `.properties` file should contain:

- Reference to a Kerberos config file (e.g., `krb5.ini` or `krb5.conf`)
- One or more service principals, i.e., reference to the `keytab` file

```
# path to kerberos configuration
kerberos.config.location=C:\\conf\\krb5.ini

# one or more service principals (principal = path to keytab file)
HTTP/MyComputer.MySSODomain.com@MYSSODOMAIN.COM=C:\\conf\\tomcat.keytab
```

Code example 101: *cm6-kerberos.properties*

```
[libdefaults]
   default_realm = MYSSODOMAIN.COM
   default_tkt_enctypes = rc4-hmac des-cbc-md5 des-cbc-crc des3-cbc-sha1
   default_tgs_enctypes = rc4-hmac des-cbc-md5 des-cbc-crc des3-cbc-sha1

[realms]
   MYSSODOMAIN.COM = {
      kdc = mywin2003srv
      admin_server = mywin2003srv:8888
   }

[domain_realm]
   .mywin2003srv = MYSSODOMAIN.COM
   mywin2003srv = MYSSODOMAIN.COM
```

Code example 102: *krb5.ini*

keytab File

Copy the `keytab` file you generated on the domain controller to the location you specified in the `cm6-kerberos.properties` config file.

⚠ You have to restart the ConSol CM server process for this change to take effect!

Configuring the CM Engineers for Kerberos

When the system property cmas-core-security, kerberos.v5.enabled has been set to “true”, the field *Kerberos Principle Name* will be available for engineer data.

You reach this screen by opening the navigation item *Engineers* in the navigation group *Access and Roles* and clicking the *Edit* button of an engineer.

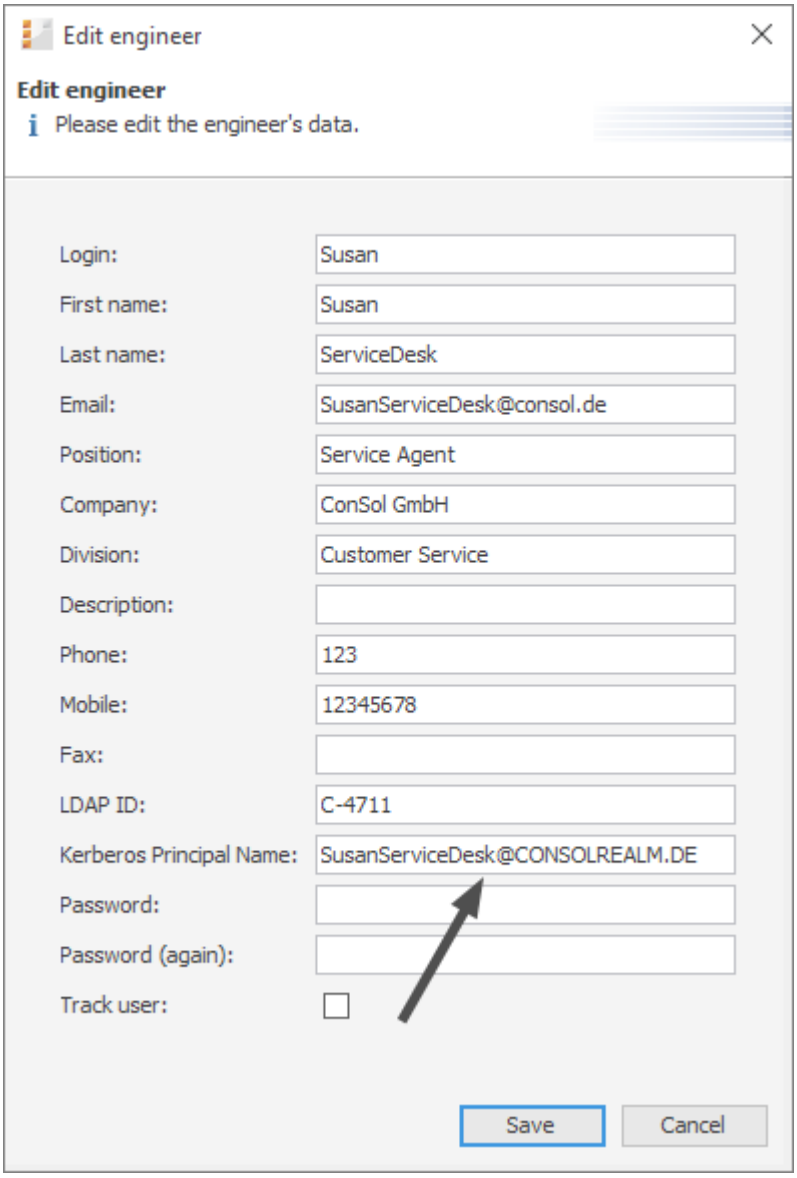


Figure 570: *ConSol CM Admin Tool - Access and Roles, Engineers: Editing the Kerberos Principal Name*

The *Kerberos Principal Name* is a user name along with the respective Kerberos realm/domain name, for example SusanServiceDesk@consol.de. This allows to authenticate users of login SusanServiceDesk that belong to different Kerberos realms (SusanServiceDesk@consol.de and SusanServiceDesk@consol.pl are in fact different users). If the field *Kerberos Principal Name* is empty for an engineer, the username (*Login*) will be used. If Kerberos authentication fails, the regular authentication mechanism (DATABASE, LDAP) will be used. In that case, single sign-on is not possible and the login dialog will be displayed for the engineer.

Operations to Be Performed on the Client Machine

Internet Explorer

Internet Explorer needs to be configured so that an automatic login is enabled. By default, this is allowed in the *medium-low* security setting, which by default is set for the *local Intranet zone*.

The following settings for login behavior are available.

Figure 571: *Internet Explorer login configuration*

Each setting and the resulting behavior:

- **Anonymous logon**
  No single sign-on is possible, the user (CM engineer) will get the ConSol CM login dialog.
- **Automatic logon only in Intranet zone**
  Single sign-on is performed automatically but only if the site is part of the *local intranet* zone.
- **Automatic logon with current user name and password**
  Single sign-on is performed automatically with the current user credentials.
- **Prompt for user name and password**
  OS displays a login dialog, user can enter OS login information, which is then used in Kerberos authentication.

Firefox

With the default settings, ***Firefox*** does not support Kerberos single sign-on. To enable single sign-on, you have to add the URI of the ConSol CM Web Client in the Firefox configuration.

To do this:

- Open `about:config`.
- Add the web server to the property `network.automatic-ntlm-auth.trusted-uris`
  (for example `http://mycm6server` if that is the URI).
  If this does not work, perform the following step:
- Add the web server to the property `network.negotiate-auth.trusted-uris`
  (for example `http://mycm6 server` if that is the URI).

You can set this property also in the file system. Open the file

`C:\Users\[USER]\AppData\Roaming\Mozilla\Firefox\Profiles\[XYZ].default\prefs.js`

and add/replace the following line:

```
user_pref("network.automatic-ntlm-auth.trusted-uris", "http://mycm6server");
```

or

```
user_pref("network.negotiate-auth.trusted-uris", "http://mycm6server");
```

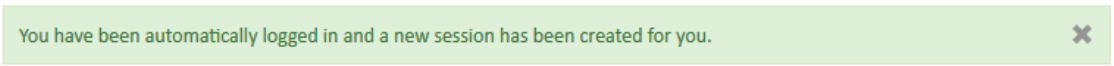
⚠ You have to restart Firefox after this change.

Using the System

Single Sign-On from the User's Point of View

An engineer using single sign-on to log into ConSol CM will notice that ...

- no ConSol CM login screen is displayed,
- instead there may be (for a short time) an intermediate text screen (which is used to gather some client data via JavaScript) which immediately forwards the user to the ConSol CM Web Client main screen.
  Here, a message is displayed:



It is still possible to login as another ConSol CM user by clicking the logout button, which will lead you to the login page, or by explicitly using the `.../cm-client/login` URL.

Multi Domains Single Sign-On

For each domain which you will enable single sign-on for, create a new domain/user and Kerberos principal and put all of them into the `cm6-kerberos.properties` file:

```
# path to kerberos configuration (think krb5.conf or krb5.ini)
kerberos.config.location=/etc/krb5.conf

# one or more service principals (principal = path to keytab file)
HTTP/MyServerMyDomain.com@MYDOMAIN.COM=/etc/krb5_mycompanycom.keytab
HTTP/MyServer.MyDEDomain@MYDOMAIN.DE=/etc/krb5_mycompanyde.keytab
```

Mapping Kerberos User Name to Engineer Name

Using Kerberos-based single sign-on, the Kerberos principal (i.e., the user's OS login) has to be mapped to a ConSol CM engineer name.

By default, this mapping is done using one of the following two ways:

- **Explicit mapping**
  Take the principal name and try to find a ConSol CM engineer who has this principal stored as *Kerberos Principal Name*. If such an engineer is found, this engineer is used.
- **Mapping via regular expression**
  The regular expression defined in the system property cmas-core-security, kerberos.v5.username.regex is taken and applied to the principal. The result of this will be taken and a ConSol CM engineer with this login will be searched:
  - First matching regular expression group (in brackets) will be used as engineer login name,
    e.g., the default property value *(.*)@.** will convert *Huber@MySSODomain.com* to *Huber*.

If further customization is needed please refer to *UsernameAdapter interface javadoc*.

Starting and Stopping Kerberos Authentication

Kerberos authentication can be started/stopped in the Admin Tool -> navigation group *Services* -> navigation item *CM Services -> Kerberos v5 authentication provider*, see section CM Services.

F.14.2 Authentication Methods for Customers in CM/Track

F.14.2.1 Available Authentication Methods

There are three possible authentication modes:

- Against the ConSol CM database
  This is called *DATABASE mode*, see Database Authentication for Customers
- Against an LDAP server
  This is called *LDAP mode*, see LDAP Authentication for Customers in CM/Track
- Against an LDAP server and the ConSol CM database
  The order can be configured. This is called *Mixed mode*, see Mixed Authentication Mode

F.14.2.2 Defining the Authentication Method

The authentication mode is specified by the system property cmas-core-security, contact.authentication.method. A change of this property does not require a server restart, and is propagated to all cluster nodes.

The possible values (see also section System Properties) and their respective system behaviors are:

- **DATABASE**
  Attempt a database login if the customer has a database password. I.e., the login and password are stored in the ConSol CM database and are thus managed by the ConSol CM engineers, or indirectly by the customers themselves when they reset their password. The customer can reset his own password, see section Password Reset Template for Customers Using CM/Track V2.
- **LDAP**
  Try authentication using the available LDAP server(s), if an LDAP ID is provided. I.e., the password is stored in the LDAP directory and cannot be changed via ConSol CM, neither by the customer nor by an engineer.
- **LDAP,DATABASE**
  First attempt authentication using the available LDAP server(s), if an LDAP ID is provided. On failure, try a database login if the customer has a database password.
- **DATABASE,LDAP**
  First attempt a database login if the customer has a database password. On failure try authentication using the available LDAP server(s) if an LDAP ID is provided.

The values are case insensitive, and commas and whitespace are ignored.

F.14.2.3 Database Authentication for Customers

Database authentication is activated by setting the system property cmas-core-security, contact.authentication.method to “DATABASE” (default value).

There are two steps which you need to perform to set up database authentication for customers using CM/Track:

- Create customer fields for the user name (login) and password in the Admin Tool (see Defining the Customer Fields for CM/Track Login and Password)
- Enter the user name and password for the actual customers in the Web Client (see Granting Access to CM/Track for Customers)

When database authentication is used, you can allow your customers to change their own passwords, see Configuring CM/Track for Password Reset by Customers.

Defining the Customer Fields for CM/Track Login and Password

The fields for login and password for a customer are regular customer fields at the contact level. Please see section Setting Up the Customer Data Model for an introduction to customer field management and GUI configuration for customer data.

Edit the fields which contain the customer data (if there are two levels: **not** the company level, but the contact level!) as demonstrated in the following example. You reach the following screen by opening the navigation group *Customers*, navigation item *Data Models*.

- Create a field for the **login** with the annotation `username` = “true”.

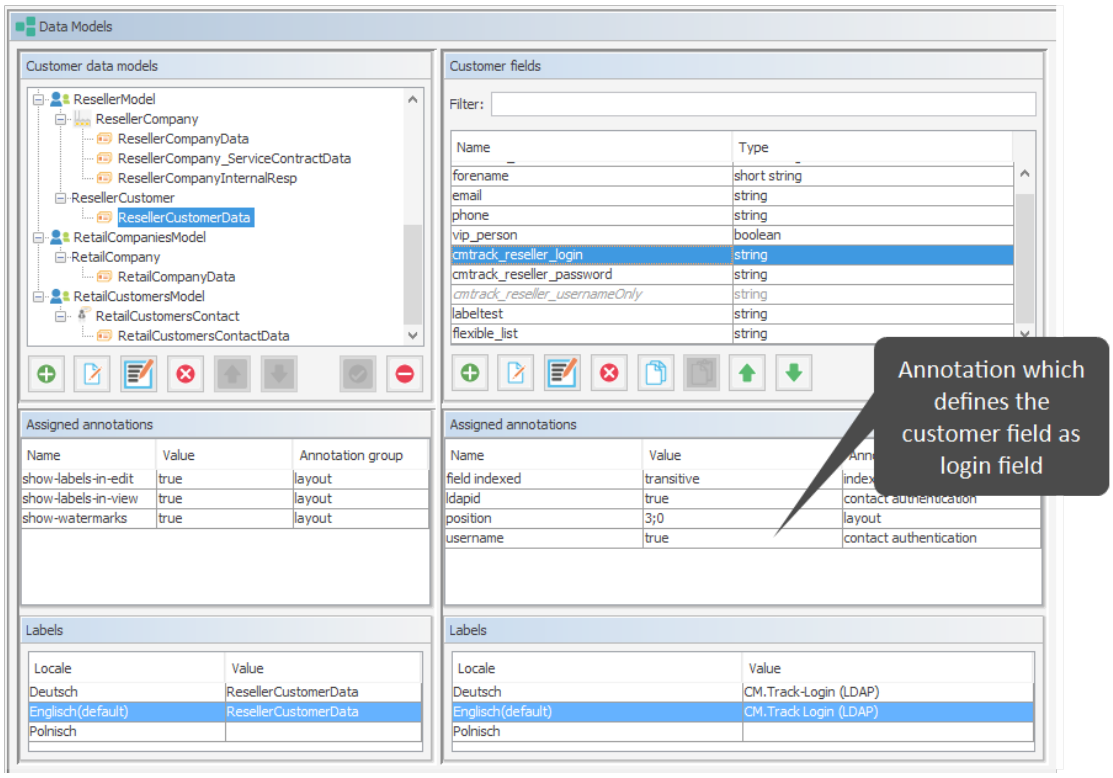


Figure 572: *ConSol CM Admin Tool - Customers, Data Models: CM/Track - Annotation 'username' for login*

> ⚠ Assigning the annotation `username` to a customer field is only possible, if there is no previous assignment of this annotation. Otherwise it will be prohibited. When assigning it, a warning dialog must be confirmed before it is executed, since it can be a longer running operation. Un-assigning the annotation must be confirmed as well, because it cannot be undone: Un-assignment delete the user name values unrecoverably from internal storage.

- Create a field for the password with the annotation `password` = “true”. The annotation `text-type` = “password” guarantees that only stars/dots are displayed in the Web Client, not the clear text password.

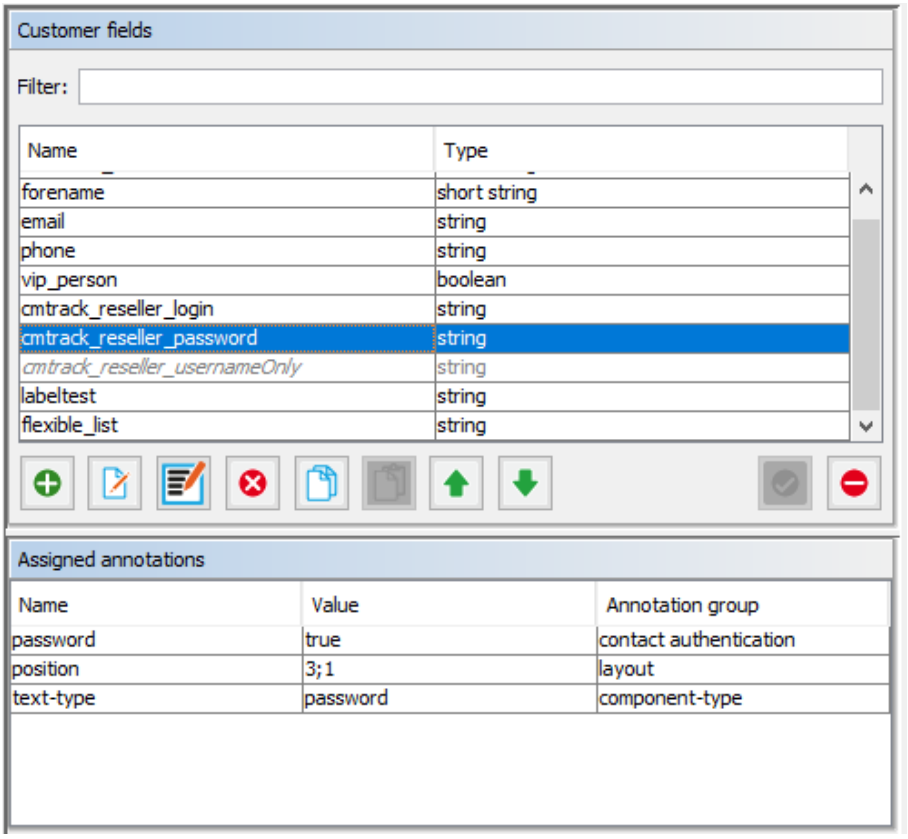


Figure 573: *ConSol CM Admin Tool - Customers, Data Models: CM/Track - Annotation for password*

> ⚠ The annotation `password` requires confirmation when assigned.
>
> In case of an update from CM versions lower than 6.11 to 6.11 and up: When this annotation is set, the system reads the plain text passwords from the original field values, encrypts them and saves the encrypted values to the internal storage. Then the original field values are deleted and thus the plain text value cannot be recovered anymore.
>
> When trying to un-assign the `password` annotation the operation must be confirmed as well, since the encrypted passwords are deleted from the internal storage. After the annotation unassignment the password information is completely lost and cannot be recovered at all.

When a scenario from a CM version lower than 6.11 is imported into a system with CM 6.11 (or higher), a transformation of user names and passwords is performed automatically. This is described in detail in section Transformation of User Name and Password Fields During Import into CM 6.11 .

Granting Access to CM/Track for Customers

The engineer working with the Web Client can then assign a user name, initial password, and a CM/Track user profile to every customer who should have access to the portal CM/Track. The user name has to be unique. This is checked by the system. You cannot enter a name a second time if this has already been assigned to another customer. The password is stored as encrypted string in the CM database. This means that an engineer can set a new password, e.g., when a customer calls and asks for this, but it is never possible to read the password from the system.

> ⓘ You, as an administrator, can define if the CM/Track user names should be case sensitive. Use the CM system property cmas-core-security, policy.track.username.case.sensitive. This is a boolean variable. When it is set to “true”, the CM/Track user names are case sensitive. Please make sure that the database collation which is in use supports case sensitive strings!

The following example shows the customer data of an example contact in the ConSol CM Web Client. You reach this screen by opening a contact data set in edit mode.

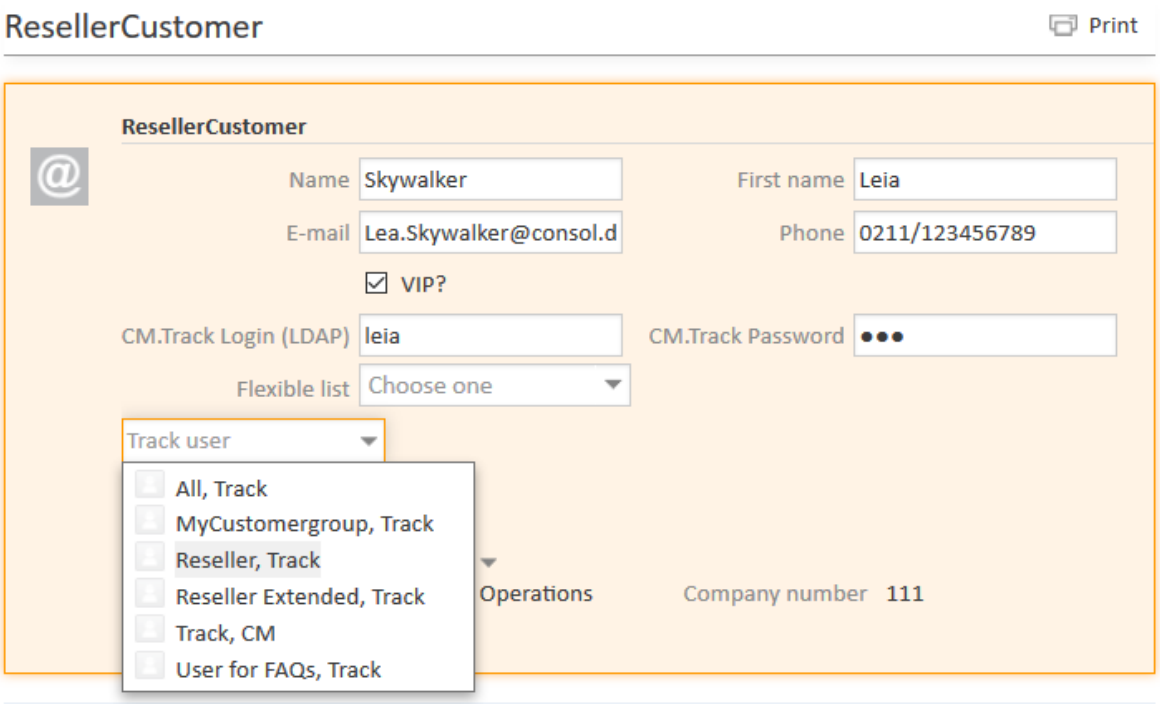


Figure 574: *ConSol CM Web Client - Contact page: CM/Track user data*

Configuring CM/Track for Password Reset by Customers

CM/Track can be configured to offer a hyperlink for customers where a customer can reset his password. This is based on the template `track-password-reset-template`. Please refer to section Password Reset Template for Customers Using CM/Track V1 / Password Reset Template for Customers Using CM/Track V2 for a detailed explanation. The password reset in CM/Track is only possible when the DATABASE mode is used. It is not possible when LDAP authentication is in operation. See section Authentication Methods for Customers in CM/Track for the portal for an explanation of all possible authentication modes.

Please note that the `From`address of the email which is sent to a customer who has requested a new password can be set using the CM system property cmas-core-security, password.reset.mail.from.

F.14.2.4 LDAP Authentication for Customers in CM/Track

Introduction to ConSol CM LDAP Authentication

ConSol CM offers LDAP authentication for CM/Track as a standard feature, i.e., instead of managing the passwords for the ConSol CM customers in the ConSol CM database, they can be retrieved from an LDAP server (like e.g., a ***Microsoft Active Directory*** server).

When customers want to log in to CM/Track, they enter their user name and password and press *Enter*. Behind the scenes, the ConSol CM server sends a request with the customer's user name and password and asks the LDAP server whether those credentials are correct.

If the credentials are correct, the approval is sent back to the ConSol CM server and the customer is logged into CM/Track.

> Please keep in mind that the LDAP connection is only used to authenticate the customer (confirm the identity). The authorization (i.e., the assignment of access permissions in the system) is done via the assignment of a CM/Track user profile in the Web Client. The CM/Track user profiles are managed in the engineer and role administration in the Admin Tool.

Please see also the following picture for an explanation of the CM/Track authentication process using LDAP.

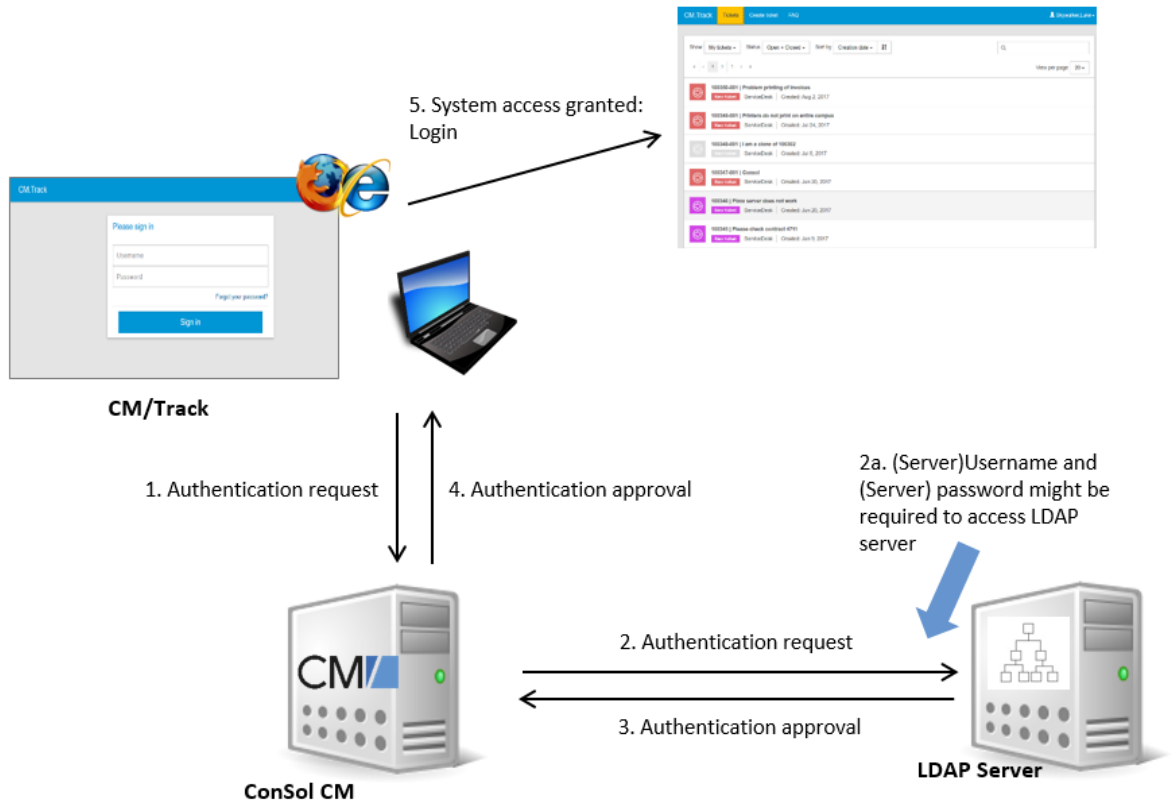


Figure 575: *ConSol CM - LDAP authentication process for CM/Track*

Configuring LDAP Authentication

LDAP authentication is activated by setting the system property cmas-core-security, contact.authentication.method to “LDAP”.

Then you have to set the required values in the system properties (navigation group *System*, navigation item *System Properties*, see System Properties).

The LDAP servers can be defined using the following system properties from the module `cmas-core-security`.

{name} is a string that you can choose to distinguish LDAP servers. It must always be set, even if only one LDAP server is configured. You should use a simple string for the {name}, not containing any keywords, like *internal* or *external*, and which does not contain special characters.

- contact.authentication.method
  LDAP
- ldap.contact.{name}.providerurl
  The property value is the address of the LDAP server in the form *ldap[s]://host:port*.
- ldap.contact.{name}.userdn
  The value is the user DN used to look up the contact DN by the LDAP ID. An anonymous account is used if the value is not set.
- ldap.contact.{name}.password
  The property contains the password to look up the contact DN by the LDAP ID. An anonymous account is used if the value is not set.
- ldap.contact.{name}.basedn
  This represents the base path to search for the contact DN by the LDAP ID, e.g., “ou=accounts,dc=mycompany,dc=de”.
- ldap.contact.{name}.searchattr
  The property value stands for the attribute to search for the contact DN by the LDAP ID, e.g., “uid”.

Initially, these system properties might not be present in your CM system. Just add them manually. Changes to any of the above system properties do not require a server restart and are propagated to all cluster nodes. The use of the placeholder {name} allows configurations to define several different LDAP servers.

- ldap.initialcontextfactory
  This is a predefined global property. If it is not set, “com.sun.jndi.ldap.LdapCtxFactory” is used as its value.

Authentication attempts against LDAP servers are made until first success, where the server order is determined by their {name} values (ascending alphabetical order of the values).

Mixed Authentication Mode

Set the system property cmas-core-security, contact.authentication.method depending on the desired order of authentication instances:

- **LDAP,DATABASE:**
  First attempt authentication using the available LDAP server(s), if an LDAP ID is provided. On failure, try a database login if the customer has a database password.
- **DATABASE,LDAP:**
  First attempt a database login if the customer has a database password. On failure try authentication using the available LDAP server(s) if an LDAP ID is provided.

The CM system will first contact the instance which is mentioned first, than the second one. For example, when the contact authentication method is set to “LDAP,DATABASE” and the customer (contact) uses the password which is only valid in the database, the login will succeed.

In `server.log` the following message will be displayed:

```
LDAP login failed: [LDAP: error code 49 - Invalid Credentials]; nested exception is
 javax.naming.AuthenticationException: [LDAP: error code 49 - Invalid Credentials]
```

Logging of LDAP Login Attempts in CM/Track

All LDAP errors encountered are logged without a stack trace using loggers with the following prefix:

- `com.consol.cmas.core.security.contact`

The stack trace of LDAP errors is not logged because failed login attempts on the first LDAP server would clutter logs if a following login on the second LDAP server succeeded.

Using LDAPS (LDAP over SSL)

Introduction

Per default, when an LDAP client accesses an LDAP server, the information is transferred in clear text. In case you want the user name and password to be transferred to the LDAP server in encrypted form, you have to set up the LDAP authentication using LDAPS.

Preparations

You have to configure the CM server machine (Java) in a way that can use certificates. One way to do this for a Linux environment is described in the following section.

1. Retrieve the certificate:
   `openssl s_client -connect dc2.mydomain.com:ldaps`
2. The answer will contain a section which starts with "---BEGIN CERTIFICATE " and ends with "END CERTIFICATE ---".
   Copy this section to a file, e.g., `/tmp/certificate2_dc2_mydomain_com.txt`
3. Import the certificate to the truststore of your machine, e.g., /home/mydirectory/mytruststore
   `$JAVA_HOME/bin/keytool -import -alias <arbitrary> -trustcacerts -keystore /home/mydirectory/mytruststore -file/tmp/certificate2_dc2_mydomain_com.txt`
   You have to enter (set) a password.
4. Enter the truststore in the ConSol CM config file in JAVA_OPTS:
   `-Djavax.net.ssl.trustStore=/home/mydirectory/mytruststore -Djavax.net.ssl.trustStorePassword=<see above>`

LDAPS Configuration in the ConSol CM Admin Tool (System Properties)

Configure the ConSol CM server as shown in the following example:

- cmas-core-security, ldap.authentication = simple
- cmas-core-security, ldap.contact.name.basedn = OU=myOU,DC=myDC
- cmas-core-security, ldap.initialcontextfactory = com.sun.jndi.ldap.LdapCtxFactory
- cmas-core-security, ldap.contact.name.password = myLDAPpw
- cmas-core-security, ldap.contact.name.searchattr = sAMAccountName
- cmas-core-security, ldap.contact.name.userdn = myLDAP_UserDN

Depending on the LDAP server configuration, use one of the following values for the server URL:

- **Standard LDAPs port**
  cmas-core-security, ldap.contact.name.providerurl = ldaps://dc2.mydomain.com:636
- **LDAPs port Global Catalogue**
  cmas-core-security, ldap.contact.name.providerurl = ldaps://dc2.mydomain.com:3269

Setting Up Customer Accounts for LDAP

There are two steps which you need to perform to set up LDAP authetication for customers using CM/Track:

- Set the required annotation for the customer field which should hold the LDAP ID in the Admin Tool.
- Enter the LDAP IDs for the actual customers in the Web Client.

When LDAP mode is used, the customer field which is used for the CM/Track user name (login) has to have two annotations:

- `username` = true
- `ldapid` = true

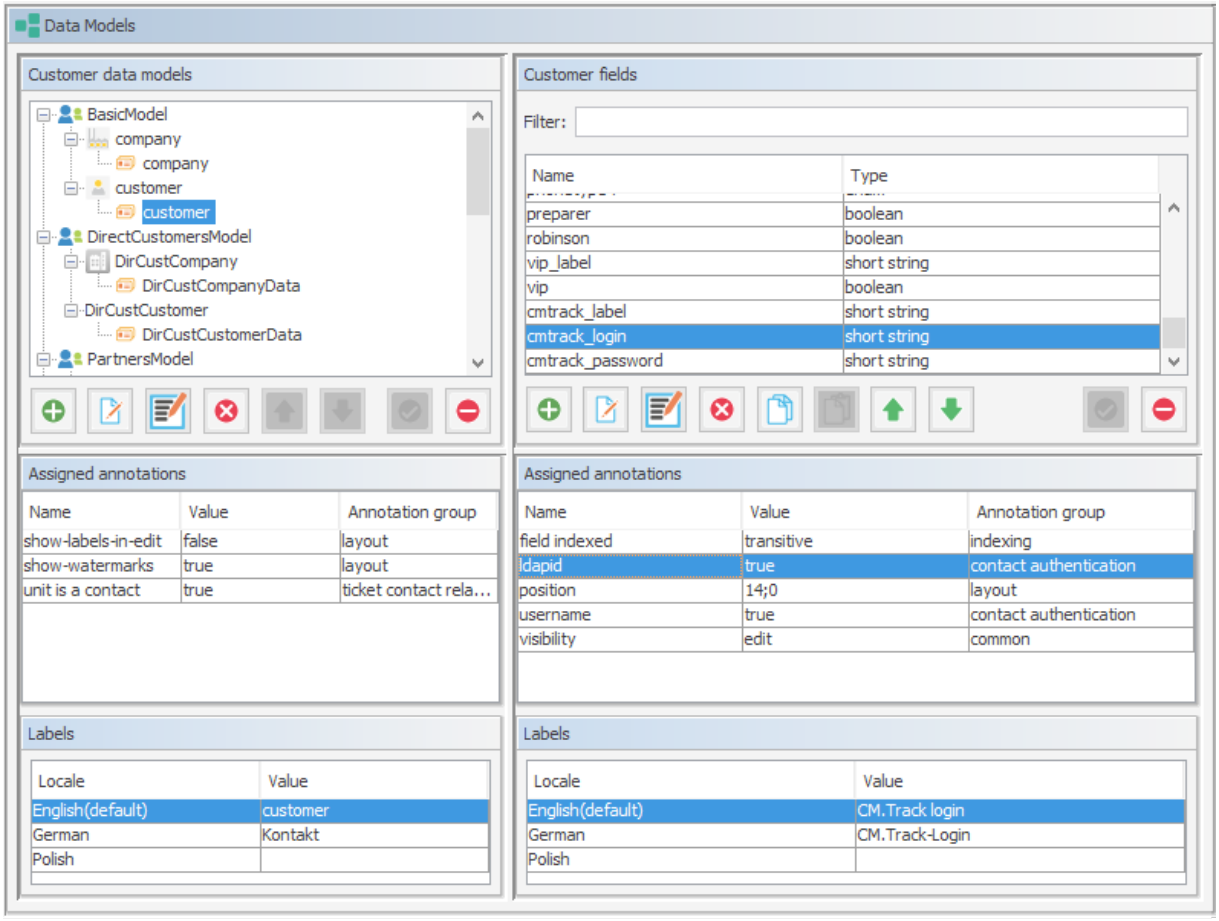


Figure 576: *ConSol CM Admin Tool - Customers, Data Models: Customer field for LDAP authentication of CM/Track users*

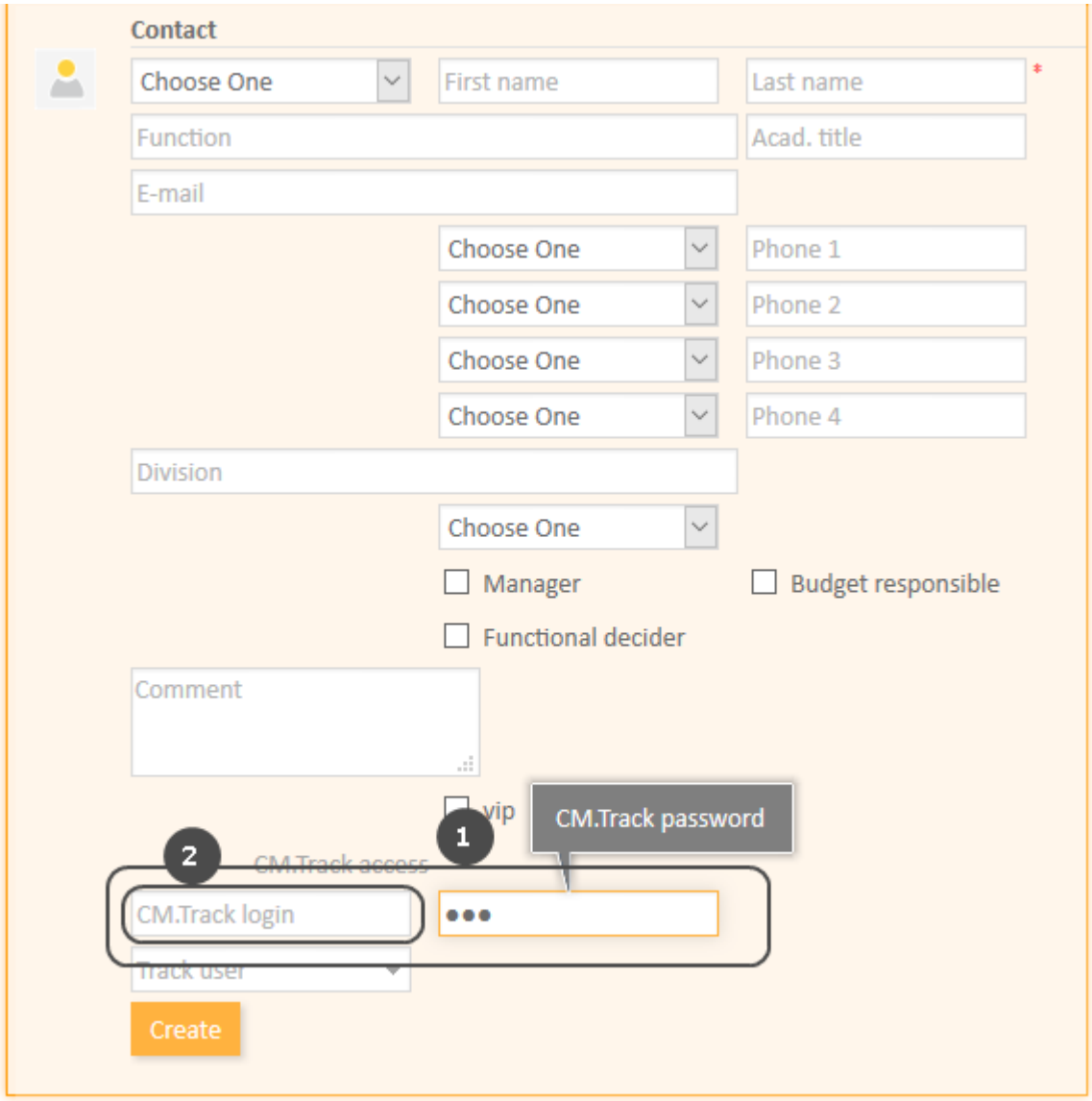


Figure 577: *ConSol CM Web Client - Field for LDAP ID in contact data*

- Used for database authentication in case of mixed mode (1)
- Used for LDAP authentication (2)

F.15 ConSol CM External Interfaces

ConSol CM is not a stand-alone system but can be integrated into your company's IT infrastructure. One possibility of integrating CM is using one of the external interfaces:

- The **REST API**
  - This is used for the implementation of CM/Track, see section CM/Track: The Customer Portal.
  - If you want to develop your own application which uses the REST API, please ask your ConSol CM consultant for a REST API documentation.
- The **Webhooks** interface
  - This offers the possibility to easily integrate CM with such applications as chat tools, shopping systems, or social media platforms.
  - See section Webhooks for details.
- **The ETL interface**
  - This provides the interface for ETL (extract- transform - load) tools. Those tools might be used to import large numbers of objects, e.g., customers, into the ConSol CM database. For a detailed explanation, please refer to the *ConSol CM ETL Manual*.

F.15.1 Webhooks

F.15.1.1 Introduction

ConSol CM Webhooks allow easy integration of ConSol CM with third-party applications, for example shopping applications, social media platforms, or chat tools. In order to communicate with those tools, the CM Webhooks module can provide several services so that you can integrate CM with several tools at the same time.

The basic steps of the information transfer are the following. Please see section Some Background Knowledge about ConSol CM Webhooks for a detailed explanation.

- The external application sends a HTTP request to ConSol CM, namely to the specific service based on the Webhooks technology.
- The request is processed by the CM Webhooks module. The functionalities of the specific service are implemented by an Admin Tool script.
- The answer is sent back to the requesting site which has to process the answer.

F.15.1.2 Example Use Case

The following example shows how you could create a new contact in ConSol CM easily by grabbing customer data in a (fictional) social media platform.

Figure 578: *Creating a new ConSol CM contact by using Webhooks service, fictional example of use with a social media platform*

F.15.1.3 Some Background Knowledge about ConSol CM Webhooks

Basic Principle

The basic principle of the Webhooks interface can be summarized as follows:

- The Webhoooks interface can provide any number of services. The only limitation is the system performance.
- A service is implemented using an Admin Tool script of type *Integration*.

- For each service, a security provider has to be defined which checks the requests. For some background information about this topic, please read section Secret Tokens. To learn how to configure the required security provider, read section Step 2: Configure the new service, namely the security providers for the service.
- If service as well as (at least one!) security provider have been set up correctly, the new service is available using the respective URL:

```
https://[CM-Server-Address]/intg/<Admin Tool script name>/service
```

- When the service has been set up correctly, requests can be processed:
  - The requesting application sends a JSON statement in a HTTP POST request to the service URL. The content type has to be “application/json”.
  - The security provider checks the request. If IP as well as token check are configured:
    - First the IP address is checked: if it is in the permitted range, the request can proceed. See also section Definition of the Valid Range of IP Addresses.
    - If the IP is not in the required range, the secret token is checked subsequently if it is correct, the request can proceed as well. See section Secret Tokens.
    - If neither of the two criteria is met, the request is rejected, and an “authorization failed” message is displayed.
  - If the security criteria are met, the request can proceed and is processed by the Admin Tool script which implements the specific service.
  - The answer is sent back to the requesting application as object of the class `IntgServiceResponse`.

    ⚠ Please note the change between CM version 6.11.1.0 and 6.11.1.1:
    In CM 6.11.1.0, Webhook integration scripts return a JSON statement, in CM versions 6.11.1.1 and up, Webhook integration scripts return an object of class `IntgServiceResponse`. This object can contain a JSON statement in the message body.

  - The requesting application has to process the answer as required.

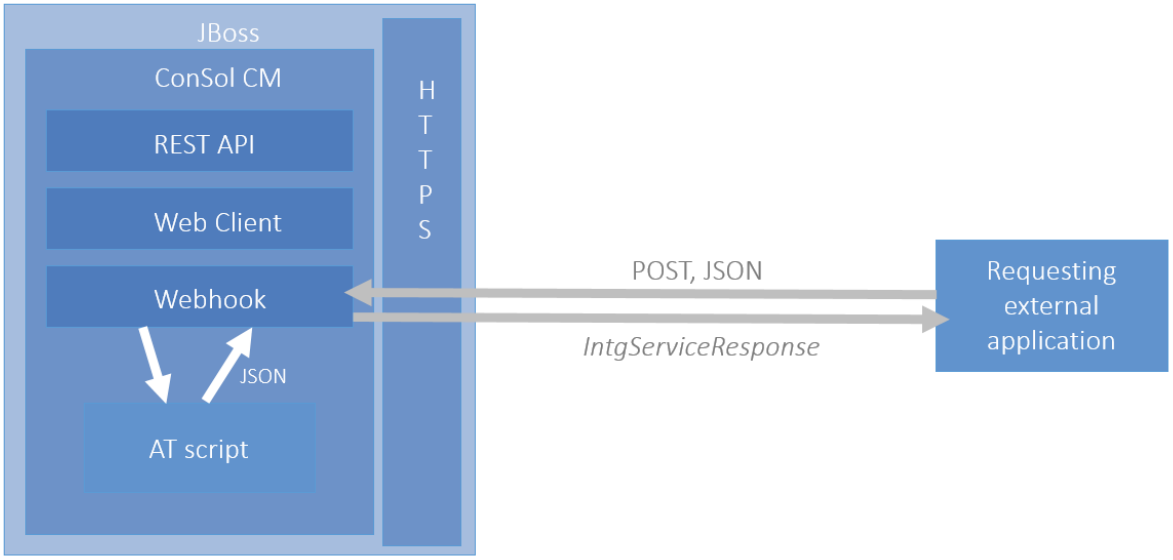


Figure 579: *Basic principle of the ConSol CM Webhooks*

The response can contain custom-defined objects of the following types:

- httpStatusCode
- httpHeaders
- body

For details, please see the following section.

> ⚠ It is highly recommended to use HTTPS as a communication protocol for transferring data using a ConSol CM Webhook.

Admin Tool Script of Type *Integration*

The script defined in the Admin Tool has to be of type *Integration*. The script implements the logic for dealing with the requests sent. The script name without the extension ".groovy" will also be the service name used for the URL to access the interface. For example, to serve the following URL, `https://10.20.3.40:8080/intg/myWebHook/service`, an *Integration* type script called "myWebHook.groovy" is present in the Admin Tool for the interface logic. When the script is executed for processing a request, it runs in an administrative context and has full system access.

A script of type *Integration* returns an object of type `IntgServiceResponse`. Creating a new Admin Tool script of this type puts a basic script template in the editor as a start which contains the code basics of returning an `IntgServiceResponse`.

An `IntgServiceResponse` can contain the following objects:

- **httpStatusCode**
  Defines the HTTP status code. Optional, if not provided, only the default status codes are used (see section below).

- **httpHeaders**
  Optional, defines the response header. Only required if you want to define a specific httpHeader, i.e., one which is not part of the supported standard headers (see section below).
- **body**
  Defines the content of the response. Optional, if not provided, the response is empty. This is the object which can contain a JSON statement, see the following code examples.

The following default HTTP status codes are supported:

- **200 (OK)**
- **401 (Unauthorized)**
  One or more security provider is enabled but the data provided by the client did not match the expected data (e.g., wrong token or IP address)
- **404 (Not Found)**
  A script for webhook with the given name could not be found / webhook is disabled / no security provider is enabled for webhook
- **405 (Method Not Allowed)**
  Any HTTP method other than POST
- **500 (Internal Server Error)**
  Any problems/exceptions on the server side (CM)

The example script below (section Step 1: Build the Admin Tool script) returns a simple JSON object with ticket number, subject, and engineer as response on a request with the ticket number. It returns "n/a" for invalid ticket numbers or a missing engineer assignment. Before returning the response, it logs the request and response content.

Definition of the Valid Range of IP Addresses

The IP-based filter, when enabled by checking the checkbox, uses a list of IP address ranges defined in the CIDR notation (Classless Inter-Domain Routing). A valid entry in this notation is an IP address followed by a slash separating it from a number of bits defining the subnet mask. The example "10.20.30.40/24" stands for the IP address "10.20.30.40" and its subnet mask "255.255.255.0". The address list for service configuration allows multiple IP ranges in this notation with each range entry on a separate line.

Secret Tokens

Using interface access with secret tokens must be enabled by checking the checkbox *Enabled* as well. Each token to be used must be added as a string on a separate line in the list *Tokens* in the last input box on the page. The checkbox *HMAC* activates the validation of the request by a mandatory hash based on the token. The HMAC (Hash-based Message Authentication Code) value is derived from the unencrypted payload and the shared secret. This code is transferred in the header of the request. The only hashing algorithm supported by ConSol CM in this case is SHA-1. MD5 is not supported!

A corresponding hashing of the response is currently not supported.

The input field *Header* allows to define the HTTP header field name which shall be used to transfer the shared secret token. The value defaults to "SECRET_TOKEN", but it is recommended to change this value, if possible. A non-standard name can mean a security improvement.

Please be aware that for a *Webhook* integration there is no other mandatory access validation beyond the security providers, specifically no login-based authentication as for engineers which imposes additional access restrictions. The interface has full system access, so securing it additionally with a proxy is mandatory for real-life usage.

When using the shared secret security provider, the token is accessible for the script from the headers map in a field named as defined in the configuration web interface under *Headers*.

Provided the default “SECRET_TOKEN” is used, it is available in the script from this HTTP header field.

This can also be returned explicitly in the response, if desired, please compare the example code snippet:

```
...
token = headers.get('secret_token')
...
JsonOutput.toJson([ticketId: ticket.getId(), token: headers.get('secret_token')])

return response
```

Please note that the service behavior is completely based on the implementation of the corresponding script, i.e., the security based on the provided tokens has to be implemented and is not system-immanent!

Configuration of Webhook Services

Before the new service can be used, it has to be configured. This has to be done using the *Configuration* page which is offered under the following URL:

```
https://[CM-Server-Address]/intg/<Admin Tool script name>/config
```

Log in with an admin account (a ConSol CM engineer with administrator permissions).

Upon successful login, the configuration page is shown with a form to enter the relevant information. For an example of a configuration page, please see the figure in section Step 2: Configure the new service, namely the security providers for the service

The following parameters can be configured:

- General information:
  - Display of the relative URL to access the *Webhooks* service
  - Checkbox to enable/disable the service generally, must be checked for the interface to operate. Please note that at least one security provider has to be configured to activate the service. The checkbox alone does not activate it!

- Security providers configuration:
  - IP-based filtering to limit access to defined IP ranges:
    - Checkbox to enable/disable IP-based filtering, must be checked to enable the access control based on the IP address
    - List with IP address ranges in CIDR notation (see below)
- Secret tokens configuration to request a shared secret for access:
  - Checkbox *Enabled*, to enable/disable access by token, must be checked for access by token
  - Checkbox for HMAC to use a secure hash for message authentication (see below)
  - HTML header name for the shared secret, by default “SECRET_TOKEN”
  - List with access tokens

Permissions

The Webhook script is executed with administrator privileges. This means that all operations performed in the script are executed with these permissions. If this is not desired, the script can include the credentials of an engineer and some code to downgrade the session to the engineer’s privileges. The method `executeWithUserPermissions` of the class `SecurityTemplate` is used for this purpose.

The following code example shows the usage of this method in a script. The session is downgraded to the permissions of the user identified by the login and password included in the JSON payload. Then, a customer search for the customer with the ID “12345” is performed with these permissions.

```
import com.consol.cmas.common.security.template.SecurityCallbackWithoutResult;
import com.consol.cmas.common.security.template.SecurityTemplate;
import groovy.json.JsonSlurper

def jsonSlurper = new JsonSlurper()
def message = jsonSlurper.parseText(payload);
SecurityTemplate.executeWithUserPermissions(message.login, message.password, new
 SecurityCallbackWithoutResult() {
  @Override
  public void doInSecurityContextWithoutResult() {
     unitService.getById(12345)
  }
});
```

Code example 103: *Example of downgrading the session to an engineer’s privileges*

F.15.1.4 Example of Implementing a Service Based on CM Webhooks

The following example shows the implementation of a service named “myWebHook”.

Step 1: Build the Admin Tool script

The script is named “myWebHook.groovy” and is stored in the script section of the Admin Tool.

```
//////////////////////////////////////////////////////////////////////
//// Variables available in script: ////
//// headers - headers as map ////
//// payload - request payload as json ////
//// Script should return json as result ////
//////////////////////////////////////////////////////////////////////
import com.consol.cmas.intg.service.IntgServiceResponse
import groovy.json.JsonSlurper
import groovy.json.JsonOutput
import com.consol.cmas.common.service.TicketService
def jsonSlurper = new JsonSlurper()
def message = jsonSlurper.parseText(payload)
def ticketname
def ticketsubject
def engineername
log.info 'PAYLOAD is NOW ' + payload
def OK = false
if (message.ticket) {
   ticket = ticketService.getByName(message.ticket)
   if (ticket) {
      ticketname = message.ticket
      log.info 'TICKETNAME is now ' + ticketname
      ticketsubject = ticket?.getSubject()
      if (ticketsubject) {
         log.info 'TICKET SUBJECT is now ' + ticketsubject
      } else {
         log.info 'NO SUBJECT'
      }
      if (ticket.engineer) {
         engineername = ticket.engineer.getFirstname() + " " +
          ticket.engineer.getLastname()
      }
      OK = true
   }
}
def response = new IntgServiceResponse()
if (OK) {
   log.info "Webhook [Search engineer for ticket] returned: {ticket: " + ticketname
    + ", subject: " + ticketsubject + ", engineer: " + engineername + "} on request
    [" + message + "]"
   response.httpStatusCode = 200
   response.body = JsonOutput.toJson([ticket: ticketname, subject: ticketsubject,
    engineer: engineername]) // optional
} else {
   log.info 'No hits found in webhook request ... '
   response.httpStatusCode = 404
   response.httpHeaders = ['Content-Language':'en', 'Warning':'Required Objects not
    found']
}
return response
```

Code example 104: *Example Admin Tool script which implements a Webhooks service (with some log.info statements for test purposes)*

As soon as the script has been implemented, the configuration of the new service can be performed. See next section.

Step 2: Configure the new service, namely the security providers for the service

Use the following URL to configure the service:

```
https://[CM-Server-Address]/intg/<Admin Tool script name>/config
```

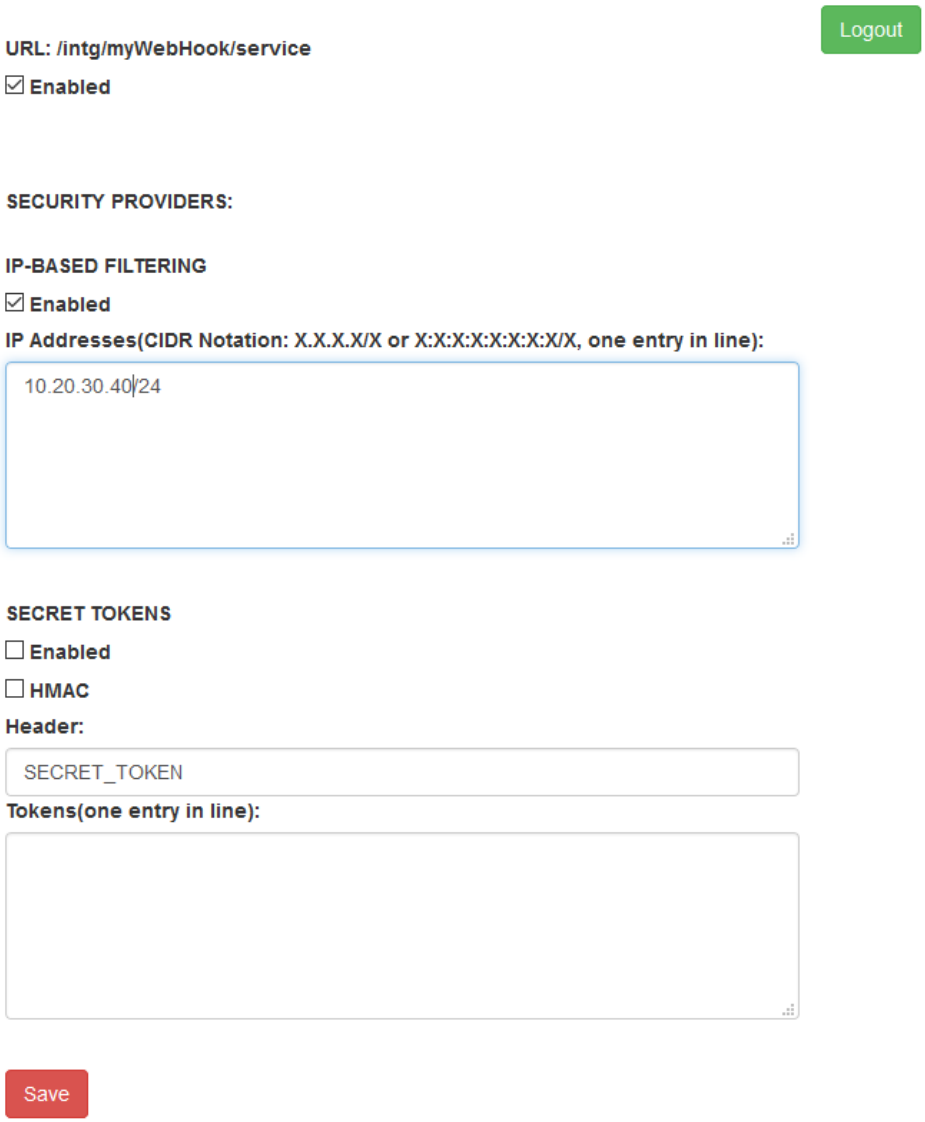


Figure 580: *Configuration page of a Webhooks-based service*

In this example, only the IP range is checked.

Step 3: Test the new service

If a machine with an IP address within the valid range sends a request to the new service, the response is a simple JSON output. You can, for example, use a REST plugin in the web browser to test the service. The *content-type* has to be “application/JSON”.

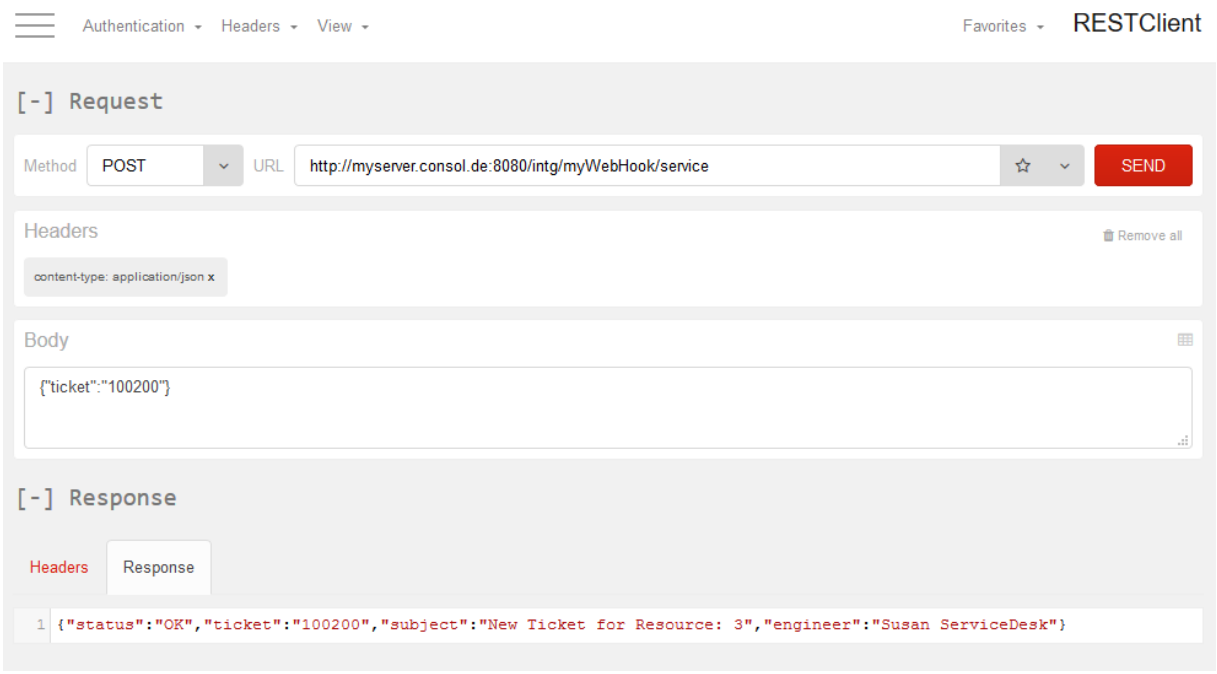


Figure 581: *Test of the Webhooks service myWebHook using a REST plugin in a web browser, ticket has been found in the system*

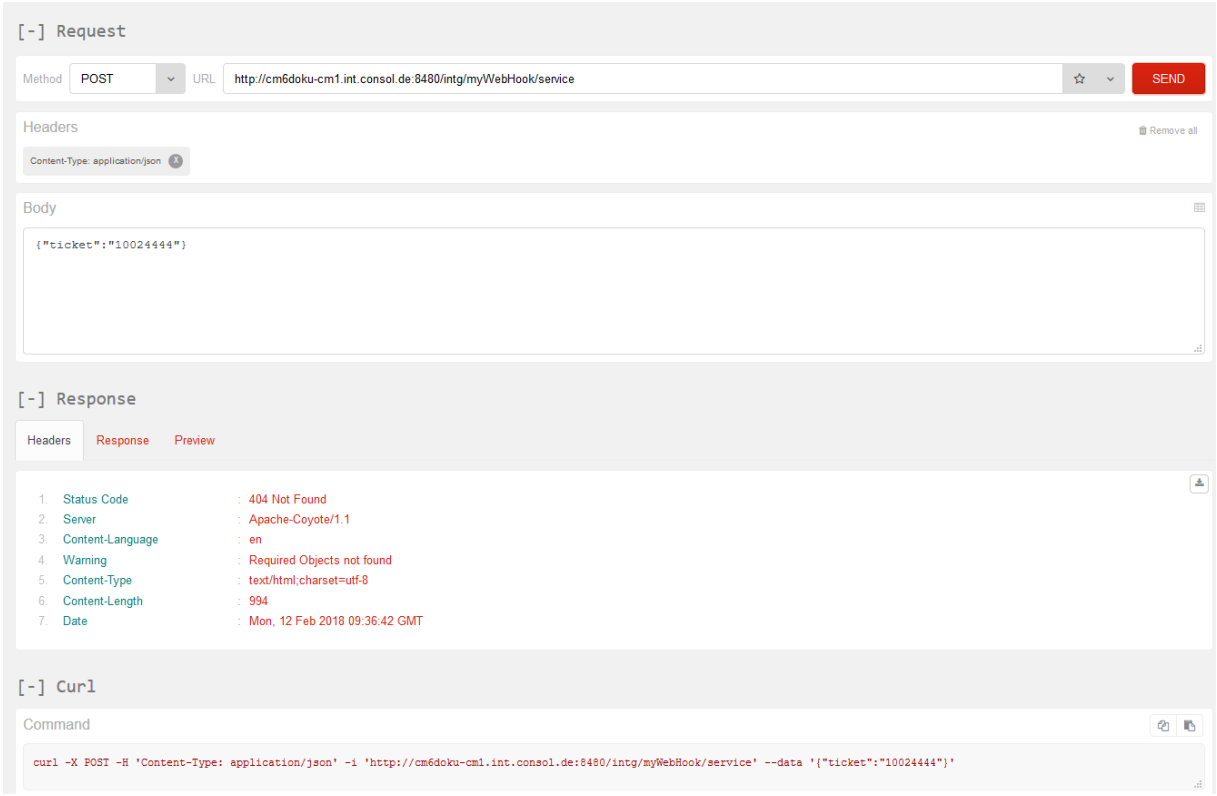


Figure 582: *Test of the Webhooks service myWebHook using a REST plugin in a web browser, ticket has not been found, custom-defined error code returned*

You can also use a command-line tool:

```
# Linux:
curl -X POST https://myserver.consol.de:8080/intg/myWebHook/service //
-H 'content-type: application/json' //
-d '{"ticket": "100200"}'

# Windows (note the different quoting/escaping for the -d option):
curl -X POST https://myserver.consol.de:8080/intg/myWebHook/service //
-H "content-type: application/json" //
-d "{\"ticket\": \"100200\"}"
```

The request payload in this example is the value of the `-d` option.

The response generated by the script and returned to the requester is a little more elaborate (Please note that this JSON has been heavily reformatted for better legibility!):

```
{
   "status" : "OK" ,
   "ticket" : "100200" ,
   "subject" : "New Ticket for Resource 3" ,
   "engineer" : "Susan ServiceDesk"
}
```

F.16 System Architecture

ConSol CM is a Java EE application. It can be operated as

- **ConSol CM-only system**, see section Basic System Architecture

or as

- **ConSol CM system with reporting infrastructure**, see section System Architecture with Reporting Infrastructure

Furthermore, ConSol CM can be installed and operated in an application server cluster. This is explained in detail in the *ConSol CM Cluster Manual*.

F.16.1 Architecture of a CM System

F.16.1.1 Introduction to ConSol CM System Architecture

ConSol CM is a ***Java EE*** (Java Enterprise Edition) application that can be run in a standard application server on Unix/Linux or Windows systems. JBoss and Oracle WebLogic are supported.

In this chapter, a short overview of the ConSol CM system architecture will be provided.

> A detailed list of supported operation systems, application servers, database systems, and other systems, as well as storage and CPU requirements is given in the current *System Requirements*.

F.16.1.2 Basic System Architecture

ConSol CM is a Java EE application which is based on the classical three-tier architecture. The ConSol CM server is deployed in an application server and accesses a relational database. Two web interfaces are available as client interfaces: the standard interface is the ConSol CM Web Client, which is used by the engineers to work on the tickets. Another web client is the ConSol CM portal, CM/Track. This provides access to the system for customers who might want to know some basic facts about the status of their tickets. The two Java applications which are used to configure ConSol CM are the Admin Tool and the Process Designer. Both can be downloaded from the ConSol CM start page using Java Web Start (JWS). JWS is a component of every recent Java edition, so no extra installation is required on the PCs or Laptops you want to use to administer the system. On the contrary - you can do this from every regular web client with a supported web browser. Please make sure that the versions of all components which are used in your company meet the system requirements.

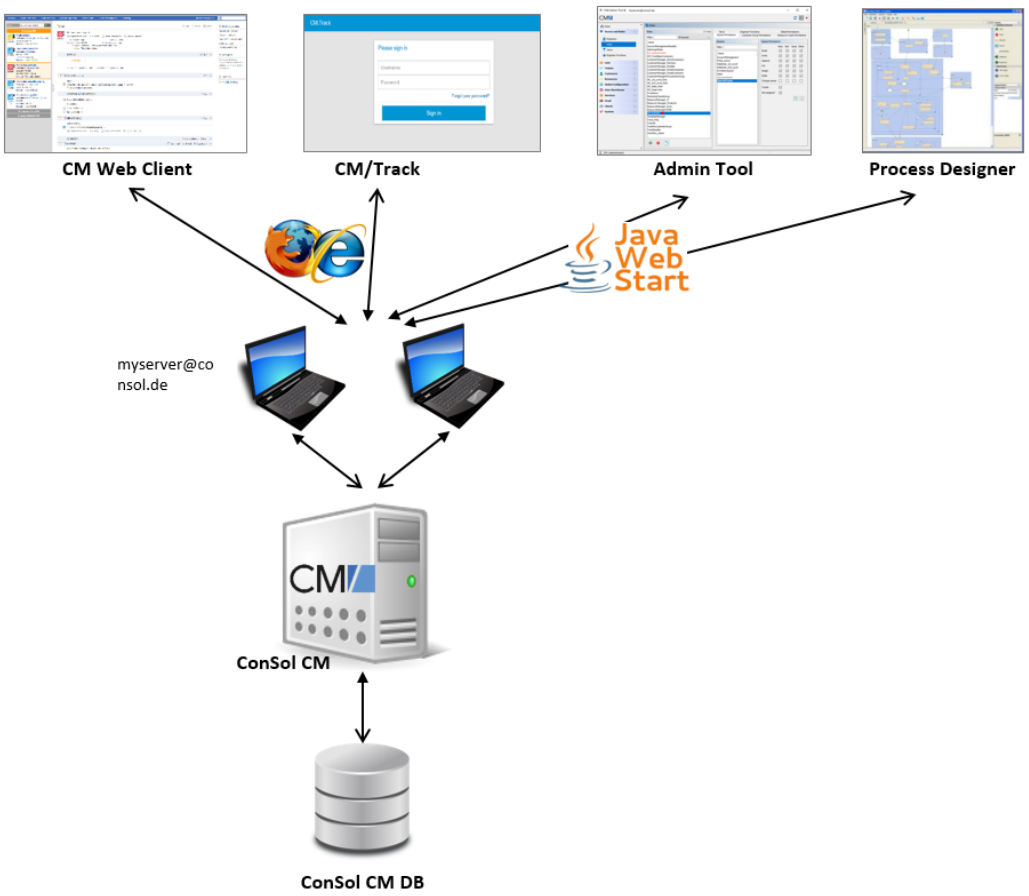


Figure 583: *ConSol CM - Basic system architecture*

CM Database

The ConSol CM database (CM DB) is a relational database which can be operated as Oracle, Microsoft SQL Server or MySQL system.

> A detailed list of supported operation systems, application servers, database systems, and other systems, as well as storage and CPU requirements is given in the current *System Requirements*.

- **Oracle**
  One database schema with one database user is used by ConSol CM.
- **Microsoft SQL**
  One database schema with one database user is used by ConSol CM.
- **MySQL**
  One database with one database user is used by ConSol CM.

F.16.1.3 System Architecture with Reporting Infrastructure

In order to allow Business Intelligence (BI) tools or other applications to build specific reports, OLAP cubes, and other analyses, ConSol CM provides a data warehouse (DWH) as one of its standard components. The DWH is a separate database (or database scheme, see below). The DWH is filled by a Java EE application called ConSol CM Reporting Framework (CMRF).

The ConSol CM standard function set comprises two components which enable reporting:

- **CMRF** (ConSol CM Reporting Framework)

  This is a Java EE application which synchronizes the ConSol CM database with the ConSol CM data warehouse (DWH). The CMRF can be deployed into the same application server as the core CM (overlay mode) or it can be run on a separate application server (standalone mode). The synchronization of CM data with the DWH is based on direct messaging. For a detailed explanation, please refer to the *ConSol CM Operations Manual*.section *Operating the Data Warehouse*.

- **DWH** (data warehouse)

  The ConSol CM DWH is a relational database which can be operated as Oracle, Microsoft SQL Server, or MySQL system. It stores the integrated/pre-processed data from the ConSol CM database.

> A detailed list of supported operation systems, application servers, database systems, and other systems, as well as storage and CPU requirements is given in the current *System Requirements*.

Separate application servers for ConSol CM and CMRF (standalone mode):

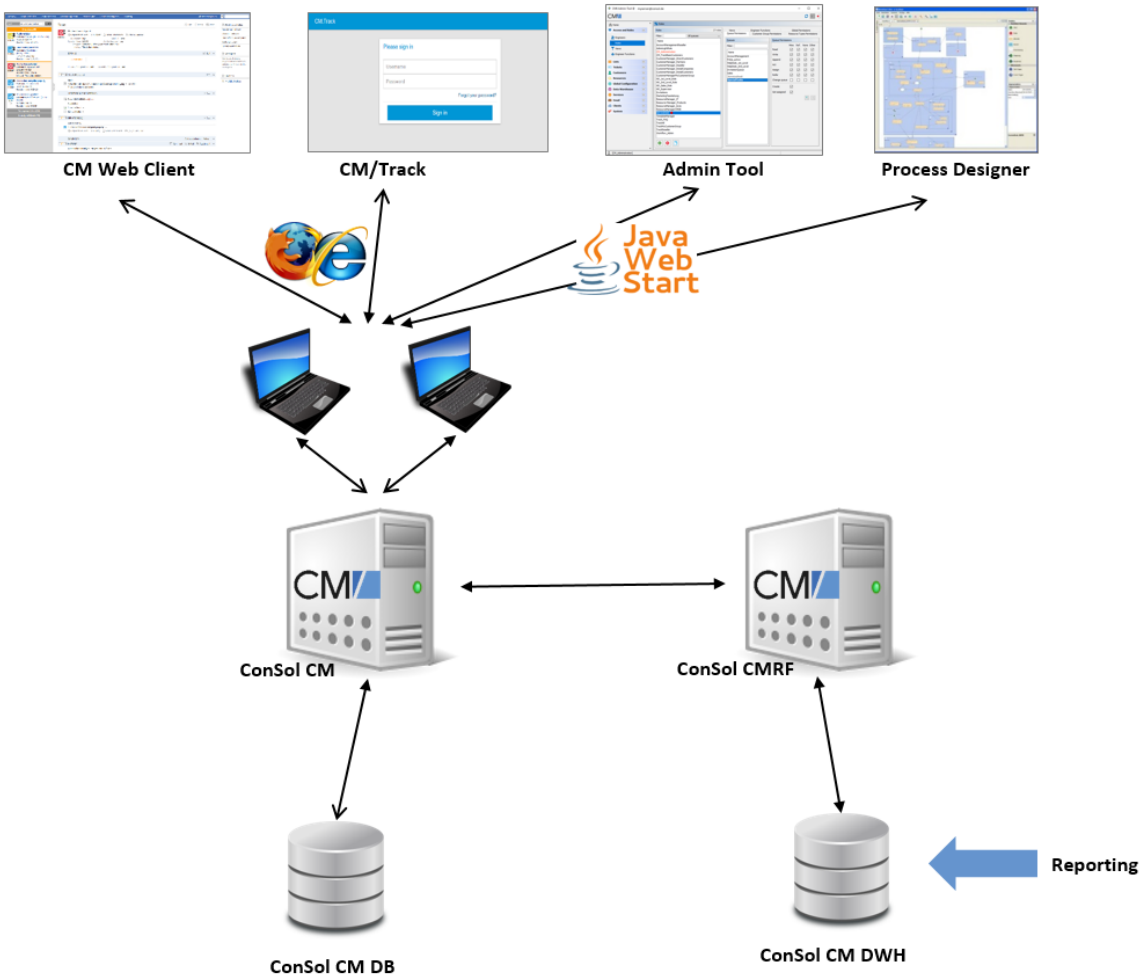


Figure 584: *ConSol CM - Infrastructure with CMRF and DWH (2 servers)*

One application server for ConSol CM and CMRF (overlay mode):

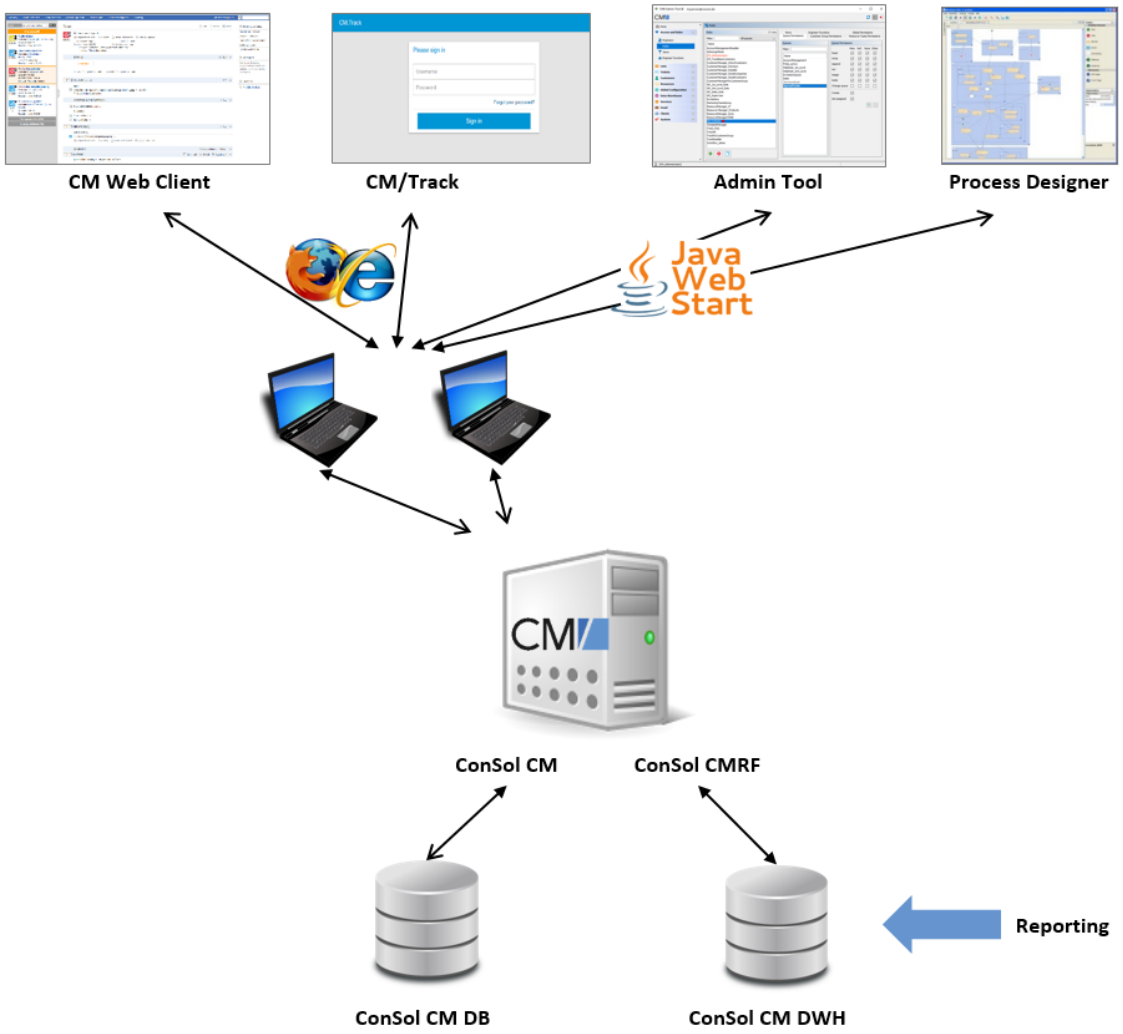


Figure 585: *ConSol CM - Infrastructure with CMRF and DWH (1 server)*

When the DWH has been established, BI (Business Intelligence) applications can be used to create reports, data cubes, and other reporting output formats. Please see the following example with the Pentaho™ BI Suite.

Separate application servers for ConSol CM and CMRF (standalone mode):

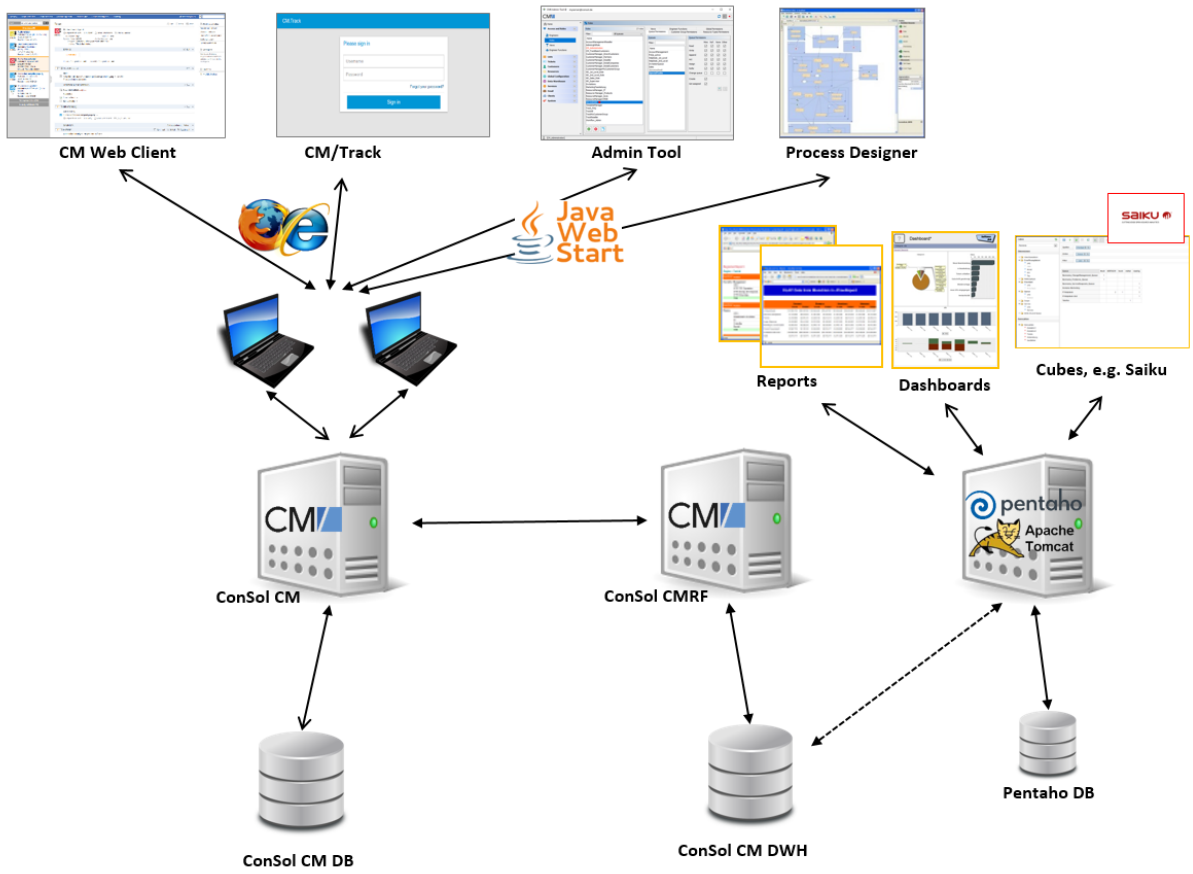


Figure 586: *ConSol CM - Reporting infrastructure (2 servers)*

One application server for ConSol CM and CMRF (overlay mode):

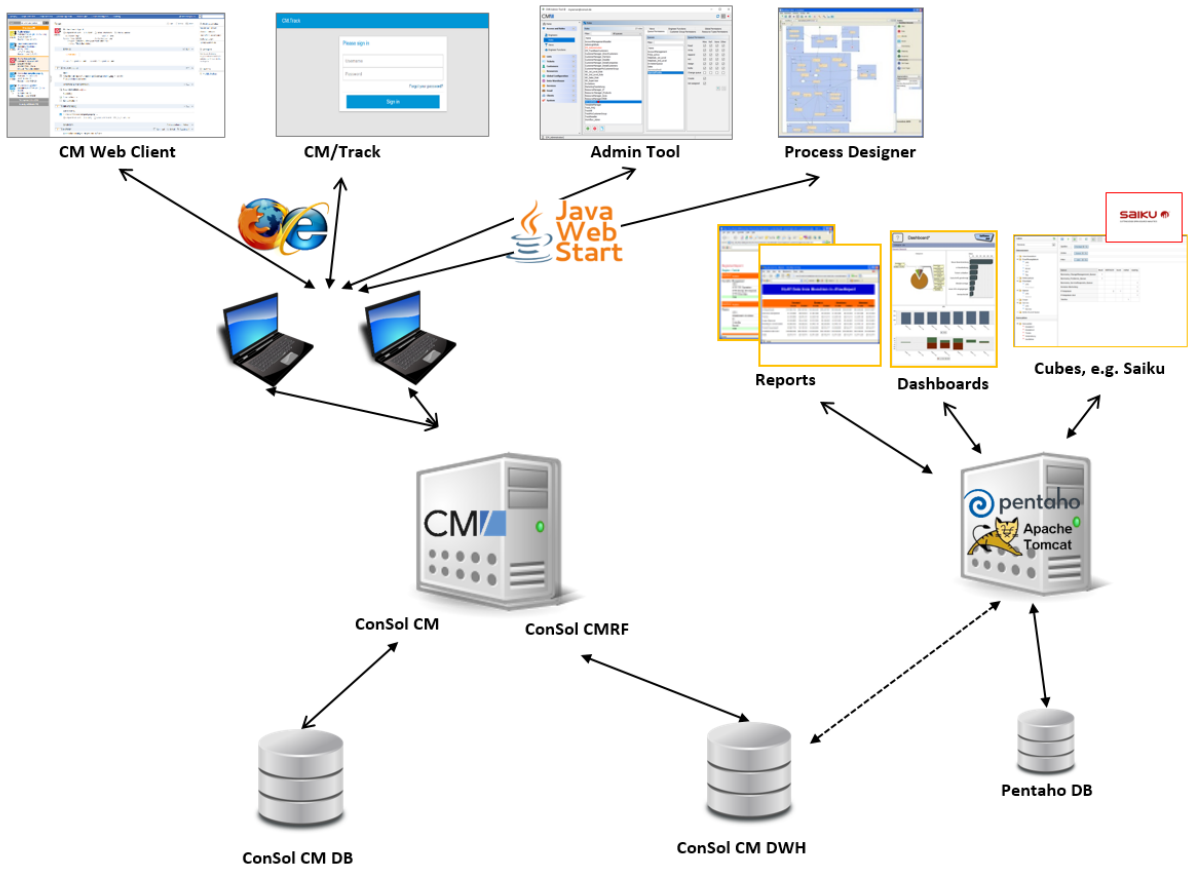


Figure 587: *ConSol CM - Reporting infrastructure (1 server)*

DWH Database

- **Oracle**
  One database scheme with one database user is used by the DWH.
- **Microsoft SQL**
  One database scheme with one database user is used by the DWH.
- **MySQL**
  One database with one database user is used by the DWH.

F.16.1.4 Components for Email Interactions

One of the core functionalities of ConSol CM is integration with mail servers. This allows ConSol CM to send and to receive emails. For the engineer, this means new tickets can easily be opened via email and the entire communication regarding a case is located in the respective ticket, including all incoming and outgoing emails.

In order to receive emails, ConSol CM connects to a mail server and retrieves emails from one or more mailboxes. ConSol CM acts like a regular email client (e.g., Thunderbird, Microsoft Outlook) and uses standard email protocols like IMAP or POP3. If you want to use the secure version, IMAPs and POPs are also supported, in which case the required certificates have to be installed on the server.

In order to send emails, ConSol CM uses an SMTP server.

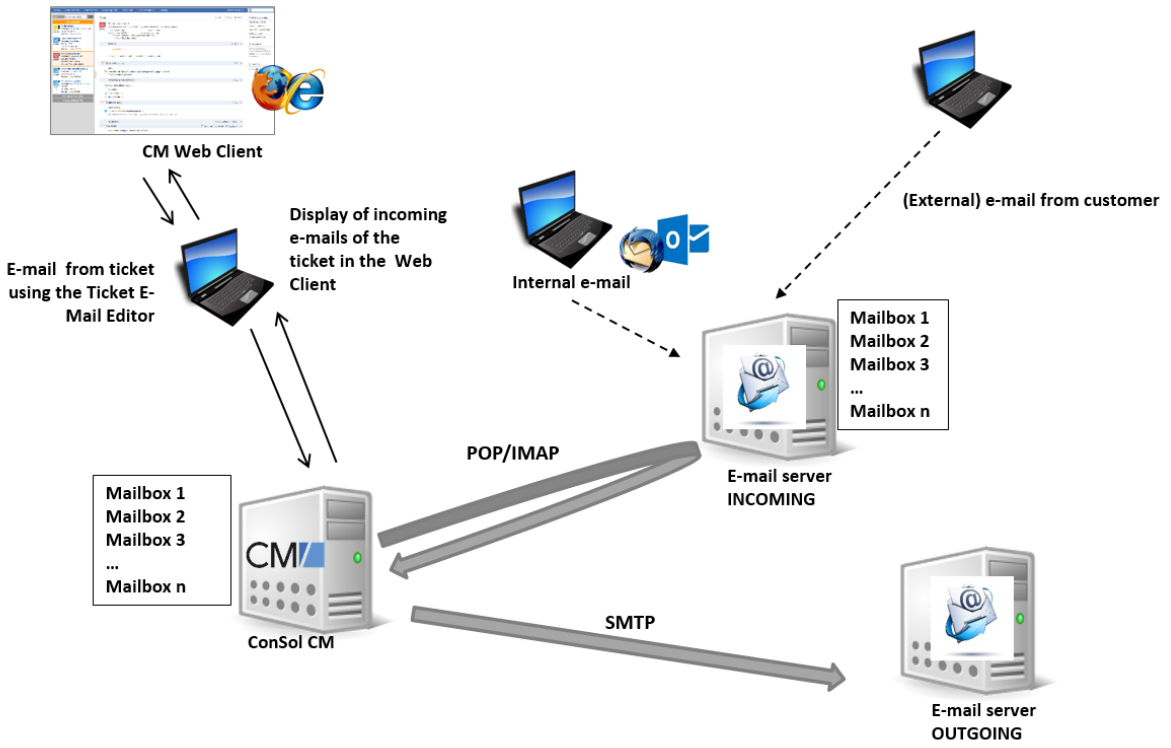


Figure 588: *ConSol CM - Mail server interactions*

F.16.1.5 Indexer

In order to perform effective searches in the database, ConSol CM builds an index for each ticket field, customer field, and resource field which should be included in a search. Furthermore, the engineer data, the ticket comments and the attachments are indexed by default. The indexes are stored in the file system. Please refer to the section ConSol CM File System Structure for an explanation of the index directory structure, and read the detailed introduction to the entire topic in the section ConSol CM Indexer.

F.16.1.6 LDAP Authentication

As standard feature, ConSol CM can use LDAP authentication in the Web Client and/or in the portal (CM/Track). Depending on the configuration of your LDAP server (e.g., Microsoft Active Directory), a user name and password might be required to establish the LDAP connection. All LDAP parameters are stored as ConSol CM system properties.

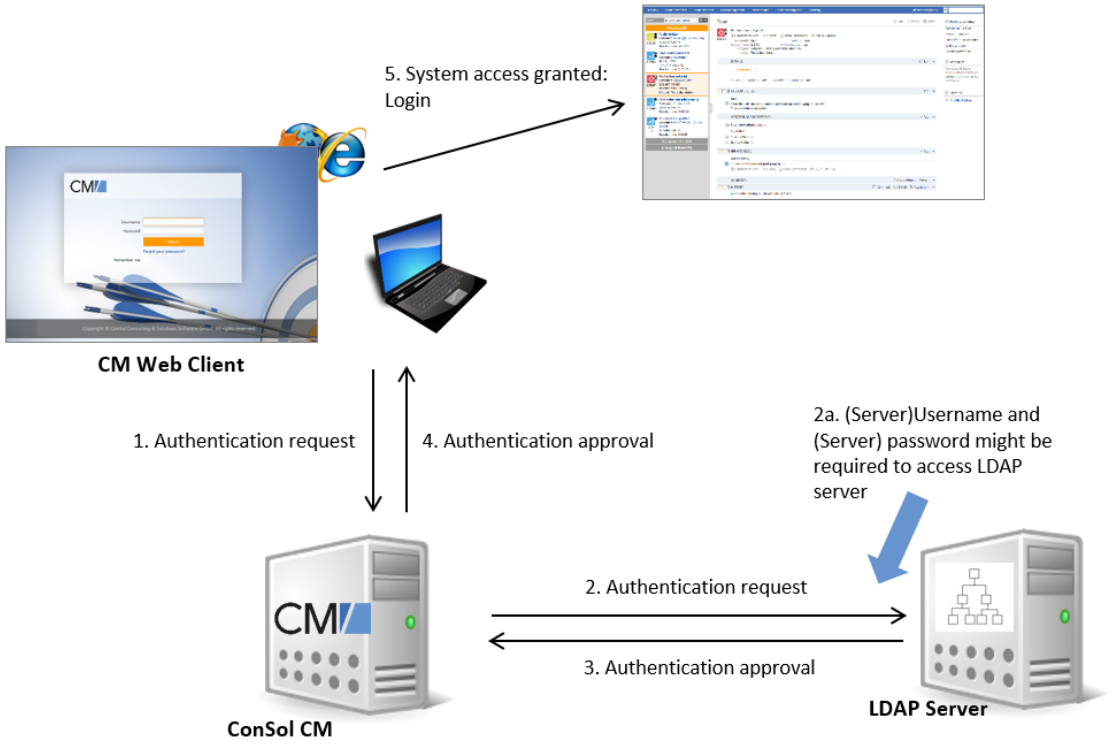


Figure 589: *ConSol CM - LDAP authentication (Web Client)*

F.16.1.7 ConSol CM File System Structure

Most of the data concerning the configuration and operation of ConSol CM is stored in the ConSol CM database. However, some data is saved in the file system in the data directory entered during system setup.

ConSol CM Data Directory

The following figure and list show examples from Windows and Linux systems:

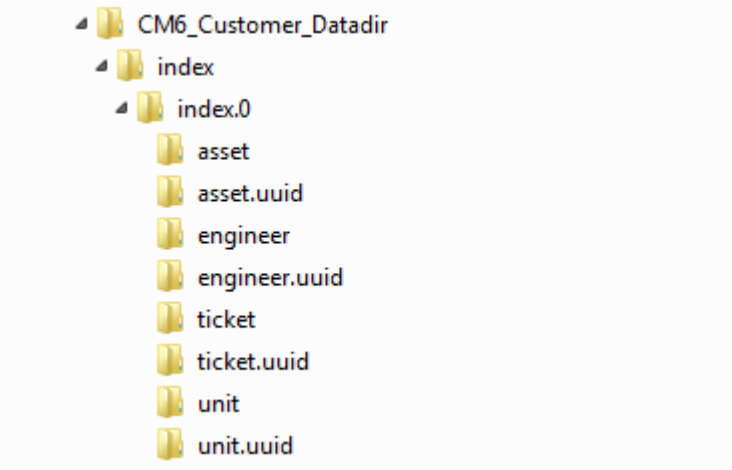


Figure 590: *ConSol CM - Data directory (Windows)*

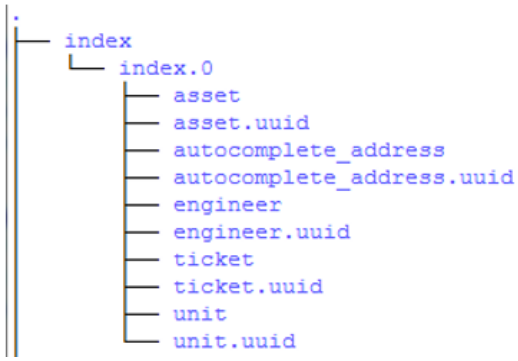


Figure 591: *ConSol CM - Data directory (Linux)*

Example directories:

- *index*

  This is the directory where all the indexes are stored (see also section ConSol CM Indexer). Be sure to include it into your regular file system backup.

  - **index.0**
    In this directory, there is a subdirectory for each required index.

JBoss 7 Application Server File Structure

The following directories are available in a JBoss EAP 6.2 / 6.4 installation of ConSol CM:

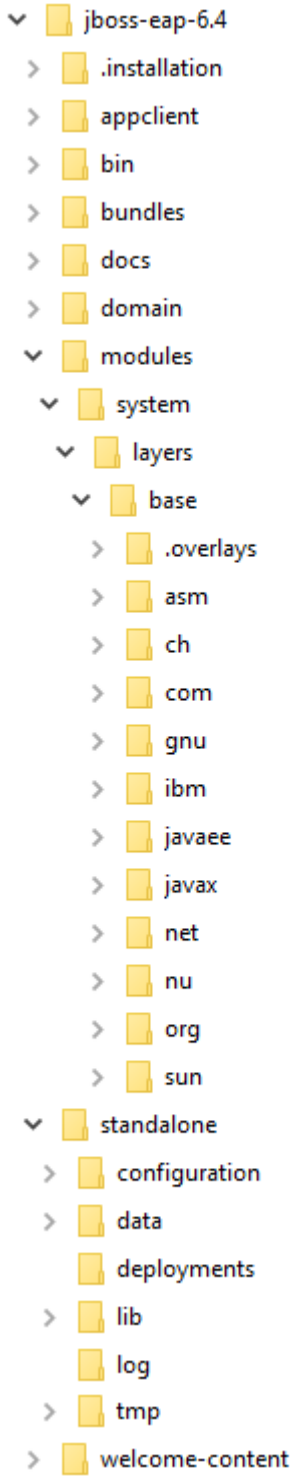


Figure 592: *ConSol CM - File structure in a JBoss EAP 6.4 system*

Example directories:

- **modules\system\layers\base**
  Subfolders contain the JDBC drivers:
  - com\microsoft\sqlserver\jdbc\main\sqljdbc4.jar
    (Microsoft SQL)
  - oracle\jdbc\main\ojdbc6-11.2.0.3.jar
    (Oracle)
  - com\mysql\jdbc\main\
    (MySQL JDBC driver destination, must be installed manually)
- **standalone**
  Configuration in standalone environments:
  - **configuration**
    Configuration of the DB connection and logging in the file `cm6.xml` or `cm6-cmrf.xml`
  - **data**
    Data for operation, e.g., `tx-operation` keys
  - **deployments**
    Deployed applications, for example `cm6.ear` and `cm-track.war`
  - **log**
    Log files, see section Log Files.
  - **tmp**
    Temporary data and also working copy of the application server files. When JBoss is stopped, it can be emptied, e.g. for error analysis and/or fixing.
- **domain**
  Configuration in domain (i.e. multi-server) environments
  - **configuration**
    Configuration of the DB connection and logging in the file `domain.xml`
  - **servers/<server-name>**/log
    Log files

Oracle WebLogic Application Server File Structure

In an Oracle WebLogic environment, ConSol CM is installed as a separate domain. ConSol CM and CMRF are *managed servers*.

Here, only some directories are explained. If you want to administer ConSol CM as a WebLogic application, please also refer to general Weblogic tutorials.

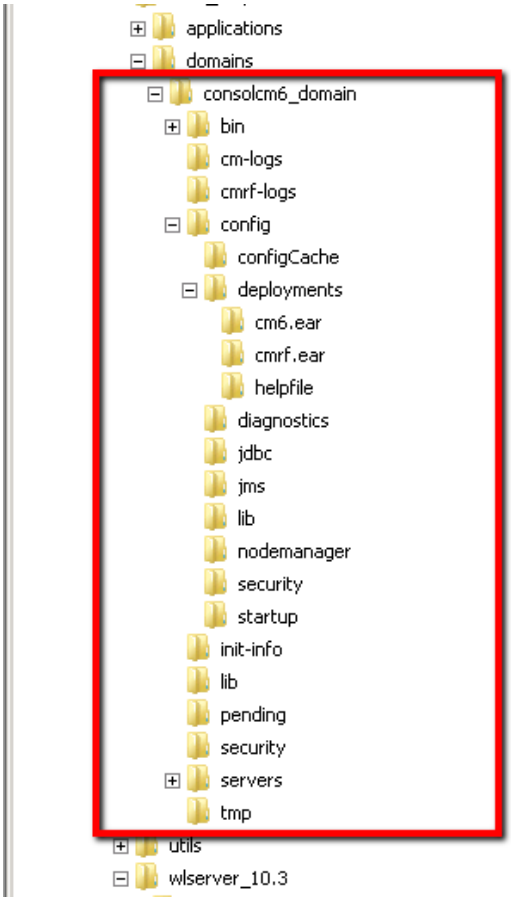


Figure 593: *ConSol CM - File structure in an Oracle WebLogic application server*

Example directories:

- **bin**
  Start/stop scripts
- **cm-logs**
  All log files except for `cmrf.log`
- **cmrf-logs**
  - **cmrf.log file**
    Log messages for the CMRF (ConSol CM Reporting Framework)
- **config**
  Configuration files
- **deployments**
  Deployed applications, i.e., here: ConSol CM and CMRF as directories

F.16.1.8 Logging and Log Files

Introduction

Log files are the main information source for the administrator about the activities of the system and potential problems of the system. The administrator should have a look at the log files on a regular basis. There may be problems that do not appear on the user interface, but are reported in the log file.

The standard path for the log files is:

- **JBoss**: <JBOSS_HOME>/standalone/log
- **WebLogic**: <WEBLOGIC_HOME>\user_projects\domains\<WEBLOGIC_DOMAIN>\cm-logs

Log Files

The location of the log files and the logging behavior can be configured in the respective .xml files:

- **log4j.xml**
  (in WebLogic, where ***Log4J*** is used as logging framework)
- **cm6.xml** or **cm6-cmrf.xml**
  (in JBoss EAP 6.2 / 6.4 standalone, where the built-in logging module of JBoss is used)
- **domain.xml**, tag <subsystem xmlns="urn:jboss:domain:logging:1.3">
  (in JBoss EAP 6.2 / 6.4 in domain mode, where the built-in logging module of JBoss is used)

Log File Types

The following log files are used:

- **access.log**
  Access to and usage of the Web Client is logged. In the current configuration, a separate `access.log` file is written every day.

Config info: How to activate the access.log

In a default installation, the following lines in the configuration file, cm6.xml or cm6-cmrf.xml, are commented out. By commenting them in, you can start access logging in your ConSol CM system.

```
<subsystem xmlns="urn:jboss:domain:web:1.5" default-
 virtualserver="default-host" native="false">
  <connector name="http" protocol="HTTP/1.1" scheme="http"
   socketbinding="http"/>
  <virtual-server name="default-host" enable-welcome-root="false">
     <alias name="localhost"/>
        <!--<access-log pattern='%h %l %u %t %r %s %b %{Referer}i %
         {User-Agent}i %S %T'>-->
          <!--<directory path="./" relative-
           to="jboss.server.log.dir"/>-->
        <!--</access-log>-->
  </virtual-server>
</subsystem>
```

- **cmrf.log**
  Messages pertaining to CMRF (ConSol CM Reporting Framework), i.e., messages regarding the data transfer operations from the ConSol CM database to the CMRF database (DWH).
- **cmweb.log**
  Messages pertaining to the ConSol CM Web Client.
- **ctx.log**
  Contains messages from the Spring Framework.
- **errors.log**
  Contains only messages that have at least the log level `ERROR`.
- **index.log**
  Messages pertaining to the Indexer.
- **mail.log**
  Contains log messages from the email subsystem.
- **operationtimes.log**
  Only used when it has been enabled. Contains timing information for requests in order to identify possible performance bottlenecks.

Config info: How to produce good output in operationtimes.log
In order to produce concise and informative output in this log file, the respective ConSol CM module has to be run in DEBUG mode. Please see configuration examples in the following code snippets.

JBoss configuration:

```
<!-- JBoss7, add these parts to your cm6.xml or cm6-cmrf.xml file -->
<size-rotating-file-handler name="OPERATION_TIMES" autoflush="true">
   <file relative-to="jboss.server.log.dir"
    path="operationtimes.log"/>
   <append value="true"/>
   <rotate-size value="300m"/>
   <max-backup-index value="6"/>
   <formatter>
      <pattern-formatter pattern="%d %-5.5p [%30.-30c] [%X{username}-%X
       {sessionId}] %m%n"/>
   </formatter>
</size-rotating-file-handler>

<logger
 category="com.consol.cmweb.client.webapp.timemeasure.log.Log4jOperati
 onLogger">
   <level name="DEBUG"/>
   <handlers>
      <handler name="OPERATION_TIMES"/>
   </handlers>
</logger>
```

Oracle Weblogic Server configuration:

```
<!-- Oracle WLS -->
<appender name="OPERATION_TIMES" class="
 org.apache.log4j.RollingFileAppender">
  <errorHandler class="
   org.apache.log4j.helpers.OnlyOnceErrorHandler"/>
  <param name="File" value=" @DOMAIN_HOME/cm-
   logs/operationtimes.log"/>
  <param name="Append" value="true"/> <param name="MaxFileSize"
   value="500MB"/>
  <param name="MaxBackupIndex" value="6"/>
  <layout class="org.apache.log4j.PatternLayout">
     <param name="ConversionPattern" value="%m\n"/>
  </layout>
</appender>

<logger
 name="com.consol.cmweb.client.webapp.timemeasure.log.Log4jOperationLo
 gger">
  <level value="DEBUG"/> <appender-ref ref="OPERATION_TIMES"/>
</logger>
```

- **operationtimes-db.log**
  Contains information about database access times of atomic operations in order to identify possible performance bottlenecks. This logging is configured using the ConSol CM system properties cmas-core-server, dao.log.threshold.milliseconds and cmas-core-server, dao.-log.username.

Config info: How to activate operationtimes-db.log after an update to 6.11.1.0
Insert the following line into the configuration file `cm6.xml` or `cm6-cmrf.xml` at the correct locations. If you are not familiar with the logging configuration, ask your CM consultant for help!

```
<subsystem xmlns="urn:jboss:domain:logging:1.5">
[...]
<size-rotating-file-handler name="DAO_OPERATIONS_LOG"
 autoflush="true">
  <level name="DEBUG"/>
  <formatter>
     <pattern-formatter pattern="%d %-5.5p [%30.-30c] [%X{context}-%X
      {sessionId}] %m%n"/>
  </formatter>
  <file relative-to="jboss.server.log.dir" path="operationtimes-
   db.log"/>
  <rotate-size value="300m"/>
  <max-backup-index value="6"/>
  <append value="true"/>
</size-rotating-file-handler>
[...]
<logger
 category="com.consol.cmas.core.server.service.aspect.DaoMetricsAspec
 t" use-parent-handlers="false">
<level name="TRACE"/>
  <handlers>
     <handler name="DAO_OPERATIONS_LOG"/>
  </handlers>
</logger>
```

- **server.log**
  The general log file that contains all messages, by default, with at least log level `INFO`. It is recommended to use the `DailyRollingFileAppender` in order to prevent the file system from filling up.
- **session.log**
  Contains messages about logins (session starts) and session timeouts of ***ConSol CM*** users.
- **sql.log**
  Contains log entries about SQL statements coming from hibernate if it is set to `DEBUG` level (by default it is set to `INFO`).
- **support_libs_errors.log**
  Contains errors which are thrown by support libs but are properly handled by the ConSol CM application (this method keeps the `server.log` clean).
- **timer-manager.log**
  Contains additional log messages written in log level `DEBUG` when workflow timers are activated or deactivated. Information about the escalation date is logged, too.

- **tx.log**
  Contains Spring Framework transactions related log messages.
- **unit-deletion.log**
  Contains information regarding the deletion of customers.
- **workflow.log**
  Information about activated/reinitialized/deactivated timers is logged with level `INFO` and all debug output related to the workflow engine is written to this dedicated file.

Admin Tool-Specific Log Files

Two log files report information which are specific for the Admin Tool. The CM logging configuration might have to be adapted during a system update from a CM version lower than 6.11 to 6.11.

- **audit.log**
  Available in ConSol CM versions 6.11 and up. In new 6.11 installations, the log file will be present automatically. If a system is updated from a CM version lower than 6.11, the respective configuration for the logger has to be entered into the configuration file (`cm6.xml` or `cm6-cmrf.xml`). Please see config info #1.
  - Reports login/logout operation and all operations which have been made using the Admin Tool:
    - configuration changes
    - creation of new objects
    - deletion of objects
    - reopening of tickets
  - Reports login/logout operation and operations which have been performed using the Process Designer, e.g.,
    - deployment of a workflow
- **transfer.log**
  Available in ConSol CM versions 6.11 and up. In new 6.11 installations, the log file will be present automatically. If a system is updated from a CM version lower than 6.11, the respective configuration for the logger has to be entered into the configuration file (`cm6.xml` or `cm6-cmrf.xml`). Please see config info #2.
  - Reports information about export / import of scenarios.

Config info #1: How to integrate audit.log into the CM configuration. Only required for updates from CM version older than 6.11 to 6.11.

Insert the following line into the configuration file `cm6.xml` or `cm6-cmrf.xml` at the correct locations. If you are not familiar with the logging configuration, ask your CM consultant for help!

```
<size-rotating-file-handler name="AUDIT" autoflush="true">
   <file relative-to="jboss.server.log.dir" path="audit.log"/>
   <append value="true"/>
   <rotate-size value="300m"/>
   <max-backup-index value="6"/>
   <formatter>
      <pattern-formatter pattern="%d %-5.5p [%30.-30c] [%X{username}-%X
       {context}-%X{sessionId}] %m%n"/>
   </formatter>


</size-rotating-file-handler>
<logger
 category="com.consol.cmas.core.server.history.method.MethodExecutionJourna
 lAspect" use-parent-handlers="false">
   <level name="TRACE"/>
   <handlers>
      <handler name="AUDIT"/>
   </handlers>
</logger>
```

Config info #2: How to integrate transfer.log into the CM configuration. Only required for updates from CM version older than 6.11 to 6.11.

Insert the following line into the configuration file `cm6.xml` or `cm6-cmrf.xml` at the correct locations. If you are not familiar with the logging configuration, ask your CM consultant for help!

```
<size-rotating-file-handler name="TRANSFER_FILE" autoflush="true">
   <file relative-to="jboss.server.log.dir" path="transfer.log"/>
   <append value="true"/>
   <rotate-size value="300m"/>
   <max-backup-index value="6"/>
   <formatter>
      <pattern-formatter pattern="%d %-5.5p [%X{username}-%X{context}-%X
       {sessionId}] %m%n"/>
   </formatter>
</size-rotating-file-handler>


<logger category="TRANSFER" use-parent-handlers="false">
   <level name="INFO"/>
   <handlers>
      <handler name="TRANSFER_FILE"/>
   </handlers>
</logger>
```

Log File Structure

In the default configuration, log file entries have the following syntax:

```
Date Timestamp Loglevel [Logger] Message
```

Example for a log file entry (successful start of ***ConSol CM*** in JBoss):

```
2017-10-11 13:52:44,526 INFO [reemarker.FreeMarkerConfigurer] [-]
 ClassTemplateLoader for Spring macros added to FreeMarker configuration
```

The components of the message:

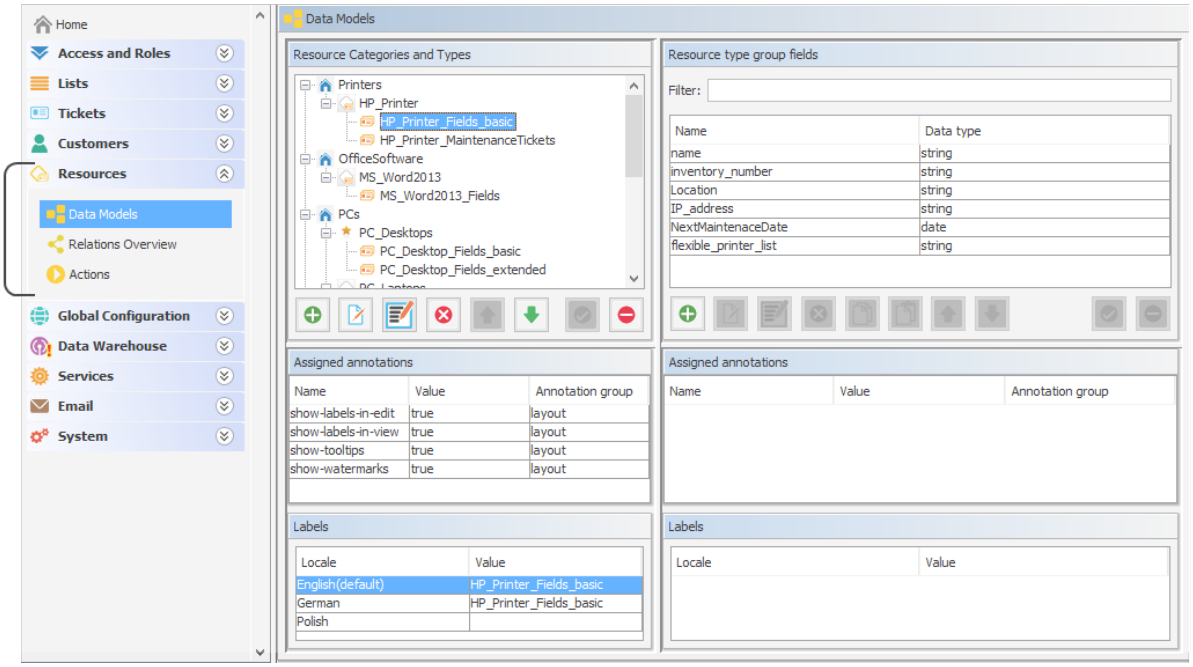
- **Date:**
  October 11th, 2017
- **Timestamp:**
  13:52:44
- **Loglevel:**
  INFO

- **Logger:**
  reemarker.FreeMarkerConfigurer
  Name of a Java class, not complete (only last 30 characters), the real name would be Free-marker.FreeMarkerConfigurer
- **Message:**
  ClassTemplateLoader for Spring macros added to FreeMarker configuration

Simple messages, and those reporting a successful operation, often have only one line.

When errors are reported (log level `ERROR`), you might find stack traces in the logs. Please contact one of our ***ConSol CM*** consultants or our ***ConSol CM*** support team for help.

G - Add-On Section



This section provides information about the optional ConSol CM add-ons:

- CM/Resource Pool
- CM/Doc
- CM/Track: The Customer Portal
- CM/Phone: CTI with ConSol CM

G.1 CM/Resource Pool

Starting with version 6.10, ConSol CM offers the add-on CM/Resource Pool. Using this module, you can extend the ConSol CM database to store data objects for any type of asset or object in your company. A service enterprise might use the Resource Pool to store and manage SLAs. For an IT Service Desk, the Resource Pool might be used to implement an IT asset database. A marketing department might store design and text templates. A reseller might reflect the product portfolio in the Resource Pool. Hence, implementing the Resource Pool for your company can be a core step in the ConSol CM customization process. To learn how to work with this module, read the following sections:

For a general introduction to the CM/Resource Pool, read the section Introduction to CM/Resource Pool.

All elements in the Admin Tool which you will work with to configure the Resource Pool are explained in section CM/Resource Pool - Admin Tool Elements.

To get an impression of how the Resource Pool objects are managed by engineers working with the Web Client, see the section A Short Introduction to CM/Resource Pool Functionality in the Web Client.

One of the first steps when you start working with the Resource Pool is the set-up of the resource model. This is covered in section CM/Resource Pool - Setting Up the Basic Resource Model.

Similar to customer templates, you can define templates which control the appearance of resource data in the Web Client. These templates are explained in section CM/Resource Pool - Templates for Resource Data.

Resources can be related. Read section CM/Resource Pool - Resource Relations to learn how to define and to work with resource relations.

For every resource type, activities can be performed. They are based on resource actions which are explained in section CM/Resource Pool - Resource Actions. Some more specific information about how to implement scripts for the Action Framework are also provided in section Scripts for the Action Framework.

As per general ConSol CM standards, engineers can have certain access permissions to resources. Those permissions are based on roles. Please read section CM/Resource Pool - Assigning Permissions for Resources for details.

The Resource Pool Dashboard provides a very convenient entry point to all Resource Pool objects. The Dashboard configuration is explained in CM/Resource Pool - The Resource Pool Dashboard.

G.1.1 Introduction to CM/Resource Pool

G.1.1.1 Introduction to CM/Resource Pool

Starting with ConSol CM version 6.10, the add-on ***CM/Resource Pool*** is available. The *Resource Pool* provides a database extension which enables ConSol CM to create and store objects other than tickets or customers. For example, you could represent your IT landscape by modeling each asset (like PCs, laptops, monitors, printers) as a resource in the Resource Pool. Similar resources (e.g., HP printers) are managed in one *resource type*, and similar resource types form a *resource category* (e.g., printers). Every specific object (e.g., the printer #4711) is an instance of a resource type (e.g., resource type *HP_Printer*). Other examples for the use of CM/Resource Pool are the use for products, facilities, machines, rooms, vehicles, contracts (e.g., SLAs), or scientific samples.

CM/Resource Pool extends ConSol CM's basic objects:

1. A ticket is an instance of a process execution.
2. A real-world customer (e.g., a person who calls the service desk to open a ticket) is an instance of a customer (unit) definition.
3. **A real-world resource (e.g., a printer or an SLA) is an instance of a resource definition.**

G.1.1.2 CM/Resource Pool at a Glance

CM/Resource Pool Structure

The Resource Pool is structured based on a two-level hierarchy and comprises any number of resource categories. Each resource category represents a subtree within the Resource Pool. You can manage any number of resource categories, i.e., any number of subtrees, within the Resource Pool. In this way, you could, for example, manage the IT landscape, as well as your product portfolio, as resources, each in one or more specific resource categories/subtrees.

Relations between resources and other ConSol CM objects can mirror various constellations, e.g., a resource of resource category *printer* and of resource type *HP printer* can have a relation to a ticket which was opened for a specific printer. Or, e.g., the printer is related to all contacts which use this printer. Relations between resources are also possible. For a detailed explanation of working with resource relations, see section CM/Resource Pool - Resource Relations.

Data fields are defined for each resource type in a way similar to the field definition for ticket data (ticket fields) and customer data (customer fields). Data fields are called *resource fields* and are always managed in groups which are called *resource field groups*. The set-up of the data model for the Resource Pool, i.e., the *resource model*, is explained in detail in section CM/Resource Pool - Setting Up the Basic Resource Model.

For each resource type, resource actions can be defined, very similar to customer actions for customer objects (companies and contacts). The resource actions will trigger scripts which perform resource-specific activities, like updating the data of a resource (e.g., providing a list with all companies which are related to the printer #4711 and create a maintenance ticket for a firmware update). Resource actions are a component of the Action Framework and are explained in detail in section CM/Resource Pool - Resource Actions.

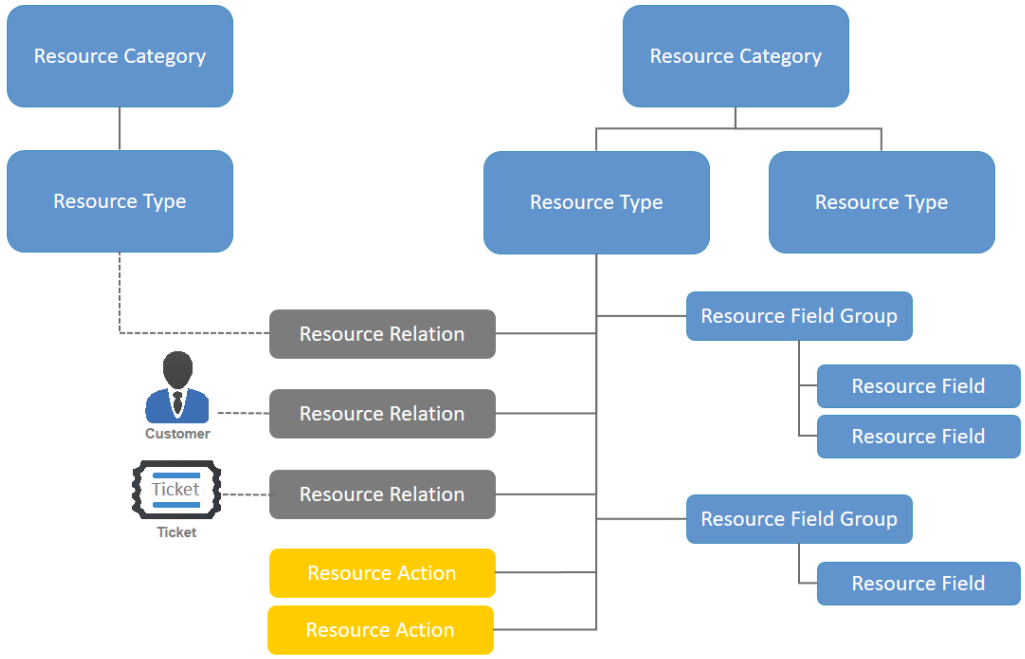


Figure 594: *CM/Resource Pool - Objects in a resource model*

Important Terms

- A **resource category**
  represents the highest hierarchy level in the Resource Pool, it can be seen as the root node of a Resource Pool subtree. For example, the Resource Pool of a company could have a resource category *Printers*. A resource category comprises one or more resource types, e.g., HP_Printers, Kyocera_Printers, Lexmark_Printers.
- A **resource type**
  represents similar resources which share the same data fields, but with differing values. For example, the resource type *HP_Printers* has the data fields (resource fields) *Model*, *IP Address*, *Inventory Number*, and *Location*. Each real-world HP printer is represented by an object of this Resource type. Lexmark printers might have the data fields (resource fields) *Model*, *IP Address*, *Location* and *SLA type*. Each real-world Lexmark printer is represented by an object of this resource type.

  > Within the Resource Pool hierarchy, there is **no inheritance**! Data fields (resource fields) are always defined as resource field groups for each specific resource type!

- A **resource field group**
  groups one or more resource fields within one resource type. On the resource page, a resource field group can be displayed as tab, just like the ticket field group for ticket data on the ticket page. Resource field groups can be annotated.

- A **resource field**
  is a data field for data of a resource type, e.g., *Model*, *IP address*, or *Location*. A resource field always is of one specific data type. Resource fields contain resource data as ticket fields contain ticket data. Resource fields can be annotated to modify their logical behavior or graphical appearance.
- The **resource model**
  is the complete data model which defines all objects within the Resource Pool. It is built using the Admin Tool, navigation group *Resources*, navigation item *Data Models*.
- A **resource relation**
  is a relation of a resource to another ConSol CM object. This can be a ticket, a customer or another resource.
- A **resource action**
  is an action defined for a resource type and available for each real-world resource object of this type. The action can be triggered automatically or manually. Manual resource actions can be selected in the Web Client on the resource page like workflow activities for tickets.
- The **resource page**
  is the page in the Web Client where all data for one real-world resource is displayed, e.g., the resource page for the HP printer #4711.
- The **resource type page**
  is the page in the Web Client which provides information about a resource type and an overview (list) of all resources of this type.
- The **Resource Pool Dashboard**
  is the overview page in the Web Client where all resource categories and resource types are displayed to which the current engineer has access.

G.1.1.3 Defining Resource Models Using the Admin Tool

To be able to create real-world resources (e.g., HP printer #4711), a resource model has to be defined, i.e., a definition of all data fields which are required for all resources of a type, e.g., of type *HP_printer*. All resource data models have to be defined using the Admin Tool. This applies to:

- Resource categories (e.g., resource category *Printers*)
- Resource types (e.g., resource type *HP Printer*)
- Resource relations (e.g., possible relation between resource type *HP Printer* and *Ticket*)
- Resource actions (e.g., resource action *Retrieve Maintenance Contract Data from ERP System*)

The following figure shows an example of a simple Resource Pool configuration with four resource categories. All steps you have to perform to configure this constellation will be explained in detail in the following sections. The Admin Tool sections involved in Resource Pool configuration are described in section CM/Resource Pool - Admin Tool Elements.

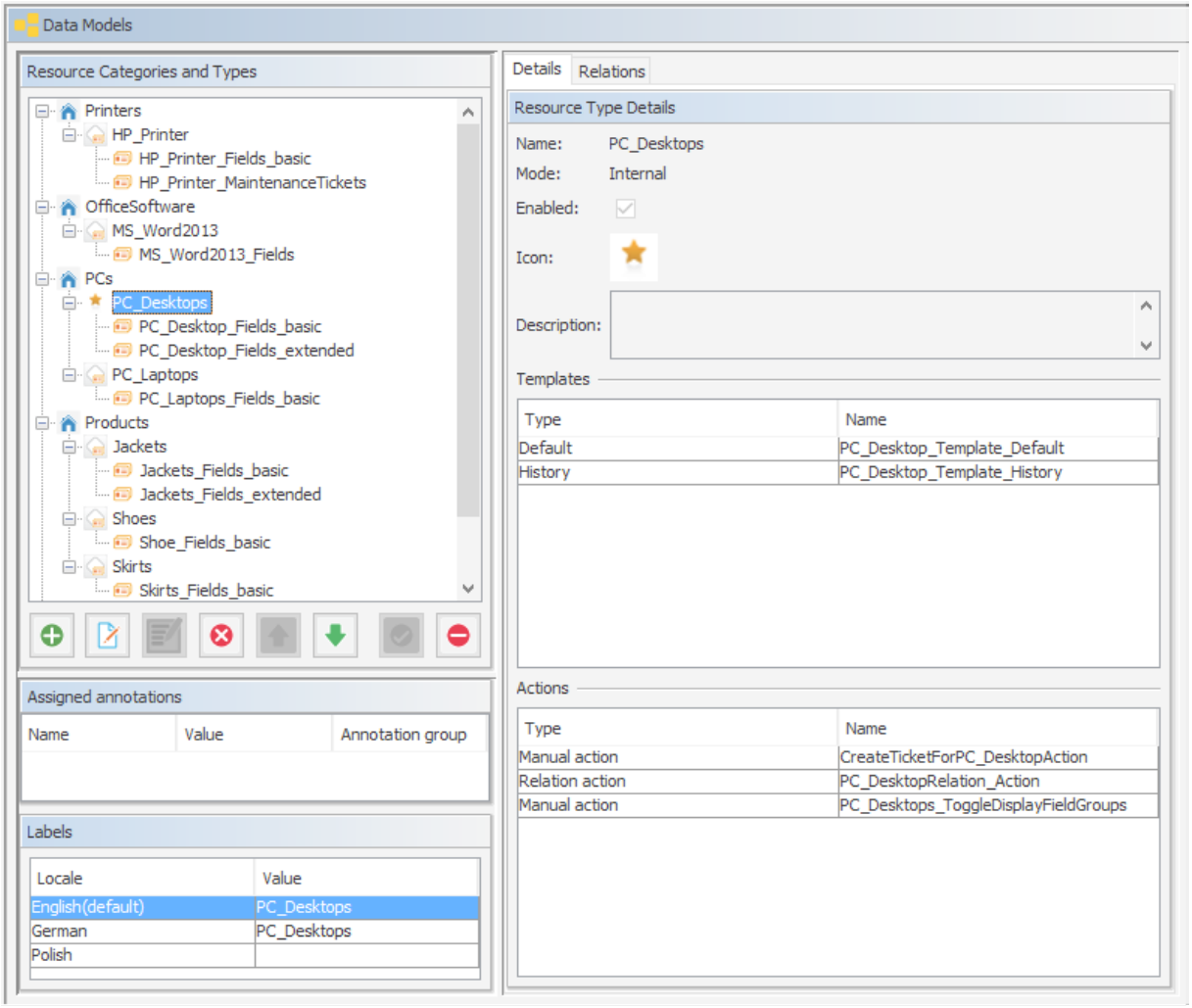


Figure 595: *ConSol CM Admin Tool - Resources, Data Models: Simple Resource Pool configuration*

Once you have defined the resources (i.e., the data model for the real resources) using the Admin Tool, the engineers can start working with real resource objects using the Web Client. This is explained briefly in section A Short Introduction to CM/Resource Pool Functionality in the Web Client. A detailed explanation is provided in the *ConSol CM User Manual*.

G.1.1.4 Licensing

CM/Resource Pool has to be licensed separately. Please ask your ConSol CM sales representative for more information.

G.1.1.5 Programming with Resource Pool Objects

The ConSol CM Java API has been extended considerably to implement the Resource Pool. The explanation of important Groovy classes and various programming examples are provided in the *ConSol CM Process Designer Manual*, CM version 6.10.

G.1.2 CM/Resource Pool - Admin Tool Elements

You will have to work with several sections of the Admin Tool to configure CM/Resource Pool:

- **Navigation group *Access and Roles***
  To assign Resource Pool permissions to roles
- **Navigation group *Lists***
  If you use MLAs and/or enums (sorted lists) in resource fields
- **Navigation group *Resources*** (see following figure)
  - **Data Models**
    Definition of the resource model, including resource relations
  - **Relations Overview**
    Contains a list of all resource relations, read-only mode, no definitions/configurations here
  - **Actions**
    Definition of resource actions
- **Navigation group *Global Configuration***
  - **Labels**
    In case you would like to modify labels of Resource Pool elements in the Web Client or define your own labels used in scripts
- **Navigation group *System***
  - **Scripts and Templates**
    - Scripts for resource actions
    - Templates for the display of resource data in the Web Client
  - **License**
    To activate a new license for CM version 6.10 with CM/Resource Pool

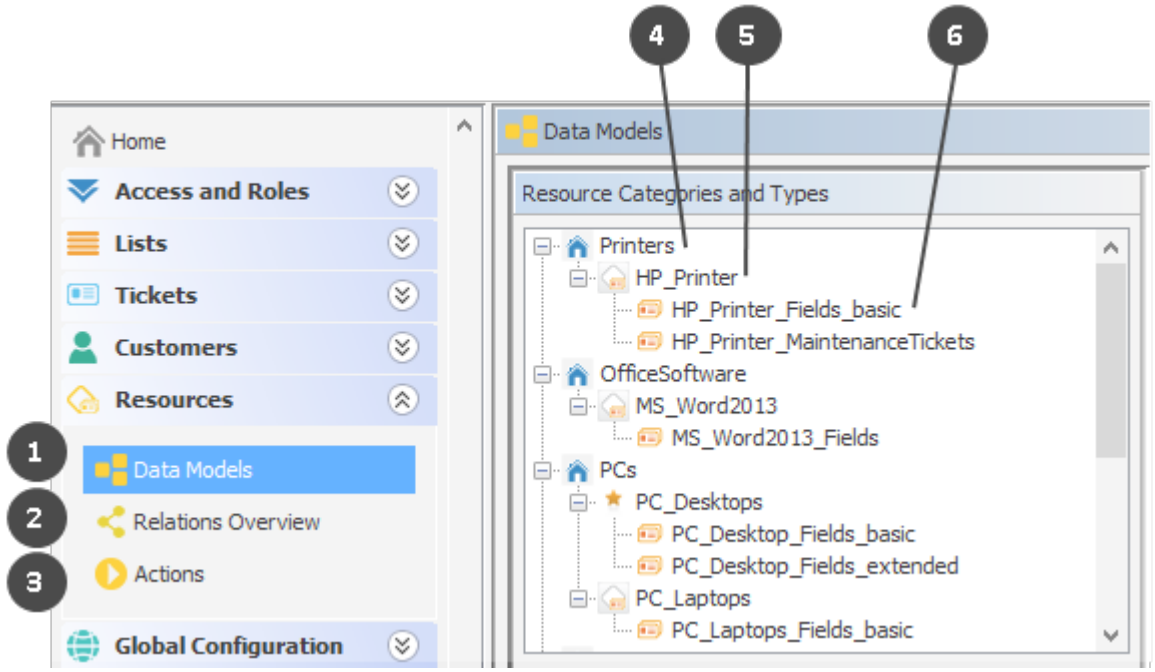


Figure 596: *ConSol CM Admin Tool - Resources, Data Models: Main elements for the configuration of CM/Resource Pool*

The main elements of CM/Resource Pool are:

- Navigation item *Data Models* to define resources and resource relations (1)
- Navigation item *Relations Overview* to display a relations list (2)
- Navigation item *Actions* to manage resource actions (3)
- Resource category (4)
- Resource type (5)
- Resource field group (6)

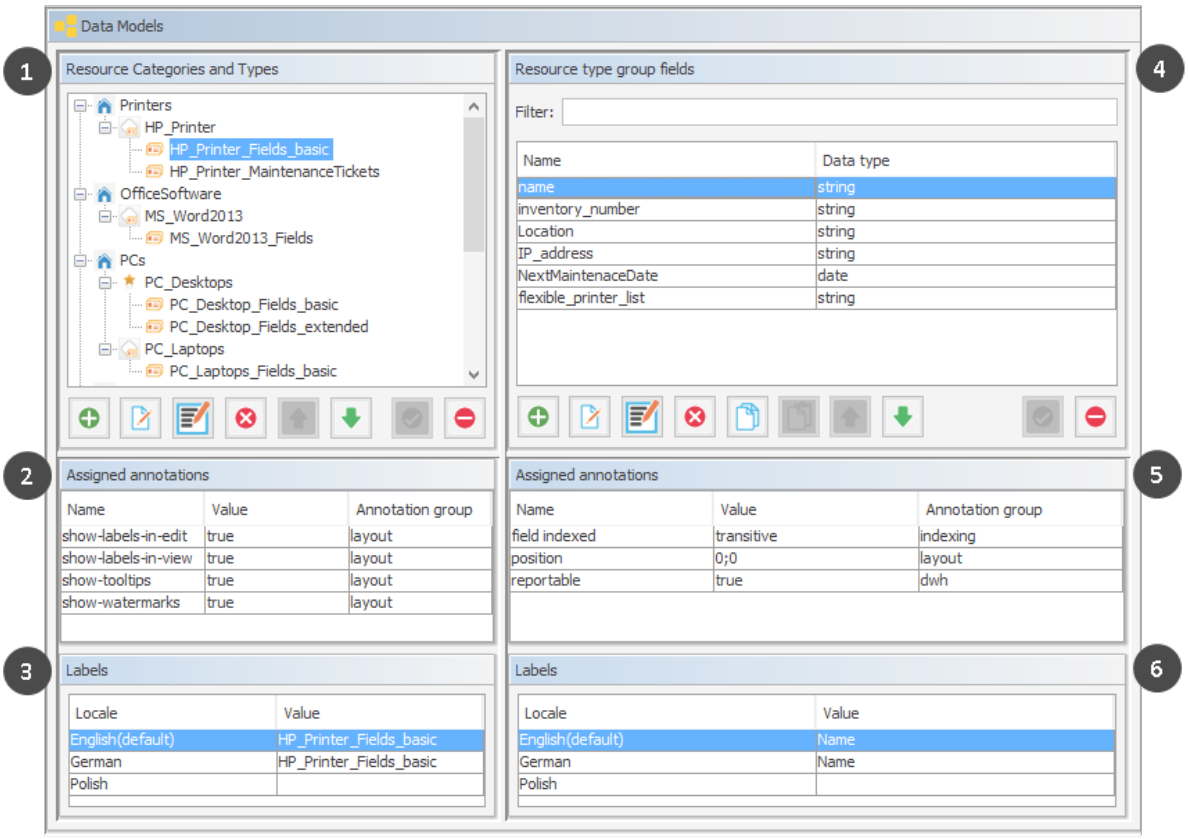


Figure 597: *ConSol CM Admin Tool - Resources, Data Models: CM/Resource Pool configuration*

The *Data Models* screen consists of six sections:

- Resource data models (1)
- Annotations for the selected resource field group (2)
- Labels for the selected resource field group (3), i.e. the localized values of the name of the selected resource field group
- Resource fields (4) of the selected resource field group
- Annotations for the selected resource field (5)
- Labels of the selected for resource field (6), i.e. the localized values of the name of the selected resource field

G.1.3 A Short Introduction to CM/Resource Pool Functionality in the Web Client

G.1.3.1 Overview of Resource Pool Operations in the Web Client

When you, as an administrator, have defined the resource data model (resource categories and the required resource types with their resource field groups and resource fields) using the Admin Tool, engineers can start working with real-world Resource Pool objects, provided they have the required access permissions. Permissions are discussed in section CM/Resource Pool - Assigning Permissions for Resources. In this section, you will get an overview of the following operations:

- Creating Resources Using the Web Client
- Working with the Resource Type Page
- Setting a Resource as Favorite
- Defining and Using Resource Relations
- Defining and Using Resource Actions
- Using the Quick Search to Find Resources
- Using the Detailed Search to Find Resources

G.1.3.2 Creating Resources Using the Web Client

In the Admin Tool, you only define the data models for the objects (developers: this is comparable to defining classes in object-oriented programming). The real objects (the instances of the defined data models) have to be defined using the Web Client where the Resource Pool Dashboard provides the required graphical user interface. The permissions to resources are managed based on resource types, using roles, according to the ConSol CM standard. An engineer only sees the resource types in the Resource Pool Dashboard if he has the required permissions. The role management for resources is explained in detail in section CM/Resource Pool - Assigning Permissions for Resources.

The following figure shows the Resource Pool Dashboard with the resource models which have been defined in the Admin Tool (see section CM/Resource Pool - Admin Tool Elements). The logged-in engineer has access permissions for all resource types.

The label for the Resource Pool in the main menu (as shown in the following figure) can be modified according to the requirements in the specific system. Examples for alternative wording could be *Asset management*, *Products*, *Machines*, or *Inventory*. Please refer to the Labels section for an explanation of how to change the label.

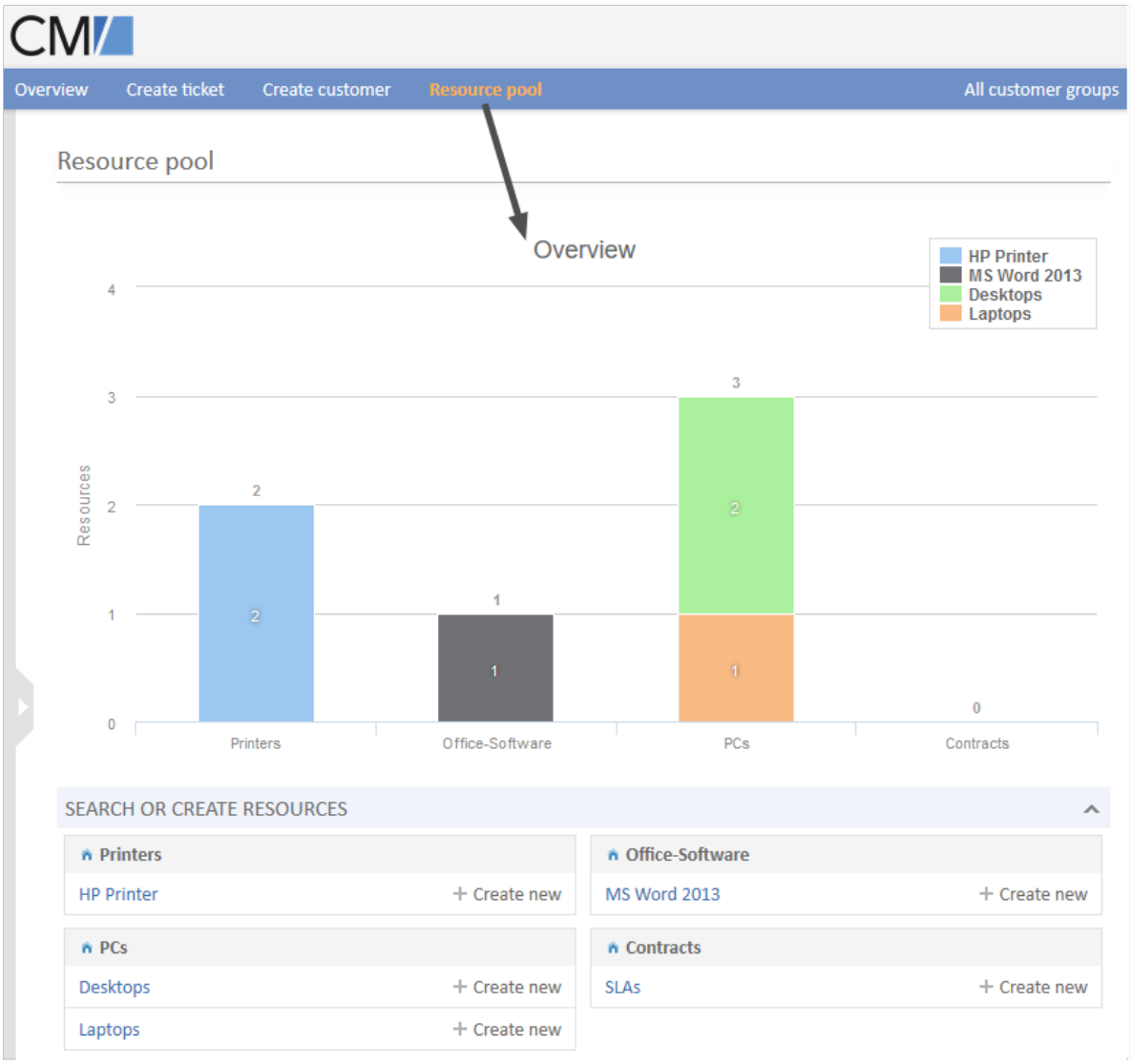


Figure 598: *ConSol CM Web Client - Resource Pool Dashboard*

Once the data models have been defined (in the Admin Tool) and you, as an engineer, have the required access permissions, you can create a real object, e.g., you can create an object for the printer #4711 which is located on the top floor.

The following figures demonstrate the required steps.

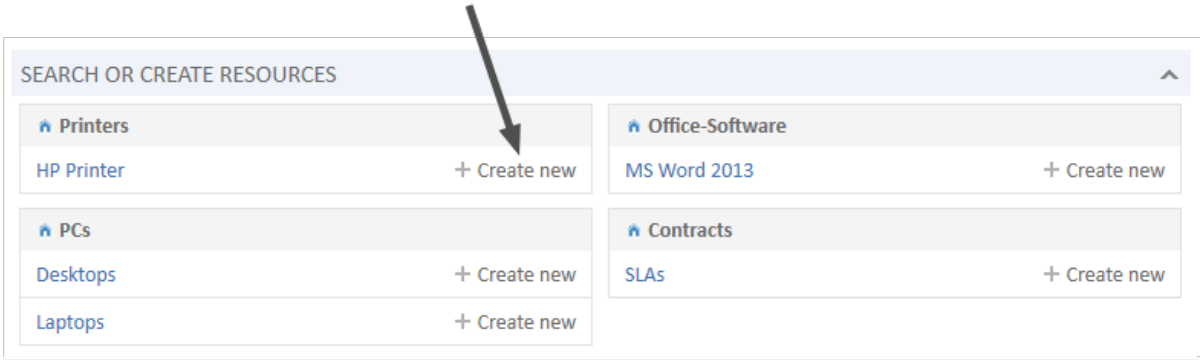


Figure 599: *ConSol CM Web Client - Resource Pool Dashboard: Create a new resource of type HP Printer, step 1*

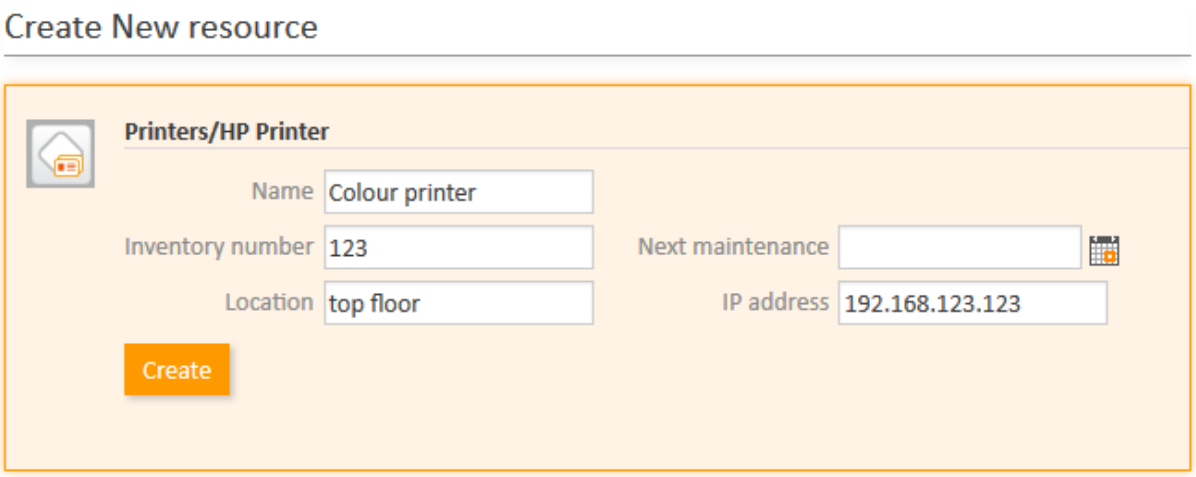


Figure 600: *ConSol CM Web Client - Create a new resource of type HP Printer, step 2*

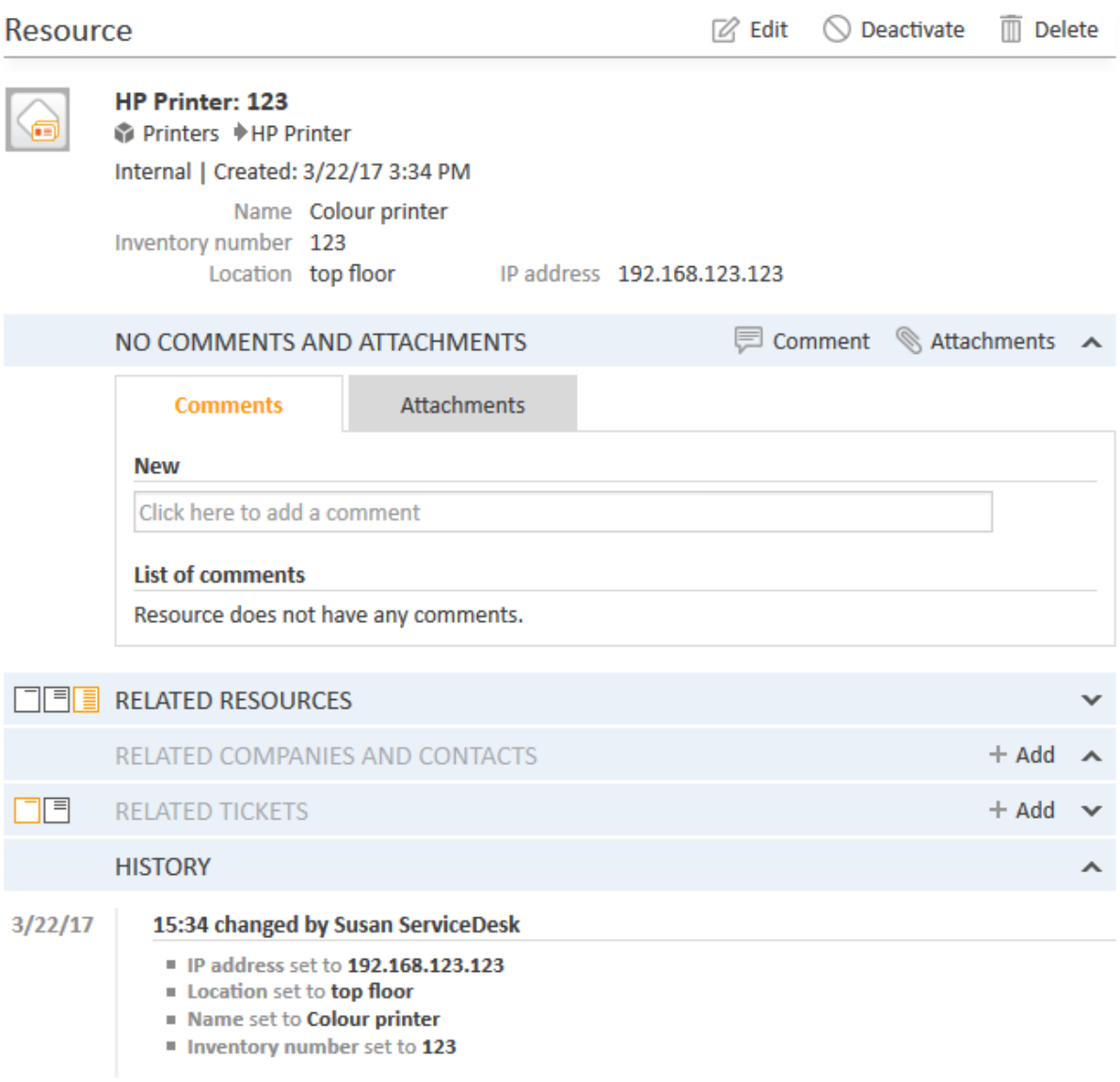


Figure 601: *ConSol CM Web Client - Resource page: Create a new resource of type HP Printer, step 3*

Now you have created a new *HP Printer* object, i.e., you have registered the printer #123 (located on the top floor). The resource page of this specific printer is displayed (see figure above).

For a detailed introduction to the work with CM/Resource Pool using the Web Client, please refer to the *ConSol CM User Manual*. Here we will only provide short explanations to show you, as an administrator, what happens when you work with the Admin Tool configuration for the ConSol CM resources.

How You, as an Administrator, Can Manipulate the Layout of the Resource Pool Dashboard Section *Search or create Resources*

- The roles of an engineer determine whether he has access permissions to a resource type. Hence, for the engineer currently logged in, only the resource types are displayed to which he

has access permissions.

- The name of a resource type is displayed as a hyperlink which leads to the resource type page. This hyperlink is only active if the engineer has the access privilege READ for the respective resource type.
- The Link *Create new* for a resource type is only displayed if the engineer has the access permission CREATE for this resource type.

How You, as an Administrator, Can Manipulate the Layout of the Resource Page

- In the top section of the resource page, the localized values of the resource fields are displayed (analog to ticket data of ticket fields in a ticket). Data of resource fields may also be displayed in tabs which represent resource field groups (also analog to tabs on a ticket page which represent ticket field groups). The resource fields are displayed according to the layout you have defined. See section CM/Resource Pool - Setting Up the Basic Resource Model for details.
- The icon for each resource of the resource type can be defined in the Admin Tool.
- The template for the display of the resource name is defined for the resource type. Here on the resource page, the *Default* template is used, see section CM/Resource Pool - Templates for Resource Data for details.
- The menu entries (links) in the top right corner of the page are only visible if the engineer has the required access permissions:
  - *Edit* link: WRITE permission
  - *Deactivate* link: DEACTIVATE / ACTIVATE permission
  - *Delete* link: DELETE permission
- If resource actions are defined for resources of this type (not shown in the figure above), they will be available on the resource page as well, just like workflow activities for tickets. See section CM/Resource Pool - Resource Actions for details. However, an engineer will only see the resource actions if he has the permission ACT for the respective resource type.
- For an engineer, the section *Comments and Attachments* might not be displayed if his roles do not have the permission *Details read*. See section CM/Resource Pool - Assigning Permissions for Resources.
- If relations to other resources, contacts or companies, or to tickets have been defined in the Admin Tool, relations might be displayed in the respective sections of the resource page if real relations have been created for the resource by an engineer. See section CM/Resource Pool - Resource Relations for details.

G.1.3.3 Working with the Resource Type Page

The resource type page provides an overview of all resources of one specific resource type. You can open the resource type page by opening the Resource Pool Dashboard and clicking on the name of the resource type.

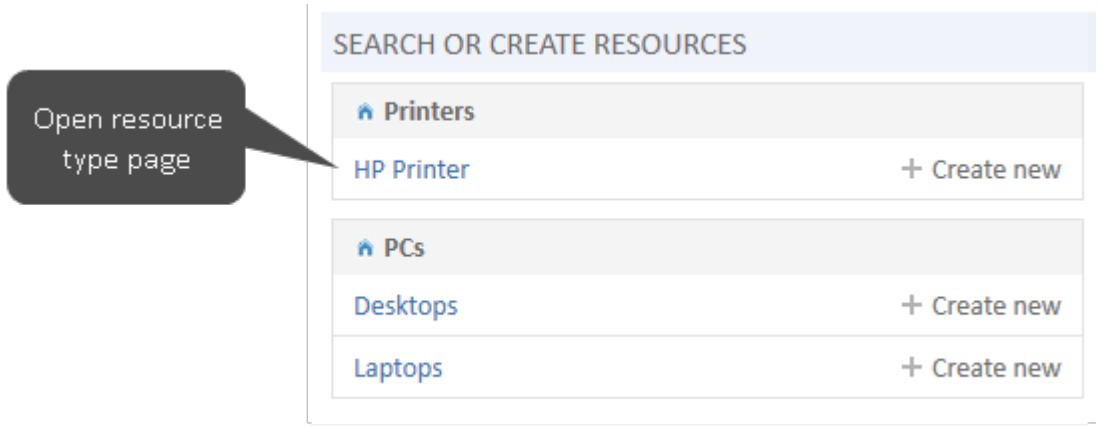


Figure 602: *ConSol CM Web Client - Opening a resource type page*

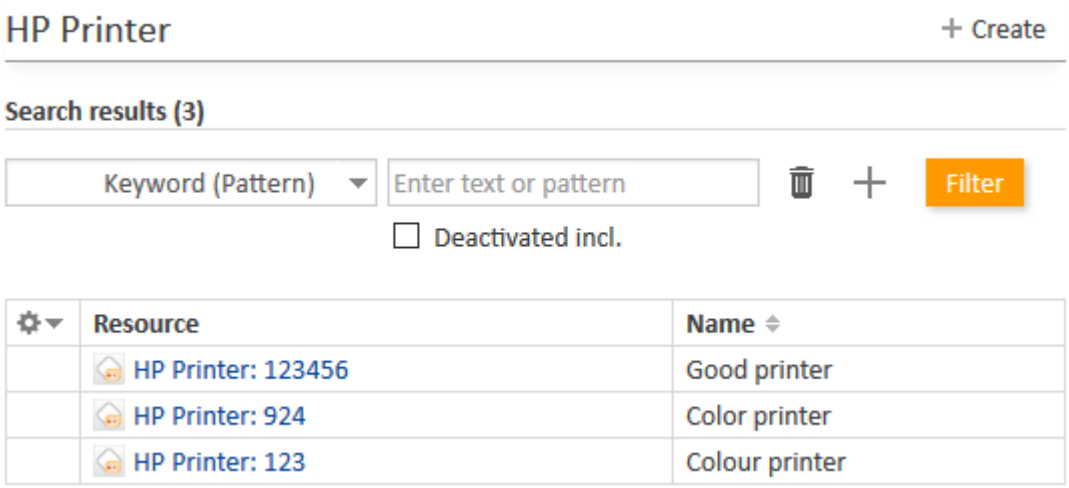


Figure 603: *ConSol CM Web Client - Resource type page*

On the resource type page, all resources of this type are listed. They can be filtered by the values of the resource fields using the search in the top section of the page.

How You, as an Administrator, Can Manipulate the Layout of the Resource Type Page

- In the drop-down menu for the search, only fields which have been indexed are available for selection. See section Search Configuration for details.
- The column headers represent the localized values of the names of the resource fields which an admin has defined. See section CM/Resource Pool - Setting Up the Basic Resource Model for details.

G.1.3.4 Setting a Resource as Favorite

Like tickets, searches and customers, resources can also be placed in the Favorites section using drag-and-drop, e.g., on the resource page.

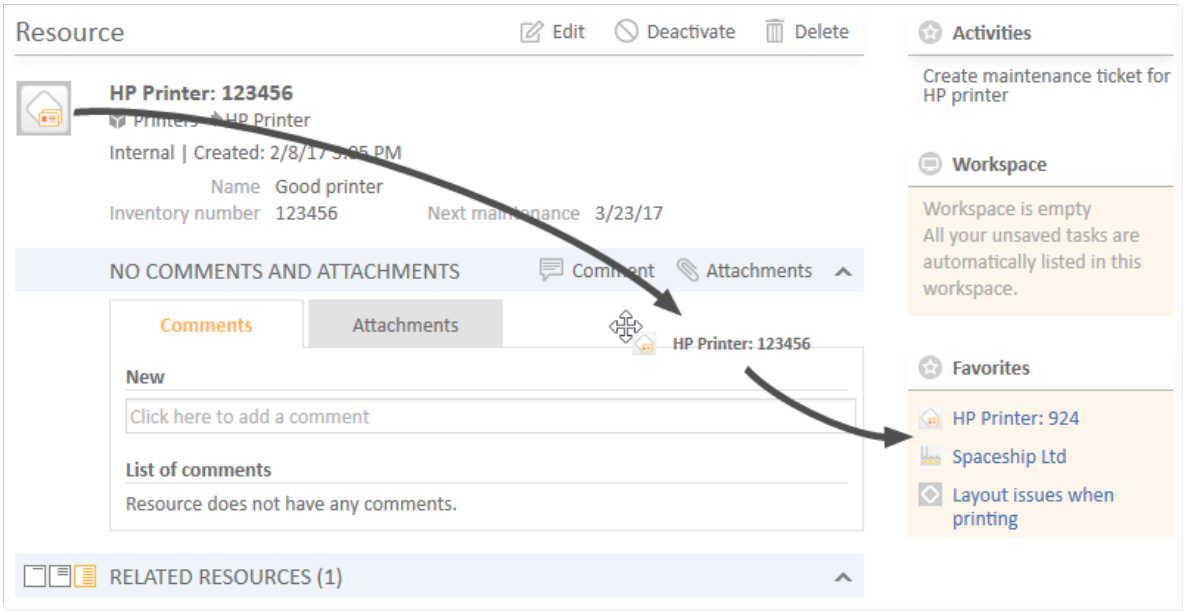


Figure 604: *ConSol CM Web Client - Resource page: Using drag-and-drop to put a resource into the Favorites section*

How You, as an Administrator, Can Manipulate the Display Format of the Resource Concerning Drag-and-Drop to Favorites

- The format for the display of the resource in the Favorites section is based on the *Default* template, as described in section CM/Resource Pool - Templates for Resource Data.
- The format for the display of the resource in the drag-and-drop box is based on the *Default* template, as described in section CM/Resource Pool - Templates for Resource Data.

G.1.3.5 Defining and Using Resource Relations

The same principle as for resource categories and resource types also applies to resource relations:

- First, the resource relation has to be defined in the data model using the Admin Tool
- Then, the real relations between resources (objects of a certain resource type) and other objects (tickets, customers or other resources) have to be created using the Web Client.

This is explained in detail in section CM/Resource Pool - Resource Relations. A short example:

To link a resource (e.g., a PC desktop) to tickets in the queue *ServiceDesk*, first, define this relation type (i.e., resource-ticket) in the Admin Tool.

Then an engineer can create a relation for a PC to a ticket, e.g., when an incident ticket involving this machine is created.

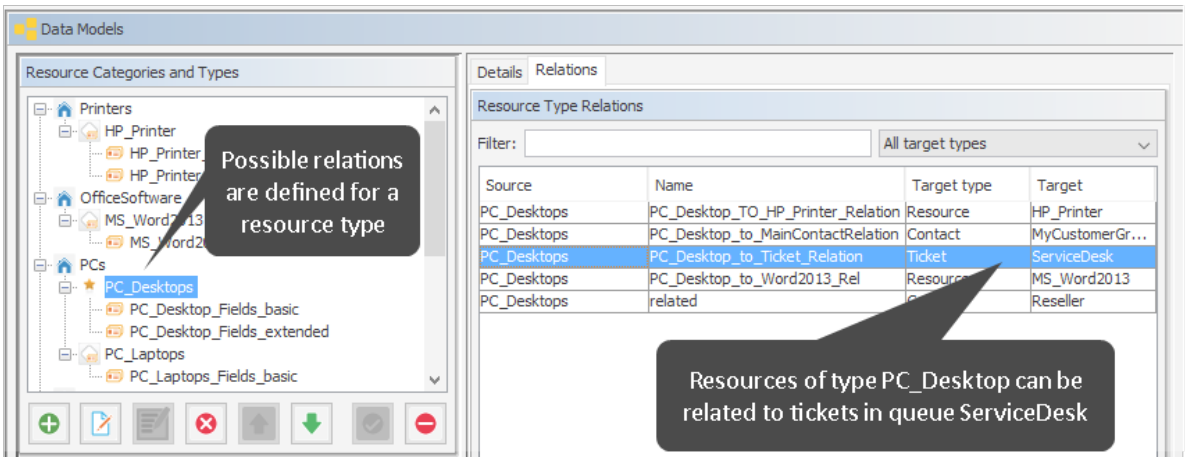


Figure 605: *ConSol CM Admin Tool - Resources, Data Models: Defining a new resource relation*

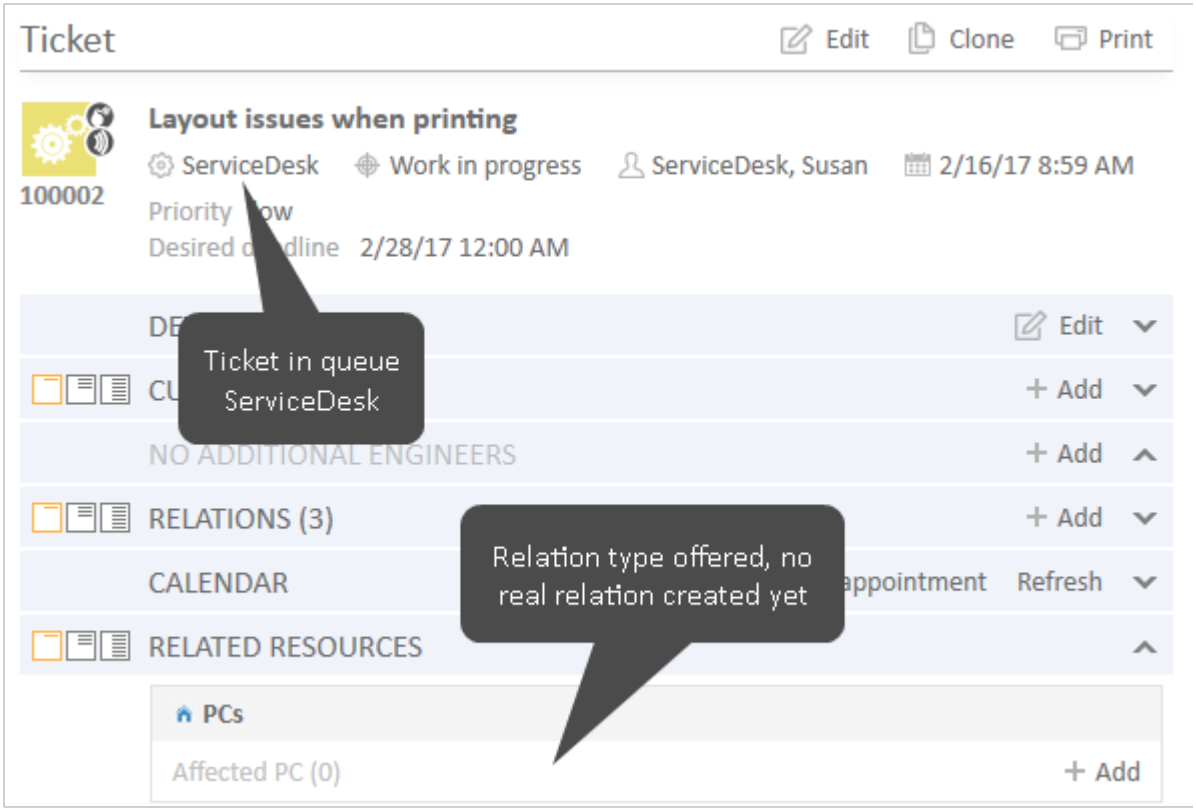


Figure 606: *ConSol CM Web Client - Ticket where a relation to a PC can be created*

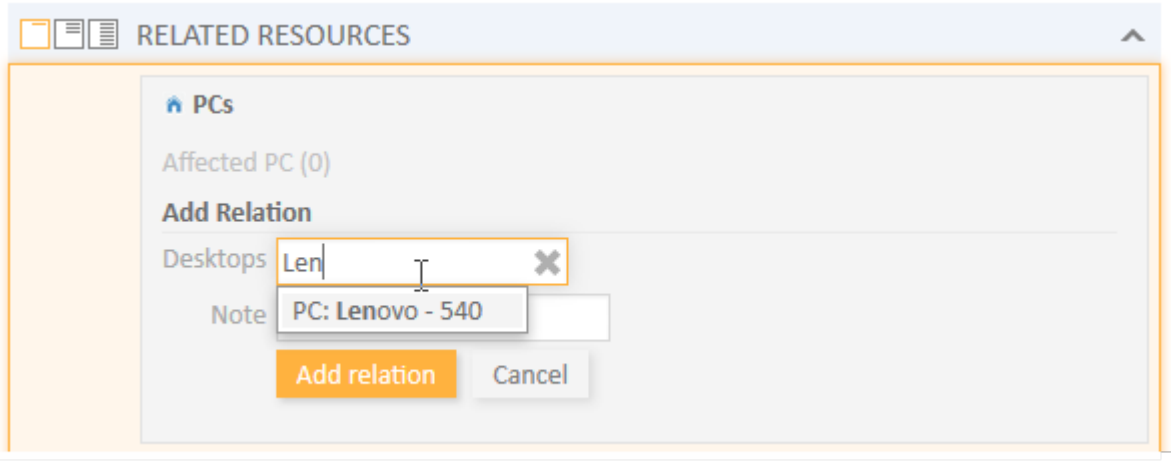


Figure 607: *ConSol CM Web Client - Creating a relation between a ticket and a PC*

Figure 608: *ConSol CM Web Client - Ticket with relation to a PC*

When the relation has been created, the counter of the relation is set to 1. The engineer can then display details of the relation by clicking on the relation name.

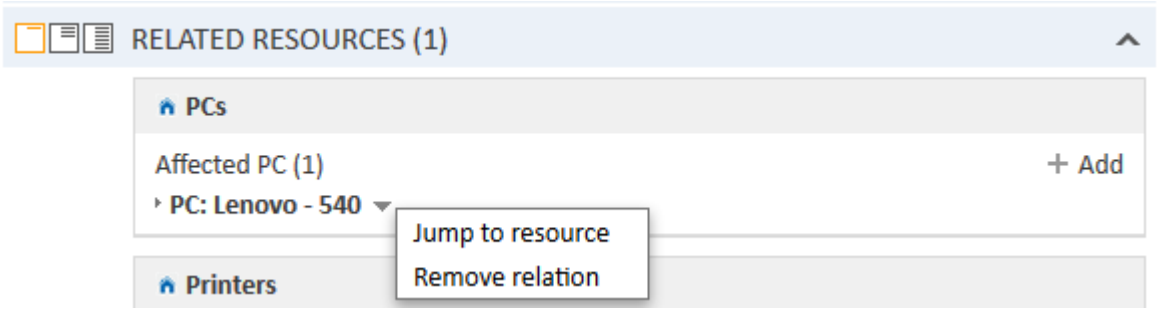


Figure 609: *ConSol CM Web Client - Details of a related resource and context menu*

The context menu of the related resource offers two options:

- **Jump to resource**
  Opens the resource page of the related resource.
- **Remove relation**
  Removes the relation of the ticket with the resource.

How You, as an Administrator, Can Manipulate the Display of Resource Relations

- The name and description of the relation which is displayed is the localized value of the relation name and description which you defined using the Admin Tool. Please note that there is one description for the source side and one for the target side of a relation.
- A note for a resource relation can only be added in the Web Client if the check box *Note field is available* has been set for the resource relation definition in the Admin Tool.
- When one resource of a type has been added to an object (e.g., to a ticket), the *Add* option (e.g., to add another resource of the type PC to the ticket) will only be available when the cardinality of the relation is set to *many-to-one, one-to-many* or *many-to-many*.

G.1.3.6 Defining and Using Resource Actions

Resource actions also have to be defined using the Admin Tool. Each resource action is based on an Admin Tool script.

Resource actions are explained in detail in section CM/Resource Pool - Resource Actions. Here, only a short example is provided.

A resource action is always defined on the navigation item *Actions* in the navigation group *Resources* and the assigned to one or more resource type(s). For example, an action is defined which should offer the possibility to create a new ticket in the queue ServiceDesk, directly from the resource page. Thus, when a customer calls to complain about a problem with the PC, the Service Desk agent can start by opening the PC's detail page for information about this asset. If necessary, the agent can directly create a new Service Desk ticket for the customer.

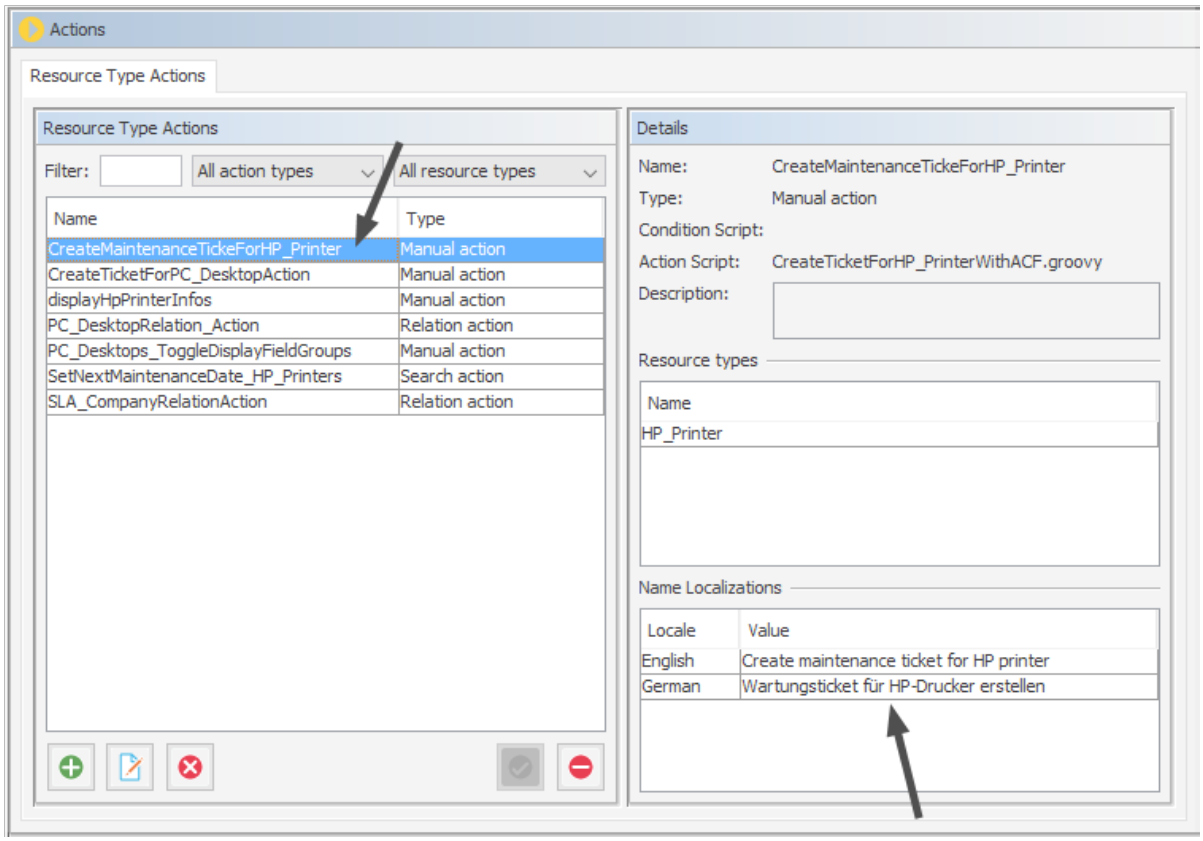


Figure 610: *ConSol CM Admin Tool - Resources, Actions: Definition of a resource action*

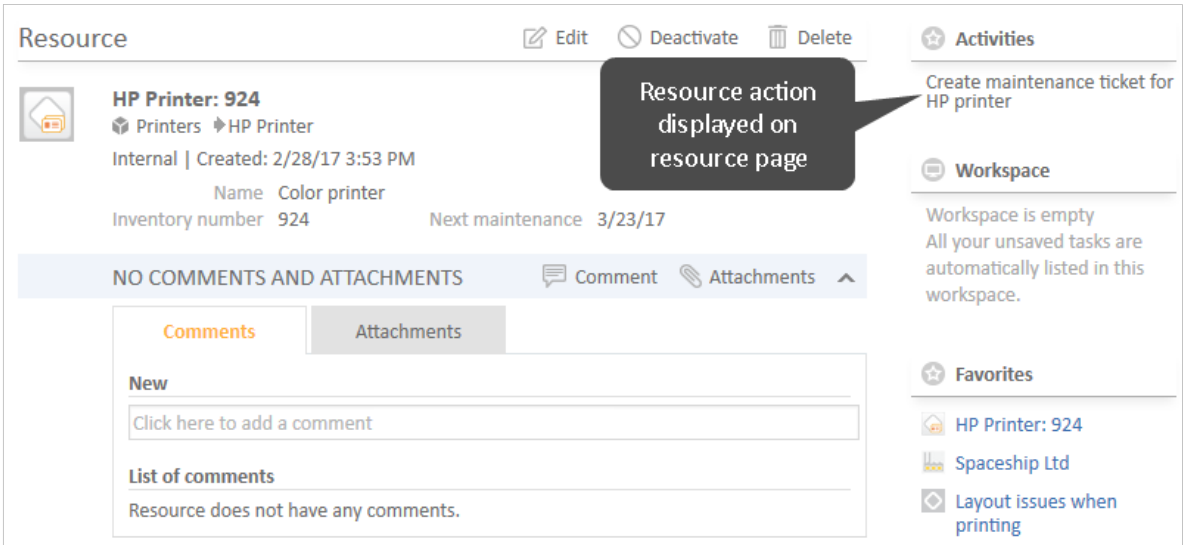


Figure 611: *ConSol CM Web Client - Resource activity (based on resource action)*

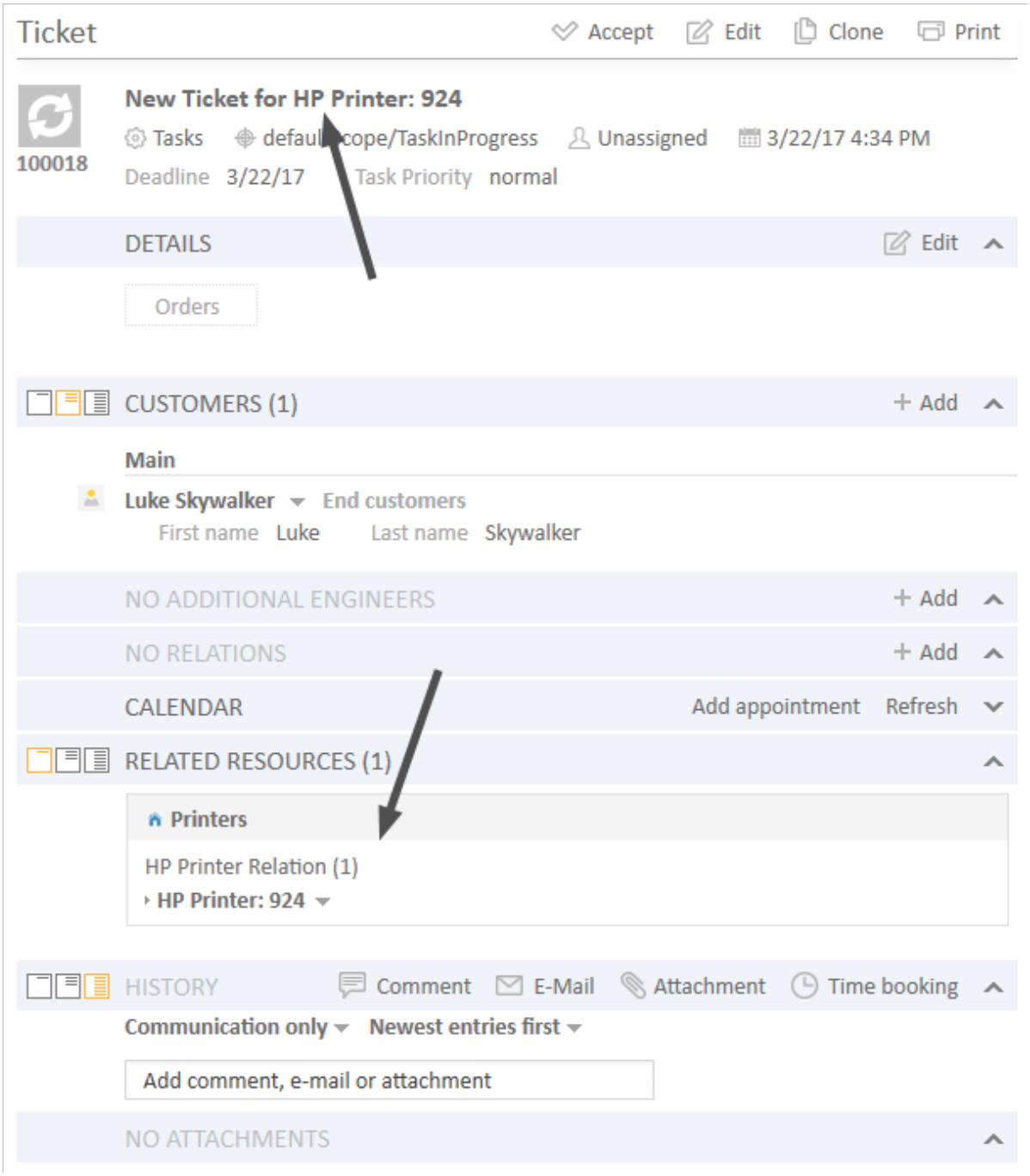


Figure 612: *ConSol CM Web Client - Task ticket created by a resource action*

How You, as an Administrator, Can Manipulate the Display of Resource Actions

- The name of the resource activity which is displayed in the Web Client is the localized name of the resource action which an administrator has defined in the *resource actions* panel. For details see section CM/Resource Pool - Resource Actions.

- The localized value of the description of the resource action will be displayed as mouse-over text for the resource activity in the Web Client.

G.1.3.7 Using the Quick Search to Find Resources

In the result list of the Quick Search, resources are displayed in a distinct section:

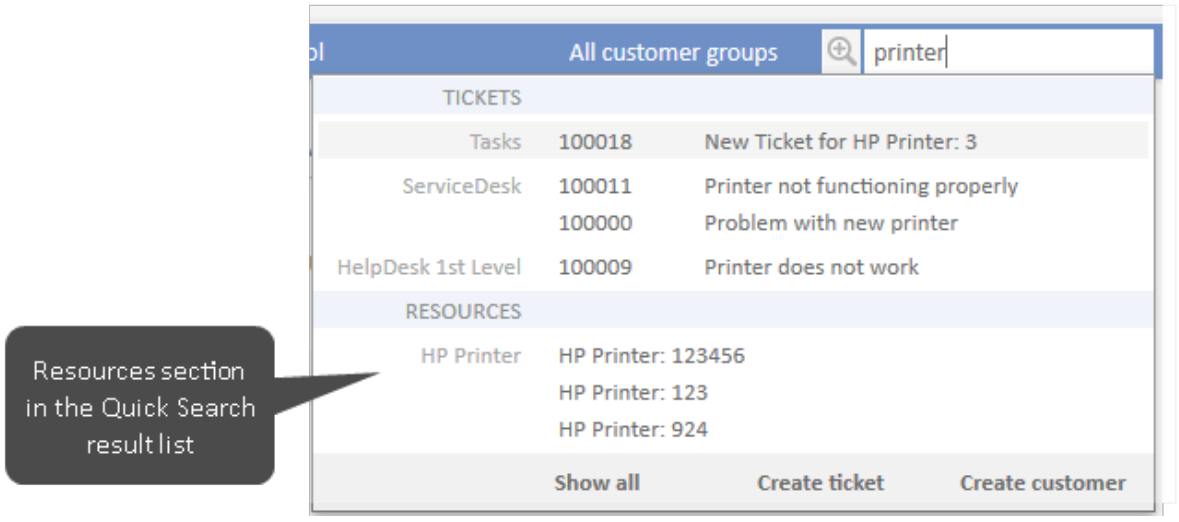


Figure 613: *ConSol CM Web Client - Result list of Quick Search with distinct resource section*

How You, as an Administrator, Can Manipulate the Display of Resources in the Quick Search

- The format for the display of the resource in the Quick Search result list section is based on the *Quick Search* template. For details see section CM/Resource Pool - Templates for Resource Data.
- The value of the system property cmweb-server-adapter, globalSearchResultSizeLimit defines the maximum number of hits displayed in the result list.

G.1.3.8 Using the Detailed Search to Find Resources

In the Detailed Search, an engineer can search for all resources of a certain type, provided that the respective resource field is indexed (the annotation `field indexed` has been set, usually to the value “transitive”).

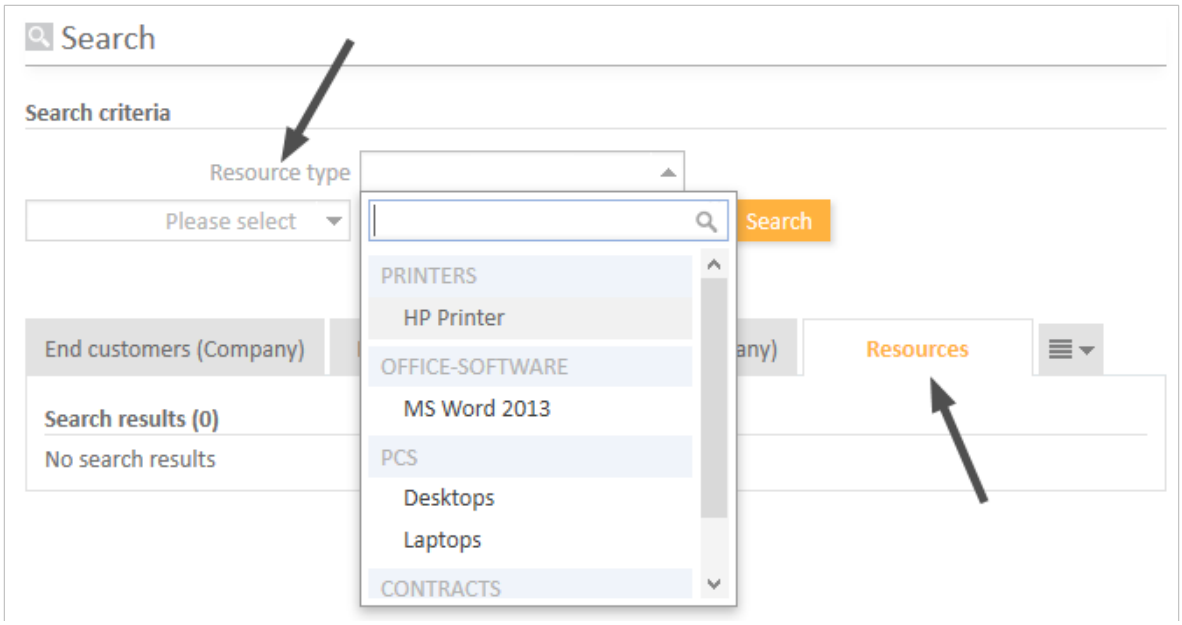


Figure 614: *ConSol CM Web Client - Detailed Search for resources*

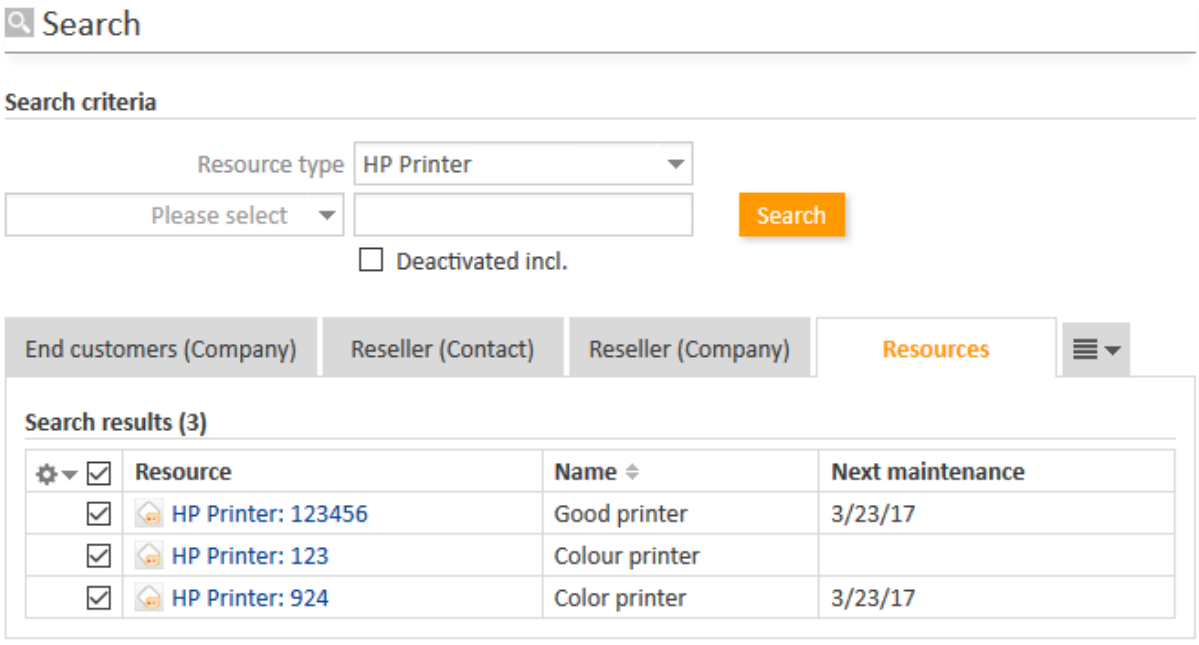


Figure 615: *ConSol CM Web Client - Result list of a Detailed Search for resources*

How You, as an Administrator, Can Manipulate the Display of Resources in the Detail Search

- On the Detailed Search page, in the drop-down menu where resource types are listed, a resource type will only appear if the respective resource field has been annotated with `field indexed` (usually with the value “transitive”).

- The value of the system property cmweb-server-adapter, searchPageSize defines the number of hits which are initially displayed

If search actions are defined for a certain resource type, the engineer will see the possible actions as Activities in the Web Client in each search result list where resources of this type are listed. Please refer to section Action Framework - Search Actions for details about this topic.

G.1.4 CM/Resource Pool - Setting Up the Basic Resource Model

G.1.4.1 Introduction

This chapter will guide you through the complete set-up of a resource model. You will learn how to

- create a resource category
- create two resource types within this resource category
- create data fields for the resources, i.e., create resource field groups for the resource types
- create resource fields within the resource field groups. i.e., create the data fields
- define templates for the display format of the resource names in the Web Client

In our example, you are an administrator who has to set up a resource model for a company which wants to manage their IT assets, as well as their products, using the Resource Pool.

The complete model will look like the one in the following figure. We will show you how to define the Printers and the Office software to demonstrate the basic principles. Then you should be able to configure all remaining resources yourself.

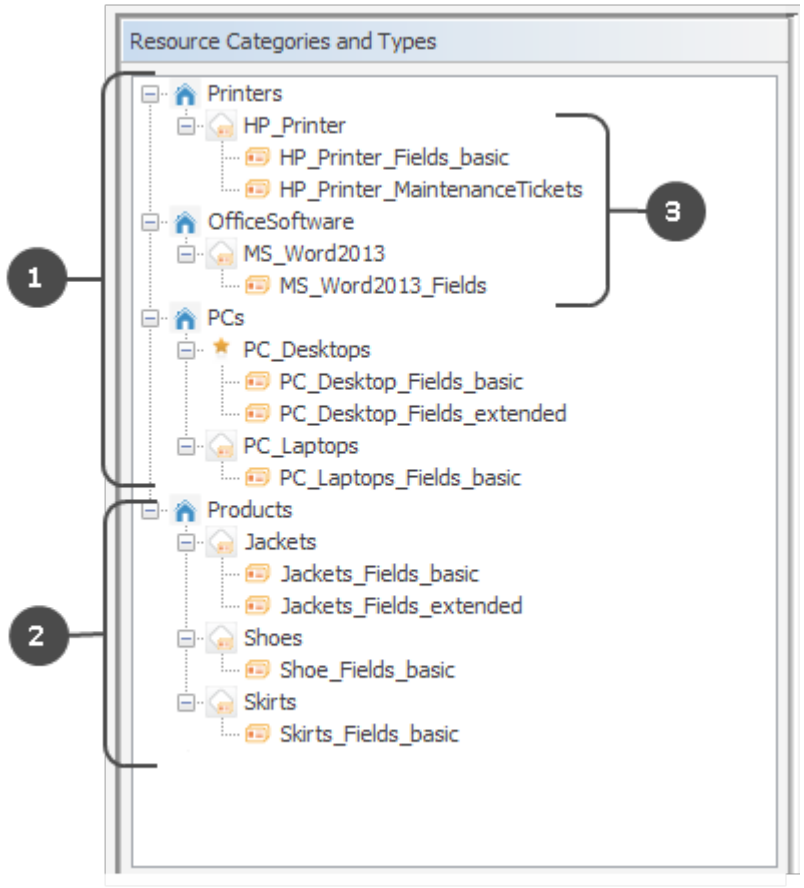


Figure 616: *ConSol CM Admin Tool - Resources, Data Models: Example of a resource model*

The above figure shows a resource pool data model with the following elements:

- Resources for IT management (1)
- Resources for product management (2)

The following example covers printers and office software (3).

G.1.4.2 Creating a Resource Category with the First Resource Type

A resource category represents the objects at the highest hierarchical level in the data model of the Resource Pool. In our example, all printers are managed in a resource category *Printers* and all office software packages are managed in a resource category called *OfficeSoftware*. The following figure shows some resource categories and resource types in the Web Client. This should provide you with an overview of the configuration in the Admin Tool and its consequences in the Web Client. Please keep in mind that the technical object names which are used in the Admin Tool are usually not the (localized) names which are displayed in the Web Client.

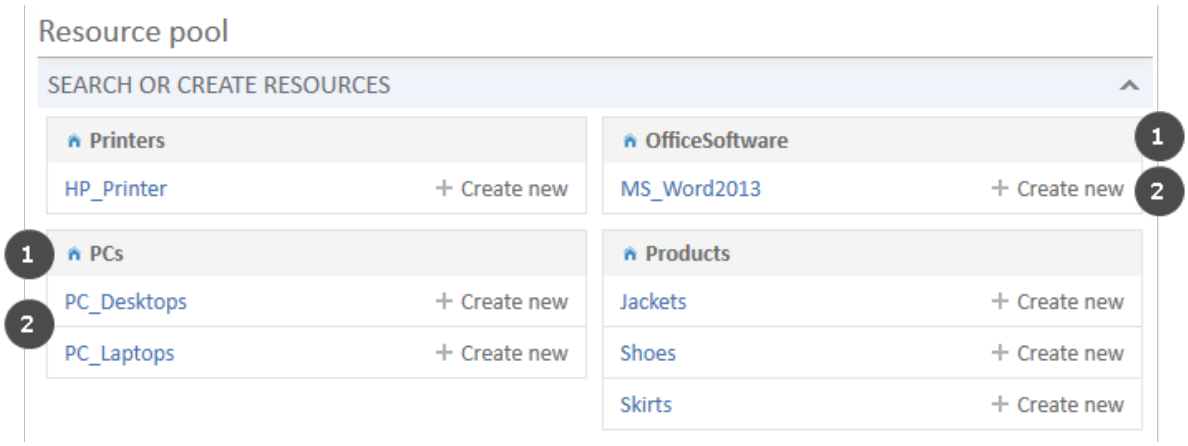


Figure 617: *ConSol CM Web Client - Resources in the Resource Pool Dashboard*

- Resource categories (1)
- Resource types (2)

The first step in a new (empty) resource model is to create a new resource category which contains one resource type. The resource type contains one resource field group.

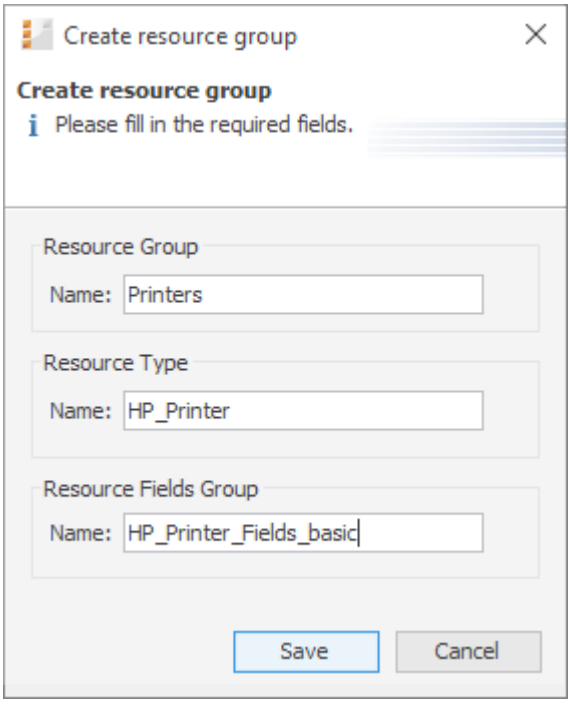


Figure 618: *ConSol CM Admin Tool - Resources, Data Models: Creating a new resource category*

To create a resource category, fill out the following fields. The localization for the names of resource category, resource type and resource field group is done using the fields on the main *Data Models* tab and is explained in section Localization of Data Fields. The descriptions are localized using the *Localize* button which is explained in section Localization of Objects in General, Type 1.

- **Resource category**, Name
  The technical name of the resource category.
- **Resource type**, Name
  The technical name of the first resource type within the resource category.
  More resource types for the resource category can be added later on.
- **Resource field group**, Name
  The name of the first resource field group (similar to a ticket field group for ticket data) within the resource type. In case the resource field group is displayed in the Group section of the resource page (annotation `show-in-group-section` = “true”), the localized name will be displayed as header of the respective tab.
  More resource field groups for the same resource type can be added later.

Click *Save* to create the resource category, type and first field group.

Once you have defined the resource category, you can add some more data for the created objects.

Editing the Resource Category

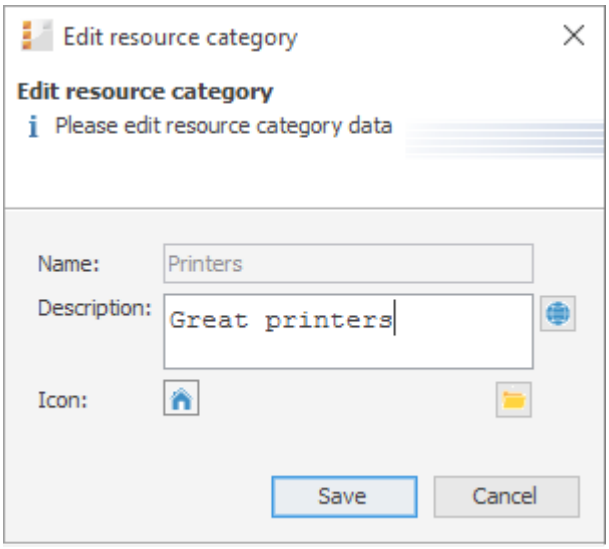


Figure 619: *ConSol CM Admin Tool - Resources, Data Models: Editing a resource category*

Mark the resource category and click the *Edit* button to edit some resource category parameters:

- **Description**
  You can enter the description directly in the pop-up window, which will become the default description, or you can use the *Localize* button to enter localized values for each language. The resource category description is displayed only in the Admin Tool, not in the Web Client.
- **Icon**
  Select one of the ConSol CM standard icons by clicking the icon, or use the file browser to upload an icon of your choice. The resource category icon will be displayed in the Web Client next to the name of the resource category. In this way, you can, for example, select a Printer icon for printers, a PC icon for PCs, etc. Allowed image file formats are: jpg, png, gif. We recommend to use 32 x 32 px as image size to achieve a reasonable size of the icon on the resource type page. In tables and at other locations on the GUI where the resource is displayed, the icon is re-sized automatically.

Figure 620: *ConSol CM Web Client - Customized icon for a resource category*

Click *Save* to commit your changes.

Editing the Resource Type

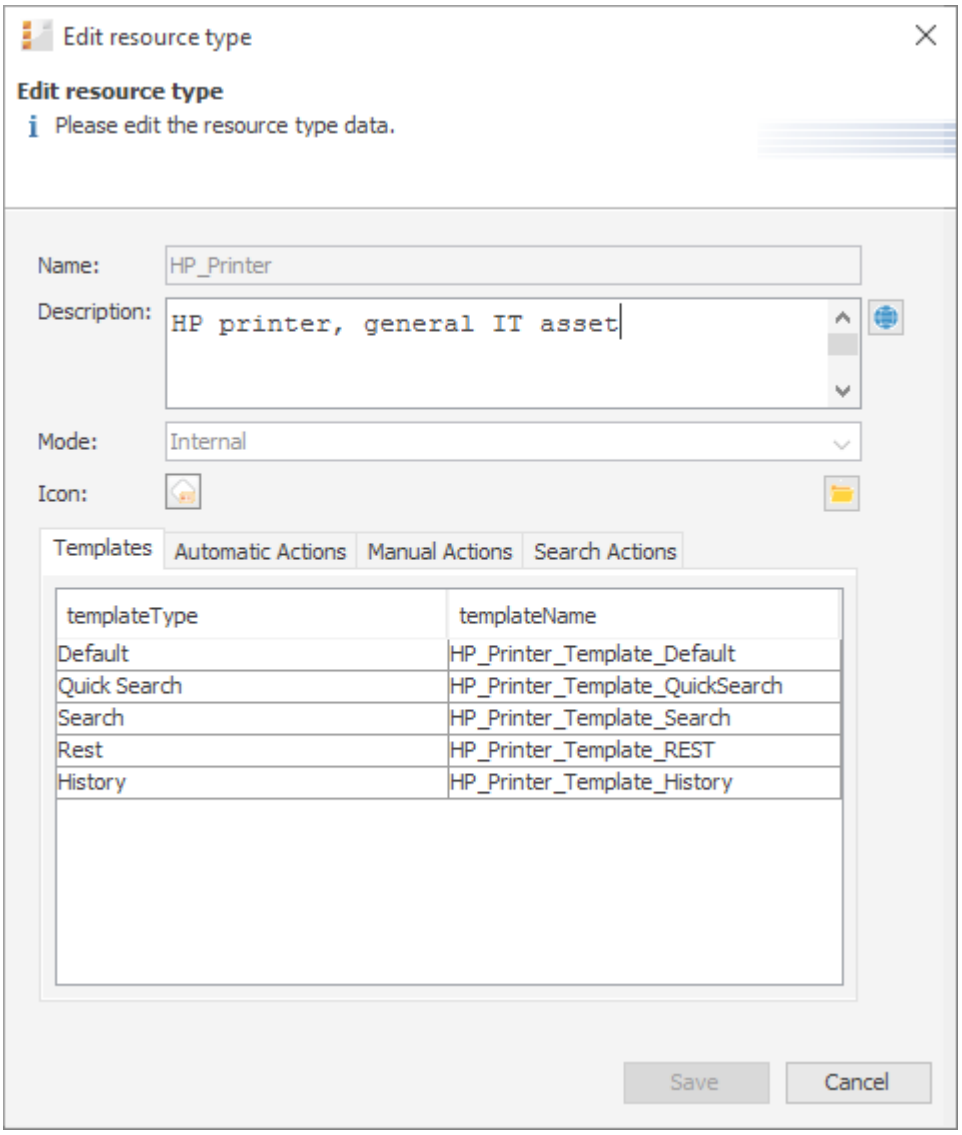


Figure 621: *ConSol CM Admin Tool - Resources, Data Models: Editing a resource type*

Mark the resource type and click the *Edit* button to edit the following resource type parameters:

- **Description**
  You can enter the description directly in the pop-up window, this will become the default description. Or you can use the *Localize* button to enter localized values for each language. The resource type description is displayed as information in the Admin Tool only, not in the Web Client.

- **Mode** (for a detailed explanation of all modes, refer to section Information about Resource Modes)
  - **Internal**
    Resource data is fully managed by ConSol CM. For details see section Internal Resources.
  - **On the fly**
    Resource data are retrieved from an external backend system whenever requested (available in ConSol CM version 6.10.3.0 and higher). For details see section On the fly Resources.
  - **Cached**
    Resource data come from an external backend system and are cached in ConSol CM. A refresh of an item can be triggered manually (available in ConSol CM version 6.10.3.0 and higher). For details see section Cached Resources.
  - **Imported**
    Resource data come from an external backend system and are stored in ConSol CM. An update requires a new bulk data import (available in ConSol CM version 6.10.3.0 and higher). For details see section Imported Resources.
- **Icon**
  Select one of the ConSol CM standard icons by clicking the icon, or use the file browser to upload an icon of your choice. The resource type icon will be displayed in the Web Client next to the name of the resource type, e.g., on the resource type page. In this way, you can, for example, select a Printer icon for printers, a PC icon for PCs etc. Allowed formats are: jpg, png, gif. We recommend to use 32 x 32 px as image size to achieve a reasonable size of the icon on the resource page. In tables and at other locations on the GUI where the resource is displayed, the icon is re-sized automatically.
- **Templates**
  Here you can define the templates for the display format of resource data in the Web Client (analog to templates for customer data). This is explained in detail in section CM/Resource Pool - Templates for Resource Data.
- **Automatic Actions**
  Here you can assign automatic resource actions to the resource type. The resource actions have to be defined first, using the *resource actions* section of the Admin Tool. See section CM/Resource Pool - Resource Actions for a detailed explanation.
- **Manual Actions**
  Here you can assign manual resource actions to the resource type. The resource actions have to be defined first, using the *resource actions* section of the Admin Tool. See section CM/Resource Pool - Resource Actions for a detailed explanation.
- **Search Actions**
  Here you can assign search actions to the resource type. The resource search actions have to be defined first using the *Scripts and Templates* section of the Admin Tool. See section Action Framework - Search Actions for a detailed explanation.

The resource relations are also defined for a resource type. This is described in detail in the section CM/Resource Pool - Resource Relations.

Information about Resource Modes

The mode of a resource type defines whether the resource data is stored in the ConSol CM system and - if required - how the transfer from the external system to ConSol CM is managed. Once the mode has been set and resources of the respective type are present in the system, the mode cannot be changed!

In ConSol CM, you have to distinguish between *internal resources* and *external resources*. The latter comprise *On the fly Resources*, *Cached Resources* and *Imported Resources*, i.e., all resources which are not completely stored in and managed by ConSol CM.

Only internal resources can be created using the Web Client. External resources can be neither created nor modified (permanently) nor deleted using the Web Client. For all external resources the complete data model has to be defined in the Admin Tool before the first data transfer can be performed. The use of *On the fly Resources* and *Cached Resources* is based on the implementation of a specific interface. Therefore, those resources can only be used with a customer-specific ConSol CM *.ear* file. The respective development project has to be built and deployed before the first data transfer takes place. The following sections provide a detailed overview of all four resource types.

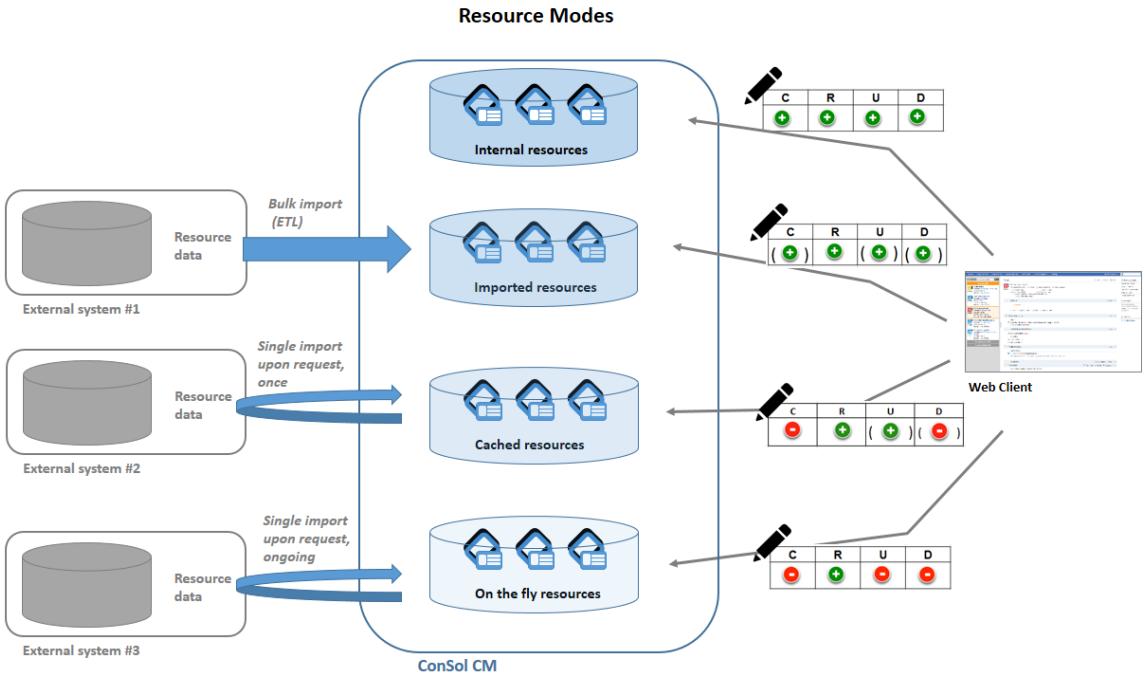


Figure 622: *Resource modes*

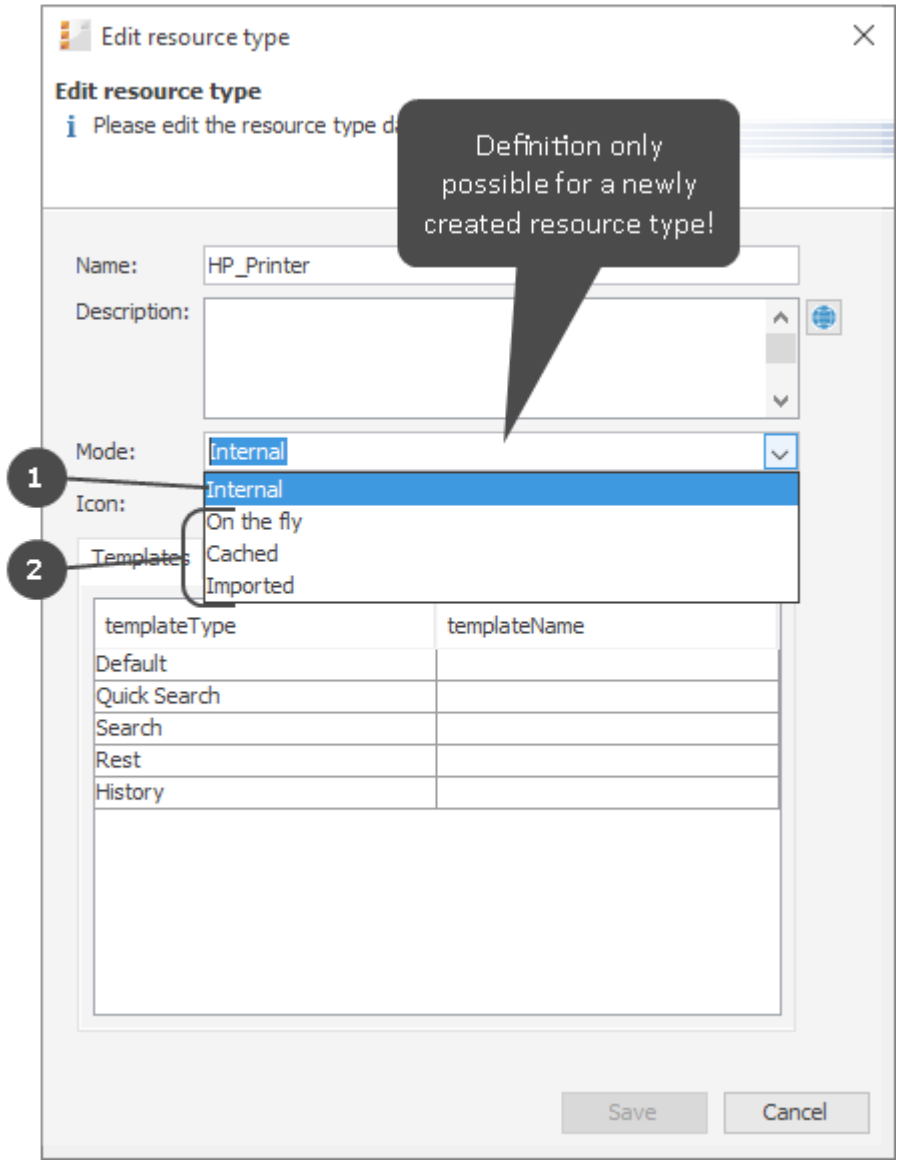


Figure 623: *ConSol CM Admin Tool - Resources, Data Models: Configuration of resource type mode*

- (1) Internal
- (2) External

Internal Resources

The resource data is fully managed by ConSol CM. Engineers who have sufficient access permissions can use the Web Client to create, modify or delete resources.

On the fly Resources

The resource data is retrieved from an external backend system whenever requested. The mode *on-the-fly* relies on a customer-specific implementation of the Java interface `ResourceExternalSource`.

An *on-the-fly* resource does not have any data which is stored in ConSol CM except for an external ID. This ID is used to retrieve data from the external system upon request.

In the Web Client:

- The Detailed Search is not available.
- In order to search CM for an on-the-fly resource, only the external ID can be used.
- There is no resource page.
- There is no resource type page
- The resource is only available/visible in the relations section of
    - tickets
    - customers
    - other resources.
- The resource data cannot be edited.

Cached Resources

The resource data originates from an external backend system. The data is transferred into ConSol CM upon the first request and is then cached in the ConSol CM database. The first request to the external system is executed when the resource is linked to another object (a ticket, a customer or another resource), i.e., when a relation is established. The mode *Cached* relies on a customer specific implementation of the Java interface `ResourceExternalSource`.

In the Web Client:

- The Detailed Search is only available for resources that have been requested at least once and are thus present in ConSol CM with their full data set.
- For resources which are already present in ConSol CM (i.e., are cached)
    - a resource page is available.
    - a resource type page is available.
    - a refresh of the resource data can be triggered manually on the Resource Page.
- In order to search ConSol CM for a *cached* resource which has not yet been requested, only the external ID can be used.
- There are no Resource Pages for resources which have not yet been cached.
- The resource is only available in the *Relations* section of tickets, customers or other resources. The resource fields are only displayed in the *Relations* section as well.
- The resource data cannot be edited.

Imported Resources

The resource data originates from an external backend system and is stored in the ConSol CM database. Once the data has been imported into ConSol CM, it is treated almost like internal data. The import into ConSol CM usually relies on an *ETL* (*Extract - Transform - Load*) job which is run once or on a certain schedule.

In the Web Client:

- The Detailed Search is fully available for imported resources.
- A resource page is available.
- A resource type page is available.
- Imported resource cannot be created using the Web Client (only via import).
- The resource data can be edited, but a new import might overwrite changes.

In order to integrate external resources into your CM system, you have to write a special Java / Groovy class which implements the interface `ResourceExternalSource` with the three methods

- PageResult<Resource> **searchByPattern**(ResourceType pType, String pPattern, int pPageSize, int pPageNumber)
- Resource **importResource**(Resource pResource)
- Resource **getByExternalId**(ResourceType pType, String pExternalId)

A detailed example for a mock class is provided in the *ConSol CM Release Notes* for version 6.10.3. You will have to implement the methods for your system to provide real data from your IT infrastructure.

Editing the Resource Field Group

A resource field group is a collection of resource fields, the data fields for resource data (analog to a ticket field group which is a collection of ticket fields for ticket data). Resource fields can never be managed as singular objects, they are always managed (e.g., faded in or out) as a resource field group.

A resource field group has the following group-specific parameters. Mark the desired resource field group and click *Edit* to edit those values (see the next section).

Editing Resource Field Group-Specific Parameters

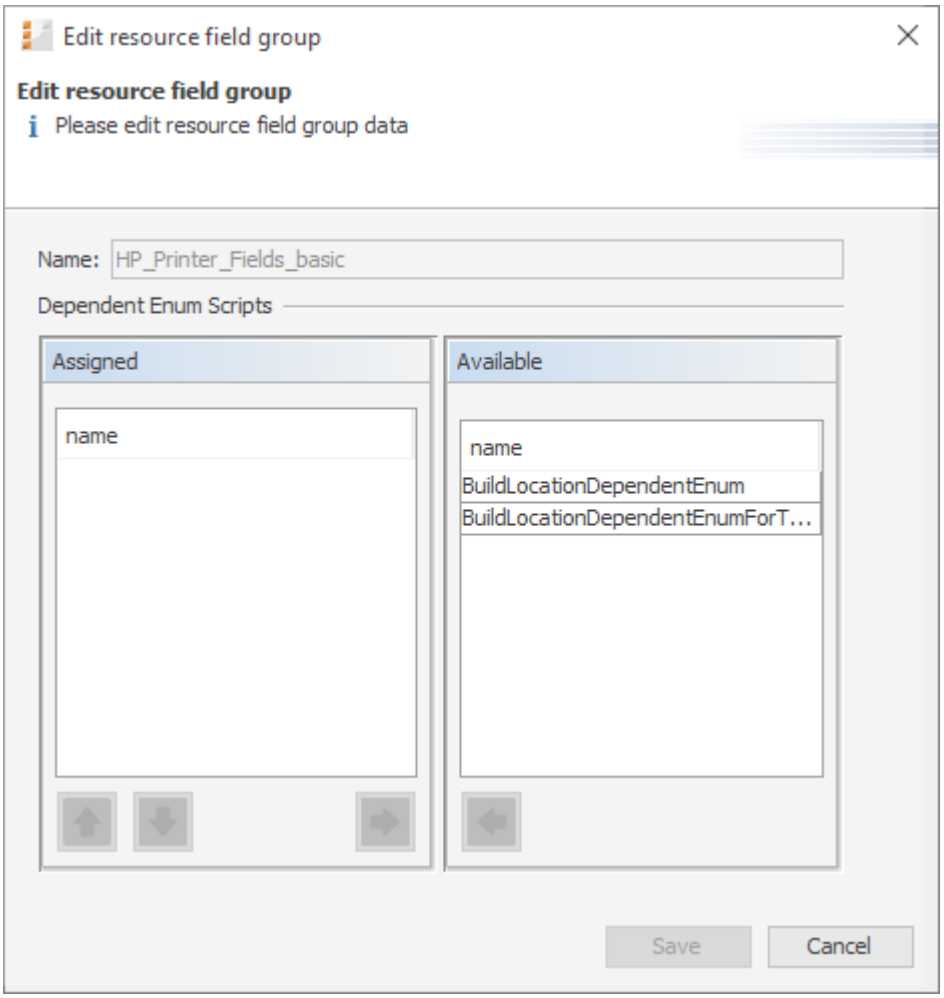


Figure 624: *ConSol CM Admin Tool - Resources, Data Models: Editing resource field group-specific parameters*

Here, you can edit the following fields:

- **Name**
  The technical name of the resource field group. This should only be changed if truly required, as changing it might have side effects on all scripts (workflow and Admin Tool) which use the resource category name. To change the term of the resource field group which is displayed in the GUI, use the mechanism which is explained in section Localization of Data Fields.
- **Dependent Enum Scripts**
  Here you can assign a Dependent Enum to the resource field group. The Dependent Enum has to be defined as an Admin Tool script first, as described in section Scripts of Type Dependent Enum.

Setting Resource Field Group Annotations

A resource field group can have annotations. Select the resource field group and click the *Annotations* button to set or unset annotations. The annotations are the same as for ticket field groups. All annotations are listed and explained in Annotations.

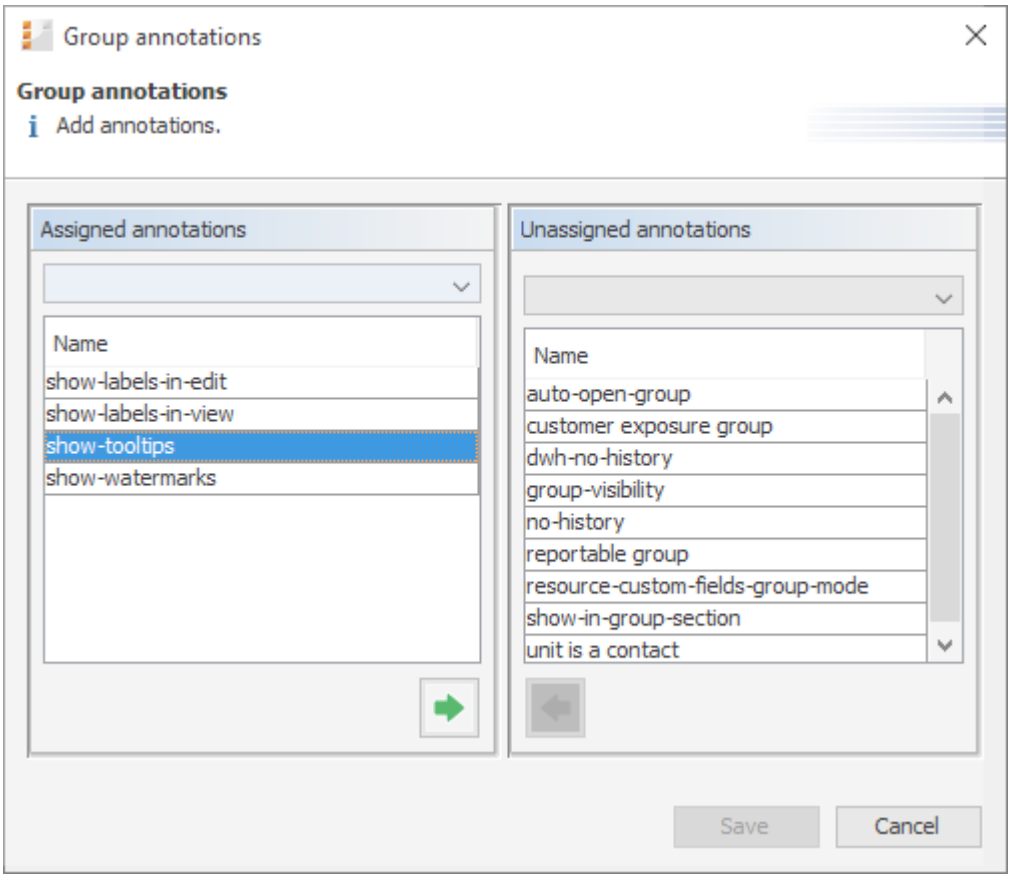


Figure 625: *ConSol CM Admin Tool - Resources, Data Models: Resource field group annotations*

Creating and Editing Resource Fields

Resource fields are containers for the data of resources (analog to ticket fields for ticket data). Resource fields always belong to a resource field group. To add or edit resource fields of a resource field group, select the group in the resource data model and enter all required data on the right-hand side of the Admin Tool.

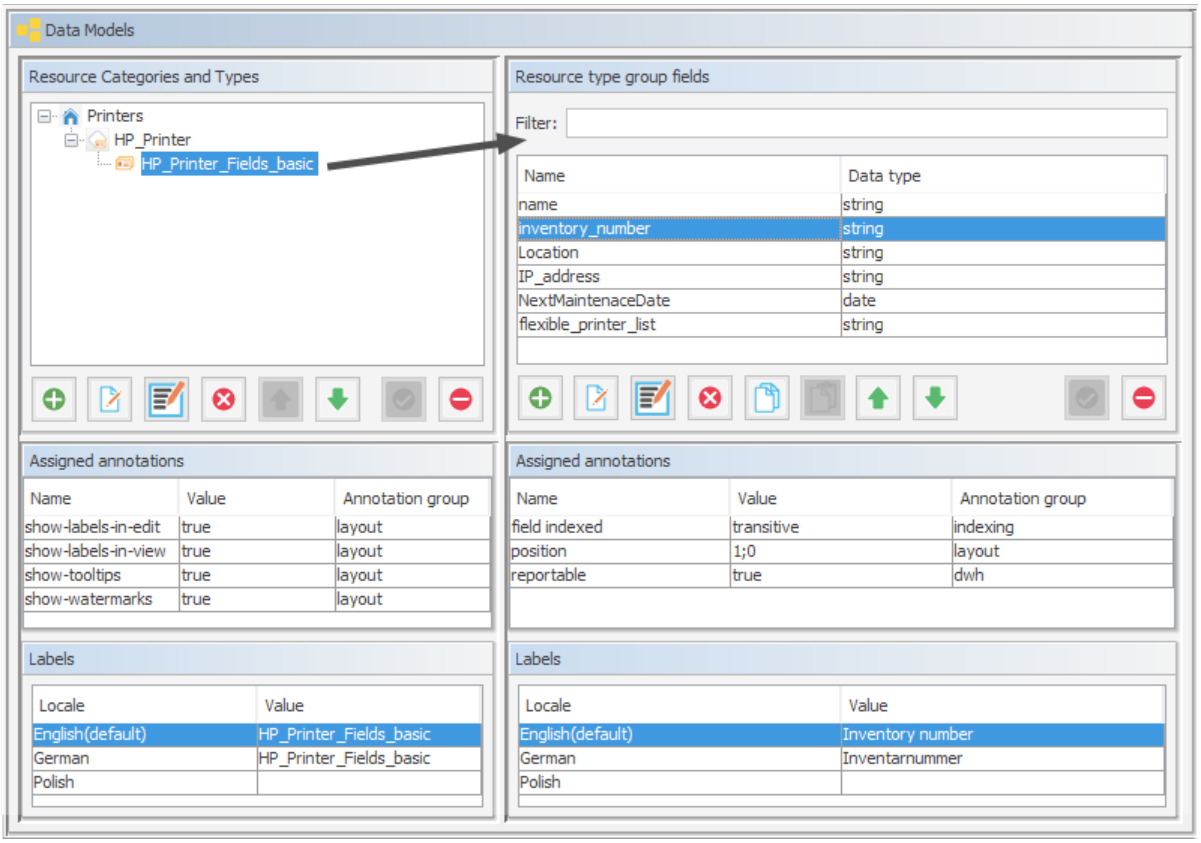


Figure 626: *ConSol CM Admin Tool - Resources, Data Models: Managing resource fields*

Creating a New Resource Field

To create a new resource field, click the *Add* button and enter the required data in the pop-up window.

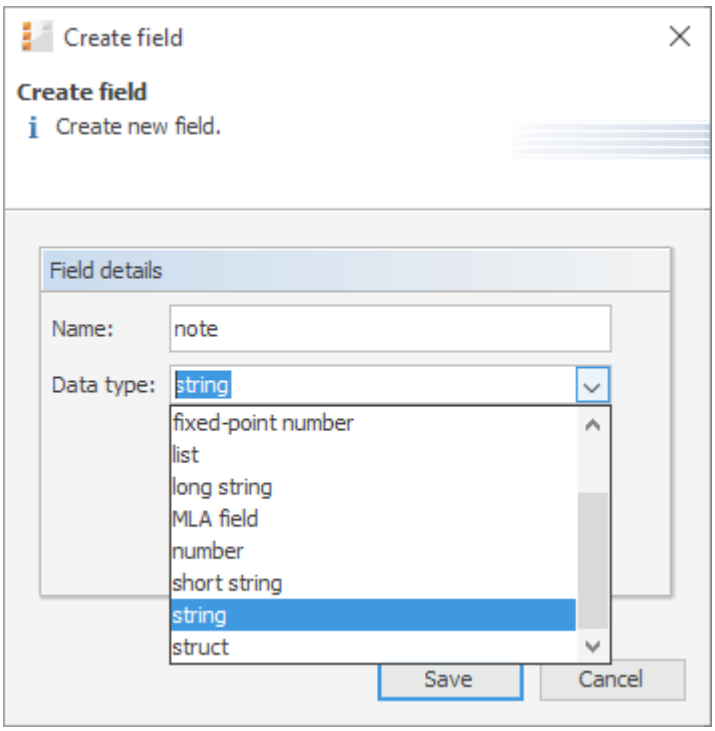


Figure 627: *ConSol CM Admin Tool - Resources, Data Models: Creating a new resource field*

Enter the following data:

- **Name**
  The technical name of the resource field. This is used in scripts and templates. The localization mechanism is explained in section Localization of Data Fields.
- **Data type**
  The data type of the resource field. You can select a data type from a drop-down list, like for ticket fields or customer fields. The following section provides detailed information about the available data types.

Types of Data Fields

The following data types are available for ticket fields, customer fields and resource fields.

- **autocomplete**
  A data field which contains a scripted autocomplete list. This is a dynamic list which is based on a script of type *Text Autocomplete*. A detailed explanation of scripted autocomplete lists is given in section Scripted Autocomplete Lists.
- **boolean**
  Values: true/false. Depending on the annotation `boolean-type`, the value is displayed as checkbox, radio buttons, or drop-down list.

If you work with scripts, either in CM workflows or in the Admin Tool, please note that the behavior of boolean fields which are represented as checkboxes, i.e. with annotation `boolean-type = checkbox` (default) is different depending on the CM version!

- **In CM versions prior to 6.9.4.0:**
  If a boolean field has not been touched, its value is “false”. If it is checked, its value is “true”, and if it is unchecked again, its value is “false ”again.
- **In version 6.9.4.0 and up:**
  If a boolean field has not been touched, its value is “NULL”. If it is checked, its value is “true”, and if it is unchecked again, its value is “false”.

Fields which have already been filled with values in the database will not be changed during an update from a version prior to 6.9.4.0 to a version 6.9.4.0 and up.

Boolean fields represented as radio buttons (annotation `boolean-type = radio`) or drop-down menu (annotation `boolean-type = select`) are always shown and behave as described for versions 6.9.4.0 and up, i.e., with “NULL ”value if untouched.

- **date**
  Format and accuracy can be set by annotations.
- **enum**
  For sorted lists. The engineer can choose one of the enum values in the Web Client. Enums and values have to be created previously within the Managing Sorted Lists: Enum Administration. Select the desired *Enum type* and *Enum group* in the fields below.
- **list**
  A data field of this data type is the first step to creating a list (one column) or a table (multiple columns) of input fields in the Web Client.
  - For a **table** the next step will be to create another field of type *struct* (see below) to contain the input of the individual list fields (which will become the columns of the table). So, if you want to create a table you have to define a field of the type *struct* first (see below) before you can add the fields for the table columns.
  - For a **simple list**, the next step will be to create fields which belong to the list. No *struct* is required.

  For all fields belonging to a list or table you have to set the dependencies in the field *Belongs to* (see below). For example, a table field (which is a regular data field) always belongs to a *struct*, a struct always belongs to a *list.*
- **struct**
  A data field of this type defines a data structure (line of a table) which groups one or multiple fields. It is the second step to building a table after you have created a field of the type *list*. Add the fields for the columns of the table in the next step. The dependencies have to be set for each field in the *Belongs to* field (see below), i.e., a *struct* always belongs to a *list*.

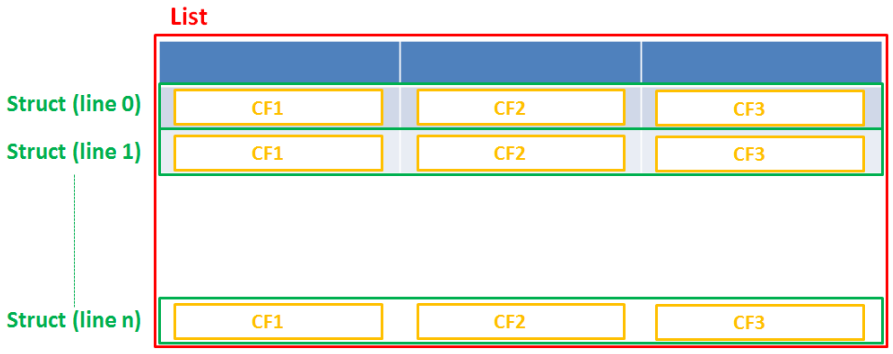


Figure 628: *Scheme: List of structs*

Technically spoken, the list is an array which contains a map (= key:value pairs) in each field.

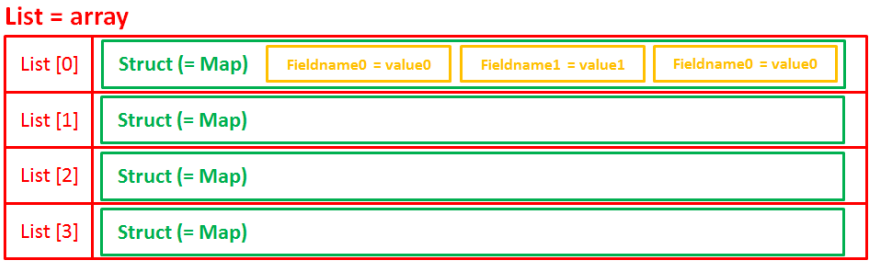


Figure 629: *List of structs, technical principle*

- **number**
  For integer values.
- **fixed point number**
  For numbers with a fractional part, e.g., currencies. You have to enter the total number of digits (*Precision*) and the number of digits that fall to the right of the decimal point (*Scale*) in the respective fields below.
- **string**
  For up to 4000 alphanumeric characters.

> ⚠ Restriction when using an Oracle database: at most 4000 bytes can be saved in UTF encoding. Starting with Oracle12c.

- **long string**
  For large objects.

  > ⚠ For *long strings* the limit depends on the database system used for ConSol CM: MS SQL Server: 2 GByte; MySQL: 4 GByte; Oracle: 16 - 64 GByte (depending on page size of tablespace).

- **short string**
  For up to 255 alphanumeric characters.

  > ⓘ For *string* fields, you can use specific annotations to fine-tune the field definition. For example, a *string* field can be defined to contain a URL which will automatically be displayed as hyperlink or can be the hook for an autocomplete list. Please read the following section.

- **contact data reference**
  Special data type used internally for referencing the contacts associated with a ticket. In FlexCDM (i.e., starting with CM version 6.9.0), this data type is no longer displayed but only used internally in the CM system.
- **MLA field**
  This data type is used for fields that contain hierarchical lists with a tree structure called *MLA* (Multi Level Attributes). The name of the field is the name of the new MLA that has to be defined within the MLA Administration. The group of the field has to be referenced when the MLA is created.

> ⚠ The data type you choose on creating a data field cannot be changed afterwards!

Details about String Fields: Use Annotations to Fine-Tune Strings

String fields are widely used for customer, ticket, and resource data and strings can be used to contain various content, for example, a text box with a comment, a simple input field with only 20 characters, a URL or a password. The fine-tuning of string fields is implemented using specific annotations which are all listed on the Annotations page. However, since work with these annotations is an every-day task of CM administrators, the most important and most commonly used annotations will be explained here as well.

How can I ...

... insert a **text box** instead of a single line?

Value for annotation `text-type`: “textarea”

The size of the text box can be adjusted, displayed as standard text box depending on web browser. Use the `field-size` annotation in case a specific size of the text box is required.

... hide the input of the fields for **passwords**?

Value for annotation `text-type`: “password”

Only dots will be displayed. This annotation does **not** define the field to contain a password! It only defines the display mode! Use the `password` annotation to define a string field to contain the CM/Track password.

... display a **hyperlink**, display the name instead of the link?

Value for annotation `text-type`: “url”

Input will be displayed as a hyperlink in view mode. String has to match a specific URL pattern:

- "^((?:mailto\:|(?:(?:ht|f)tps?)\://)1\S+)(?: (?:\|)?(.*))?$"

First part of the string is the link (url), second part is the name which should be displayed.

Example: "http://consol.de ConSol"

... display a **file link**?

Value for annotation `text-type`: “file-url”

Input will be displayed as a link to a file on the file system. The web browser has to allow/support those links!

Example: Enabling file:// URLs in a Firefox browser

Add the following lines to either the configuration file `prefs.js` or to `user.js` in the user profile. On a Windows system usually in a folder like `C:\Users\<USERNAME>\AppData\Roaming\Mozilla\Firefox\Profiles\uvubg4fj.default`

- user_pref("capability.policy.localfilelinks.checkloaduri.enabled", "allAccess");
- user_pref("capability.policy.localfilelinks.sites", "http://cm-server.domain.com:8080");
- user_pref("capability.policy.policynames", "localfilelinks");

Alternatively a Firefox browser add-on like ***Local Filesystem Links*** can be installed for better access to the referenced files and folders.

The link will also be displayed as tooltip.

The URL is correctly formed if the following conditions are met:

- It starts with `file:` followed by regular slashes:
  - three slashes "///" for files on the same computer as the browser (alternatively "//localhost/") or
  - two slashes followed by the server name followed by another slash for files on file servers accessible from the computer running the browser.
- These are followed by the full path to the file ending with the file name.
- The path on Microsoft Windows systems is also written with forward slashes instead of backslashes.

- The drive letter of a local path on Microsoft Windows systems is noted as usual, for example `C:`.
- Paths with spaces and special characters like "{, }, ^, #, ?" need to be percent encoded ("%20" for a space for example) for Microsoft Windows systems.

Example URLs:

- file://file-server/path/to/my/file.ext
- file:///linux/local/file.pdf
- file:///C:/Users/myuser/localfile.doc

See also the explanation about `file-url` in the section text-type

... define a **label**?

Value for annotation `text-type`: “label”

This will be a read-only field which is displayed in gray, use the ***label-group*** annotation to link label and input fields which belong together. Please take a look at the annotations for labels (`show-label-in-edit`, `show-label-in-view`) before implementing special label fields!

... define a field for the **valid email addresses**?

Value for annotation `email`: “true”

The field may only contain valid email addresses. Input will be validated according to standard email format <name>@<domain>.

... define a scripted autocomplete list?

Value for the annotation `text-type` = “autocomplete”

Optional: value for the annotation `autocomplete-script` = <name of the respective script>

A scripted autocomplete list is used to provide a drop-down menu which is filled dynamically using the input the engineer has provided so far. For example, when the user types "Mil", the possible values "Miller", "Milberg", and "Milhouse" are displayed as list and the engineer can select the one required for the field. You know this behavior from other autocomplete fields, e.g., the search for engineers for a ticket or the search for customers while creating a ticket. However, in these cases, CM generates the list automatically. The behavior cannot be influenced or customized. Scripted autocomplete lists, on the contrary, can be implemented by the CM administrator. The values are based on a result set which is dynamically created. The result set can contain strings, engineers, customers (Units), and resources.

A detailed description of scripted autocomplete lists is provided in section Scripted Autocomplete Lists.

Setting Resource Field Annotations

The characteristics of a resource field, e.g., the visibility, the position in the Web Client, and whether the field should be indexed or reportable, are defined using annotations. You might know this principle from your work with ticket field annotations.

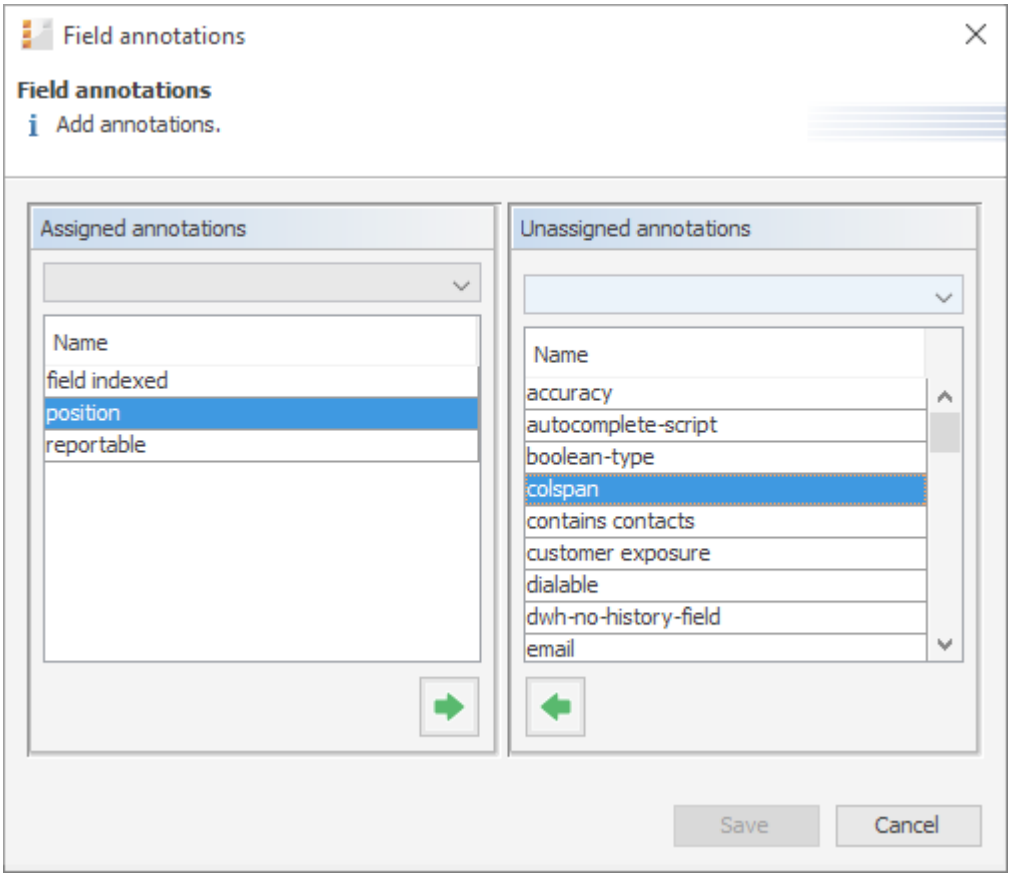


Figure 630: *ConSol CM Admin Tool - Resources, Data Models: Defining resource field annotations*

The annotations which are available are the same as for ticket fields and are listed and explained in detail in Annotations.

The `position` annotation marks the position of the field on

- the form which is used to create a new resource. See section A Short Introduction to CM/Resource Pool Functionalities in the Web Client, Resource Create Form.
- the resource page which is explained in section A Short Introduction to CM/Resource Pool Functionalities in the Web Client, Resource Page.

As with ticket layout, you can place the resource fields in the GUI by using the matrix principle and setting the position accordingly:

0;0	0;1	0;2
1;0	1;1	1;2
2;0	2;1	2;2
n;0	n;1	n;2

Figure 631: *Example matrix for position annotation*

G.1.4.3 Copying Resource Fields

You do not necessarily have to create every data field using the pop-up menu. You can also copy an existing data field. This might be a ticket field, customer field, or resource field. Select the respective data field in the data field group and click the *Copy* button. Then navigate to the data field group where the copied field should be placed. This might be a ticket field group, a customer field group, or a resource field group. You can copy a field from one object type (e.g. ticket) to another (e.g. customer). Click into the list of data fields of the target field group and click the *Paste* button.

Every type of field can be copied, from simple string fields over enums to entire lists of structs. To copy a list of structs or a list of fields, mark and copy only the list, the child objects (e.g., structs with fields) will be copied automatically.

These things will be copied:

- almost all annotations with their values
- all localized values

These things will not be copied:

- the following annotations
  - *ldapid*: This annotation can be only once in a data model. Duplication would cause an invalid model.
  - *username*: This annotation can be only once in a data model. Duplication would cause an invalid model.
  - *password*: This annotation can be only once in a data model. Duplication would cause an invalid model.
  - *position*: The value for this annotation must be unique, so the annotation is duplicated, but the value is removed from the copy.

- *ticket-list-position*: This annotation should be used very specifically and, thus, it should be present only for very few fields. Duplicating it would potentially multiply it although it should not be used for the majority of copies.

Naming of the copied field(s):

- Copy to other field group: field name is used
- Copy to same field group: <field name>_copy is used

Besides the Copy/Paste buttons you can also use the keyboard shortcuts:

- Copy: CTRL-C
- Paste: CTRL-V

G.1.4.4 Using Scripted Field Visualization for Resource Fields

Using scripted field visualization, you can enhance the display of data in resource fields. Please see section Scripts of Type Field Visualization for details.

G.1.4.5 Creating Further Resource Types within a Resource Category

To add a new resource type within a resource category, mark another resource type of the desired resource category in the list and click the *Add* button. Then fill in all required data as described in the section Editing the Resource Type.

G.1.4.6 Creating Resource Field Groups and Resource Fields

To create a new resource field group within a resource type, select another resource field group of the desired resource type in the list and click the *Add* button. Then enter all the required data as described in the section Editing the Resource Field Group.

G.1.4.7 Creating Further Resource Categories

To create a new resource category (OfficeSoftware in our example), select another resource category in the list and click the *Add* button.

Enter all the data as for the first resource category, see section Editing the Resource Category.

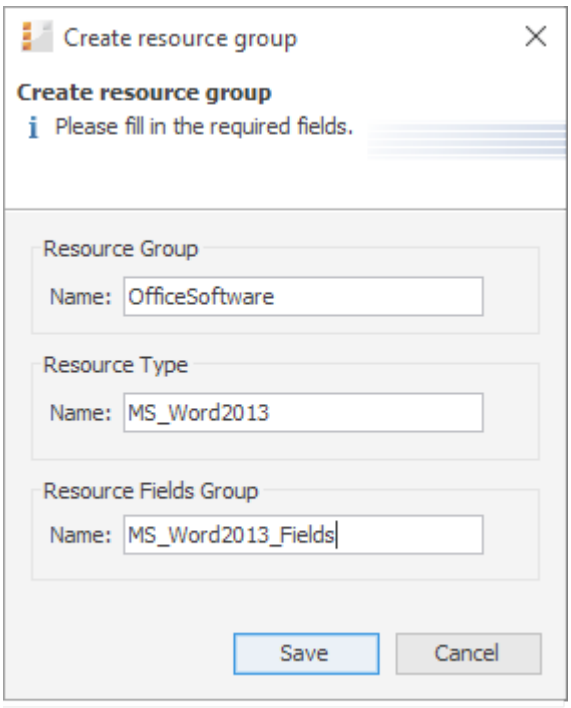


Figure 632: *ConSol CM Admin Tool - Resources, Data Models: Adding a new resource category*

To fill the new resource category with resource types and resource fields, proceed as described in the previous sections.

G.1.4.8 Example for a Complete Resource Model

In our example, the following resource model has been created. It will be used as a basis for later examples in this manual.

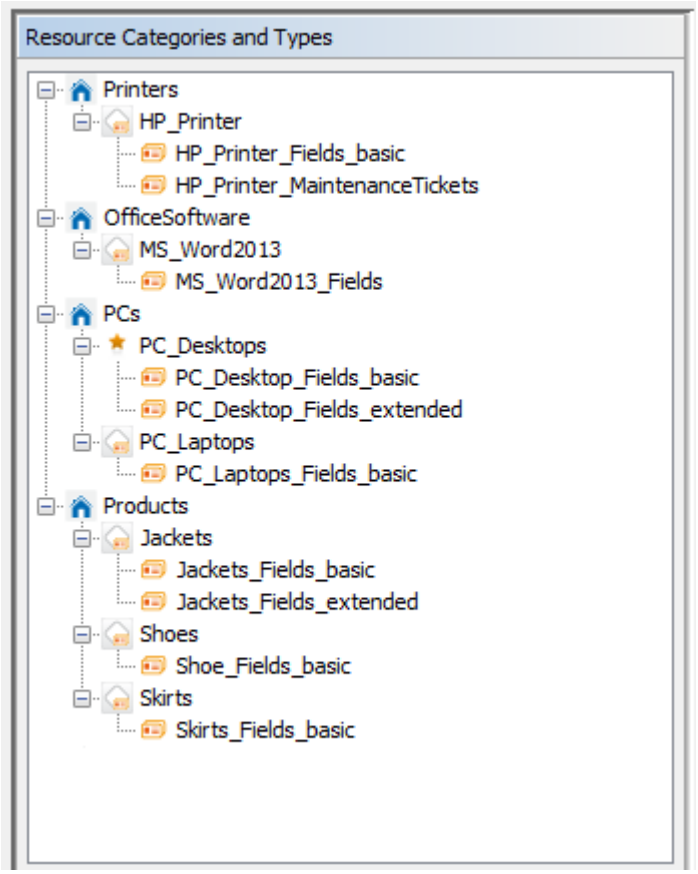


Figure 633: *ConSol CM Admin Tool - Resources, Data Models: Example for a resource model*

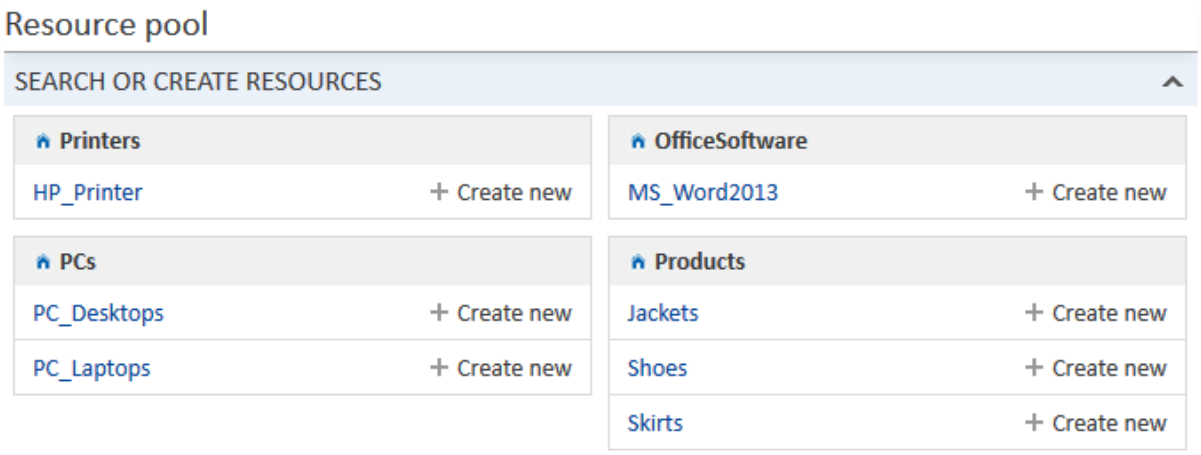


Figure 634: *ConSol CM Web Client - Example for a resource model (in Resource Pool Dashboard)*

G.1.4.9 Defining Resource Relations

This is explained in detail in the section CM/Resource Pool - Resource Relations.

G.1.4.10 Defining Resource Actions

This is explained in detail in the section CM/Resource Pool - Resource Actions.

G.1.4.11 Assigning Access Permissions for Resources to Engineers Using Roles

This is explained in detail in the section CM/Resource Pool - Assigning Permissions for Resources.

G.1.5 CM/Resource Pool - Templates for Resource Data

G.1.5.1 Introduction to Using Templates for the Display of Resource Data

In the ConSol CM Web Client, resource data sets are displayed in short form at various locations, based on templates. For example, in the ticket history, the resource name, inventory number and location might be required, whereas in the Quick Search only the name and inventory number should be displayed. This section will show you where short forms are used and how the respective templates are configured using the Admin Tool.

Templates are always defined for a resource type. In our example, *HP_Printer* is a resource type, i.e., the templates will be valid for all real-world HP printers which are stored in the ConSol CM system.

The configuration is based on the following principle (very similar to the display of customer data):

- A template for a specific location in the Web Client is assigned to a resource type in the resource model definition (navigation group *Resources*, navigation item *Data model*).
- The referenced template must be the name of a template which is stored in the navigation item *Scripts and Templates*, navigation group *System* of the Admin Tool. You, as an administrator, are free to assign names to the templates. All you have to make sure of is that the referenced template name in the resource type definition and the template name in the *Scripts and Templates* section are identical.

G.1.5.2 Creating and Editing Resource Data Format Templates

To create and edit resource data format templates, edit the desired resource type in the Admin Tool and enter the names of the templates. Name them however you like resp. following any naming conventions defined by your organization.

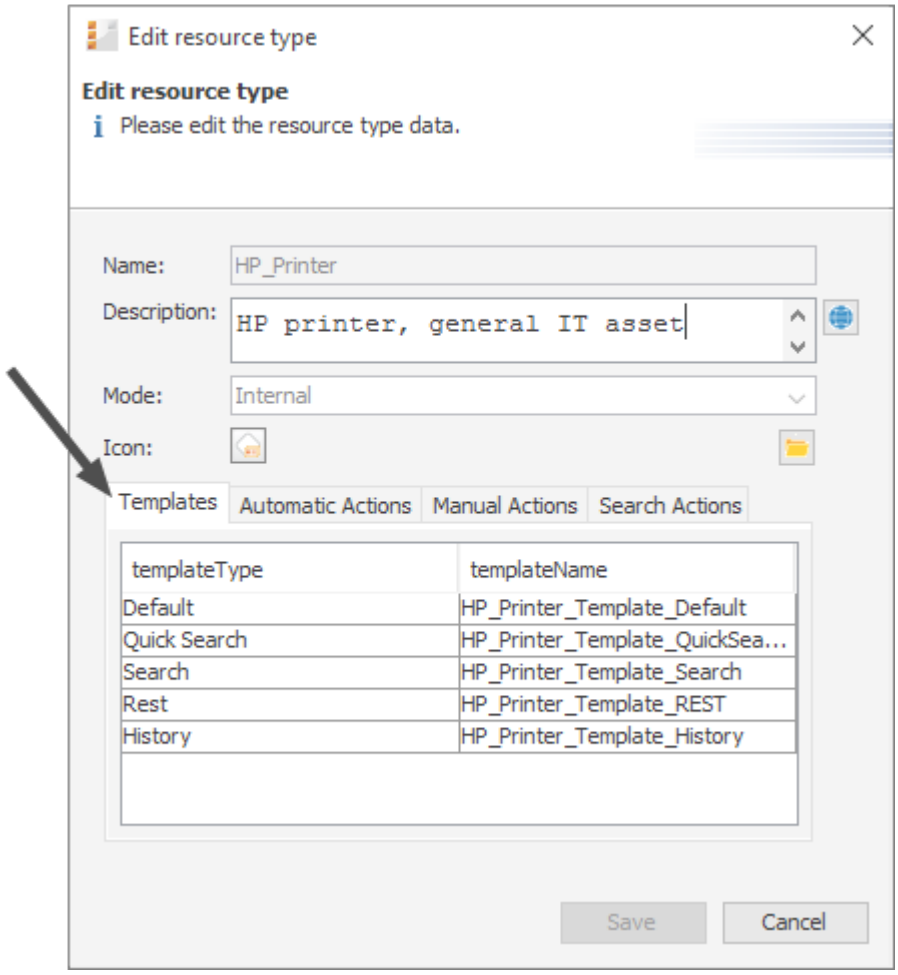


Figure 635: *ConSol CM Admin Tool - Resources, Data Models: Resource data format template definition for a resource type*

In the next step, you have to create the respective templates and store them in the *Scripts and Templates* section, as described in the following section.

G.1.5.3 Coding Resource Data Format Templates

General Principle

The templates are written in ***FreeMarker*** notation. For detailed information, please refer to the FreeMarker web site.

Within the templates, you work with three object types:

1. `resource`: this is the current resource object
2. the technical name of the resource field group
3. the technical names of the resource fields

Please note that there might be more than one resource field group within a resource type!

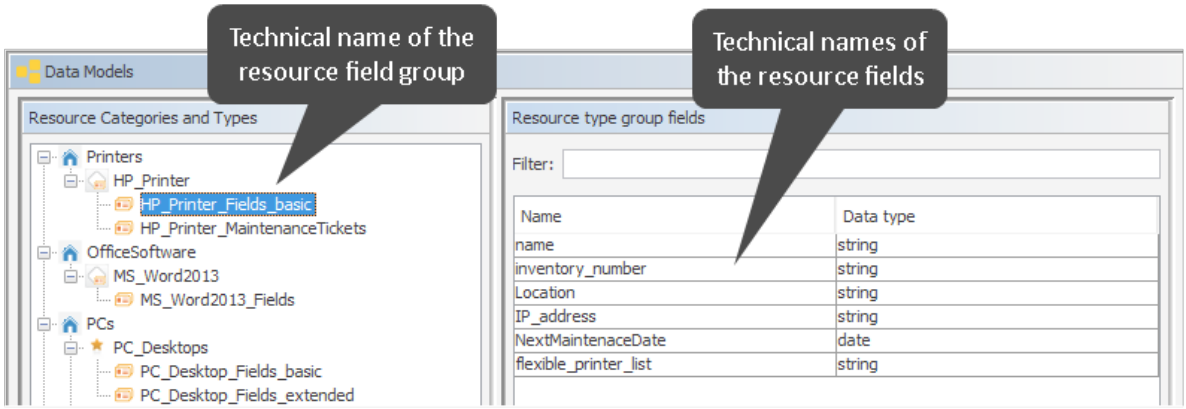


Figure 636: *ConSol CM Admin Tool - Resources, Data Models: Objects used in resource data template definition*

⚠ The resource templates must be single-row! They must not contain line-breaks!

Examples for Templates

```
// Example 1
// Resource Field group is named vehicleProperties, Resource Fields are named
 vehicleIdentification and vehicleLicense
${resource.get("vehicleProperties", "vehicleIdentification") + ' (' + resource.get
 ("vehicleProperties", "vehicleLicense") + ')'}
```

```
// Example 2
// // Resource Field group is named publicTransportProperties, Resource Fields is
 named passName
${resource.get("publicTransportProperties", "passName")}
```

Localizing Enum Values in Resource Templates

Starting with ConSol CM version 6.10.5.4, enum values can be localized, i.e. you can have the localized value displayed in the Web Client. For example, an enum *SLA_country* contains a list of countries where an SLA might be valid. In the Admin Tool, technical values are used and a localized value has to be defined for each desired language. In the Web Client, the correct country name in the respective browser locale should be displayed. If the locale of the browser is not configured explicitly, the standard CM locale will be used.

The following example works with the *SLA_country* enum.

```
SLA: ${resource.getFieldValue("SLA_Fields_basic","SLA_Name")!} (${localize
  (resource.getFieldValue("SLA_Fields_basic","SLA_country"))!})
```

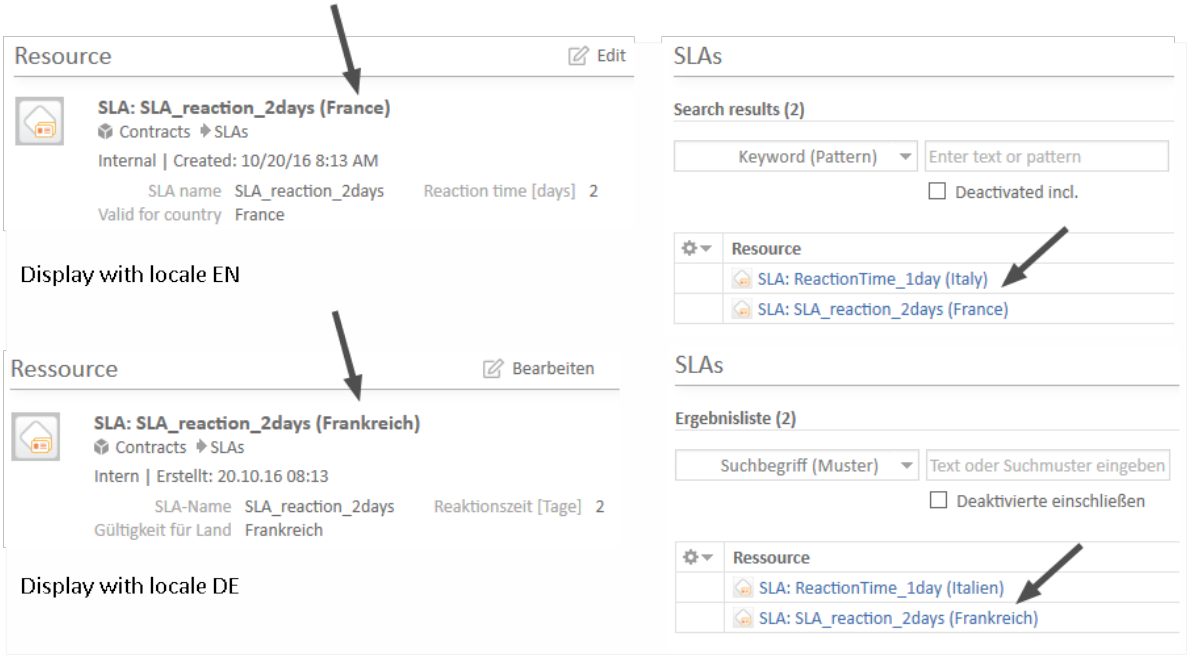


Figure 637: *ConSol CM Web Client - Display of an enum value based on a resource template*

G.1.5.4 Template Types

The following Template types can be defined:

- Default
- Quick Search
- Search
- Rest
- History

Default

This template must always be defined. If it is not, there will be an error in the Web Client (*unknown* is displayed as name of all resources of this type). The template will be used for all locations in the Web Client for which no other templates have been defined and will also be used as REST template for CM/Track in case there is no dedicated Rest template. The templates which are described in the subsequent sections will override the default template for the specific Web Client location.

```
HP Printer: ${resource.getFieldValue("HP_Printer_Fields_basic","inventory_
 number")!}
```

The template is used for

- Header of the Resource Page
- Dragged Resource in Drag-and-Drop Operations
- Favorites

Quick Search

This template defines the format of the resource data in the result of the Quick Search. For an example see section Quick Search.

Search

This template defines the format of the resource data in the result of the search (in suggestion lists) for resources when a resource is linked to another object, e.g., a ticket. For an example see section Search.

Rest

This template defines the format of the resource data in the result of the REST API. In the standard configuration, no resource data are displayed in the ConSol CM portal, CM/Track, which is based on the REST API. This template will only be effective when you address the REST API directly, e.g., for programming with ConSol CM interfaces. For an example see section Rest.

History

This template defines the format of the resource data in the result of the ticket, customer and resource history, e.g., when a relation to or from a resource was added. For an example see section History Sections of Resource Page, Customer Page and Ticket.

G.1.5.5 Web Client Locations for the Use of Templates

The following locations are specified:

- Header of the Resource Page
- Dragged Resource in Drag-and-Drop Operations
- Favorites
- Quick Search
- Search
- Rest
- History Sections of Resource Page, Customer Page and Ticket

Header of the Resource Page

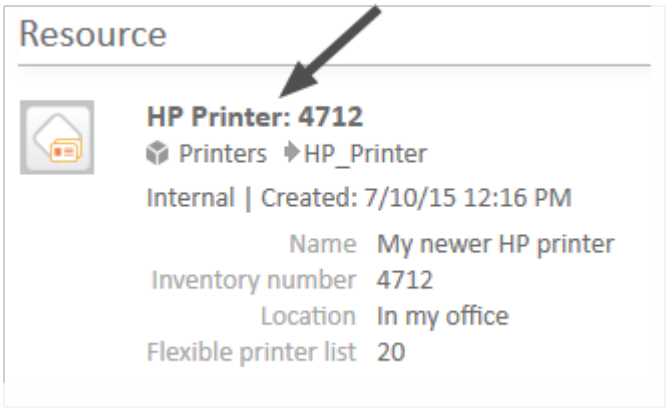
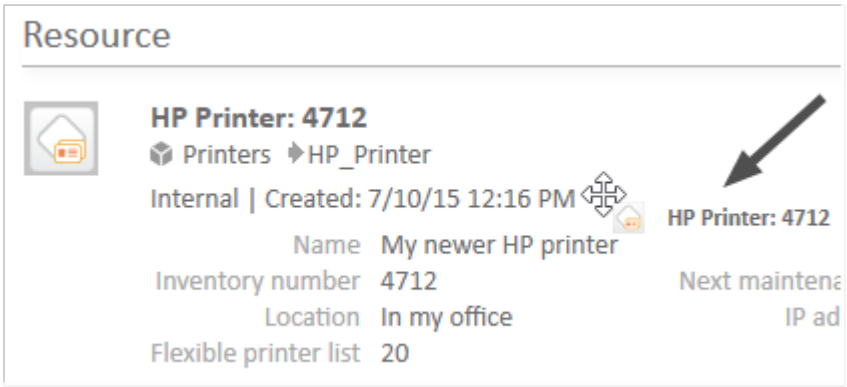


Figure 638: *ConSol CM Web Client - Header of the resource page*

Used template: *Default* (see example above)

Dragged Resource in Drag-and-Drop Operations



ConSol CM Web Client - Dragged resource in drag-and-drop operations

Used template: *Default* (see example above)

Favorites

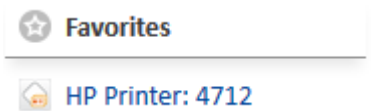


Figure 639: *ConSol CM Web Client - Resource name in Favorites*

Used template: *Default* (see example above)

Quick Search

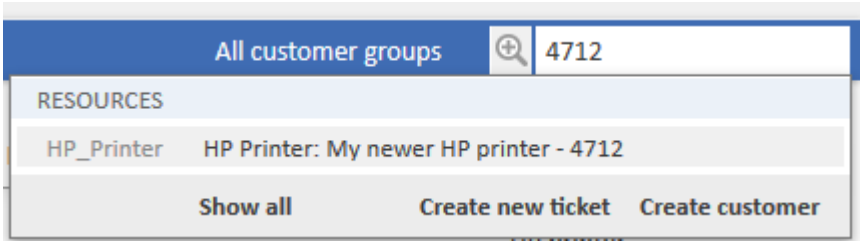


Figure 640: *ConSol CM Web Client - Results for Quick Search (resource)*

Used template: *Quick Search*, if this is defined. If not, *Default*.

```
HP Printer: ${resource.getFieldValue("HP_Printer_Fields_basic","name")!} -
 ${resource.getFieldValue("HP_Printer_Fields_basic","inventory_number")!}
```

Search

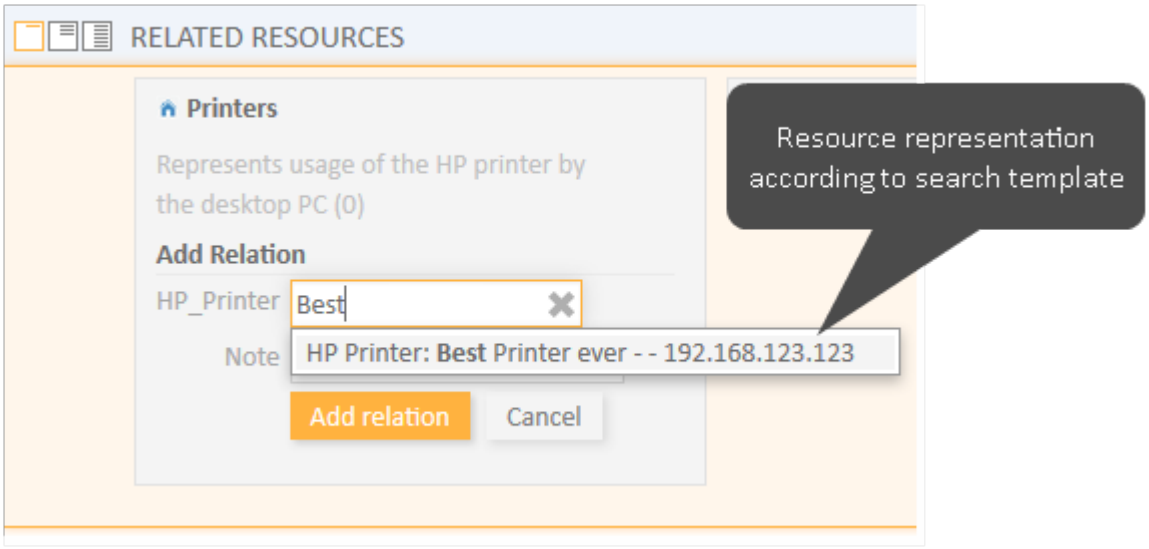


Figure 641: *ConSol CM Web Client - Resource in suggestion list, format based on search template*

Used template: *Search*, if this is defined. If not, *Default*.

```
HP Printer: ${resource.getFieldValue("HP_Printer_Fields_basic","name")!} - -
 ${resource.getFieldValue("HP_Printer_Fields_basic","IP_address")!}
```

History Sections of Resource Page, Customer Page and Ticket

The history template for a resource will be displayed, for example, in the ticket history for the extended level, if *Show all entries* has been selected.

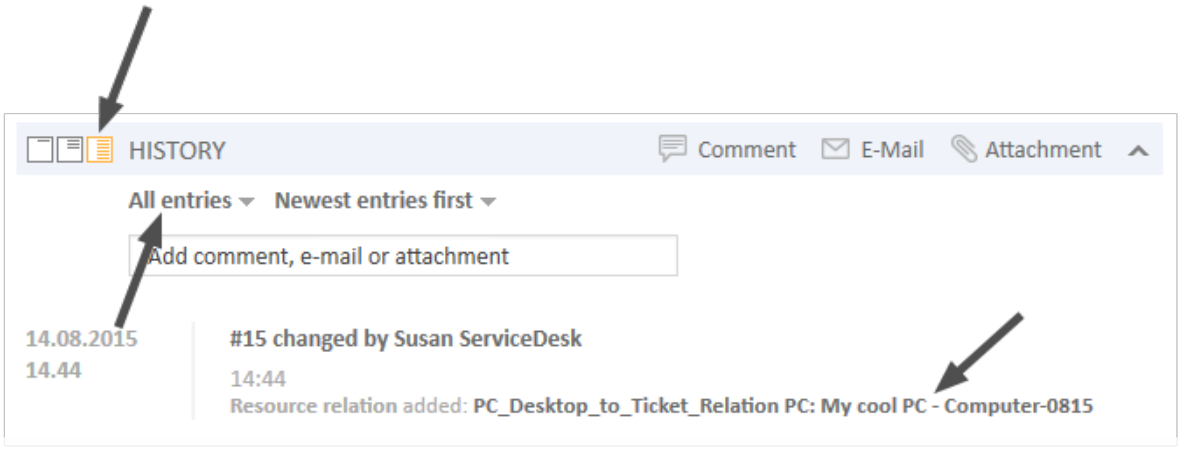


Figure 642: *ConSol CM Web Client - History template for resource data*

Used Template: *History*, if this is defined. If not, *Default*.

REST

The following example shows a REST client request for resource data and the resulting answer of the ConSol CM server via REST API. The value within the `<mark>` tag in the XML output is the resource information which is formatted using the REST template. In the example, the resource name (My new HP Printer) and the inventory number (4712) are part of the template.

```
http://cm6doku-cm1.int.consol.de:8480/restapi/resources
```

Code example 105: *GET request to retrieve the resources in the system*


```
{
    "lastPageNumber": "0",
    "pageNumber": "0",
    "pageSize": "14",
    "totalNumberOfElements": "14",
    "resource":
    [
        {
            "@uri": "http://cm6doku-cm1.int.consol.de:8480/restapi/resources/13",
            "@id": "13"
        },
        {
            "@uri": "http://cm6doku-cm1.int.consol.de:8480/restapi/resources/6",
            "@id": "6"
        },
        {
            "@uri": "http://cm6doku-cm1.int.consol.de:8480/restapi/resources/4",
            "@id": "4"
        },
        {
            "@uri": "http://cm6doku-cm1.int.consol.de:8480/restapi/resources/9",
            "@id": "9"
        },
```


Figure 643: *REST API - List of all resources*

```
http://cm6doku-cm1.int.consol.de:8480/restapi/resources/2.xml
```

Code example 106: *GET request to retrieve resource with ID 2*

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<resource id="2" uri="http://cm6doku-cm1.int.consol.de:8480/restapi/resources/2">
  <mark>HP Printer: My newer HP printer - 4712</mark>
  <resourceType uri="http://cm6doku-cm1.int.consol.de:8480/restapi/resourcetypes/HP_Printer?v=ITyTXos9hx4du4xOqfwzSg%3D%3D"/>
  <accessModeDate>1436523378885</accessModeDate>
  <modificationDate>1490263215984</modificationDate>
  <groups>
    <group name="HP_Printer_MaintenanceTickets">
      <definition uri="http://cm6doku-cm1.int.consol.de:8480/restapi/definitions/groups/HP_Printer_MaintenanceTickets?v=hcfixibBva3Bcb4ibFiPfA%3D%3D"/>
      <fields>
        <field xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="listFieldElement" class="LIST" name="MaintenanceTicketsList" group="HP_Printer_MaintenanceTickets">
          <name>Maintenance tickets</name>
          <uuid>56f9e101-df16-11e6-8068-b93703e2234d</uuid>
          <value/>
        </field>
      </fields>
    </group>
```

Figure 644: *REST API - Details of one selected resource (ID 2)*

```
HP Printer: ${resource.getFieldValue("HP_Printer_Fields_basic","name")!} -
 ${resource.getFieldValue("HP_Printer_Fields_basic","inventory_number")!}
```

Used template: *REST*, if defined. If not, *Default*.

G.1.6 CM/Resource Pool - Resource Relations

G.1.6.1 Introduction

A resource can have relations to other ConSol CM objects. In contrast to customer relations, no hierarchy levels are defined. All relations are simple references. A resource can have zero or more relations to the following types of ConSol CM objects:

- **Ticket**
  E.g., an incident ticket has a ticket-resource relation to the printer which caused a problem.
- **Company**
  E.g., a company has a company-resource relation to a specific SLA.
- **Contact**
  E.g., a contact has a contact-resource-relation to the terminal server access rights.
- **Customer** (i.e., company **or** contact of a customer group)
  E.g., a resource has a resource-customer relation to a customer group to reflect that customers in this customer group are all potential receivers of a newsletter.
- **Resource**
  E.g., a laptop has a resource-resource relation to all connected/configured printers.

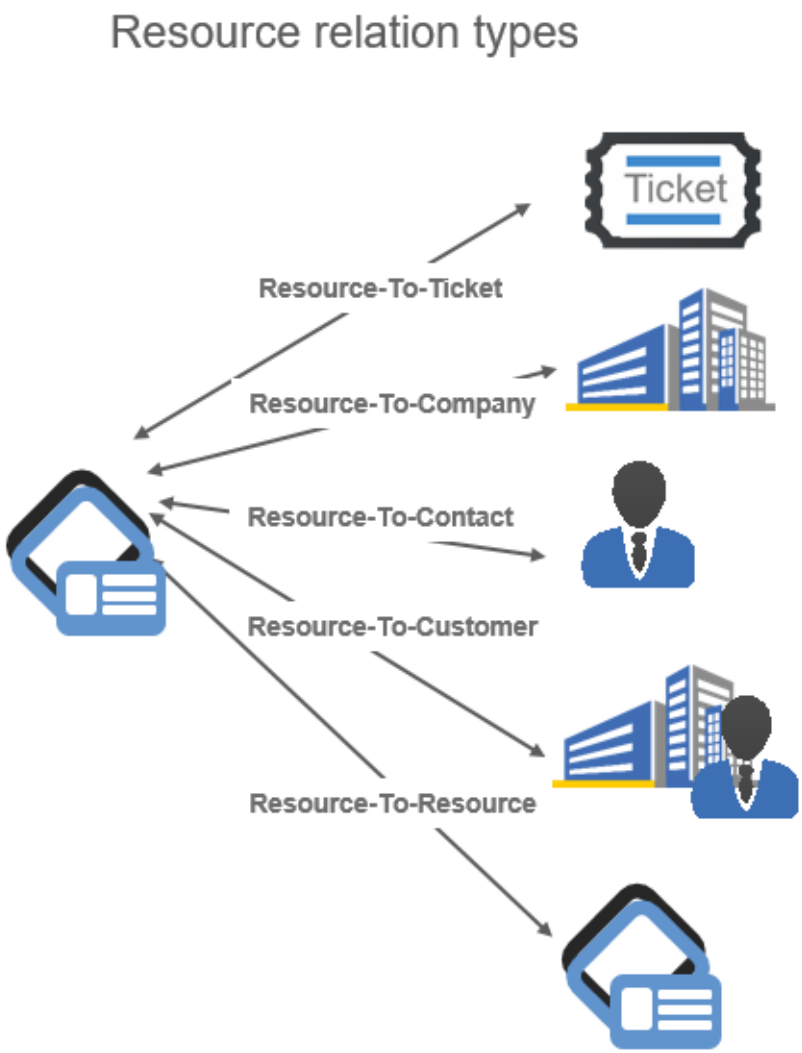


Figure 645: *ConSol CM resource relation types*

For a resource relation, the multiplicity (cardinality) has to be defined, i.e., for each relation, how many objects of the selected type may participate in a relation must be defined.

The following configurations are possible for the resource relation multiplicity (cardinality):

- **One-to-one**
  Each resource can only be related to a single target object.
- **One-to-many**
  Each resource can be related to many different target objects (e.g., one laptop can have resource-resource relation to one, two, or more printers).
- **Many-to-one**
  Many resources can be related to one single target object.
- **Many-to-many**
  Many resources can be related to many different target objects.

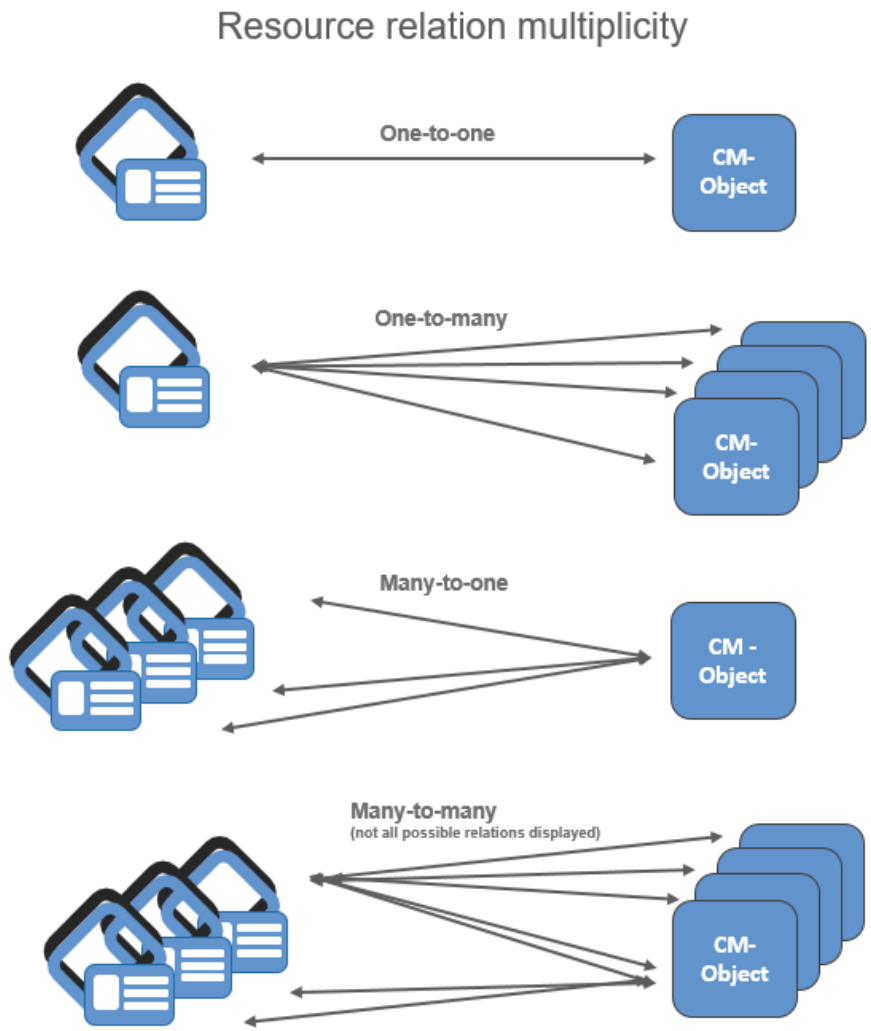


Figure 646: *ConSol CM resource relation multiplicity*

G.1.6.2 Creating Resource Relations in the Data Model Using the Admin Tool

Resource relations are always defined for a resource type, e.g., for the resource type *HP_Printer*.

To create new resource relations or to edit resource relations, open the resource type panel in the Admin Tool (navigation group *Resources*, navigation item *Data Models*).

> Please note that you cannot create or edit resource relations using the navigation item *Relations Overview*. This panel is only used for the display of relations and for defining the order of relations for display in the Web Client!

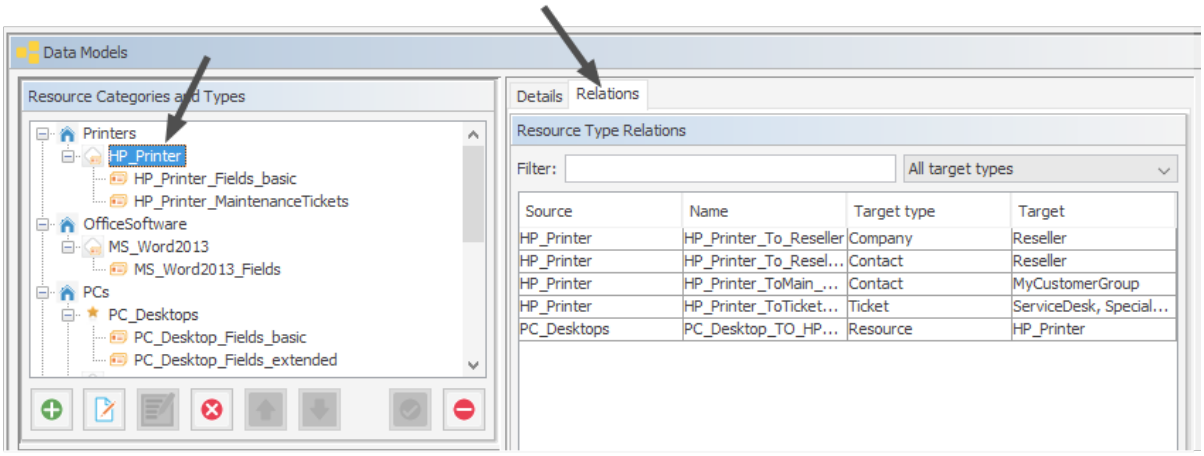


Figure 647: *ConSol CM Admin Tool - Resources, Data Models: Creating and editing resource relations*

To define a new resource relation, click the *Add* button and fill in the required fields in the pop-up window.

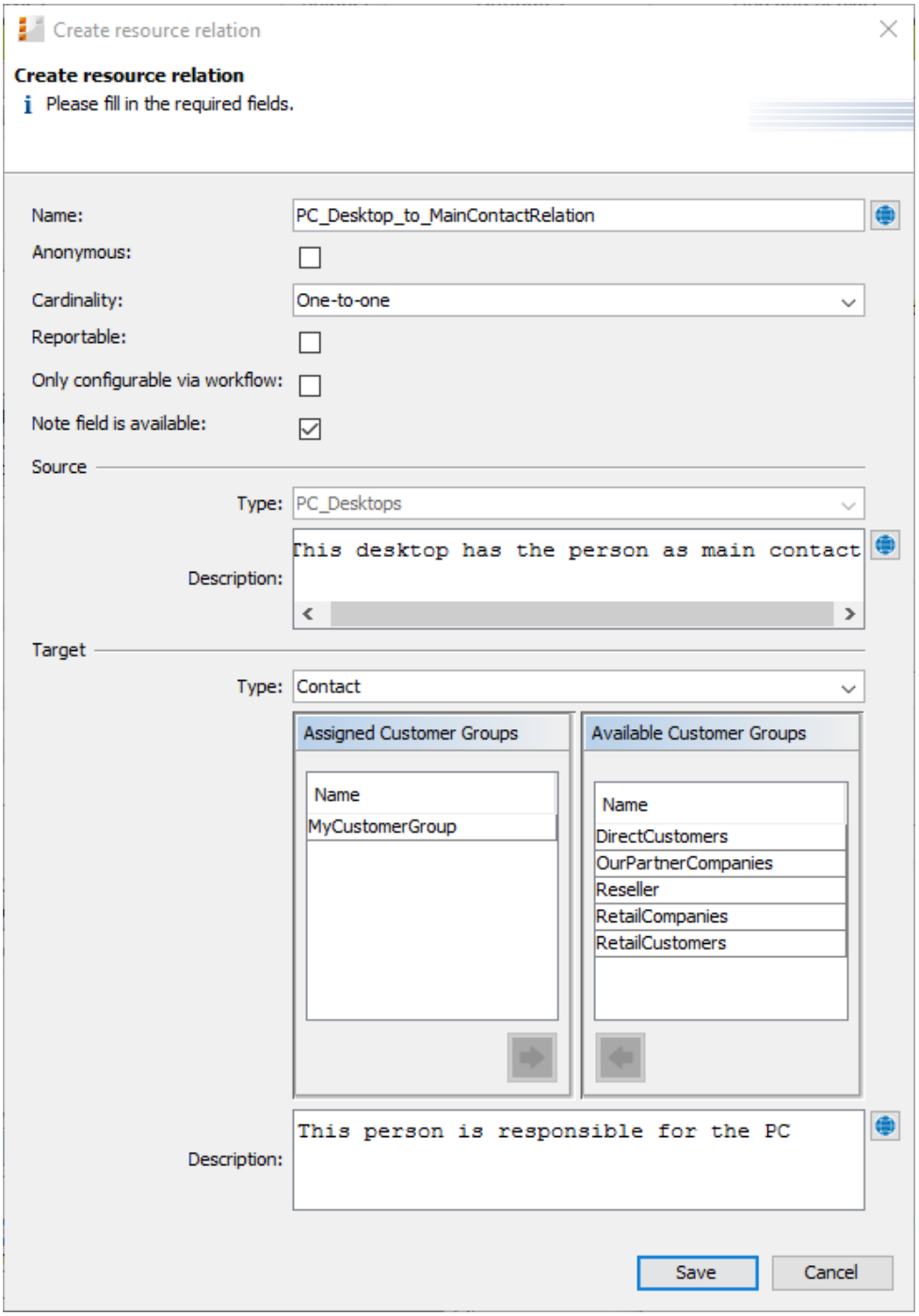


Figure 648: *ConSol CM Admin Tool - Resources, Data Models: Creating a new resource relation*

- **Name**
  The technical name of the resource relation. You can localize it using the *Localize* button. This name will be displayed in the Web Client when a resource relation of this type has been established. For a detailed explanation of the localization mechanism, please refer to section Localization of Objects in General, Type 1.

- **Anonymous**
  If this checkbox is selected, *related* will be displayed as the description for the relation in the Web Client. Use this checkbox if you do not want to have a description displayed. (Since you cannot leave the name field empty, the only other way would be to display the technical name of the relation.)
- **Multiplicity**
  Select the multiplicity (cardinality) for the relation (see the explanation of cardinality above).
- **Reportable**
  If this checkbox is checked, the relations of this type will be transferred to the DWH.
- **Only configurable via workflow**
  If this checkbox is checked, the relation can only be set / removed using workflow scripts. The relation will not be available in the respective menus in the Web Client.
- **Note field is available**
  If this checkbox is checked, a note field will be displayed for each resource when a new resource relation is established (see the following two figures)

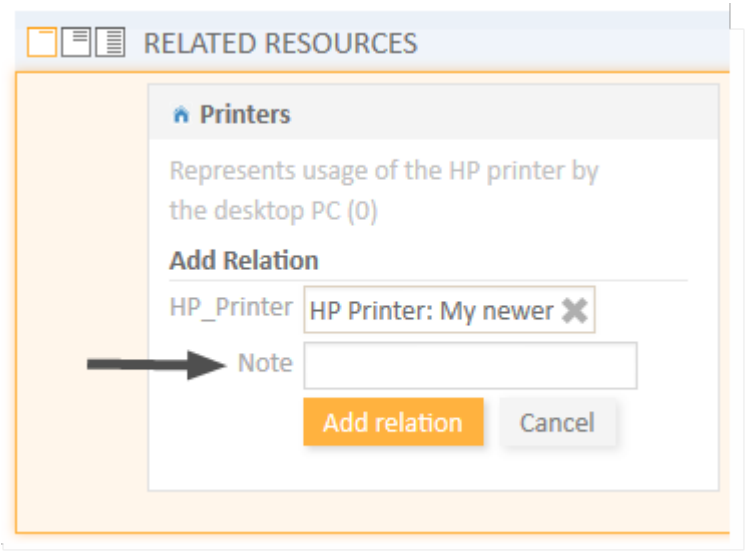


Figure 649: *ConSol CM Web Client - Establishing a resource-resource relation with Note field*

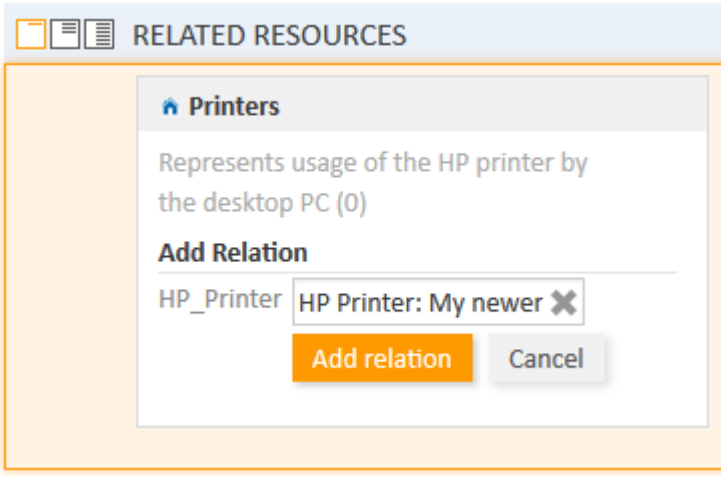


Figure 650: *ConSol CM Web Client - Establishing a resource-resource relation without Note field*

- **Source**
  - **Type**
    The current resource. The value is pre-selected and cannot be modified.
  - **Description**
    The description of the relation. It will be displayed in the Web Client as a header for any relation of this type on the source side. The description can be localized using the *Localize* button.For a detailed explanation of the localization mechanism, please refer to section Localization of Objects in General, Type 1.
- **Target**
  - **Type**
    The object type of the target of this relation. For an explanation, please see the section about relation types.
  - **Assignment table**
    You can assign objects of the selected type to the relation here, i.e., the resource relation will only be available for these objects. Depending on the selected target object type, there will be different objects which can be selected in the assignment table:
    - Target type **Resource**: assigned/available resource types
    - Target type **Customer**: assigned/available customer groups
    - Target type **Contact**: assigned/available customer groups
    - Target type **Company**: assigned/available customer groups
    - Target type **Ticket**: assigned/available queues
  - **Description**
    The description of the relation. It will be displayed in the Web Client as a header for any relation of this type on the target side. The description can be localized using the *Localize* button.

G.1.6.3 Relations Overview

The navigation item *Relations Overview* provides a list of all defined relations and provides the possibility to define the display order of the relations in the Web Client.

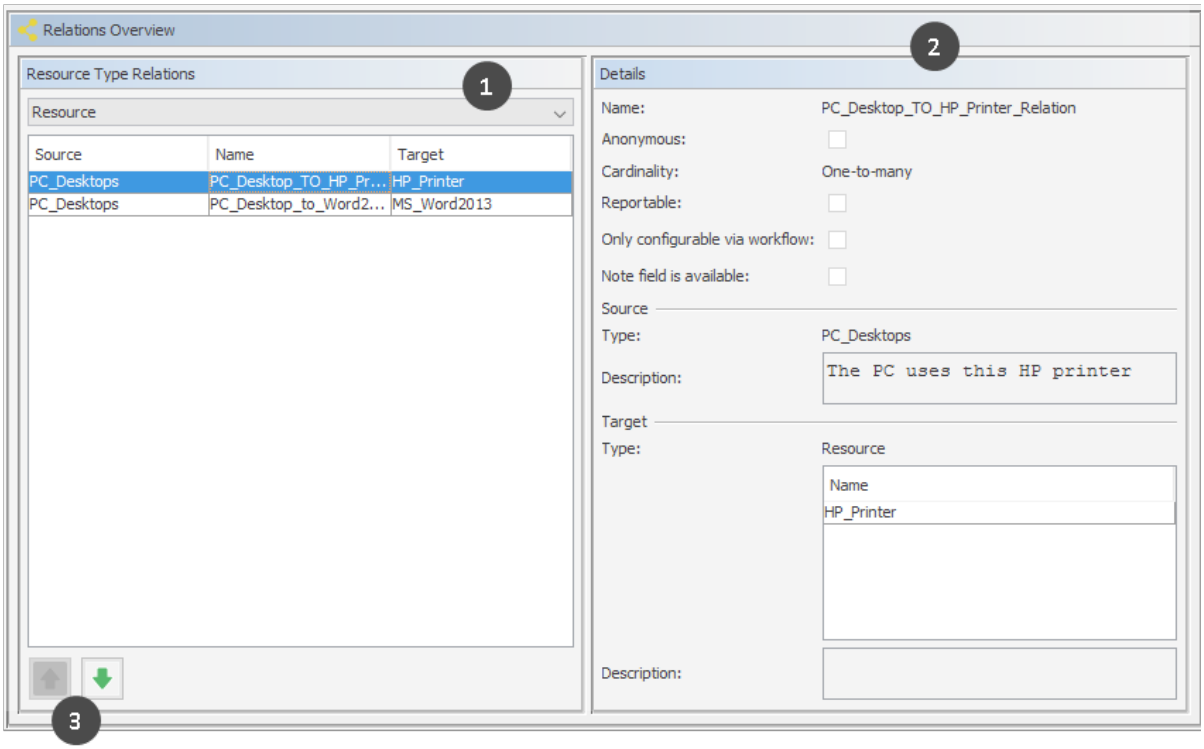


Figure 651: *ConSol CM Admin Tool - Resources, Relations Overview*

Use the filter (drop-down menu) in the top row to select the target object type (1). Only relations with this target object type will be displayed in the list. Please refer to the section about relation types for details about possible target object types.

Please note the filter above the list. You have to select one resource relation type:

- Resource
- Customer
- Contact
- Company
- Ticket

Only relations of this type will be displayed in the list.

Two functions are available in the *Relations Overview*:

- **Display relation details**
  Select a relation. Its details will be displayed on the right-hand side in read-only mode (2).

- **Modify display order**
  Select a relation and use the *Up* and *Down* arrows to move it up/down within the list. This will define the position of the relation in the list when more than one relation is available for selection in the Web Client (3).

G.1.6.4 Creating Resource Relations Using the Web Client

Resource relations can be established on several pages in the Web Client, depending on the source object.

- Ticket-to-X relations can be created on the ticket page.
- Resource-to-X relations can be created on the resource page.
- Contact-to-X relations can be created on the contact page.
- Company-to-X relations can be created on the company page.

For a short explanation of working with resource relations, see section A Short Introduction to CM/Resource Pool Functionality in the Web Client. A detailed explanation of all resource-related Web Client functions is provided in the *ConSol CM User Manual*.

G.1.6.5 Configuration of Resource Relations in the Web Client

Some parameters related to the display mode of resource relations can be configured using Page Customization. These are:

- The default limit for the number of resource relations before the table filter control elements are displayed.
- The sorting strategy for resource relations.

For details about parameters affecting customer (unit) pages, please see section UnitResourceRelationof the Page Customization chapter.

For details about parameters affecting the resource page, please see section TicketRelation of the Page Customization chapter.

For details about parameters affecting the ticket page, see section resourceRelations of the Page Customization chapter.

G.1.7 CM/Resource Pool - Resource Actions

G.1.7.1 Introduction

Resource actions are a component of the ConSol CM Action Framework. Resource actions are actions which can be performed for a resource, i.e., an object which is stored in the Resource Pool. The actions can be performed automatically by the system or manually, triggered by an engineer who has the required permissions. You might want to apply resource actions for use cases like the following:

- Create a maintenance ticket for a printer.
- Find all companies which use a certain SLA.

You can use the following types of resource actions:

- **Automatic** actions which are performed by the system after one of the following resource operations:
  - CREATE
  - UPDATE
  - DELETE
  - RELATION
  - SEARCH
- **Manual** actions which are performed by the engineer using *Activities* links on a resource page in the Web Client (similar to *Workflow activities* for tickets). Manual actions are executed for the resource which is displayed.

⚠ Please keep in mind that only engineers who have at least one role with the following access permissions for the respective resource type are allowed to use the resource actions, i.e., only then will the Activities link be displayed in the Web Client:

- Act

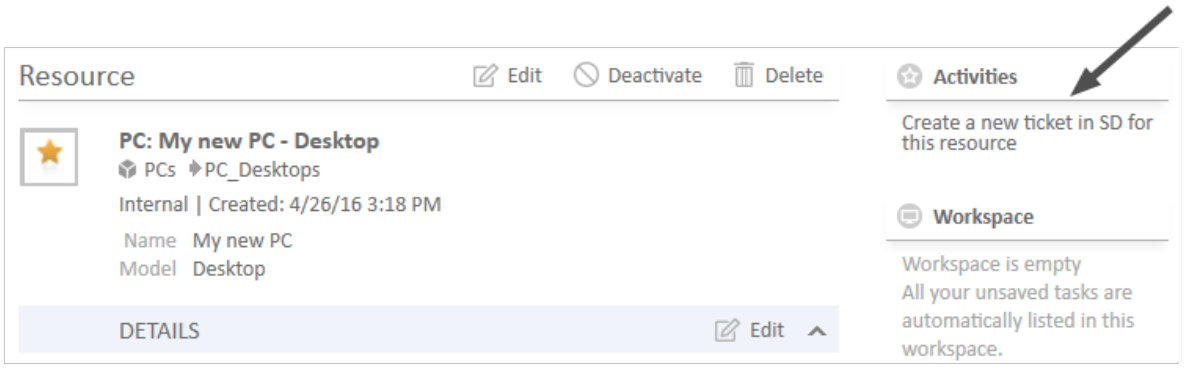


Figure 652: *ConSol CM Web Client - Resource action (on resource page)*

Resource actions are defined as Groovy scripts which are stored in the *Script and Template* section of the Admin Tool.

The execution of resource actions can be controlled using condition scripts, i.e., you can implement a condition script which is executed before the action script of the resource action. The action script is only executed if the condition script returns “true”.

So there are two types of scripts you have to deal with when you use ConSol CM resource actions:

- **Resource action script**
  Defines the action which should be performed.
- **Resource condition script**
  Defines one or more conditions for the execution of the action script. Has to return “true” or “false”. If “false” is returned the action script is not executed. If this is a manual action, it is not displayed as *Activity* in the Web Client.

When you want to implement a resource action you have to proceed in three steps:

1. Create a script for the resource action (either an action script only or an action script and a condition script).
2. Create the resource action(s) which use(s) the script(s).
3. Assign the resource action(s) to the resource type(s) where they should be available.

In the following sections, all three steps are explained in detail.

G.1.7.2 Creating Resource Actions Using the Admin Tool

Step 1: Write the Resource Action Script

Create a new Admin Tool script of type *Resource action*. If required, create another script of type *Resource condition*.

For a detailed explanation of Admin Tool scripts in general, please refer to section Admin Tool Scripts.

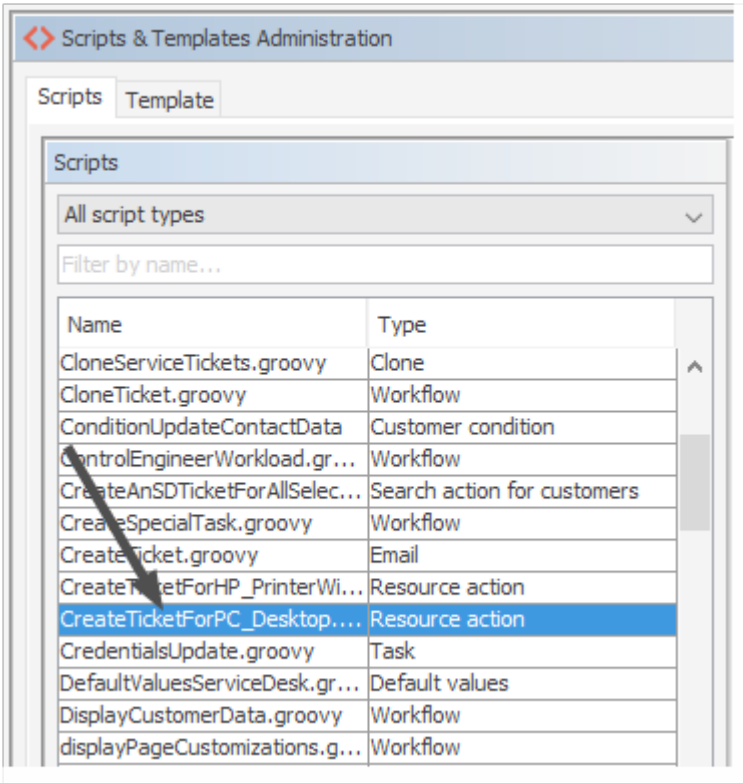


Figure 653: *ConSol CM Admin Tool - System, Scripts and Templates: Resource action script*

```
// this script creates a new ticket for the resource from which the activity is
 executed, i.e., creates new ticket and links it to resource
// resource - ticket relation must be configured beforehand!

import com.consol.cmas.common.model.ticket.Ticket
import com.consol.cmas.common.model.customfield.Unit
import com.consol.cmas.common.model.resource.*
import com.consol.cmas.common.service.resource.*
import com.consol.cmas.common.model.ticket.Queue
import com.consol.cmas.common.model.resource.meta.*

println 'CreateTicketForResource.groovy started ...'

Ticket newtic = new Ticket()

Queue qu = queueService.getByName("ServiceDesk")
newtic.setQueue(qu)

newtic.setSubject("New Ticket for Resource: " + resource.getId())
newtic.set("helpdesk_standard.priority","low")

// use main contact person of the resource as main contact for the ticket
Unit maincont = new Unit()
```

```
def crit = new ResourceRelationWithTargetUnitCriteria()
crit.setResource(resource)
List<ResourceRelationWithTargetUnit> cont_list =
 resourceRelationService.getByCriteria(crit)
if (cont_list.size() == 0) {
   //cmweb.rp.resource.action.no_contact_set has to be configured as label in the
    AT
   return actionScriptResultFactory.getPostAction(PostActionType.FAILURE,
    "cmweb.rp.resource.action.no_contact_set")
} else {
   def cont_rel = cont_list[0]
   maincont = cont_rel.getTargetUnit()
}

ticketService.createWithUnit(newtic,maincont)

println 'New Ticket created for resource with ID' + resource.getId()

// link ticket to resource

def resRelationDefCriteria = new ResourceRelationDefinitionCriteria()
resRelationDefCriteria.addDefinitionName("PC_Desktop_to_Ticket_Relation")
def s_res_type = resource.getResourceType()
resRelationDefCriteria.addSourceResourceType(s_res_type)
resRelationDefCriteria.addTargetQueue(qu)

//log.info "resRelationDefCriteria = " + resRelationDefCriteria
//log.info "resRelationDefCriteria.definitionName = " +
 resRelationDefCriteria.getDefinitionsNames()

def resRelationDef = resourceRelationDefinitionService.getByCriteriaUniqueResult
 (resRelationDefCriteria)
def resRelation = new ResourceTicketRelation(resRelationDef, resource, newtic)
// log.info "resRelation" + resRelation
resourceRelationService.create(resRelation)
```

Code example 107: *Resource action script*

Step 2: Create Resource Action(s) Which Use(s) the Script

To create, edit, or delete resource actions, open the navigation item *Actions* in navigation group *Resources* in the Admin Tool.

To create or add a new action click the *Add* button and fill-in the required data in the pop-up window (the pop-up window is the same for adding and for editing a resource action).

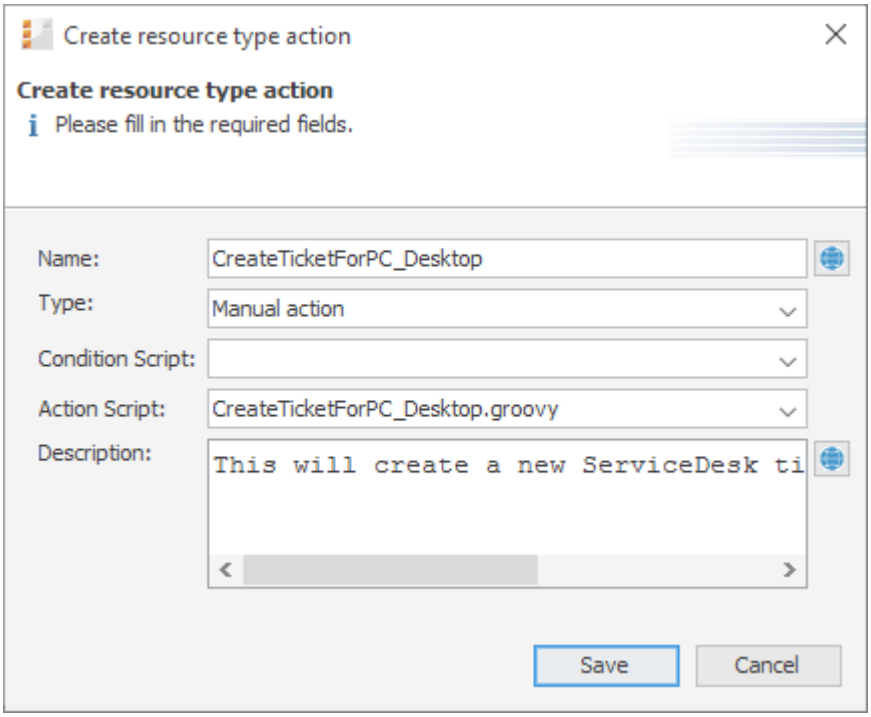


Figure 654: *ConSol CM Admin Tool - Resources, Actions: Creating a resource action*

- **Name**
  The technical name of the resource action. You can localize the value by using the *Localize* button. The localized name will be displayed in the Web Client. For a detailed explanation of the localization mechanism, please refer to section Localization of Objects in General, Type 1.
- **Type**
  The resource action type. Select one of the following types. Once a type has been defined, it cannot be modified afterwards (when you edit an existing resource action).
  - **Create**
    This action will be executed automatically when the resource is created.
  - **Update**
    This action will be executed automatically when the resource is updated, i.e., when the data has been modified (either manually or automatically) and is saved again.
  - **Delete**
    This action will be executed automatically when the resource is deleted.
  - **Relation**
    This script will be executed automatically when a relation to or from a resource of this type is
    - created
    - deleted

    (The script will not be executed when the comment of a relation is changed.)

    - **Manual**
      The resource action is displayed as *Activity* in the Web Client and can be executed only manually. If a Resource Condition Script is implemented, the activity will only be displayed in the Web Client if the condition script has returned “true”.
    - **Search**
      The resource action is a search action. Actions of this type are explained in section Action Framework - Search Actions.
- **Condition Script**
  In case a condition script should be executed before the action script, the name of the condition script must be entered here. Use the drop-down menu to select from a list with all scripts of type *Resource condition* which are stored in the Scripts section of the Admin Tool. The action script will only be executed if the condition script returns “true”. If there is no condition, just leave this field empty.
- **Action Script**
  The name of the action script which should be executed. Use the drop-down menu to select from a list with all scripts of type *Resource action* which are stored in the *Scripts* section of the Admin Tool.
- **Description**
  Enter the description which should be displayed as mouse-over in the Web Client (for manual actions only).

Save the action. Then you can assign it to resource types. Please see following step.

Step 3: Assign Resource Actions to Resource Types

To assign pre-defined Resource Execution and/or resource condition scripts to resource types, the respective manual and/or automatic actions have to be assigned to a resource type. Open the navigation item *Data Models* in navigation group *Resources* in the Admin Tool. Select the resource type you would like to edit and click the *Edit* button to open the pop-up window where you can assign the resource actions. An action might contain only a resource action script, or a resource condition script and a resource action script.

In the following example (next figure), a manual resource action is assigned to the resource type *PC_Desktop*.

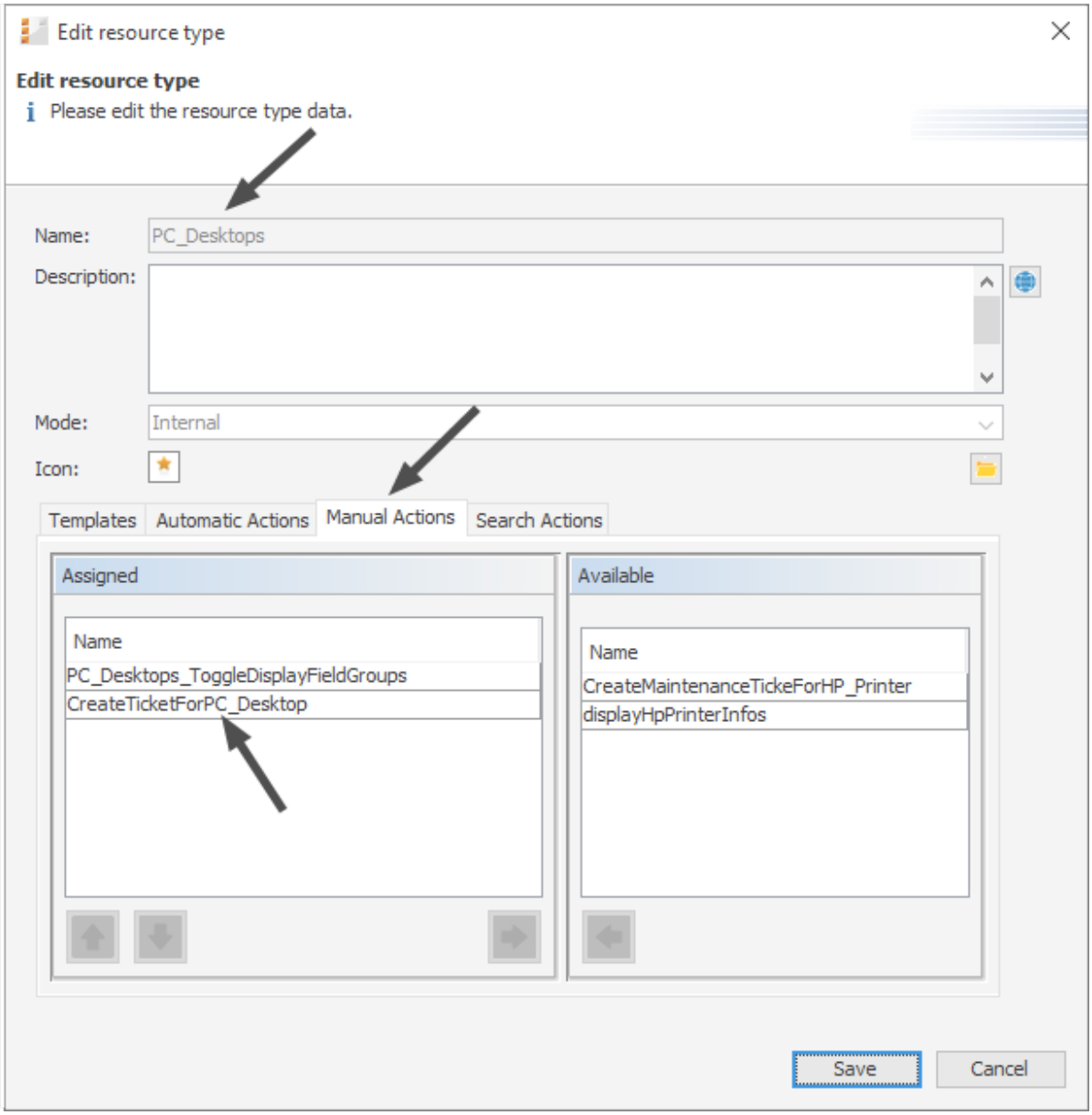


Figure 655: *ConSol CM Admin Tool - Resources, Data Models: Assigning manual resource actions to a resource type*

You can assign resource actions of the following action types:

- **Automatic Actions**
  Those actions will be executed automatically when the respective type of event (Create, Update, Delete, Relation) has been triggered. Select an action for each type which is required. Only the actions with the correct type (which have been defined as resource actions, see step 2) are listed in each drop-down menu. For example, for an automatic action of type *Create*, only resource actions of type *Create* (which have been defined in navigation item *Actions*) will be available.

- **Manual Actions**
  Those actions are displayed as activities on the resource page in the Web Client and have to be executed manually. An activity is only displayed if either no Resource Condition Script is present or if the respective resource condition script has returned “true”.
- **Search Actions**
  See section Action Framework - Search Actions.

G.1.7.3 Using Resource Actions in the Web Client (as an Engineer)

As an engineer (user), two resource action types are relevant for you because they are available as activities in the Web Client:

- **Manual**
  Manual actions are offered in the Web Client similar to workflow activities for a ticket. Please see *Example 1* in the next section.
- **Search**
  See section Action Framework - Search Actions.

The *CREATE*, *UPDATE*, *RELATION*, and *DELETE* actions run in the background.

G.1.7.4 Programming Resource Execution and Resource Condition Scripts

Please read the section about Scripts for the Action Framework for a general introduction about important principles, classes, and methods for execution and condition scripts.

G.1.7.5 Examples for Resource Actions

Example 1: Simple Manual Action

Use case: The engineer should be able to create a new Service Desk ticket directly from a resource page of a PC. The new ticket should be related to the resource (PC). The main contact of the new Service Desk ticket should be the person who is responsible for the PC. This is implemented as *resource-contact* relation in the resource type *PC_Desktops*. To implement the resource action, perform the following steps.

Write the resource action script:

```
// this script creates a new ticket for the resource from which the activity is
 executed, i.e., creates new ticket and links it to resource
// resource - ticket relation must be configured beforehand!

import com.consol.cmas.common.model.ticket.Ticket
import com.consol.cmas.common.model.customfield.Unit
import com.consol.cmas.common.model.resource.*
import com.consol.cmas.common.service.resource.*
import com.consol.cmas.common.model.ticket.Queue
import com.consol.cmas.common.model.resource.meta.*
import com.consol.cmas.core.server.service.action.*

println 'CreateTicketForResource.groovy started ...'
```

```
Ticket newtic = new Ticket()

Queue qu = queueService.getByName("ServiceDesk")
newtic.setQueue(qu)


def subj = resource.get("PC_Desktop_Fields_basic.name")

// newtic.setSubject("New Ticket for Resource: " + resource.getId())
newtic.setSubject("New Ticket for Resource: " + subj)

newtic.set("helpdesk_standard.priority","low")

// use main contact person of the resource as main contact for the ticket
Unit maincont = new Unit()
def crit = new ResourceRelationWithTargetUnitCriteria()
crit.setResource(resource)
List<ResourceRelationWithTargetUnit> cont_list =
 resourceRelationService.getByCriteria(crit)
if (cont_list.size() == 0) {
   workflowApi.addValidationError("ERRROR","No contact set!")
} else {
   def cont_rel = cont_list[0]
   maincont = cont_rel.getTargetUnit()
}

ticketService.createWithUnit(newtic,maincont)

println 'New Ticket created for resource with ID' + resource.getId()

// link ticket to resource

def resRelationDefCriteria = new ResourceRelationDefinitionCriteria()
resRelationDefCriteria.addDefinitionName("PC_Desktop_to_Ticket_Relation")
def s_res_type = resource.getResourceType()
resRelationDefCriteria.addSourceResourceType(s_res_type)
resRelationDefCriteria.addTargetQueue(qu)

log.info "resRelationDefCriteria = " + resRelationDefCriteria
log.info "resRelationDefCriteria.definitionName = " +
 resRelationDefCriteria.getDefinitionsNames()

def resRelationDef = resourceRelationDefinitionService.getByCriteriaUniqueResult
 (resRelationDefCriteria)
def resRelation = new ResourceTicketRelation(resRelationDef, resource, newtic)
log.info "resRelation" + resRelation
resourceRelationService.create(resRelation)

// go to new ticket
return actionScriptResultFactory.getPostAction(PostActionType.GOTO_TICKET, newtic)
```

Code example 108: *Resource action script for PC_Desktops to create a new Service Desk ticket for responsible PC contact*

Create a resource action based on the script:

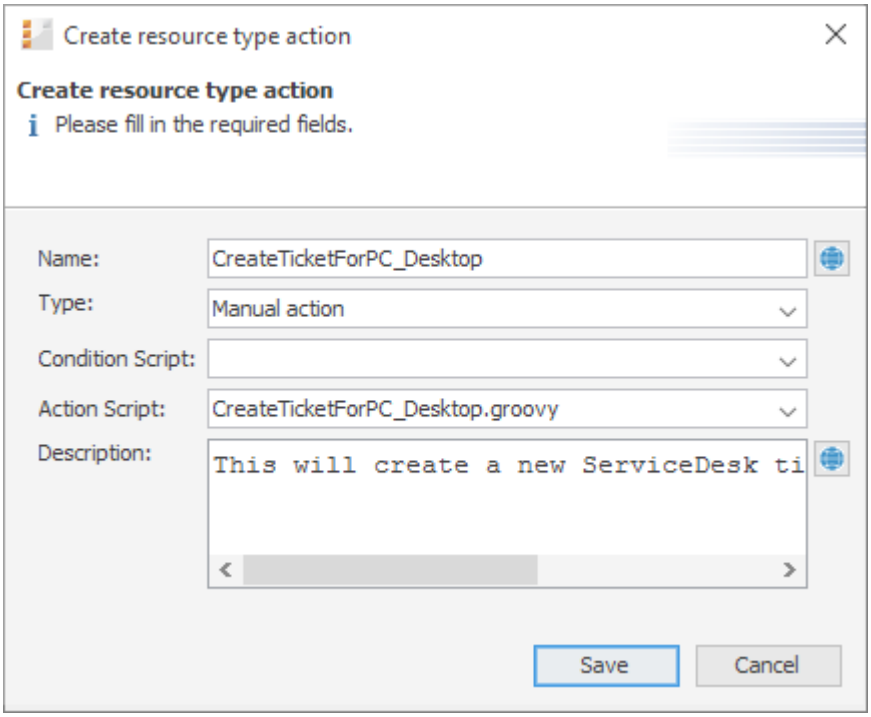


Figure 656: *ConSol CM Admin Tool - Resources, Actions: Creating a new resource action*

Assign the action to the correct resource type:

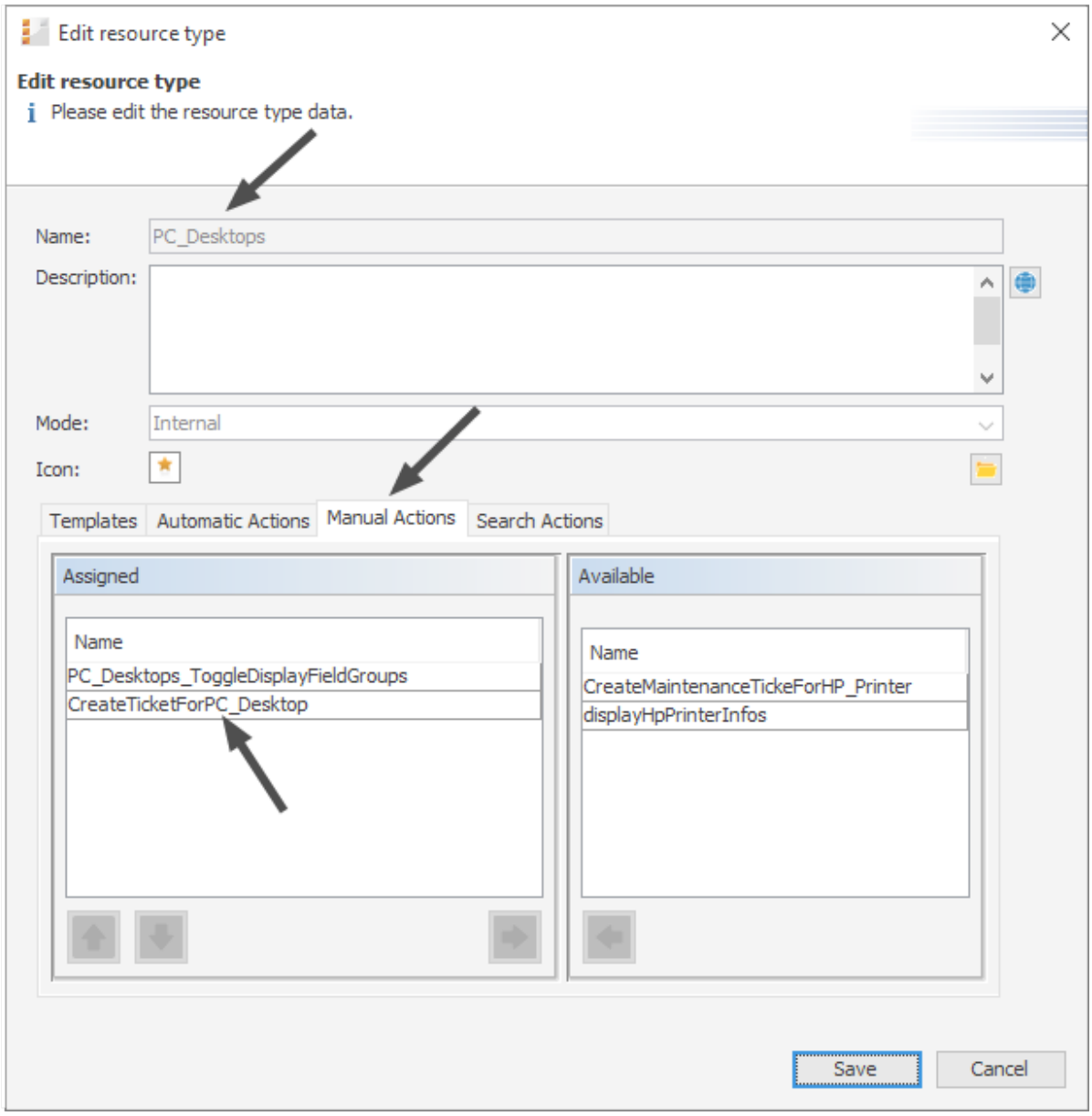


Figure 657: *ConSol CM Admin Tool - Resources, Data Models: Assign the resource action to the correct resource type*

Check the functionality using the Web Client:

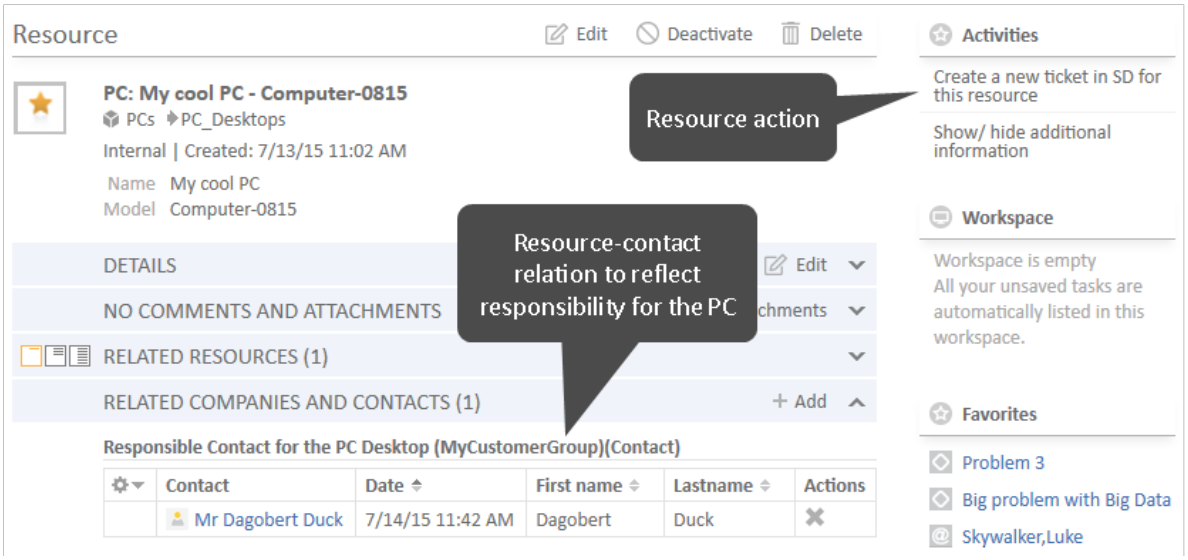


Figure 658: *ConSol CM Web Client - Resource page with the resource action*

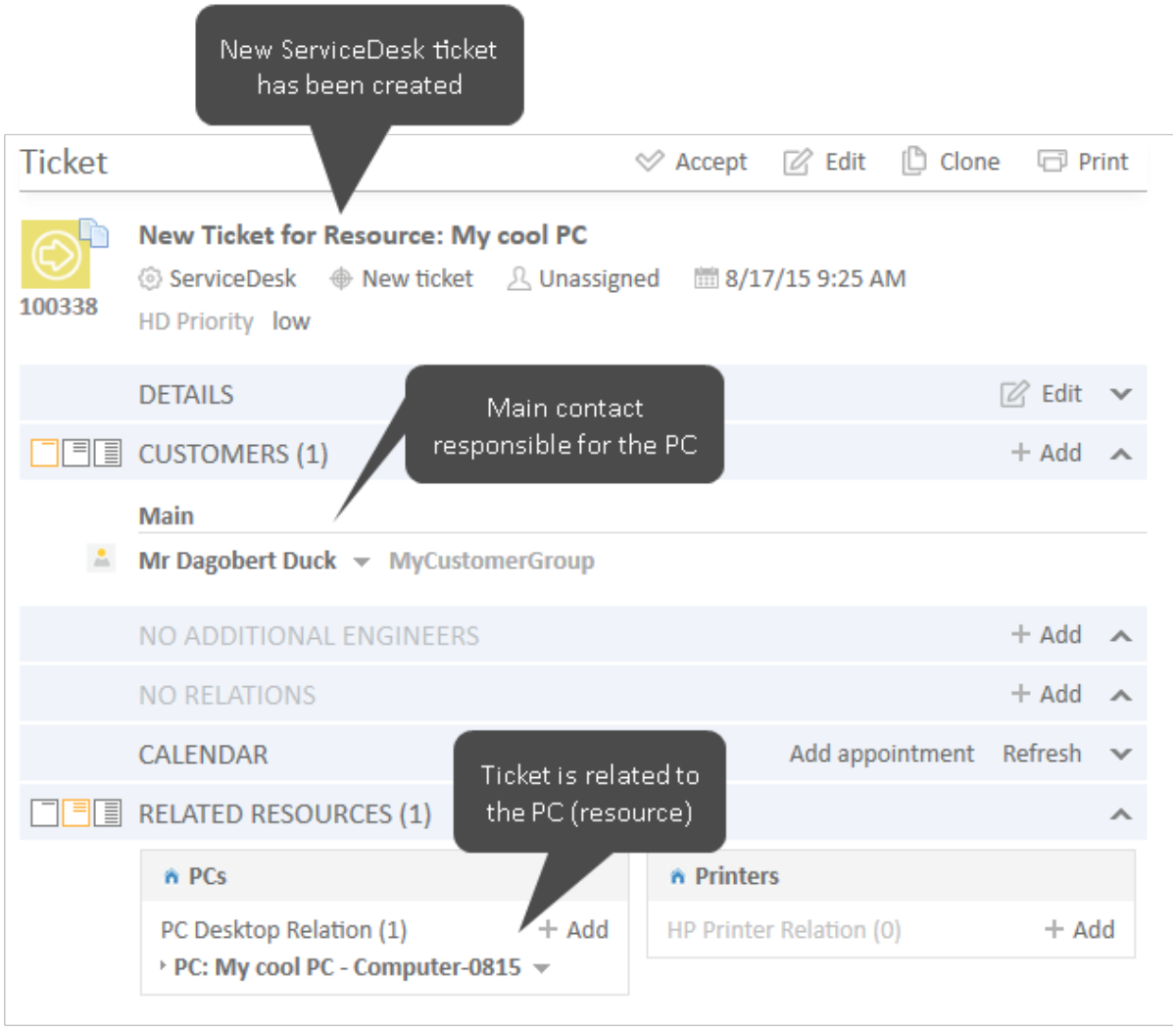


Figure 659: *ConSol CM Web Client - New Service Desk ticket created by the resource action*

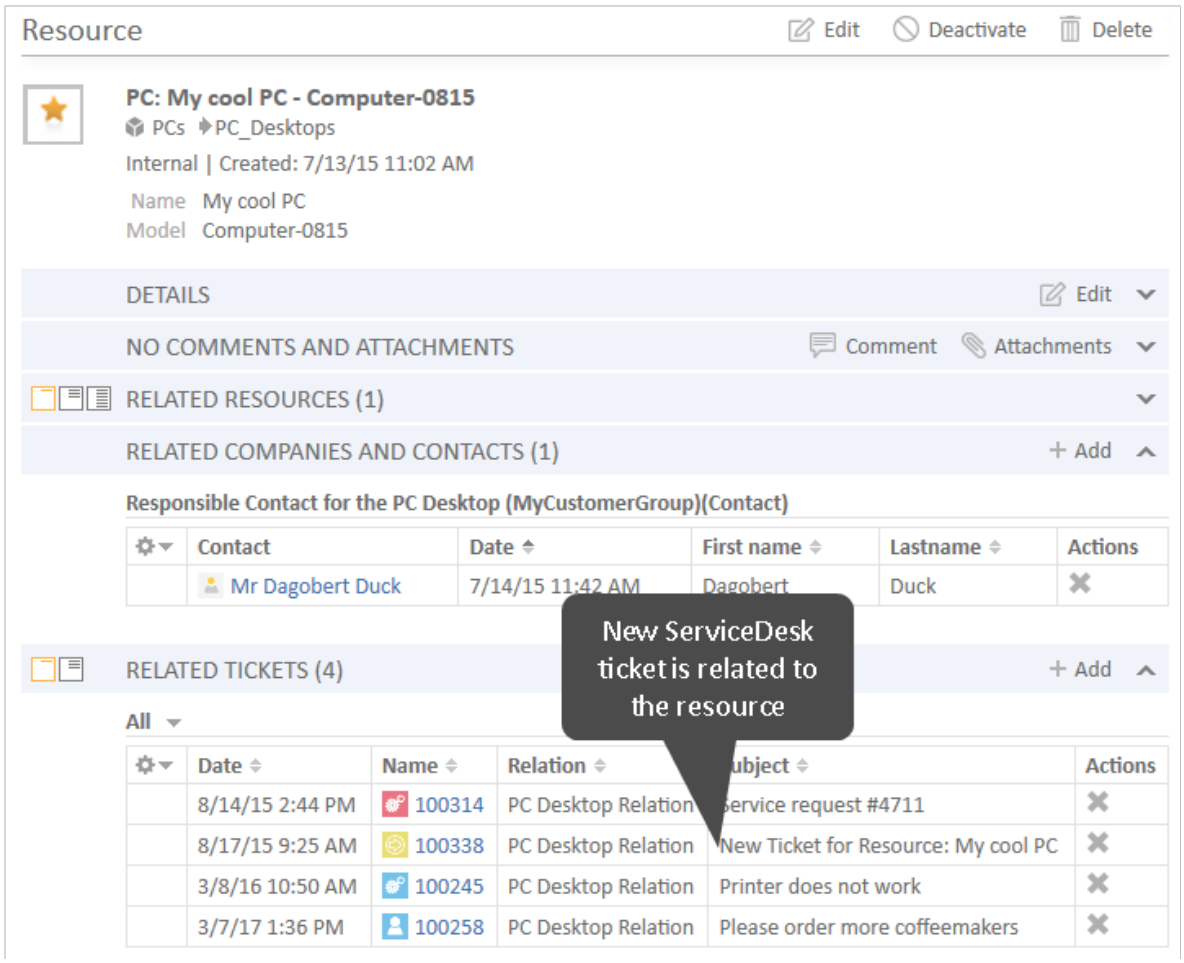


Figure 660: *ConSol CM Web Client - Resource page with one or more ticket relations (new Service Desk ticket)*

Example 2: Open a Maintenance Ticket from a Resource Using an ACF

The Action Framework offers the possibility to open an ACF (Activity Control Form) when a ticket is created. The ACF is used to gather data for the following workflow activity, i.e., the ticket can be created and moved through the first workflow step very easily. The following example demonstrates this functionality. A maintenance ticket should be created for a resource (an HP printer). During the (sub-) process, an ACF should be offered to ask for data which will then be used during the next workflow activity.

To implement the resource action, perform the following steps.

Write the resource action script:

```
// this script creates a new ticket for the resource from which the activity is
 executed, i.e., creates new ticket and links it to resource
// resource - ticket relation must be configured beforehand!
// CreateTicketForHP_PrinterWithACF.groovy
```

```
import com.consol.cmas.common.model.ticket.Ticket
import com.consol.cmas.common.model.customfield.Unit
import com.consol.cmas.common.model.resource.*
import com.consol.cmas.common.service.resource.*
import com.consol.cmas.common.model.ticket.Queue
import com.consol.cmas.common.model.resource.meta.*
import com.consol.cmas.core.server.service.action.*
import com.consol.cmas.common.service.*

println 'CreateTicketForHP_PrinterWithACF.groovy started ...'

Ticket newtic = new Ticket()

Queue qu = queueService.getByName("SpecialTasks")
newtic.setQueue(qu)

newtic.setSubject("New Ticket for HP Printer: " + resource.getId())
newtic.set("SpecialTasks_Fields.SpecialTasksPrio","normal")

// use main contact person of the resource as main contact for the ticket
Unit maincont = new Unit()
def crit = new ResourceRelationWithTargetUnitCriteria()
crit.setResource(resource)
List<ResourceRelationWithTargetUnit> cont_list =
 resourceRelationService.getByCriteria(crit)
if (cont_list.size() == 0) {
   log.info("ERRROR in script CreateTicketForHP_PrinterWithACF -- No contact set!")
} else {
   def cont_rel = cont_list[0]
   maincont = cont_rel.getTargetUnit()
}

ticketService.createWithUnit(newtic,maincont)

println 'New Ticket created for resource with ID' + resource.getId()

// link ticket to resource

def resRelationDefCriteria = new ResourceRelationDefinitionCriteria()
resRelationDefCriteria.addDefinitionName("HP_Printer_ToTicket_Relation")
def s_res_type = resource.getResourceType()
resRelationDefCriteria.addSourceResourceType(s_res_type)
resRelationDefCriteria.addTargetQueue(qu)

log.info "resRelationDefCriteria = " + resRelationDefCriteria
log.info "resRelationDefCriteria.definitionName = " +
 resRelationDefCriteria.getDefinitionsNames()

def resRelationDef = resourceRelationDefinitionService.getByCriteriaUniqueResult
 (resRelationDefCriteria)
def resRelation = new ResourceTicketRelation(resRelationDef, resource, newtic)
log.info "resRelation" + resRelation
```

```
resourceRelationService.create(resRelation)

// go to new ticket, but fill ACF before

def executionContext = activityFormDefinitionService.getExecutionContext(newtic,
 "defaultScope/TaskInProgress/Aufgabe_annehmen")

if (!executionContext) {
   return actionScriptResultFactory.getPostAction(PostActionType.FAILURE,
    "action.fail.wrong.activity")
}

// Modify entities from the execution context - not the original ones
// - since the user may still press cancel.
executionContext.ticket.add("SpecialTasks_Fields","Deadline", new Date());

return actionScriptResultFactory.getPostAction(PostActionType.GOTO_TICKET, newtic,
 executionContext);
```

Code example 109: *Resource action script which opens a ticket and uses an ACF*

Create a resource action based on the script.

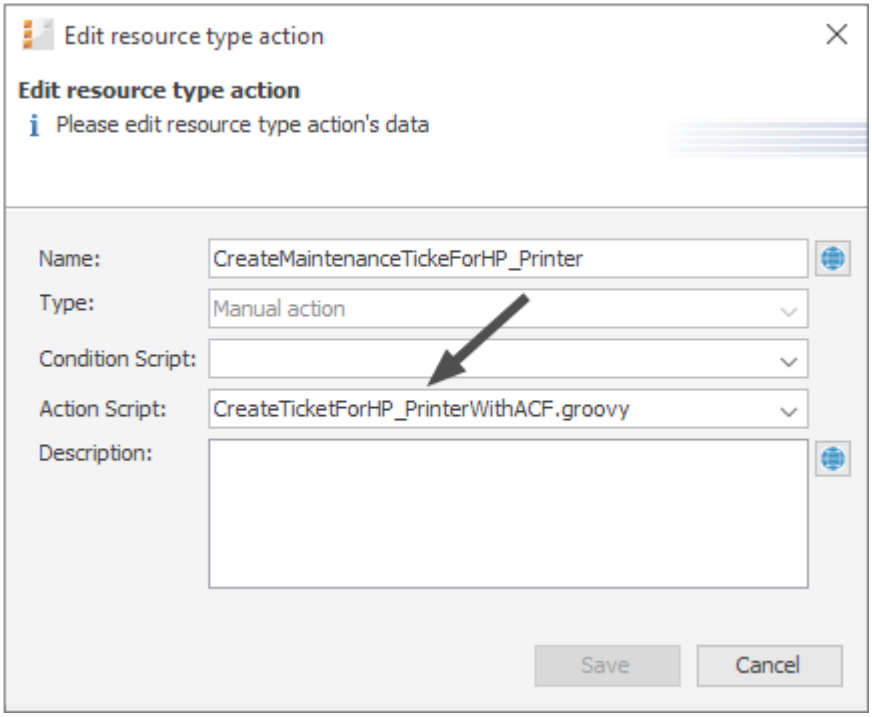


Figure 661: *ConSol CM Admin Tool - Resources, Actions: Create the resource action for the HP Printer maintenance ticket*

Assign the action to the correct resource type:

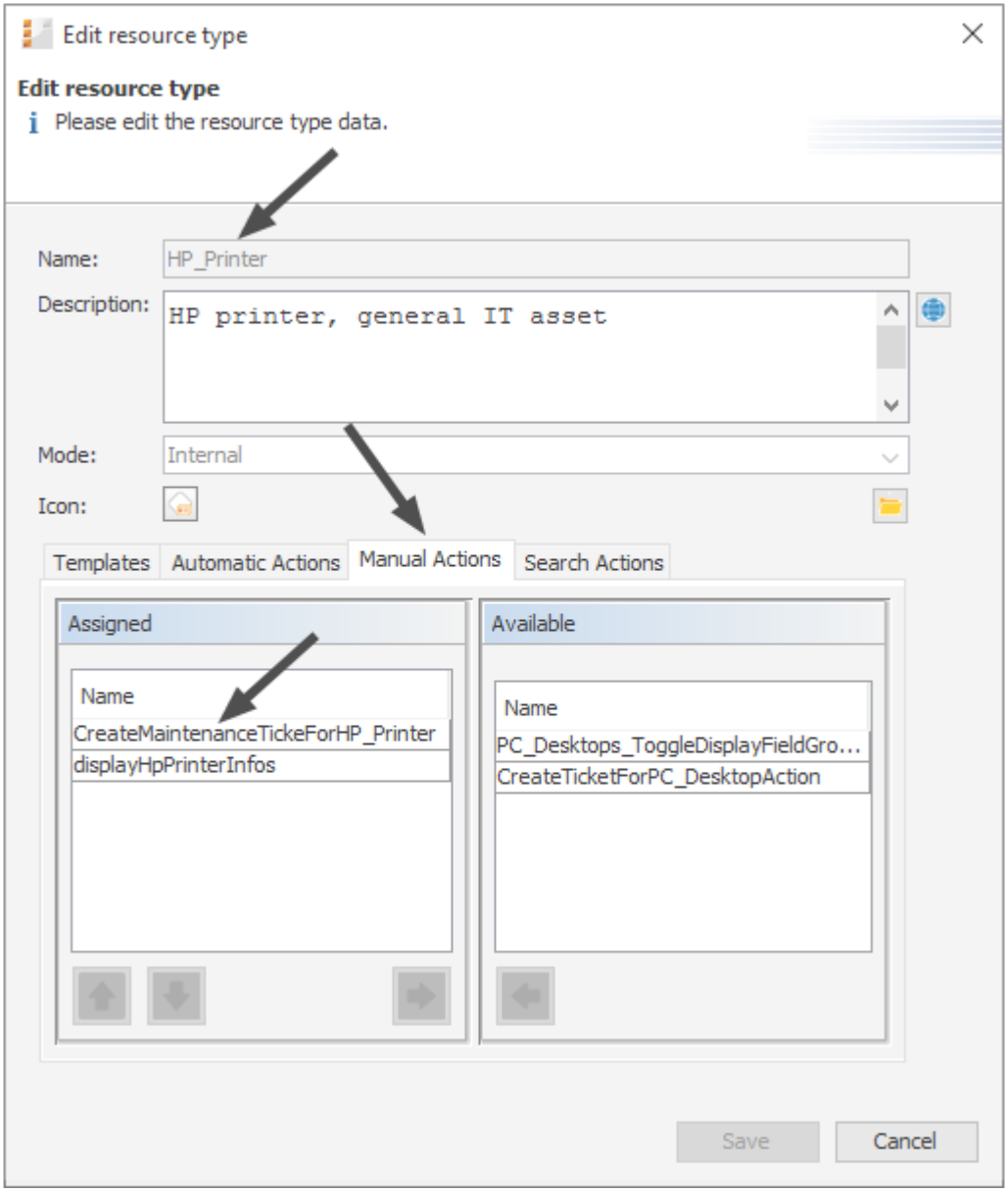


Figure 662: *ConSol CM Admin Tool - Resources, Data Models: Assigning the resource action to the correct resource type (HP Printer)*

Check the functionality using the Web Client:

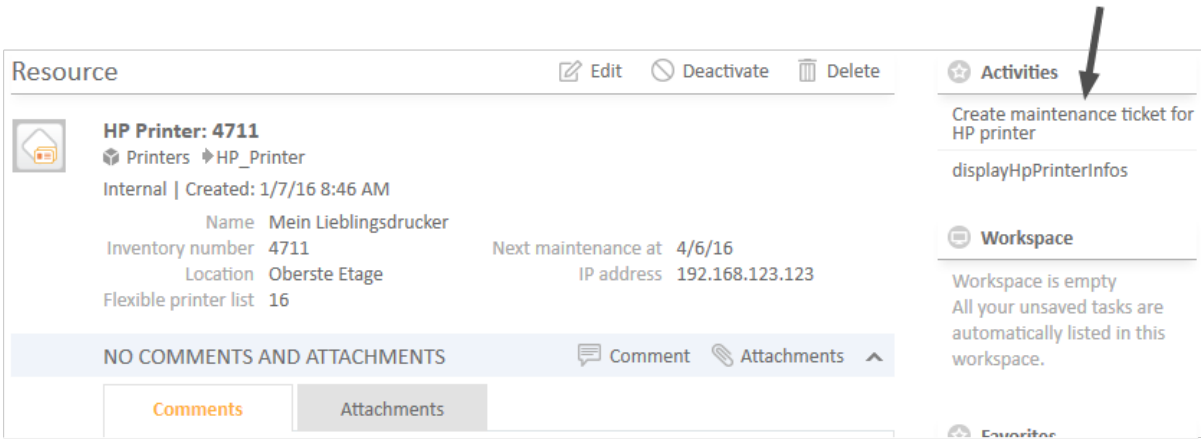


Figure 663: *ConSol CM Web Client - Resource action for HP Printer*

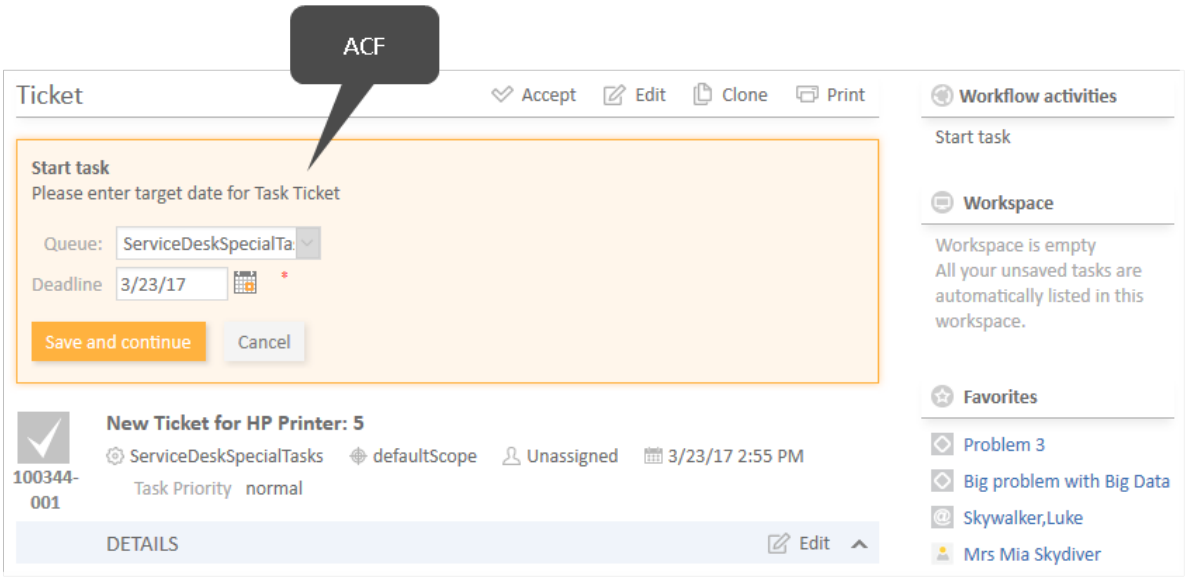


Figure 664: *ConSol CM Web Client - New maintenance ticket for resource (HP Printer), ACF*

Working with the Changes Object in Resource Update Actions

Starting with CM version 6.10.5.4, it is possible to monitor the changes which have been performed during a resource update action. (The same applies to unit Update actions, explained in section Working with the Changes Object in Customer Update Actions).

To find out which changes have been performed use the object of class `ResourceChanges` in resource actions.

Please remember, the Update script will be executed:

- in an explicit *Update* action
- when comments or attachments are added
- when comments or attachments are removed

There are two methods of the `ResourceChanges` object which provide information about the changed data:

- `getCustomFieldChangeInfo()`
  provides information about changes of resource data (in resource fields)
- `getContentChangeInfo()`
  provides information about changes in the resource history (comments, attachments)

Since the method return parameters contain rather complex components, we recommend to read the API doc of the `ResourceChanges` class. The following code provides an example for a script where a `resourceChanges` object is used.

```
/**
* Available script variables:
* Manual action:
* resource - Resource for which action is executed
*
* Create, Delete Action:
* resource - Resource for which action is executed
*
* Update Action:
* resource - Resource for which action is executed
* changes - ResourceChanges object containing information about changes done for
 resource entity
*
* Relation action:
* resource - Resource for which action is executed (resource-resource)
* relation - Relation object
* resourceExternalId - External resource id (resource, unit, ticket - ext resource)
*/

// Update Action Script displayPC_DesktpChangesInLog.groovy for resources in PC_
 Desktops

import com.consol.cmas.common.model.content.unit.UnitCommentEntry
import com.consol.cmas.common.model.content.unit.UnitAttachmentEntry

log.info 'Resource (PC_Desktop) data have been UPDATEd!'
// Are there any changes?
if (changes) {
   log.info 'Yes, changes have been made to unit'
   log.info 'Changes object is a ' + changes.class
}

// Have Custom Fields been changed? If yes - which?
if (changes.customFieldChangeInfo) {
```

```
   log.info 'Yes, changes have been made to Custom Fields (Resource Fields)'
   log.info changes.customFieldChangeInfo
   log.info changes.customFieldChangeInfo.each { k, v ->
   log.info "Changed field: ${k.groupName}/ ${k.fieldName}"
   log.info "New value: ${v.value.value}"
   log.info "Old value: ${v.previousValue.value}"
}
} else {
   log.info 'No changes to Custom Fields'
}

// Have comments or attachmenst been changed? If yes - which?

log.info changes.contentChangeInfo

if (changes.contentChangeInfo){
   log.info 'Yes, changes have been made in detail section'
   if (changes.contentChangeInfo.value){
      log.info changes?.contentChangeInfo.each { ctEntry ->
         if (ctEntry?.value[0] instanceof UnitCommentEntry){
            log.info 'A comment has been added.'
            log.info 'Old value: ' + ctEntry?.previousValue
            log.info 'New value: ' + ctEntry.value[0]?.text
            log.info 'Made by the engineer ' + ctEntry.value[0]?.engineer?.name
            log.info 'Creation date of the comment: ' + ctEntry.value
              [0]?.creationDate
         } else if (ctEntry?.value[0] instanceof UnitAttachmentEntry){
            log.info 'An attachment has been added.'
            log.info 'Old value: ' + ctEntry?.previousValue
            log.info 'New value text: ' + ctEntry.value[0]?.text
            log.info 'New value file name: ' + ctEntry.value[0]?.filename
         }
      }
   } else {
      log.info 'Entry has been deleted.'
   }
}
```

Code example 110: *Resource Update script where changes are monitored and printed out to server.log*

When for a resource of type *PC_Desktops*, the content of the two resource fields *modell* and *name* are modified, the following text is printed into the `server.log` file.

```
PC_DesktpChangesInLog.groovy] [Susan-] Resource (PDC_Desktop) data have been
 UPDATEd!
PC_DesktpChangesInLog.groovy] [Susan-] Yes, changes have been made to unit
PC_DesktpChangesInLog.groovy] [Susan-] Changes object is a class
 com.consol.cmas.common.model.resource.history.ResourceChanges
PC_DesktpChangesInLog.groovy] [Susan-] Yes, changes have been made to Custom Fields
 (Resource Fields)
PC_DesktpChangesInLog.groovy] [Susan-] {(modell,PC_Desktop_Fields_
 basic)=Modification{value=AbstractField{key=(modell,PC_Desktop_Fields_basic),
 value=Computer-0815-01}, previousValue=AbstractField{key=(modell,PC_Desktop_
 Fields_basic), value=Computer-0815}}, (name,PC_Desktop_Fields_basic)=Modification
 {value=AbstractField{key=(name,PC_Desktop_Fields_basic), value=My cool PC11},
 previousValue=AbstractField{key=(name,PC_Desktop_Fields_basic), value=My cool
 PC}}}
PC_DesktpChangesInLog.groovy] [Susan-] Changed field: PC_Desktop_Fields_basic/
 modell
PC_DesktpChangesInLog.groovy] [Susan-] New value: Computer-0815-01
PC_DesktpChangesInLog.groovy] [Susan-] Old value: Computer-0815
PC_DesktpChangesInLog.groovy] [Susan-] Changed field: PC_Desktop_Fields_basic/ name
PC_DesktpChangesInLog.groovy] [Susan-] New value: My cool PC11
PC_DesktpChangesInLog.groovy] [Susan-] Old value: My cool PC
```

Code example 111: *Log output from the script above*

G.1.8 CM/Resource Pool - Assigning Permissions for Resources

G.1.8.1 Introduction to Permissions Concerning Resources

According to the ConSol CM basic principle, engineers can only access resources if they have the required permissions, i.e., if he has at least one role which has the required permissions. Resource permissions are granted at the resource type level, e.g., members of the role *ResourceManager_IT* can access the resource types *HP_Printer*, *MS_Word2013*, *PC_Desktops*, and *PC_Laptops*.

G.1.8.2 Assigning Resource Permissions Using the Admin Tool

To manage resource permissions, open the tab *Resource Types Permissions* (navigation group *Access and Roles*, navigation item *Roles*) in the Admin Tool.

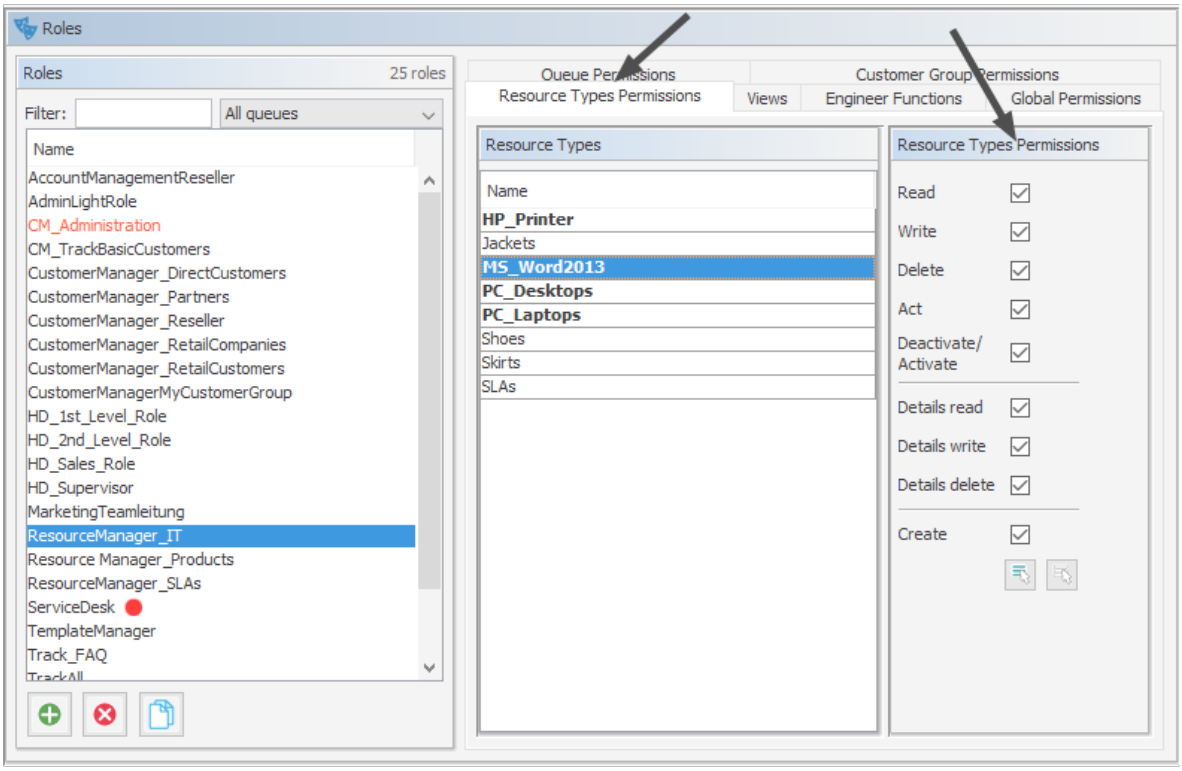


Figure 665: *ConSol CM Admin Tool - Access and Roles, Roles: Assigning resource permissions*

For a detailed description of the permissions, see section Tab Resource Types Permissions.

G.1.9 CM/Resource Pool - The Resource Pool Dashboard

G.1.9.1 Introduction

The Resource Pool Dashboard provides an overview of all objects in the Resource Pool. As per ConSol CM principles, only the objects are displayed which are covered by the access permissions of the engineer who is using the dashboard.

The Resource Pool Dashboard always contains the overview of all resources (section *Search or create Resources*, see the following figure). It might also contain some charts and/or tables which display figures/reports about resources.

Open the Resource Pool Dashboard by clicking *Resource pool* (or the label which has been configured in your CM system, see section Labels for details) in the main menu:

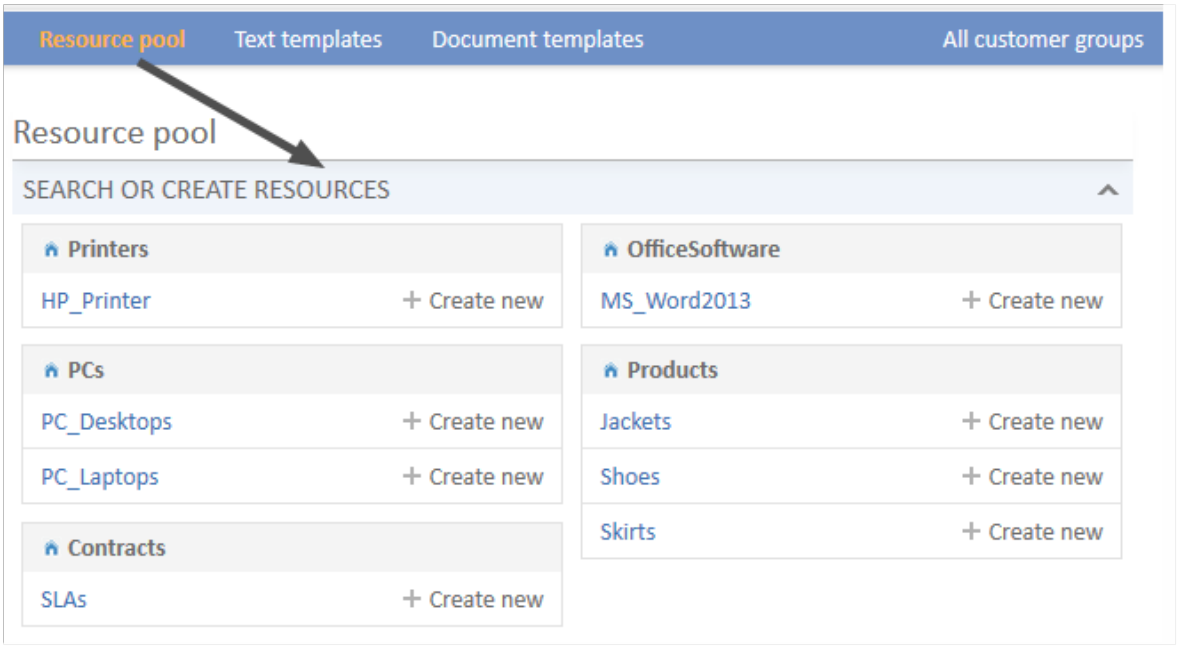


Figure 666: *ConSol CM Web Client - Resource Pool Dashboard without reports*

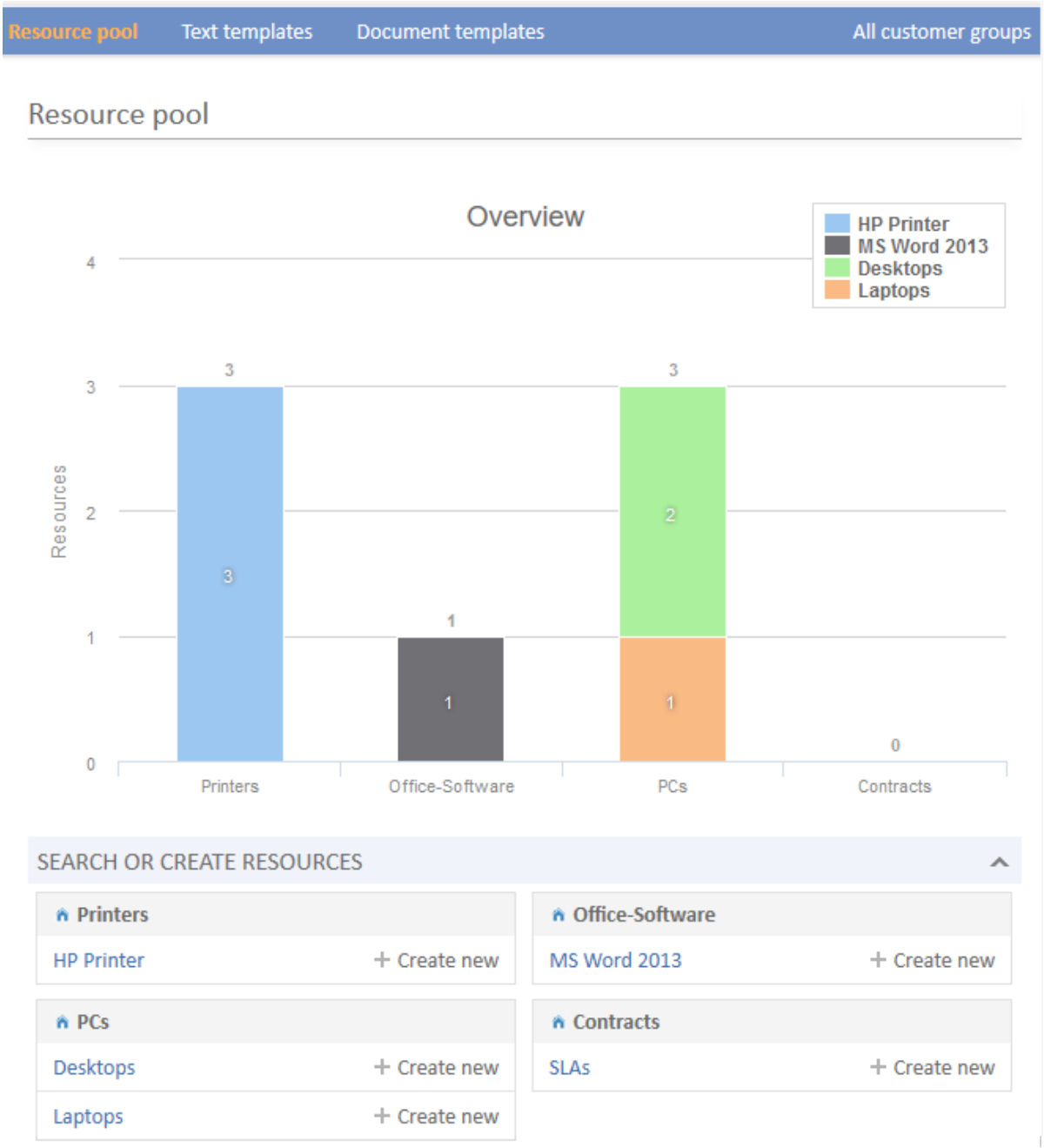


Figure 667: *ConSol CM Web Client - Resource Pool Dashboard with report*

The Resource Pool, including the Resource Pool Dashboard, is a new feature which has been introduced in ConSol CM version 6.10. However, after a system update from a version previous to 6.10, only the section *Search or create Resources* will be displayed (if the license for CM/Resource Pool is present). Any Resource Pool Dashboard report (chart and/or table widgets) has to be configured manually.

G.1.9.2 Configuring Reports for the Resource Pool Dashboard

When you configure reports with graphs and tables (or both) for the Resource Pool Dashboard, you follow the same process as for configuring the Web Client Dashboard (see section Page Customization for the Web Client Dashboard).

The **Resource Pool Dashboard** is configured using *page customization*.

Log in as *admin*, open the *Resource pool* page and select *Enable page customization* in the main menu. Besides other attributes which are not relevant for the Dashboard (and are explained in section Page Customization), the following (Resource Pool Dashboard-relevant) elements can be configured (i.e., attributes can be set). Each of the three elements is represented by a subtree in the page customization tree.

1. **widgetsGrid / resourceDashboard**
   The Resource Pool Dashboard can be switched on or off here. If a valid value is entered in the layout field, the Dashboard is displayed. If the field is empty, no Dashboard is shown.
   The following configuration can be performed here:
   a. The Dashboard layout, i.e., the layout of the grid on which the Dashboard is based (see section Definition of the Overall Dashboard Layout), this comprises:
      i. The widgets which should be displayed.
      ii. The order and placement of those widgets on the Dashboard page.
2. **chartWidget / resourceDashboard** (only available if chart widgets are present)
   a. The definition/layout for all chart widgets in the *chartWidget* subtree.
   b. Each widget is represented by one node which has the name of the widget, e.g., *chartWidget / resourceDashboard / ResourceDashboardOverview1*.
   c. A new chart widget is added when its name has been added in the *layout* value.
   d. Attributes can be defined for all chart widgets on the level *chartWidget* or *chartWidget / resourceDashboard* or they can be configured for one chart widget individually using the values of the attributes for the chart widget, e.g., *chartWidget / resourceDashboard / ResourceDashboardOverview1*.
3. **tableWidget / resourceDashboard** (only available if table widgets are present)
   a. The definition of all table widgets in the *tableWidget* subtree.
   b. Each widget is represented by one node which has the name of the widget, e.g., *tableWidget / resourceDashboard / ResourceDashboardOverview2_table*.
   c. A new table widget is added when its name has been added in the *layout* value.
   d. Attributes can be defined for all table widgets on the level *tableWidget* or *tableWidget / resourceDashboard* or they can be configured for one table widget individually using the values of the attributes for the table widget, e.g., *tableWidget / resourceDashboard / ResourceDashboardOverview2_table*.

The following figure provides an example page customization tree with subtrees relevant for the Resource Pool Dashboard. A detailed explanation is provided in the following sections.

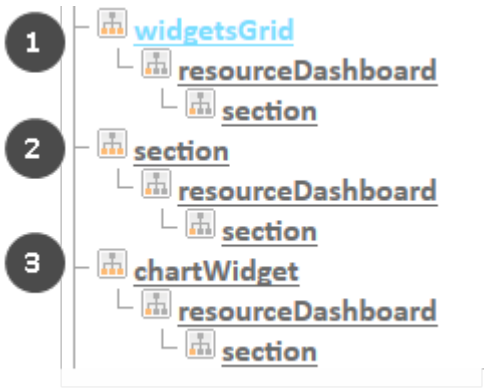


Figure 668: *ConSol CM Web Client - Page Customization subtree for Resource Pool Dashboard configuration*

- Overall layout for the resource pool dashboard (1)
- Dashboard overview section with all resources (default) (2)
- Definition of layout for all chart widgets in the resource pool dashboard (3)

G.1.9.3 Definition of the Overall Dashboard Layout

The overall layout of the entire Resource Pool Dashboard is defined using the page customization attribute *widgetsGrid / resourceDashboard*.

Attributes:

- **layout**
  This defines the layout of the entire dashboard on the Resource Pool Dashboard page based on the following:
  - Each row of the dashboard grid is represented as an array of elements: [x,y,z]. A new widget object will be added to the page customization tree automatically when it is added as value in the *layout* attribute, e.g., when the value *has been* [*ResourceDashboardOverview1:Chart*] and the value is *now* [*ResourceDashboardOverview1:Chart*, *ResourceDashboardOverview1*, *ResourceDashboardOverview2_table:Table*], a new table widget named *ResourceDashboardOverview2_table* will appear in the page customization tree (see the figure above). In the same way, widgets can be removed from the dashboard - just remove the name and type of the widget in the *layout* value. After saving and reloading the page all layout changes are available in the tree for further configuration.
  - A widget is described by its name and its type, separated by a colon, e.g., *ResourceDashboardOverview1:Chart*. The name for a specific widget must be unique.
  - The grid starts with the upper left corner (0,0) and it is built up row after row, e.g., a layout value with two pairs of [ ] brackets will represent two rows as shown in the figure and code below.
  - *null* is a reserved keyword for an empty cell.

- The type of a widget is *Chart* or *Table*. The type has to be indicated only at the first appearance of the widget's name. Afterwards it can be omitted, e.g., [*ResourceDashboardOverview1:Chart*, *ResourceDashboardOverview1*, *ResourceDashboardOverview1*] for a three-column chart.
- The widget can occupy multiple adjacent rows and columns.
- The dashboard can be completely disabled by removing the value from the attribute *layout*.

The following figure and code show the organization of an example grid and its representation in the page customization. This is a simple example with only one chart. For a more complex example which is shown for the Web Client Dashboard, please take a look at the respective section in Page Customization for the Web Client Dashboard.

Figure 669: *Organization of an example grid (1 row, 1 column only) for Page Customization for the Resource Pool Dashboard*

The value of the respective *layout* attribute would be:

```
[ResourceDashboardOverview1:Chart]
```

If you work with only one chart or table, it is not necessary to indicate more than one column since the entire width of the page will be filled with the widget.

G.1.9.4 Configuration of Widgets

Configuration Script for Widgets

Each chart widget and each table widget has a configuration script which will provide the data for the chart or table (e.g., connect to the database and execute the SQL statements to retrieve the data and to render the data correctly in the next step). This script is a Groovy script which is stored in the Admin Tool section *Scripts and Templates* and is referenced by its name. The script has to be of type *Page customization*. Select the widget in the page customization and enter the script's name:

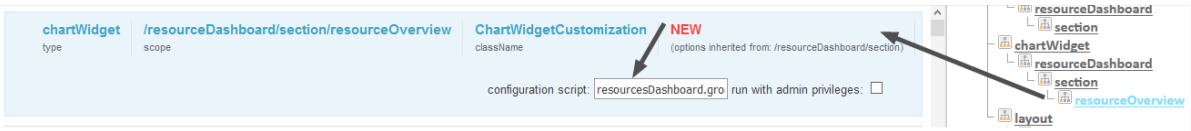


Figure 670: *ConSol CM Web Client - Script definition for a chart widget*

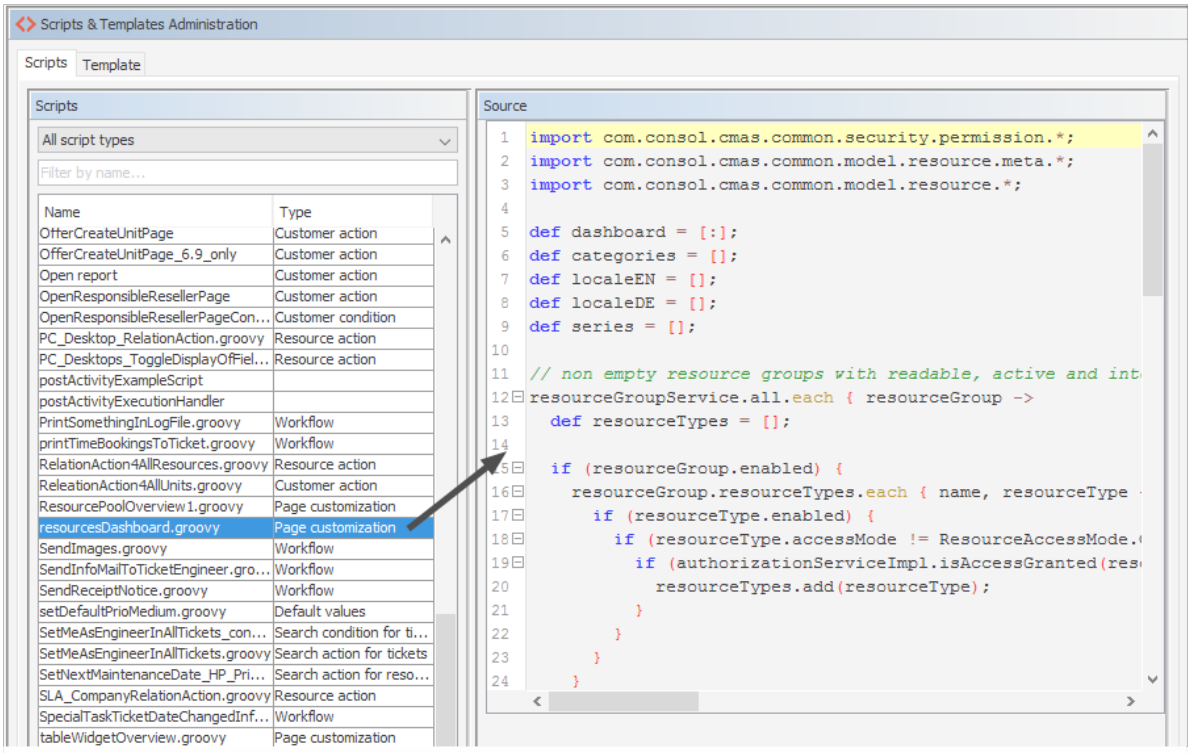


Figure 671: *ConSol CM Admin Tool - System, Scripts and Templates: Admin Tool script for a widget of the Resource Pool Dashboard*

The configuration script of a widget is the place where the statements are defined which retrieve the required data from the ConSol CM system and where the widget layout is defined. The execution of this Groovy script is a core part of the customization. The script must return a map of variables which correspond to the defined widget properties.

The script overwrites the configuration data provided in the page customization. The values are not merged!

The script thus will override any widget attribute value set in the page customization, so please make sure that the desired attribute is not set within the script if you want to use the page customization for attribute settings.

A **script** which is associated with a widget is usually executed with user (= engineer) permissions. However, sometimes values have to be used which are not available in the engineer context, e.g., escalated tickets (of all engineers) in a certain queue. In order to execute a script with admin permissions, select the check box *run with admin privileges*. Please keep in mind that the results of the Java or Groovy methods which retrieve the data will vary depending on the context. For example, the method `ticketService.getAll()` will return only tickets for which the current engineer has at least read permissions, but will return all tickets in the system when executed as admin.

The **chart** representation in the Resource Pool Dashboard is based on the Highcharts library. Thus, for chart widgets, the Admin Tool script has to return a HashMap containing the attributes which should be set (see the return statement in the code example below).

The **table** representation in the Resource Pool Dashboard is based on the Datatables library. Thus, for table widgets, the Admin Tool script has to return a HashMap containing the attributes which should be set.

⚠ Please note: Very complex scripts can decrease system performance!!!

The following example shows the script `ResourceDashboardOverview1.groovy`.

```
import com.consol.cmas.common.security.permission.*;
import com.consol.cmas.common.model.resource.meta.*;
import com.consol.cmas.common.model.resource.*;

def dashboard = [:];
def categories = [];
def localeEN = [];
def localeDE = [];
def series = [];

// non empty resource groups with readable, active and internal resource types
resourceGroupService.all.each { resourceGroup ->
   def resourceTypes = [];

   if (resourceGroup.enabled) {
      resourceGroup.resourceTypes.each { name, resourceType ->
         if (resourceType.enabled) {
            if (resourceType.accessMode != ResourceAccessMode.ON_THE_FLY) {
               if (authorizationServiceImpl.isAccessGranted(resourceType,
                [ResourceTypePermissionType.READ] as Set)) {
                  resourceTypes.add(resourceType);
               }
            }
         }
      }
   };

   if (resourceTypes) {
      Collections.sort(resourceTypes);
      dashboard.put(resourceGroup, resourceTypes);
      categories.add('_(\'' + resourceGroup.name + '\')');
```

```
   }
};

// add localization
dashboard.each { resourceGroup, resourceTypes ->
   def resourceGroupEN = localizationService.getLocalizedProperty
    (ResourceGroup.class, "name", resourceGroup.getId(), Locale.ENGLISH);
   def resourceGroupDE = localizationService.getLocalizedProperty
    (ResourceGroup.class, "name", resourceGroup.getId(), Locale.GERMAN);
   if (resourceGroupEN) {
      localeEN.add("${resourceGroup.name}:${resourceGroupEN}" as String);
   }
   if (resourceGroupDE) {
      localeDE.add("${resourceGroup.name}:${resourceGroupDE}" as String);
   }
   resourceTypes.each { resourceType ->
      def resourceTypeEN = localizationService.getLocalizedProperty
       (ResourceType.class, "name", resourceType.getId(), Locale.ENGLISH);
      def resourceTypeDE = localizationService.getLocalizedProperty
       (ResourceType.class, "name", resourceType.getId(), Locale.GERMAN);
      if (resourceTypeEN) {
         localeEN.add("${resourceType.name}:${resourceTypeEN}" as String);
      }
      if (resourceTypeDE) {
         localeDE.add("${resourceType.name}:${resourceTypeDE}" as String);
      }
   };
};

// prepare data
Map<Long, Long> counters =
 resourceService.getResourceTypesIdsToActiveResourcesCountersMapping();

dashboard.eachWithIndex { resourceGroup, resourceTypes, index ->
   resourceTypes.each { resourceType ->
      def count = counters.get(resourceType.id);
      if (count != null && count > 0) {
         def data = [null] * categories.size;
         data[index] = count;
         series.add("{ name: _('${resourceType.name}'), data:[${data.join(',')}] }"
          as String);
      }
   };
};

return [
   visible: "true",
   chart: "{ type: 'column' }",
   title: "{ text: 'Overview' }",
   legend: "{ layout: 'vertical', align: 'right', verticalAlign: 'top', floating:
    true, maxHeight: 200, backgroundColor: 'white', borderColor: '#CCC',
    borderWidth: 1, shadow: false, navigation: { animation: true } }",
   xAxis: "{ categories: [${categories.join(',')}] }" as String,
```

```
   yAxis: "{ allowDecimals: false, min: 0, title: { text: 'Resources' },
    stackLabels: { enabled: true, style: { fontWeight: 'bold', color: 'gray' } }
    }",
   tooltip: "{ headerFormat: '<b>{point.key}: </b>', pointFormat: '
    {point.stackTotal}<br><span style=\"color:{series.color}\">?</span>
    {series.name}: {point.y}'",
   plotOptions: "{ column: { stacking: 'normal', dataLabels: { enabled: true,
    zIndex: 5, color: 'white', style: { textShadow: '0 0 3px black' } } } }",
   series: "[ ${series.join(',')} ]" as String,
   localization: "{ de: {${localeDE.join(',')}}, en: {${localeEN.join(',')}}}" as
    String
];
```

Code example 112: *ResourceDashboardOverview1.groovy*

The following chart is defined by the script above.

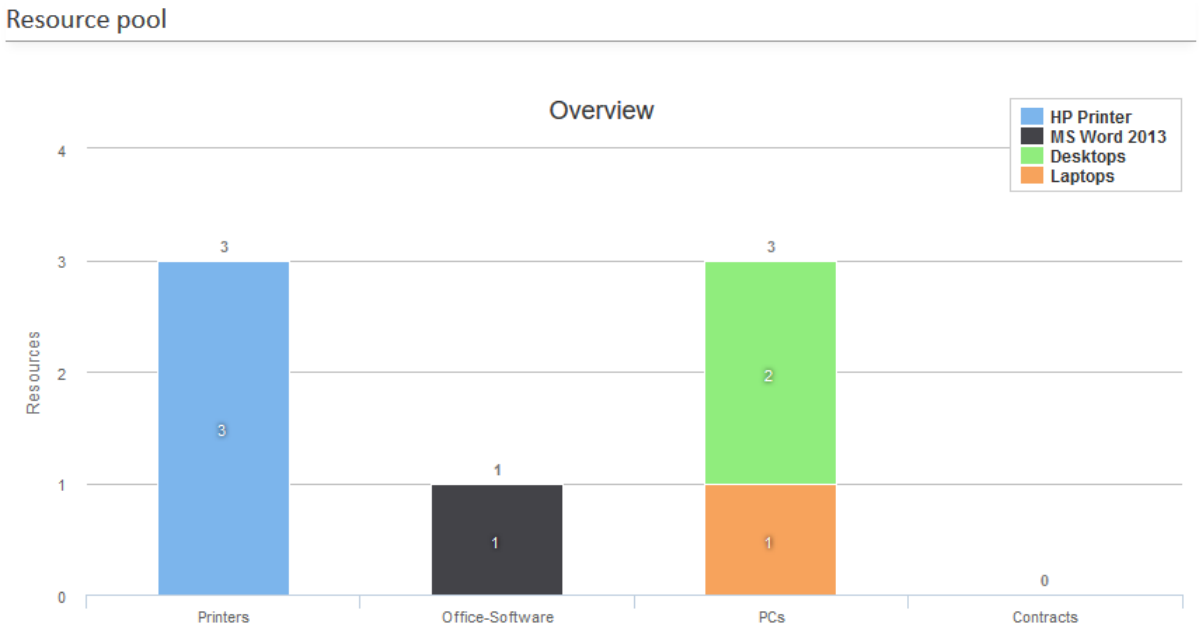


Figure 672: *Example chart widget*

Configuration Attributes for Widgets

These are the same as for the Web Client Dashboard. Please refer to section Configuration Attributes for Widgets.

G.2 CM/Doc

CM/Doc is a ConSol CM add-on that allows you to create document templates. These templates can be used to create documents directly from the business management process. CM/Doc supports Microsoft Word documents and (in CM versions 6.10.1 and up) OpenOffice documents.

Please see section CM/Doc for a detailed description of this add-on.

G.3 CM/Track: The Customer Portal

G.3.1 Overview

G.3.1.1 Introduction

The customer portal ***CM/Track*** allows customers to log in to the ConSol CM system. Like the ConSol CM Web Client, CM/Track is a web-based application, i.e., the customer only needs a standard web browser for access to the portal.

Technically, the data for CM/Track is retrieved using a REST (Representational State Transfer, see Glossary.

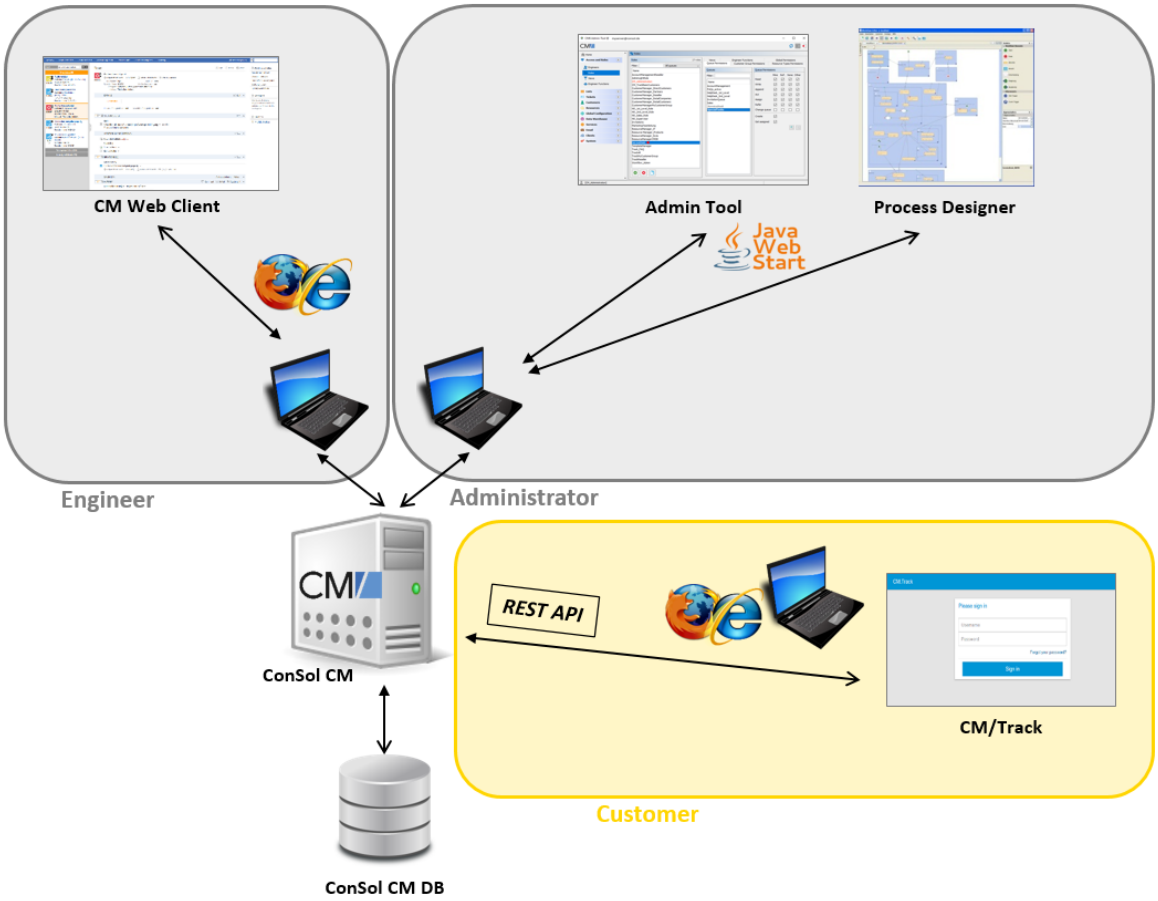


Figure 673: *ConSol CM system architecture with CM/Track*

In a standard environment, a customer can perform the following operations via CM/Track:

- See a list of his tickets.
- See a list of all tickets of his company (if this has been configured).
- Add comments and/or attachments for a ticket.

- Open/Create a new ticket.
- Search the FAQs for solutions.

G.3.1.2 CM/Track Versions

There are two versions of CM/Track:

- CM/Track V1
  Available as portal solution in ConSol CM versions 6.10.4 and less.
- CM/Track V2
  Available as (new) portal solution in ConSol CM versions 6.10.5 and up.

The functionalities and the user interface are very similar in both versions, however, to provide an easy access to the version of your CM system and to avoid any miscommunications, you will find two separate sections in the Admin Manual.

If you run CM/Track V1 and perform an update to CM version 6.10.5 (or higher), you can continue operating V1. Of course you could also migrate to V2, which would include adapting V2 in the same way V1 was adapted, if you do not use the standard flavor of CM/Track. Please note that there is **no automatic update** from V1 to V2, since the two are separate web applications which are deployed in the application server!

We recommend to consider operating (and if required: migrating to) V2 because this version provides extended security features and will be part of future versions of ConSol CM.

G.3.2 Authentication Methods for Customers in CM/Track

G.3.2.1 Available Authentication Methods

There are three possible authentication modes:

- Against the ConSol CM database
  This is called *DATABASE mode*, see Database Authentication for Customers
- Against an LDAP server
  This is called *LDAP mode*, see LDAP Authentication for Customers in CM/Track
- Against an LDAP server and the ConSol CM database
  The order can be configured. This is called *Mixed mode*, see Mixed Authentication Mode

G.3.2.2 Defining the Authentication Method

The authentication mode is specified by the system property cmas-core-security, contact.authentication.method. A change of this property does not require a server restart, and is propagated to all cluster nodes.

The possible values (see also section System Properties) and their respective system behaviors are:

- **DATABASE**
  Attempt a database login if the customer has a database password. I.e., the login and password are stored in the ConSol CM database and are thus managed by the ConSol CM engineers, or indirectly by the customers themselves when they reset their password. The customer can reset his own password, see section Password Reset Template for Customers Using CM/Track V2.
- **LDAP**
  Try authentication using the available LDAP server(s), if an LDAP ID is provided. I.e., the password is stored in the LDAP directory and cannot be changed via ConSol CM, neither by the customer nor by an engineer.
- **LDAP,DATABASE**
  First attempt authentication using the available LDAP server(s), if an LDAP ID is provided. On failure, try a database login if the customer has a database password.
- **DATABASE,LDAP**
  First attempt a database login if the customer has a database password. On failure try authentication using the available LDAP server(s) if an LDAP ID is provided.

The values are case insensitive, and commas and whitespace are ignored.

G.3.2.3 Database Authentication for Customers

Database authentication is activated by setting the system property cmas-core-security, contact.authentication.method to “DATABASE” (default value).

There are two steps which you need to perform to set up database authentication for customers using CM/Track:

- Create customer fields for the user name (login) and password in the Admin Tool (see Defining the Customer Fields for CM/Track Login and Password)
- Enter the user name and password for the actual customers in the Web Client (see Granting Access to CM/Track for Customers)

When database authentication is used, you can allow your customers to change their own passwords, see Configuring CM/Track for Password Reset by Customers.

Defining the Customer Fields for CM/Track Login and Password

The fields for login and password for a customer are regular customer fields at the contact level. Please see section Setting Up the Customer Data Model for an introduction to customer field management and GUI configuration for customer data.

Edit the fields which contain the customer data (if there are two levels: **not** the company level, but the contact level!) as demonstrated in the following example. You reach the following screen by opening the navigation group *Customers*, navigation item *Data Models*.

- Create a field for the **login** with the annotation `username` = “true”.

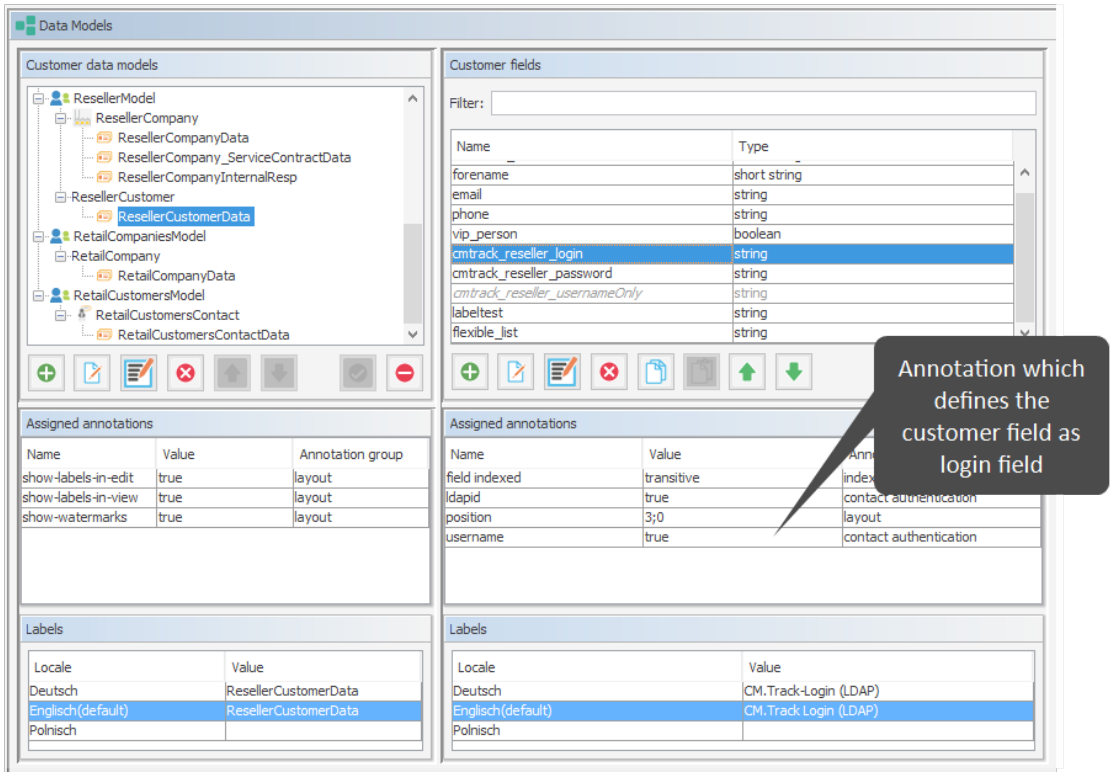


Figure 674: *ConSol CM Admin Tool - Customers, Data Models: CM/Track - Annotation 'username' for login*

> ⚠ Assigning the annotation `username` to a customer field is only possible, if there is no previous assignment of this annotation. Otherwise it will be prohibited. When assigning it, a warning dialog must be confirmed before it is executed, since it can be a longer running operation. Un-assigning the annotation must be confirmed as well, because it cannot be undone: Un-assignment delete the user name values unrecoverably from internal storage.

- Create a field for the password with the annotation `password` = “true”. The annotation `text-type` = “password” guarantees that only stars/dots are displayed in the Web Client, not the clear text password.

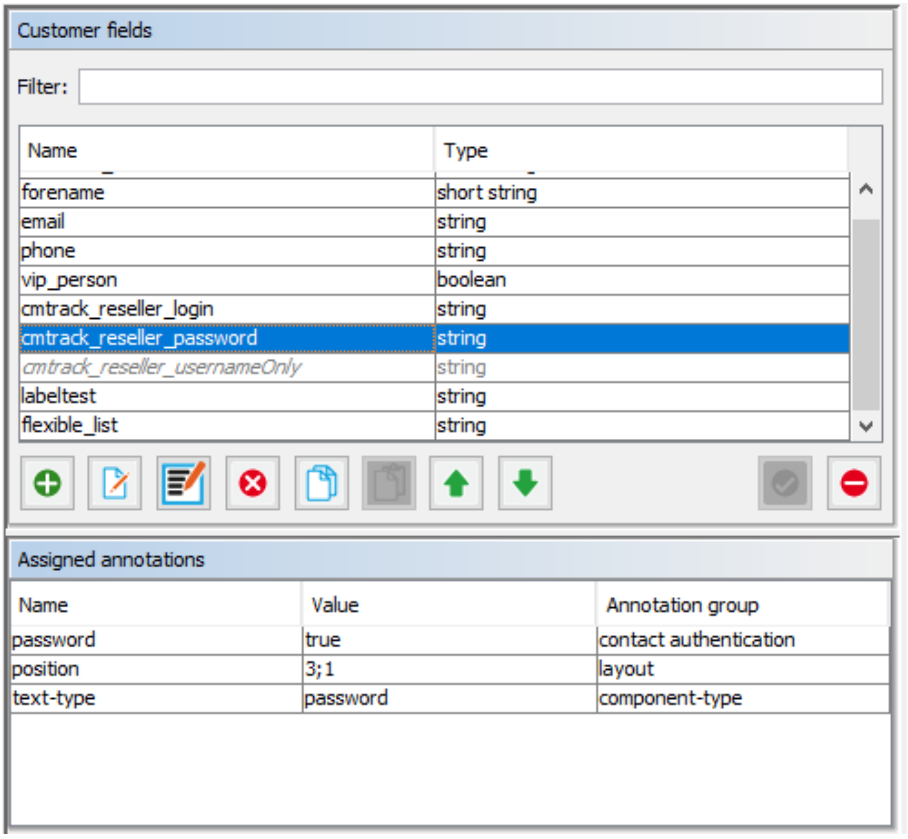


Figure 675: *ConSol CM Admin Tool - Customers, Data Models: CM/Track - Annotation for password*

> ⚠ The annotation `password` requires confirmation when assigned.
>
> In case of an update from CM versions lower than 6.11 to 6.11 and up: When this annotation is set, the system reads the plain text passwords from the original field values, encrypts them and saves the encrypted values to the internal storage. Then the original field values are deleted and thus the plain text value cannot be recovered anymore.
>
> When trying to un-assign the `password` annotation the operation must be confirmed as well, since the encrypted passwords are deleted from the internal storage. After the annotation unassignment the password information is completely lost and cannot be recovered at all.

When a scenario from a CM version lower than 6.11 is imported into a system with CM 6.11 (or higher), a transformation of user names and passwords is performed automatically. This is described in detail in section Transformation of User Name and Password Fields During Import into CM 6.11 .

Granting Access to CM/Track for Customers

The engineer working with the Web Client can then assign a user name, initial password, and a CM/Track user profile to every customer who should have access to the portal CM/Track. The user name has to be unique. This is checked by the system. You cannot enter a name a second time if this has already been assigned to another customer. The password is stored as encrypted string in the CM database. This means that an engineer can set a new password, e.g., when a customer calls and asks for this, but it is never possible to read the password from the system.

> ⓘ You, as an administrator, can define if the CM/Track user names should be case sensitive. Use the CM system property cmas-core-security, policy.track.username.case.sensitive. This is a boolean variable. When it is set to “true”, the CM/Track user names are case sensitive. Please make sure that the database collation which is in use supports case sensitive strings!

The following example shows the customer data of an example contact in the ConSol CM Web Client. You reach this screen by opening a contact data set in edit mode.

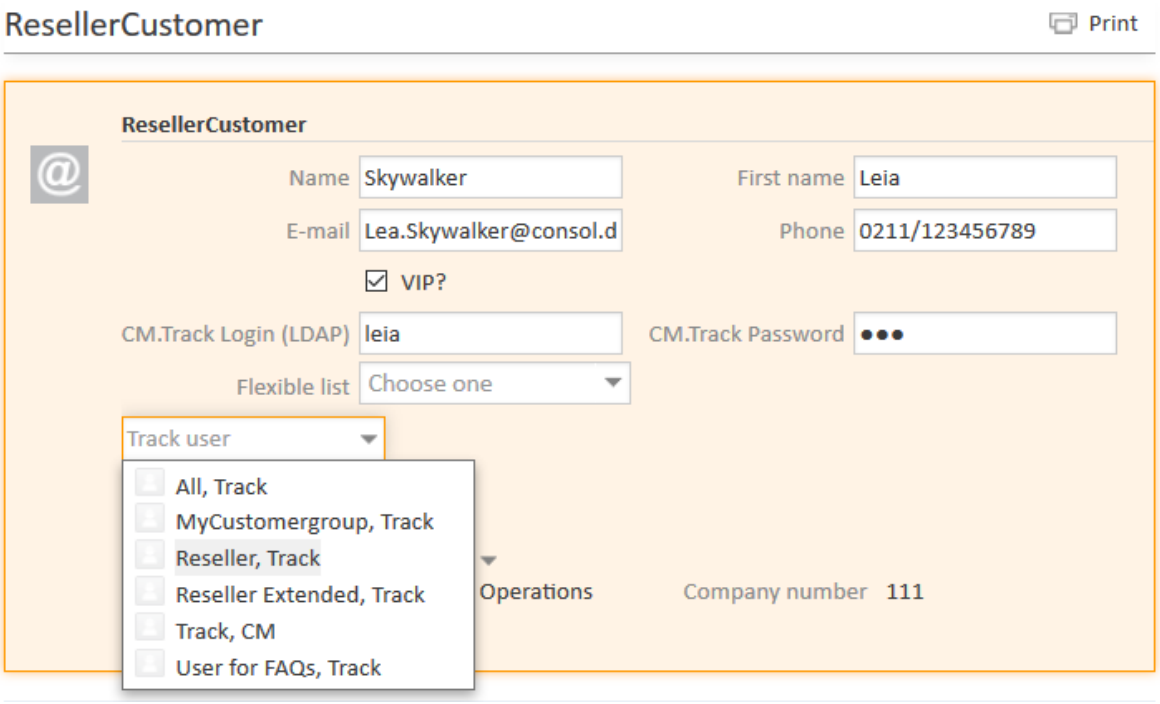


Figure 676: *ConSol CM Web Client - Contact page: CM/Track user data*

Configuring CM/Track for Password Reset by Customers

CM/Track can be configured to offer a hyperlink for customers where a customer can reset his password. This is based on the template `track-password-reset-template`. Please refer to section Password Reset Template for Customers Using CM/Track V1 / Password Reset Template for Customers Using CM/Track V2 for a detailed explanation. The password reset in CM/Track is only possible when the DATABASE mode is used. It is not possible when LDAP authentication is in operation. See section Authentication Methods for Customers in CM/Track for the portal for an explanation of all possible authentication modes.

Please note that the `From`address of the email which is sent to a customer who has requested a new password can be set using the CM system property cmas-core-security, password.reset.mail.from.

G.3.2.4 LDAP Authentication for Customers in CM/Track

Introduction to ConSol CM LDAP Authentication

ConSol CM offers LDAP authentication for CM/Track as a standard feature, i.e., instead of managing the passwords for the ConSol CM customers in the ConSol CM database, they can be retrieved from an LDAP server (like e.g., a ***Microsoft Active Directory*** server).

When customers want to log in to CM/Track, they enter their user name and password and press *Enter*. Behind the scenes, the ConSol CM server sends a request with the customer's user name and password and asks the LDAP server whether those credentials are correct.

If the credentials are correct, the approval is sent back to the ConSol CM server and the customer is logged into CM/Track.

> Please keep in mind that the LDAP connection is only used to authenticate the customer (confirm the identity). The authorization (i.e., the assignment of access permissions in the system) is done via the assignment of a CM/Track user profile in the Web Client. The CM/Track user profiles are managed in the engineer and role administration in the Admin Tool.

Please see also the following picture for an explanation of the CM/Track authentication process using LDAP.

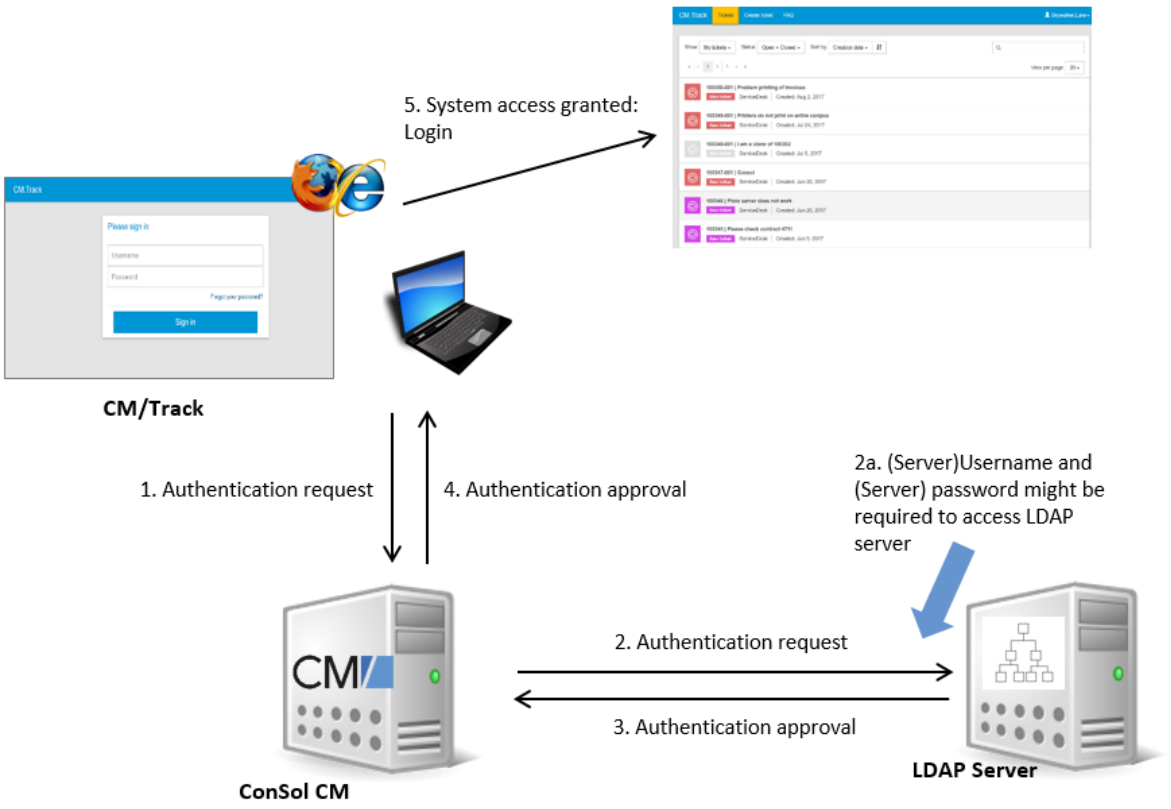


Figure 677: *ConSol CM - LDAP authentication process for CM/Track*

Configuring LDAP Authentication

LDAP authentication is activated by setting the system property cmas-core-security, contact.authentication.method to “LDAP”.

Then you have to set the required values in the system properties (navigation group *System*, navigation item *System Properties*, see System Properties).

The LDAP servers can be defined using the following system properties from the module `cmas-core-security`.

{name} is a string that you can choose to distinguish LDAP servers. It must always be set, even if only one LDAP server is configured. You should use a simple string for the {name}, not containing any keywords, like *internal* or *external*, and which does not contain special characters.

- contact.authentication.method
  LDAP
- ldap.contact.{name}.providerurl
  The property value is the address of the LDAP server in the form *ldap[s]://host:port*.
- ldap.contact.{name}.userdn
  The value is the user DN used to look up the contact DN by the LDAP ID. An anonymous account is used if the value is not set.
- ldap.contact.{name}.password
  The property contains the password to look up the contact DN by the LDAP ID. An anonymous account is used if the value is not set.
- ldap.contact.{name}.basedn
  This represents the base path to search for the contact DN by the LDAP ID, e.g., “ou=accounts,dc=mycompany,dc=de”.
- ldap.contact.{name}.searchattr
  The property value stands for the attribute to search for the contact DN by the LDAP ID, e.g., “uid”.

Initially, these system properties might not be present in your CM system. Just add them manually. Changes to any of the above system properties do not require a server restart and are propagated to all cluster nodes. The use of the placeholder {name} allows configurations to define several different LDAP servers.

- ldap.initialcontextfactory
  This is a predefined global property. If it is not set, “com.sun.jndi.ldap.LdapCtxFactory” is used as its value.

Authentication attempts against LDAP servers are made until first success, where the server order is determined by their {name} values (ascending alphabetical order of the values).

Mixed Authentication Mode

Set the system property cmas-core-security, contact.authentication.method depending on the desired order of authentication instances:

- **LDAP,DATABASE:**
  First attempt authentication using the available LDAP server(s), if an LDAP ID is provided. On failure, try a database login if the customer has a database password.
- **DATABASE,LDAP:**
  First attempt a database login if the customer has a database password. On failure try authentication using the available LDAP server(s) if an LDAP ID is provided.

The CM system will first contact the instance which is mentioned first, than the second one. For example, when the contact authentication method is set to “LDAP,DATABASE” and the customer (contact) uses the password which is only valid in the database, the login will succeed.

In `server.log` the following message will be displayed:

```
LDAP login failed: [LDAP: error code 49 - Invalid Credentials]; nested exception is
 javax.naming.AuthenticationException: [LDAP: error code 49 - Invalid Credentials]
```

Logging of LDAP Login Attempts in CM/Track

All LDAP errors encountered are logged without a stack trace using loggers with the following prefix:

- `com.consol.cmas.core.security.contact`

The stack trace of LDAP errors is not logged because failed login attempts on the first LDAP server would clutter logs if a following login on the second LDAP server succeeded.

Using LDAPS (LDAP over SSL)

Introduction

Per default, when an LDAP client accesses an LDAP server, the information is transferred in clear text. In case you want the user name and password to be transferred to the LDAP server in encrypted form, you have to set up the LDAP authentication using LDAPS.

Preparations

You have to configure the CM server machine (Java) in a way that can use certificates. One way to do this for a Linux environment is described in the following section.

1. Retrieve the certificate:
   `openssl s_client -connect dc2.mydomain.com:ldaps`
2. The answer will contain a section which starts with "---BEGIN CERTIFICATE " and ends with "END CERTIFICATE ---".
   Copy this section to a file, e.g., `/tmp/certificate2_dc2_mydomain_com.txt`
3. Import the certificate to the truststore of your machine, e.g., /home/mydirectory/mytruststore
   `$JAVA_HOME/bin/keytool -import -alias <arbitrary> -trustcacerts -keystore /home/mydirectory/mytruststore -file/tmp/certificate2_dc2_mydomain_com.txt`
   You have to enter (set) a password.
4. Enter the truststore in the ConSol CM config file in JAVA_OPTS:
   `-Djavax.net.ssl.trustStore=/home/mydirectory/mytruststore -Djavax.net.ssl.trustStorePassword=<see above>`

LDAPS Configuration in the ConSol CM Admin Tool (System Properties)

Configure the ConSol CM server as shown in the following example:

- cmas-core-security, ldap.authentication = simple
- cmas-core-security, ldap.contact.name.basedn = OU=myOU,DC=myDC
- cmas-core-security, ldap.initialcontextfactory = com.sun.jndi.ldap.LdapCtxFactory
- cmas-core-security, ldap.contact.name.password = myLDAPpw
- cmas-core-security, ldap.contact.name.searchattr = sAMAccountName
- cmas-core-security, ldap.contact.name.userdn = myLDAP_UserDN

Depending on the LDAP server configuration, use one of the following values for the server URL:

- **Standard LDAPs port**
  cmas-core-security, ldap.contact.name.providerurl = ldaps://dc2.mydomain.com:636
- **LDAPs port Global Catalogue**
  cmas-core-security, ldap.contact.name.providerurl = ldaps://dc2.mydomain.com:3269

Setting Up Customer Accounts for LDAP

There are two steps which you need to perform to set up LDAP authetication for customers using CM/Track:

- Set the required annotation for the customer field which should hold the LDAP ID in the Admin Tool.
- Enter the LDAP IDs for the actual customers in the Web Client.

When LDAP mode is used, the customer field which is used for the CM/Track user name (login) has to have two annotations:

- `username` = true
- `ldapid` = true

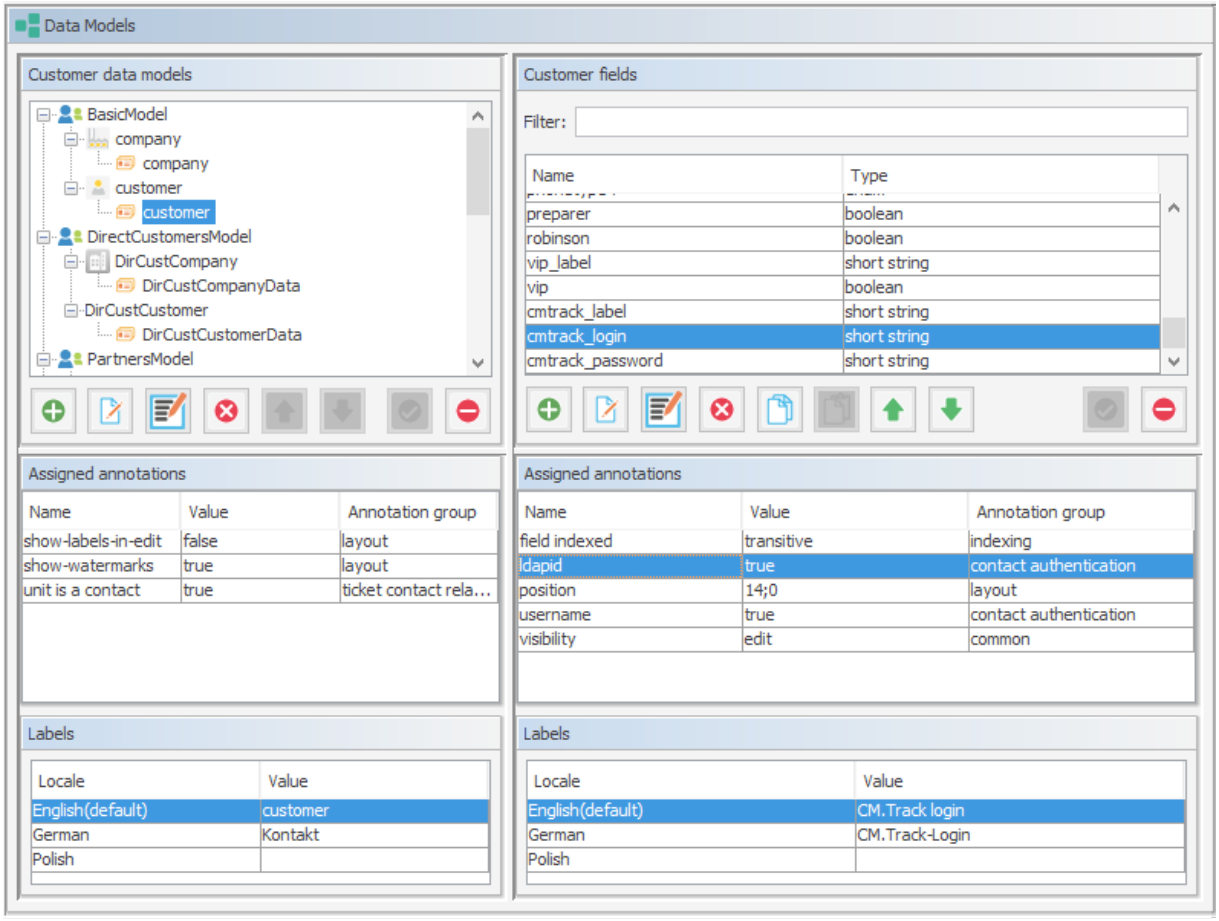


Figure 678: *ConSol CM Admin Tool - Customers, Data Models: Customer field for LDAP authentication of CM/Track users*

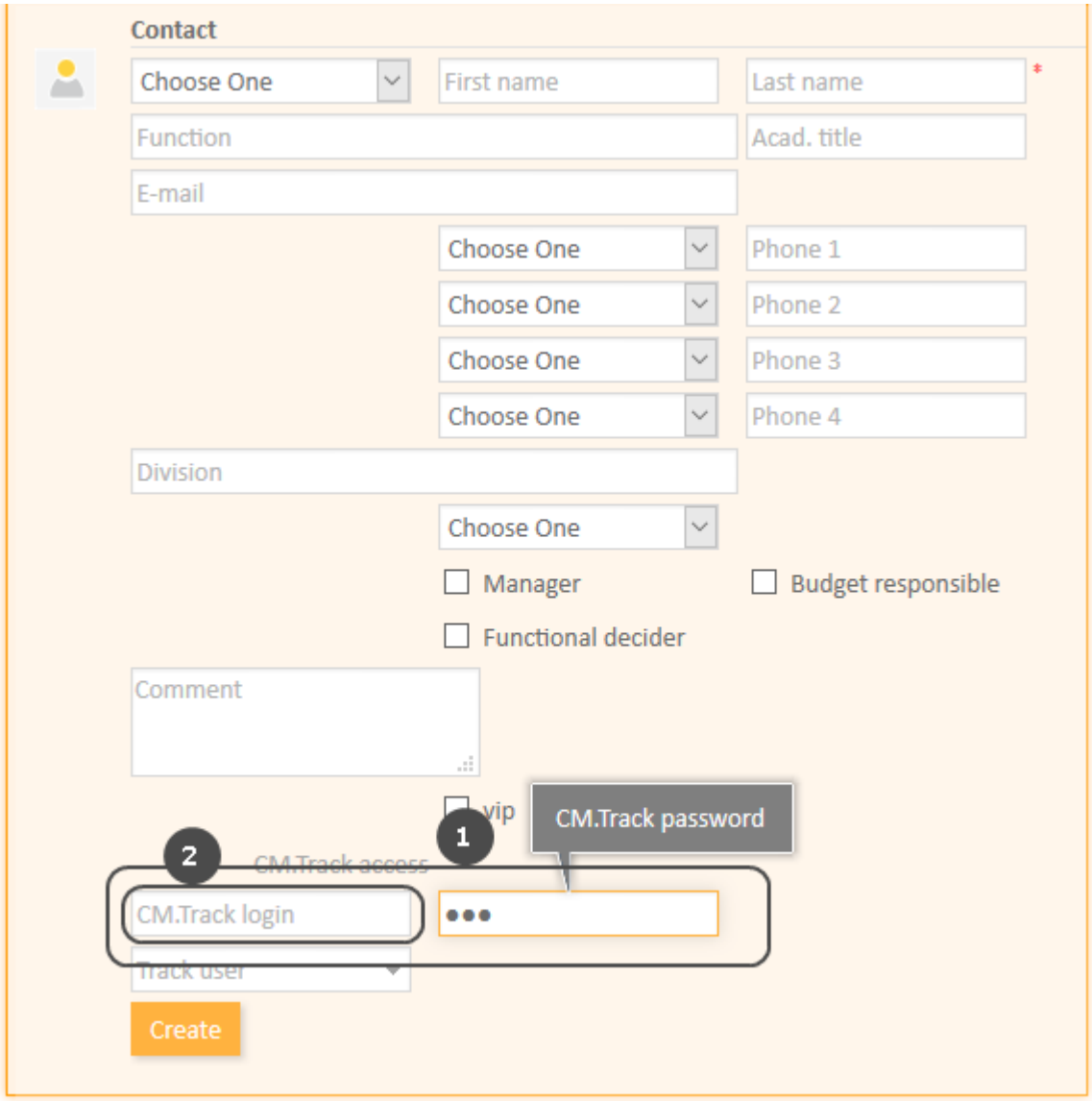


Figure 679: *ConSol CM Web Client - Field for LDAP ID in contact data*

- Used for database authentication in case of mixed mode (1)
- Used for LDAP authentication (2)

G.3.3 CM/Track V1

See the following sections for topics which concern CM/Track V1:

- Customer authentication methods in CM/Track (both V1 and V2):
  See section Authentication Methods for Customers in CM/Track.
- General system access to CM/Track for customers:
  See section CM/Track V1: System Access for CM/Track Users (Customers).
- Using the portal for FAQs:
  See section CM/Track V1: FAQs in CM/Track.

G.3.3.1 CM/Track V1: System Access for CM/Track Users (Customers)

In the following chapter you will find detailed information about how to configure your ConSol CM system to grant access to CM/Track (the ConSol CM portal) to your customers.

CM/Track V1 is a ConSol CM add-on which has to be purchased separately.

Please note that for every CM/Track user (i.e., user profile), a ConSol CM license is required. Since numerous customers can log in to CM/Track using one user profile, you do not need a ConSol CM license for every customer. Please refer to section License Management for details.

Precondition

CM/Track V1 is part of every default shipment of ConSol CM in versions 6.10.4 and below, so there are no new files that have to be deployed. However, the default function set of CM/Track provides basic functionalities (e.g., viewing a ticket list, creating a new ticket, seeing ticket details) and the pages have a CM standard layout. In order to use CM/Track as a powerful portal for customer access to the system, the layout should be adapted to a company's CD (corporate design), a process called *skinning*. The forms and lists which are displayed for the customer might be modified and/or extended. Please contact our consulting team or your account manager if you would like to adjust CM/Track for your company in an optimal way.

CM/Track Technical Background

The portal CM/Track is based on the ***REST API*** of ConSol CM. Please refer to the separate document *ConSol CM REST API Documentation* for details.

General Principle of System Access via CM/Track

A customer who wants, or should, have access to your ConSol CM system using the portal CM/Track has to have a login and a password. Both can be initially provided by the engineer who edits the customer data using the ConSol CM Web Client, or the values can be imported automatically into the database.

The fields for the login and password of customers are customer fields which are defined like any other customer field and which have special annotations. If you are not familiar with customer fields, please refer to section Customer Field Management and GUI Design for Customer Data.

The access permissions of the customer are defined by assigning a user profile to the customer's account. The user profiles are managed by the ConSol CM administrator using the Admin Tool.

Defining the User Profiles/Access Permissions for CM/Track

As one of the first steps, you have to define user profiles, i.e., profiles of access permissions to CM/Track. A CM/Track user profile is defined like a regular engineer (please see section Engineer Administration for details), but is marked as *Track*.

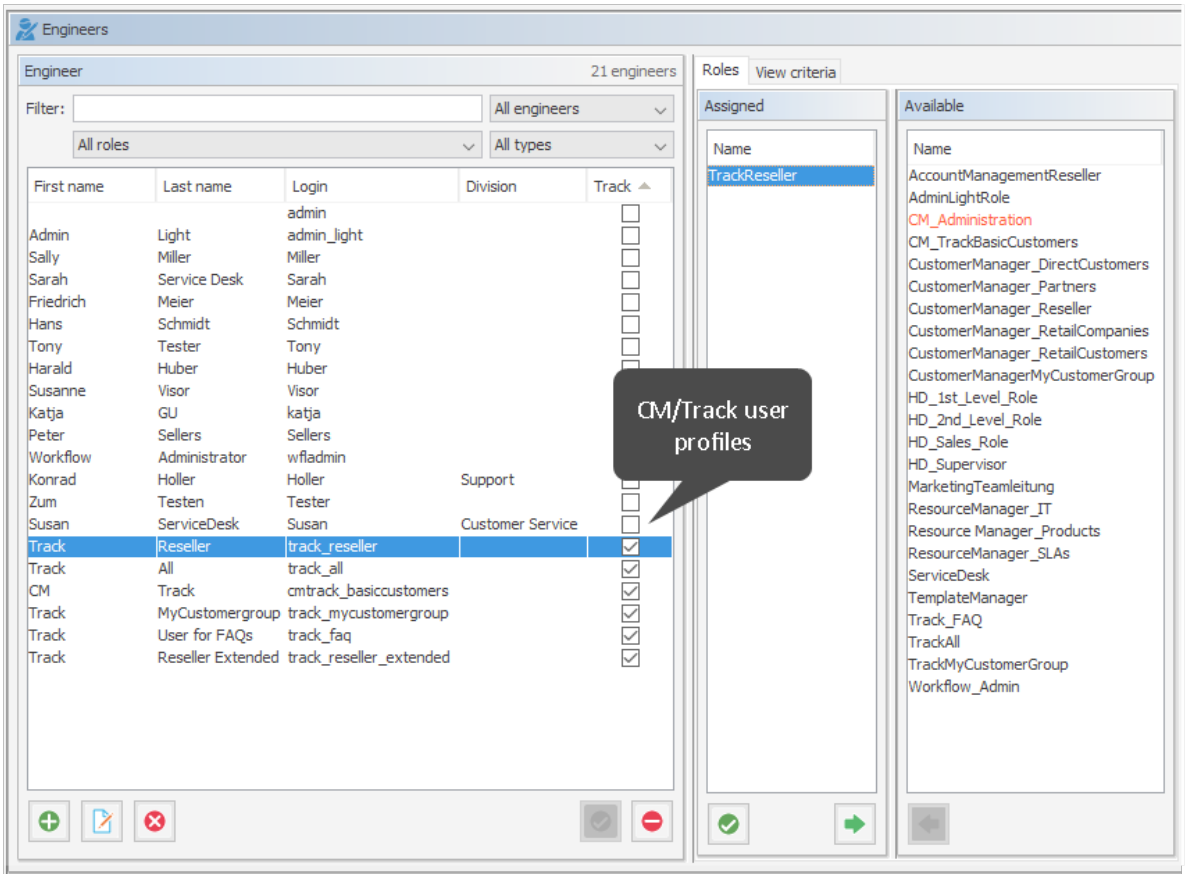


Figure 680: *ConSol CM Admin Tool - Access and Roles, Engineers: User profile name for CM/Track*

The user profile is then assigned to one or more roles to define the access permissions to queues and customer groups. For example you can set up a user profile (engineer) *cmtrack_basiccustomers* that has the role *TrackAll*. This role has read/write/append permissions for the queues *Helpdesk_2nd_Level* and *ServiceDesk* and has customer group permissions for three customer groups. Please note that

- always queue **and** customer group permissions have to be granted to allow ticket access via CM/Track for customers
- you must assign matching queue and customer group permissions, i.e. assign queues where the respective customer groups have been assigned (see section Queue Administration).
- read permissions for customer groups will be sufficient in most (standard) cases, since it is not possible to edit customer data using the portal.

In our example, the role *TrackAll* has read access to all customer groups. In your system, it might be required to create different CM/Track roles with access to different customer groups. For a detailed introduction to role administration, please refer to section Role Administration.

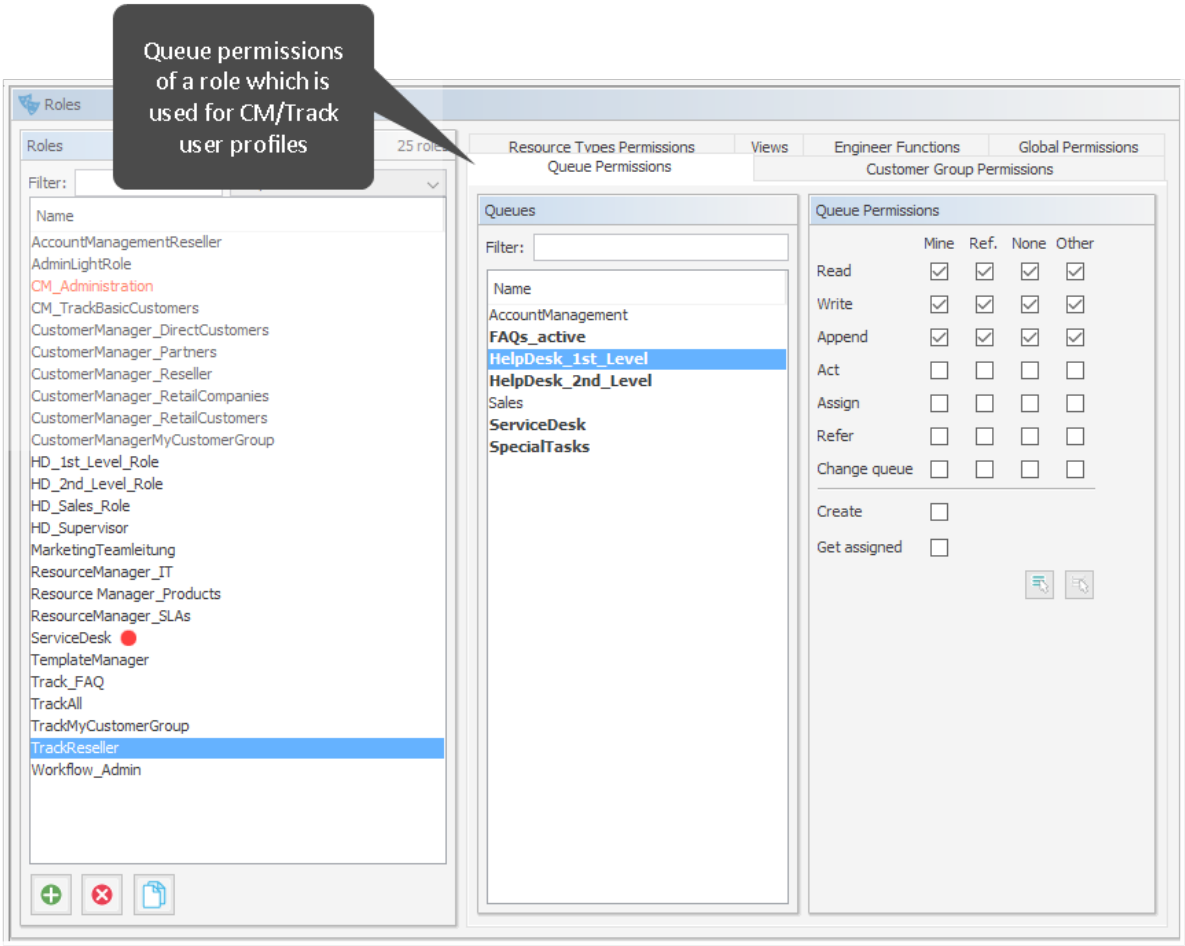


Figure 681: *ConSol CM Admin Tool - Access and Roles, Roles: User profile for CM/Track, queue permissions*

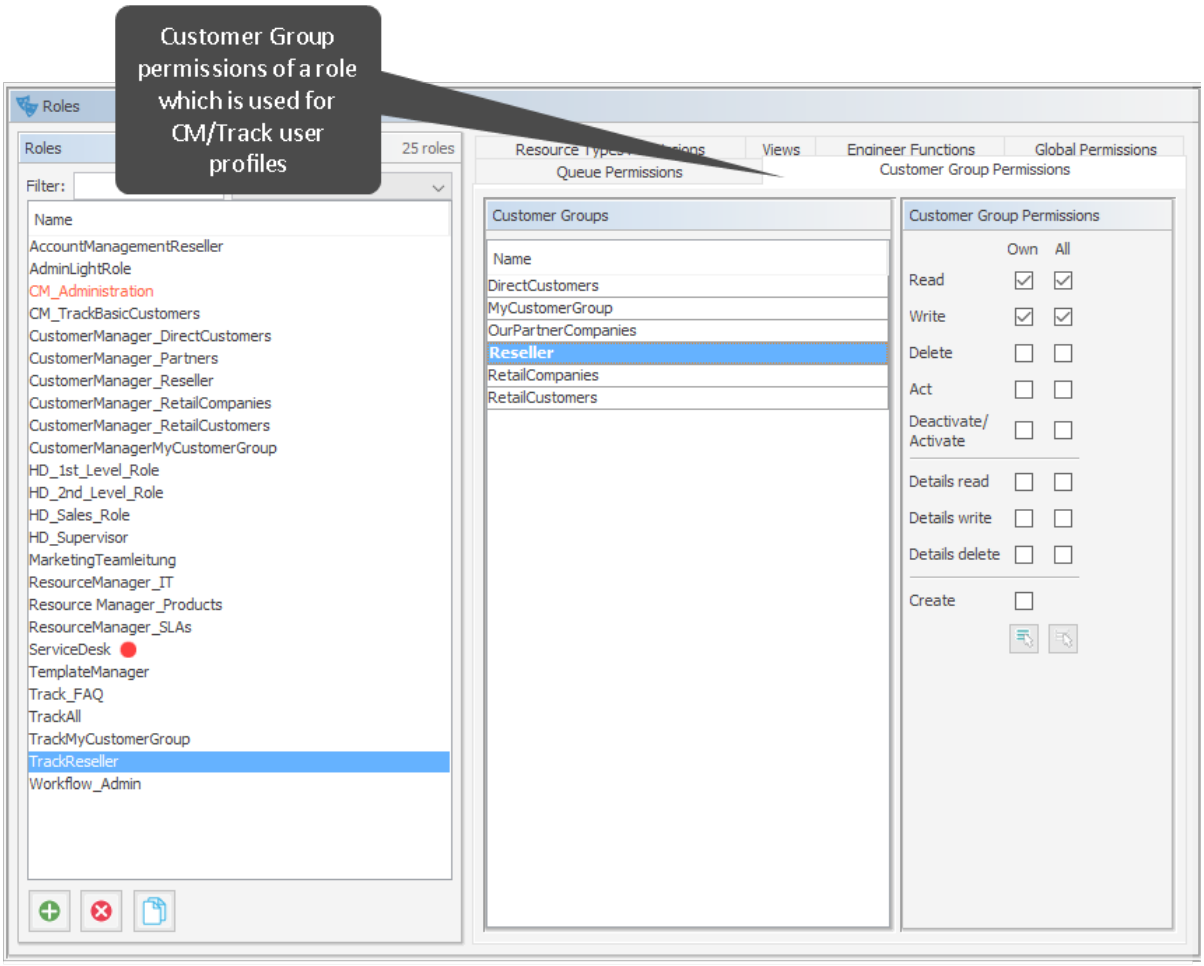


Figure 682: *ConSol CM Admin Tool - Access and Roles, Roles: User profile for CM/Track, customer group permissions*

With this approach, a customer with the CM/Track user profile *cmtrack_basiccustomers* can only see and add comments to tickets from those queues. Another user profile might have access to *Sales* tickets and/or to an *FAQ* queue.

Defining the Customer Fields for CM/Track Login and Password

The fields for login and password for a customer are regular customer fields at the contact level. Please see section Setting Up the Customer Data Model for an introduction to customer field management and GUI configuration for customer data.

Edit the fields which contain the customer data (if there are two levels: **not** the company level, but the contact level!):

- One field for the login has to be created, annotation `username` = “true”.

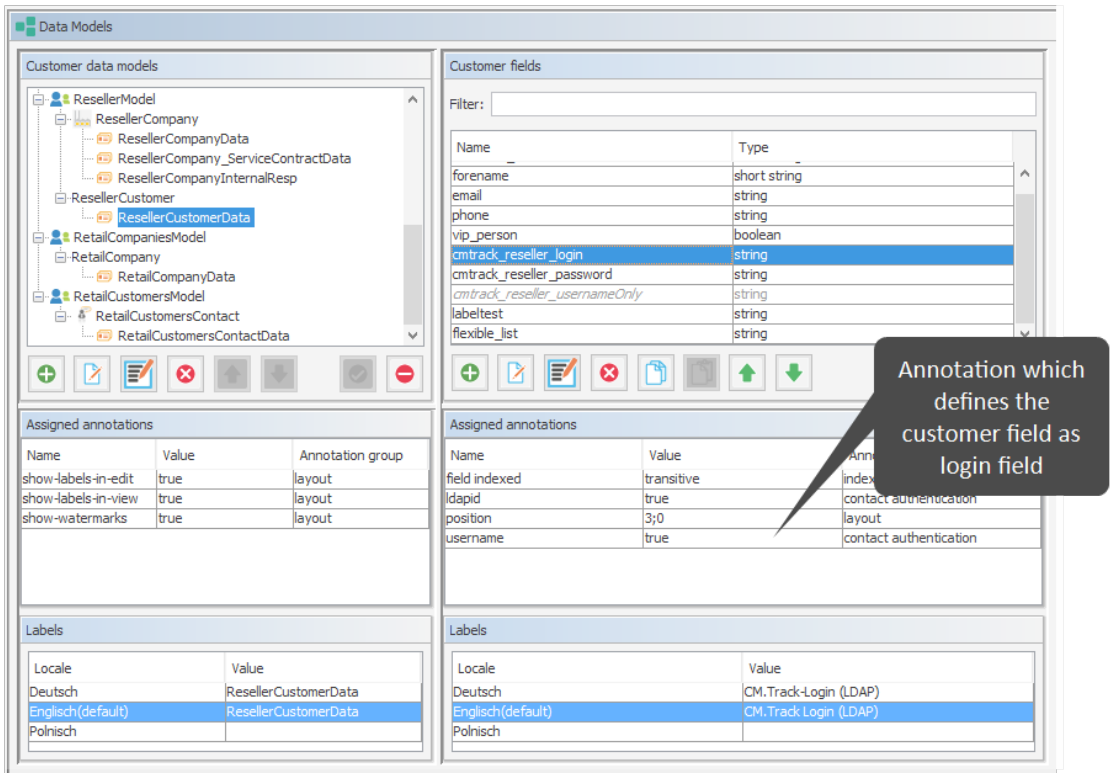


Figure 683: *ConSol CM Admin Tool - Customers, Data Models: Annotation for login for CM/Track*

- One field for the password has to be created, annotation `password` = “true”. The annotation `text-type` = “password” guarantees that only stars/dots are displayed in the Web Client, not the clear text password.

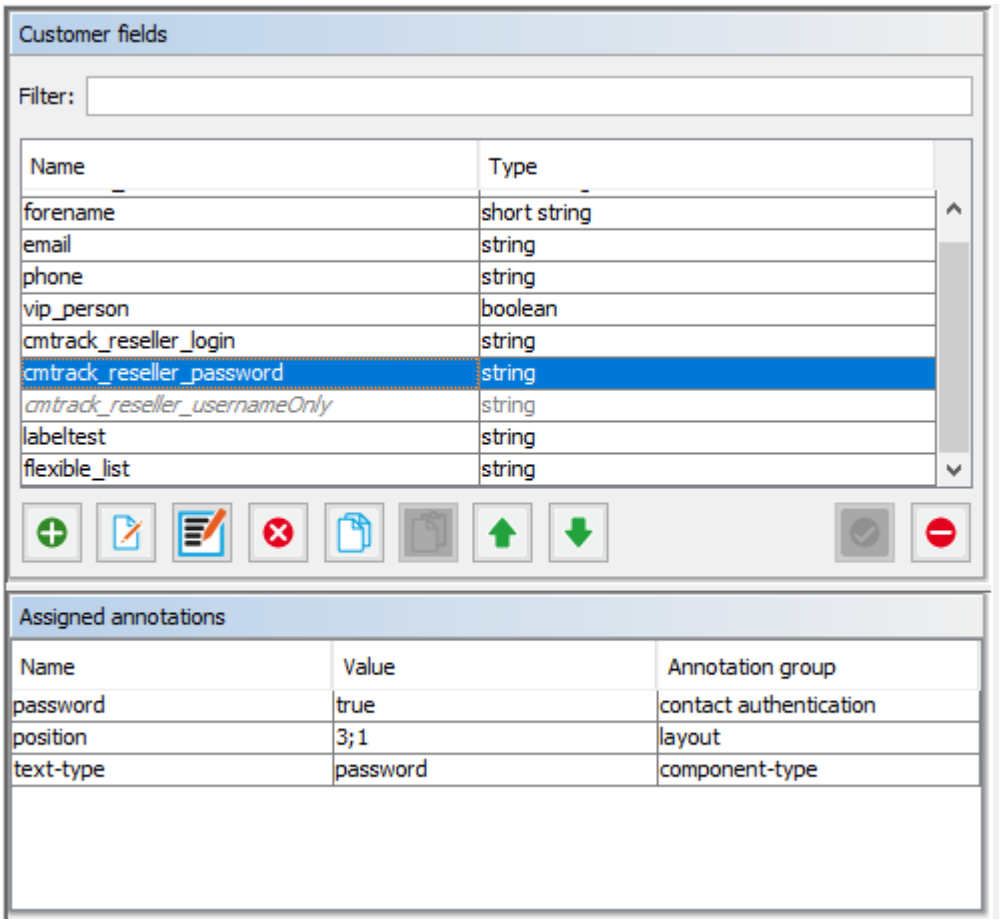


Figure 684: *ConSol CM Admin Tool - Customers, Data Models: Annotation for password for CM/Track*

Granting Access to CM/Track for Customers

The engineer working with the Web Client can then assign a user name, initial password, and a CM/Track user profile to every customer who should have access to the portal CM/Track.

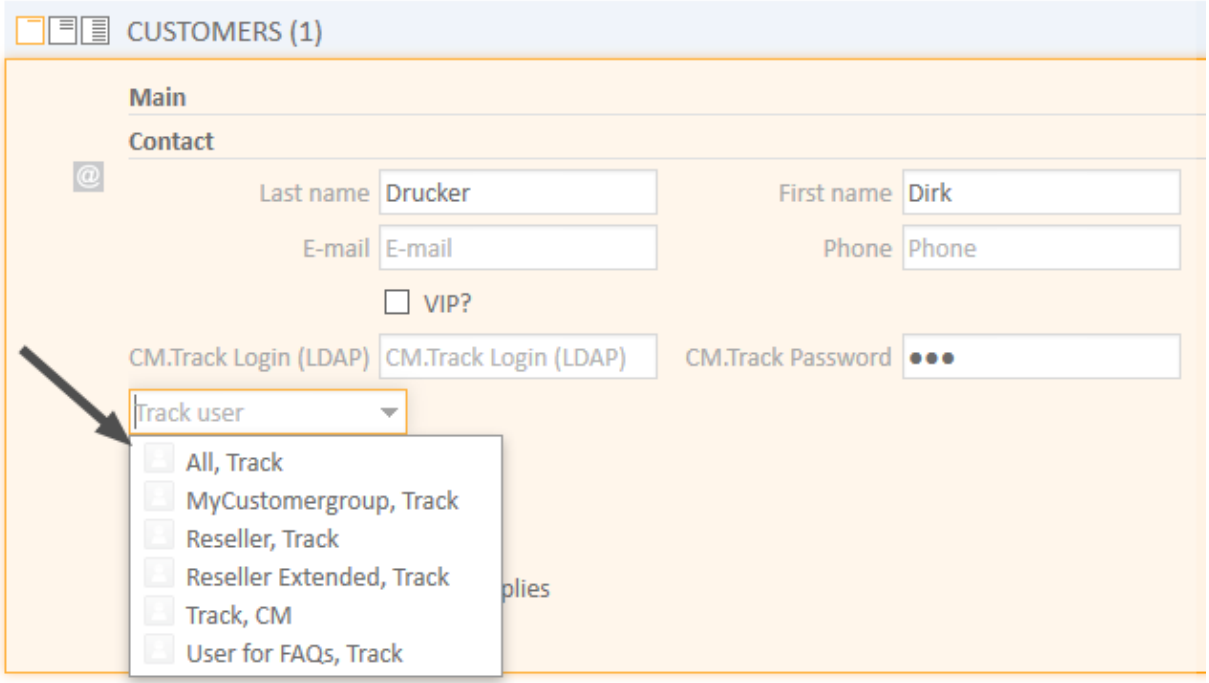


Figure 685: *ConSol CM Web Client - CM/Track users*

Customer Login to the System

Then customers can log in to the system and see their tickets. Please refer to the *ConSol CM User Manual*, section *CM/Track* for a detailed explanation on how to work with ConSol CM as a customer.

There are two mechanisms for performing user authentication:

- simple authentication
- LDAP authentication

Please refer to section Authentication Methods for Customers in CM/Track for details.

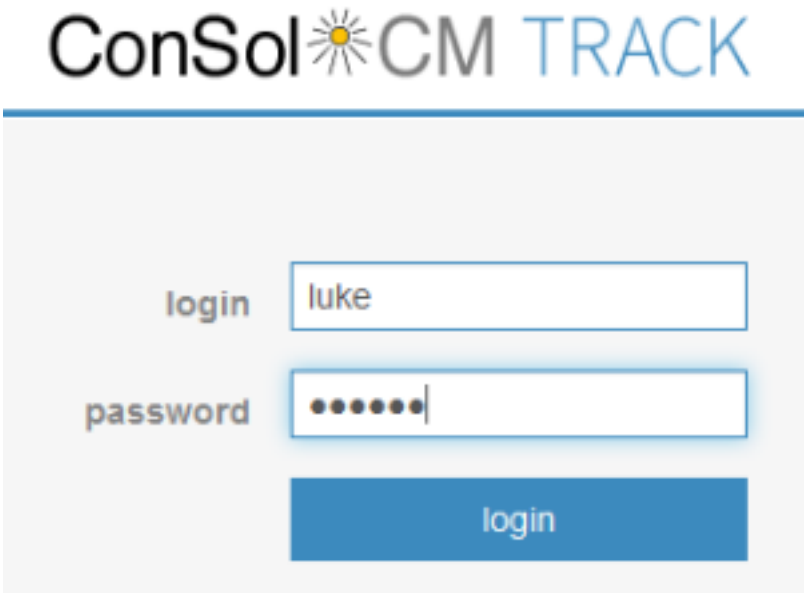


Figure 686: *ConSol CM/Track - Customer login*

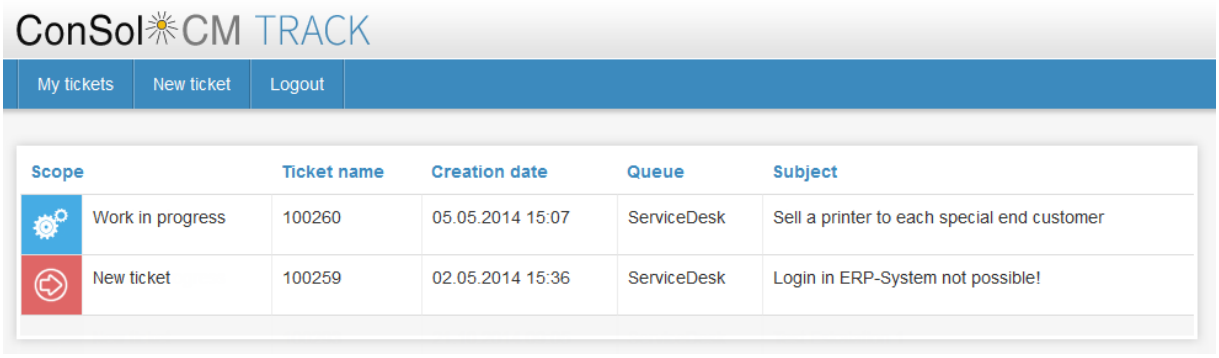


Figure 687: *ConSol CM/Track - Ticket list*

Extended Customer Permissions to See Company Tickets

In some cases it might be required that customers log in to the ConSol CM portal CM/Track and have to have access not only to their personal tickets but to all tickets of their company. In this case, the role for the CM/Track user (user profile) should be assigned the permission *Access tickets of the own company* under *Track User Permissions*. Please refer to the section Role Administration for a detailed explanation.

Configure CM/Track for Password Reset by Customers

Starting with ConSol CM version 6.10.0, CM/Track can be configured to offer a hyperlink for customers where a customer can reset his password. This is based on the template `track-password-reset-template`. Please refer to section Password Reset Template for Customers Using CM/Track V1 for a detailed explanation.

The password reset in CM/Track is only possible when the DATABASE mode is used. It is not possible when LDAP authentication is in operation. See section Authentication Methods for Customers in CM/Track for an explanation of all possible authentication modes.

G.3.3.2 CM/Track V1: FAQs in CM/Track

Introduction to FAQs in CM/Track

If you use CM/Track as a portal where your customers can access their tickets or the tickets of their company, you might consider offering an FAQ (Frequently Asked Questions) search to this clientele. This has proven very helpful in help desk or service desk environments where customers can check if the problem they face has occurred before and if there is a known solution. They only need to contact the service desk and/or open a new ticket if they do not find any help in the FAQ, saving time for both customer and the service team. It might also be employed in other environments where you would like to offer this service.

In ConSol CM, every FAQ is treated as a ticket. Any queues which should be available as FAQ queues via CM/Track have to be defined as special FAQ queues because customers are usually allowed to see only their own tickets or tickets from their company, but FAQ tickets do not belong to any specific customer. They can be accessed by every customer who logs in with a user profile that has access to the FAQ queues. Here, only read access has to be granted.

Configuring the ConSol CM System to Allow FAQ Search in CM/Track

As a first step you have to create an FAQ workflow (please see the *ConSol CM Process Designer Manual* for details) and create an FAQ queue that is marked as a queue for frequently asked questions (checkbox *FAQ*). In addition, you have to assign at least one class of text which has the option *Customer readable* selected (see Classes of Text for details) to the queue. The comments which should be visible to the customers in CM/Track must be marked with this class of text in the Web Client.

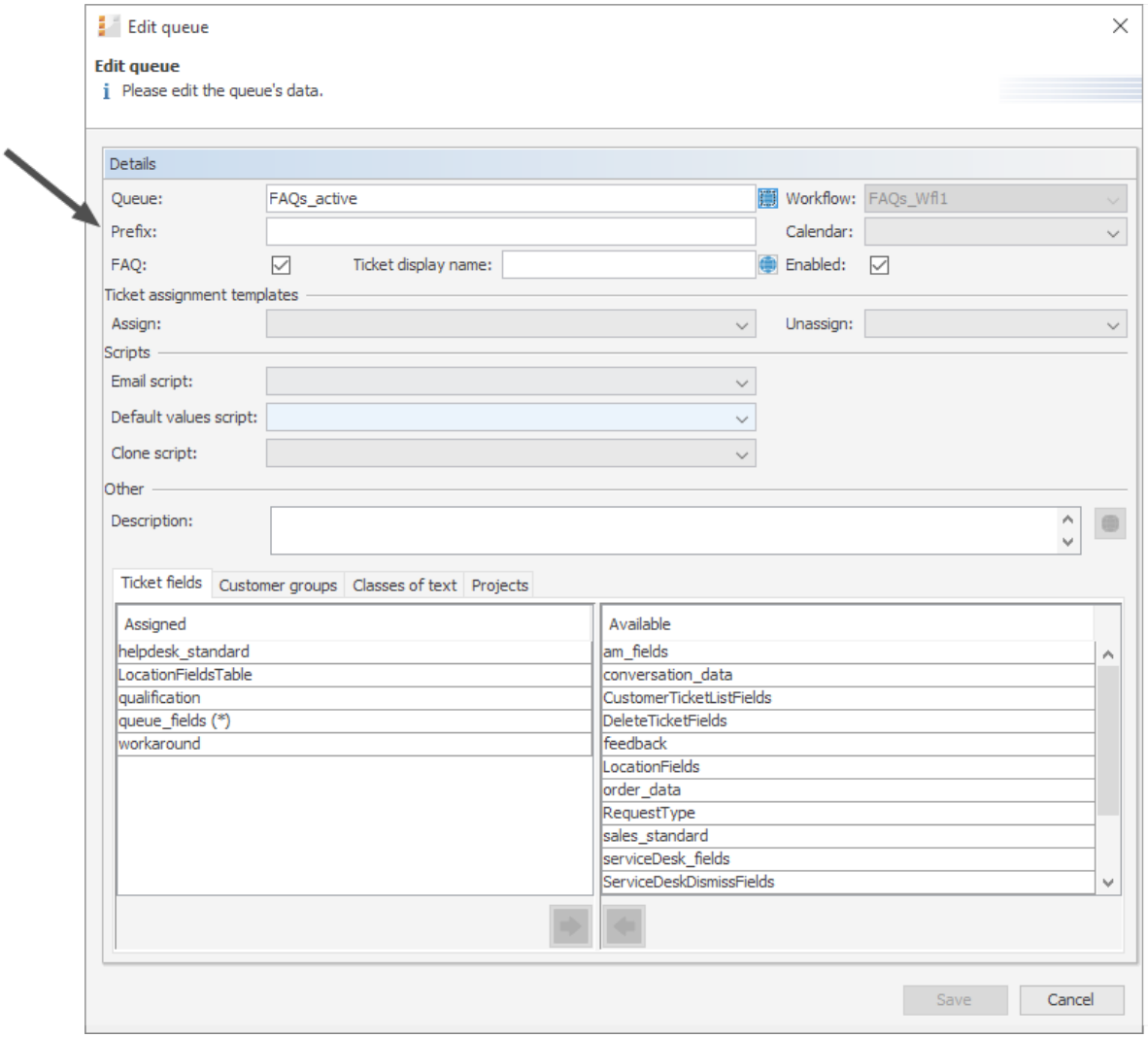


Figure 688: *ConSol CM Admin Tool - Global Configuration, Queues*

Then a role has to be defined which can access the FAQ queue in read-only mode. Please keep in mind that this role also needs read access to the customer group under which you have located the FAQ queue tickets.

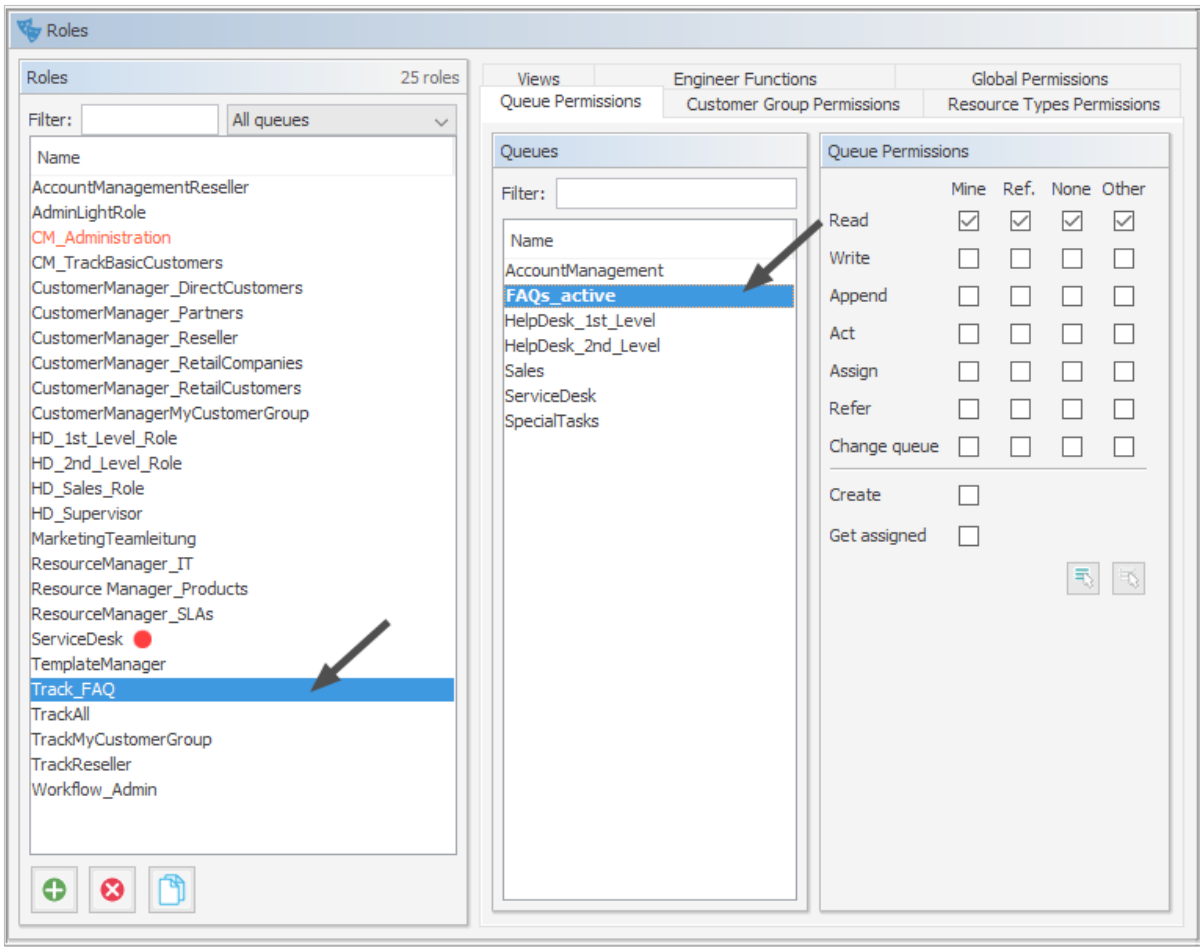


Figure 689: *ConSol CM Admin Tool - Access and Roles, Roles: Role for FAQ*

Then this new role has to be assigned to the user (profile) which is used as CM/Track access user (see section CM/Track V1: System Access for CM/Track Users (Customers)).

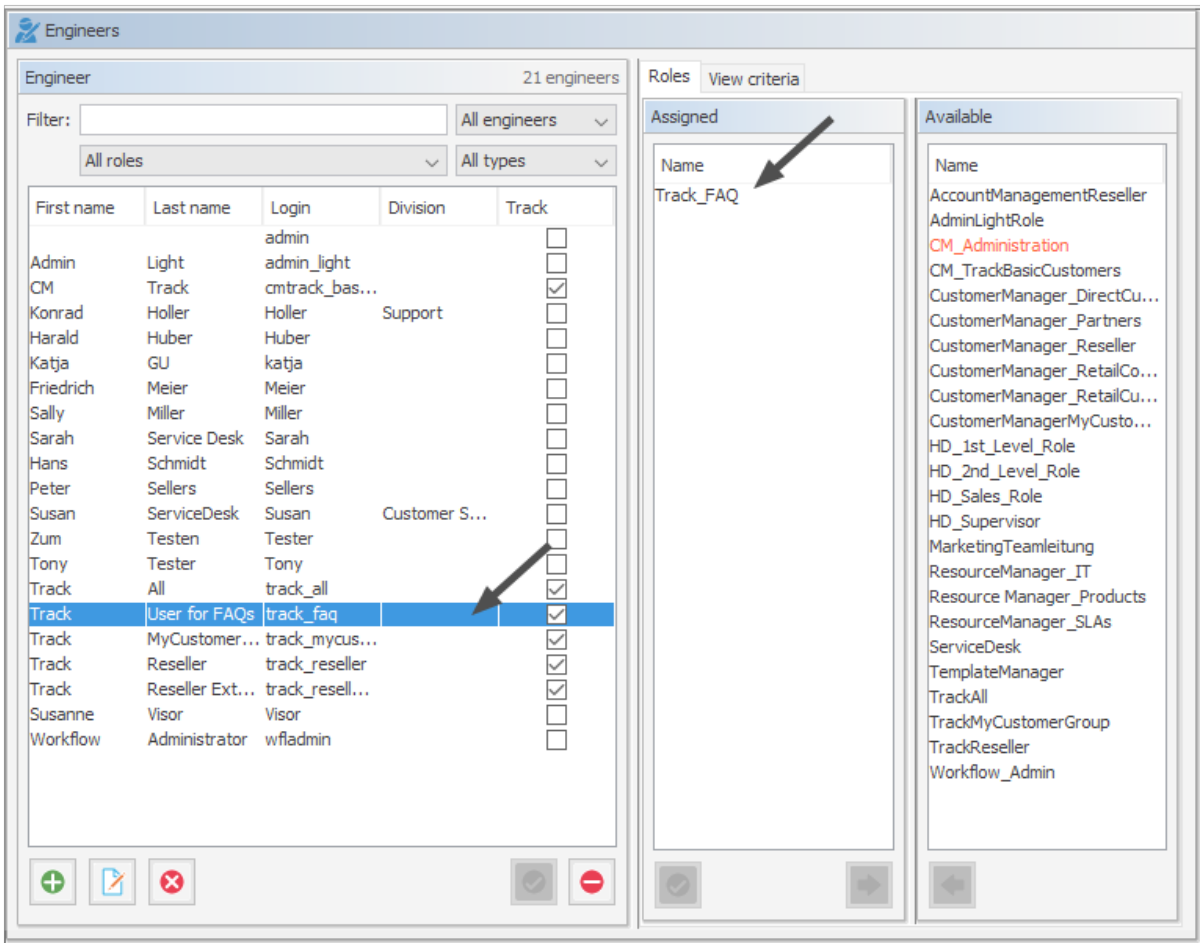


Figure 690: *ConSol CM Admin Tool - Access and Roles, Engineers: Engineer for FAQ*

FAQ Search in CM/Track from a Customer's Point of View

A customer can search the FAQ queue using a search pattern. A list with the search results is displayed. By opening one ticket from the list, the fields of the tickets are displayed. Results might include a solution, as in the following example, or other service information.

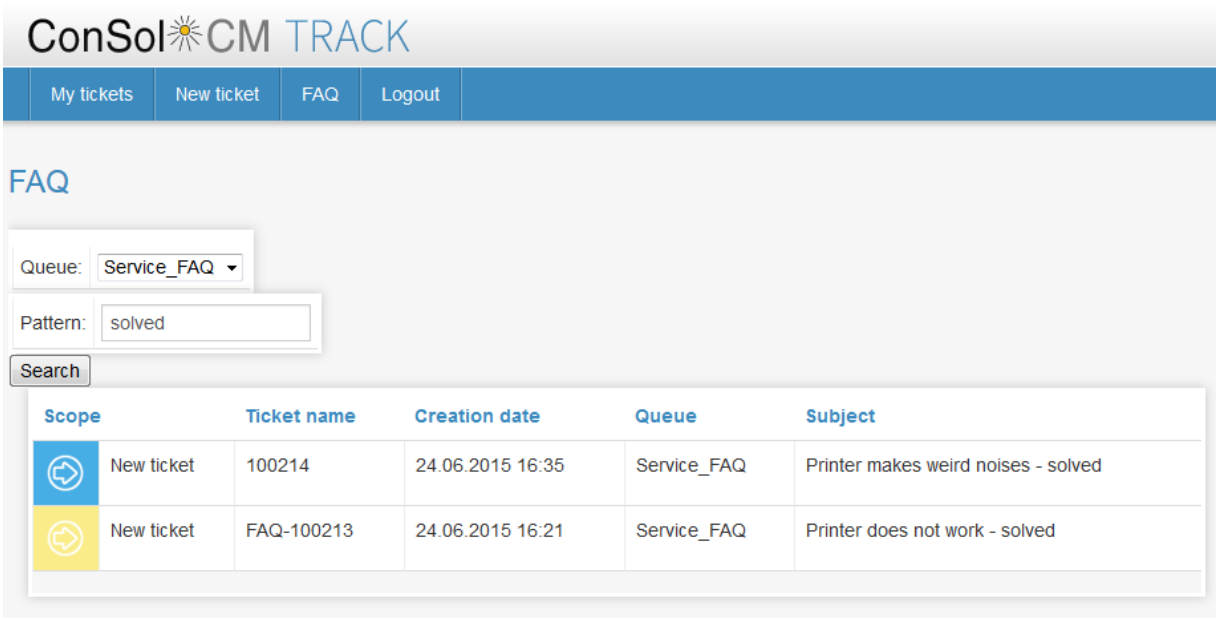


Figure 691: *ConSol CM/Track - Example for FAQ search (1)*

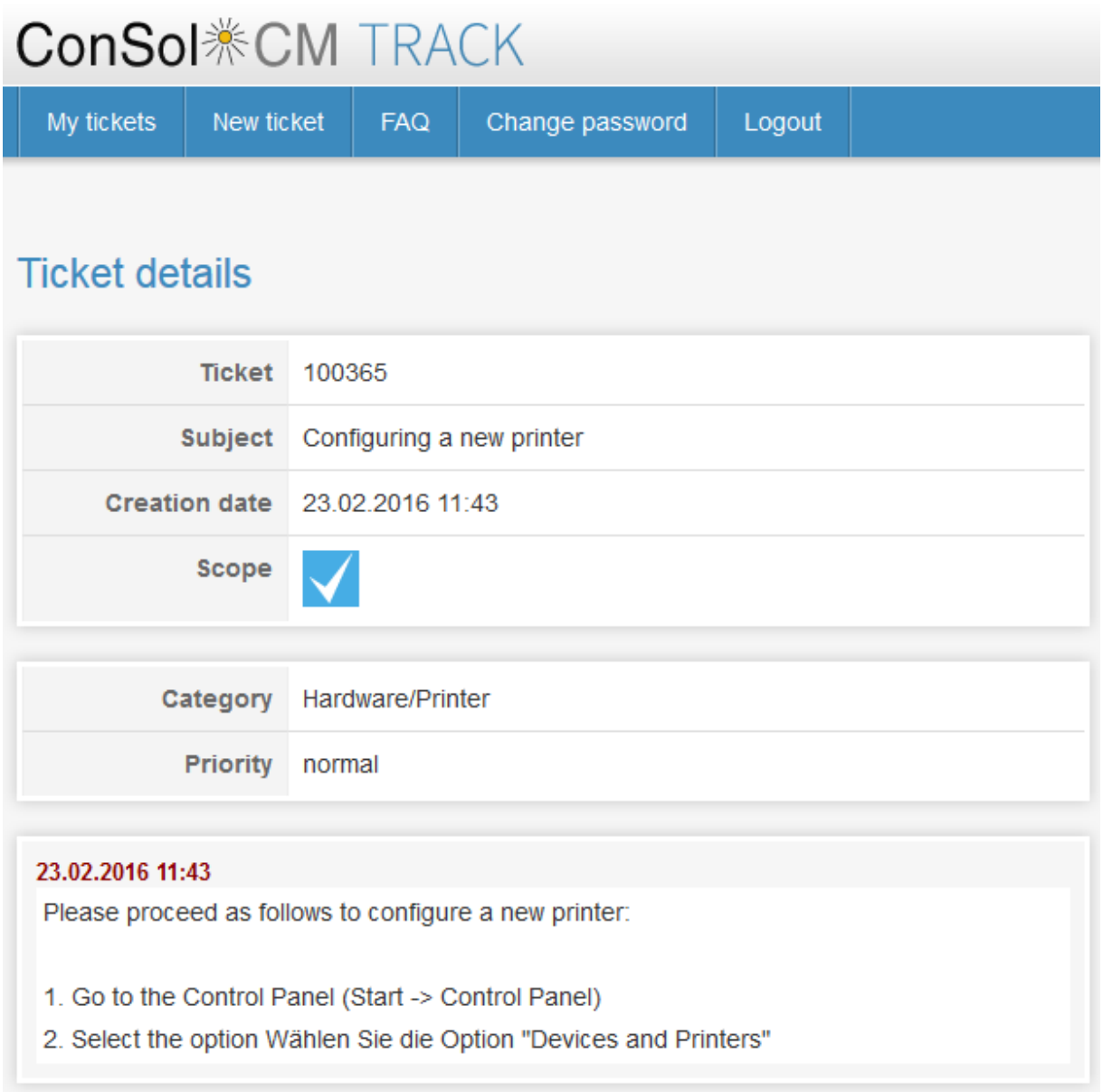


Figure 692: *ConSol CM/Track - Example for FAQ search (2)*

More Complex Solutions for Managing FAQs

Using Two FAQ Queues: FAQ Management and Active FAQs

Instead of using only one FAQ queue, two queues might be used. One can be an FAQ management queue where tickets can be placed manually or be transferred from help or service desk queues. An FAQ manager checks the FAQ and edits the ticket if required. Then the ticket is placed in the queue for active FAQs. Here it can be accessed by customers. After a certain period of time, or when the FAQ manager decides the FAQ should no longer be available, it is transferred back to the FAQ management queue. It may later be re-activated or closed.

Setting Up Two (or More) Parallel FAQ Environments Using Track Users

By creating more than one FAQ queue (or a pair of FAQ queues) and creating the respective CM/Track user profiles, it is possible to provide FAQs for different customer groups. For example, for one customer group technical help desk questions and answers are provided, whereas for the other customer group support and update information is provided. Of course, there can also be a CM/Track user profile which has access to both FAQ environments.

G.3.4 CM/Track V2

See the following sections for topics which concern CM/Track V2:

- Configuring the Welcome Page of CM/Track V2:
  see section Configuration of the Welcome Page
- Customer authentication methods in CM/Track (both V1 and V2):
  see section: Authentication Methods for Customers in CM/Track.
- General system access to CM/Track V2 for customers:
  see section: CM/Track V2: System Access for CM/Track Users (Customers).
- Using the portal for FAQ:
  see section: CM/Track V2: FAQs in CM/Track.
- Managing the availability of ticket fields in CM/Track V2:
  see section: CM/Track V2: Data Availability For Customers

G.3.4.1 CM/Track V2: System Access for CM/Track Users (Customers)

In the following chapter you will find detailed information about how to configure your ConSol CM system to grant access to CM/Track V2 (the ConSol CM portal) to your customers.

> CM/Track V2 is a ConSol CM add-on which has to be purchased separately.
>
> Please note that for every CM/Track user (i.e., user profile), a ConSol CM license is required. Since numerous customers can log in to CM/Track using one user profile, you do not need a ConSol CM license for every customer. Please refer to section License Management for details.

Precondition

CM/Track V2 is not part of every default shipment of ConSol CM. When you (your company) have purchased the license, you will receive a .war file which has to be deployed in the application server. Some minor modifications then have to be made in CM configuration files in the application server in order to operate CM/Track V2. This is all are explained in the *ConSol CM Setup Manual*.

The default function set of CM/Track provides basic functionalities (e.g., viewing a ticket list, creating a new ticket, seeing ticket details) and the pages have a CM standard layout. In order to use CM/Track as a powerful portal for customer access to the system, the layout should be adapted to a company's CD (corporate design), a process called *skinning*. The forms and lists which are displayed for the customer might be modified and/or extended. Please contact our consulting team or your account manager if you would like to adjust CM/Track for your company in an optimal way.

CM/Track V2 Technical Background

The portal CM/Track is based on the ***REST API*** of ConSol CM. Please refer to the separate document *ConSol CM REST API Documentation* for details.

General Principle of System Access via CM/Track V2

A customer who wants, or should, have access to your ConSol CM system using the portal CM/Track has to have a login and a password. Both can be initially provided by the engineer who edits the customer data using the ConSol CM Web Client, or the values can be imported automatically into the database.

The fields for the login and password of customers are customer fields which are defined like any other customer field and which have special annotations. If you are not familiar with customer fields, please refer to section Customer Field Management and GUI Design for Customer Data.

The access permissions of the customer are defined by assigning a CM/Track user profile to the customer's account. The CM/Track user profiles are managed by the ConSol CM administrator using the Admin Tool.

Defining the User Profiles/Access Permissions for CM/Track V2

As one of the first steps, you have to define CM/Track user profiles, i.e., profiles of access permissions to CM/Track. A CM/Track user profile is defined like a regular engineer (please see section Engineer Administration for details), but is marked as *Track*.

The following example shows some CM/Track user profiles in the Admin Tool. You reach this screen by opening the navigation group *Access and Roles*, navigation item *Engineers*.

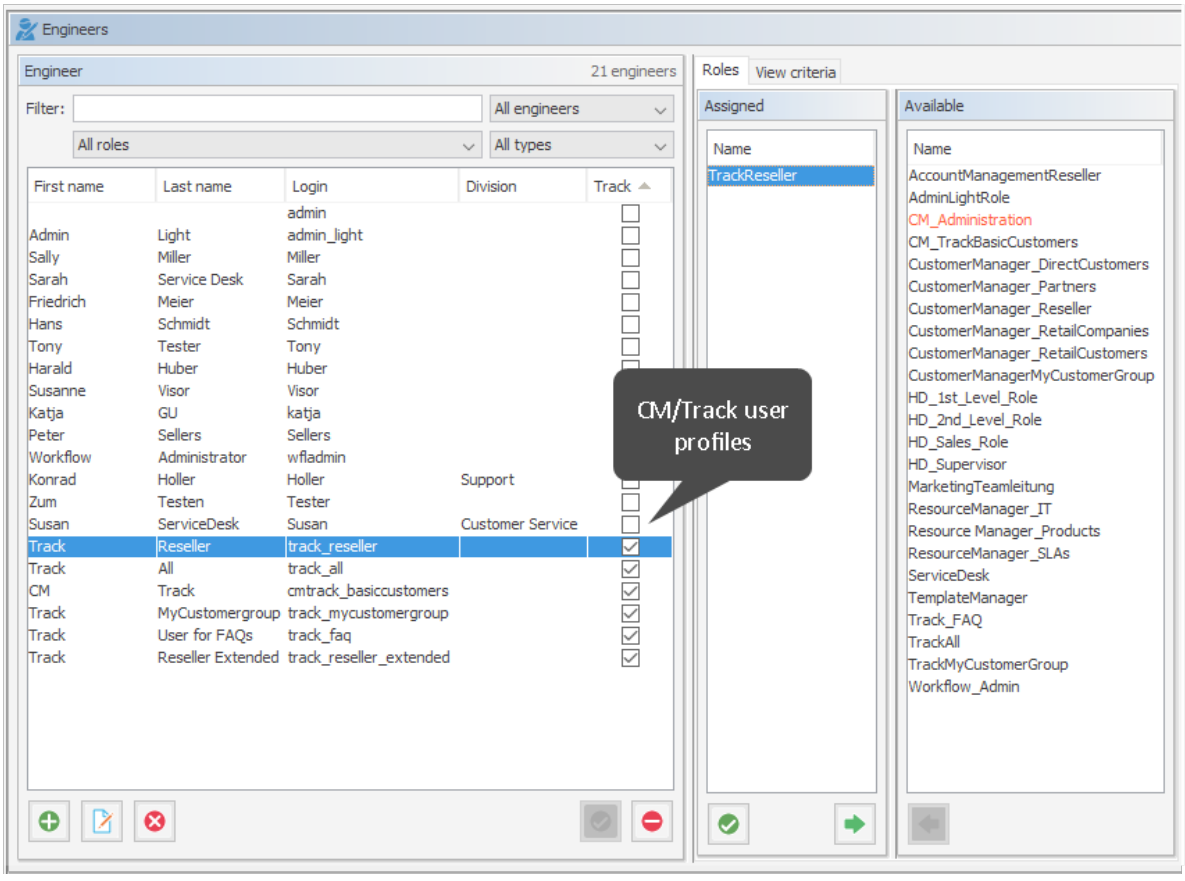


Figure 693: *ConSol CM Admin Tool - Access and Roles, Engineers: User profile name for CM/Track*

One or more roles are then assigned to the user profile to define the access permissions to queues and customer groups. For example you can set up a user profile (engineer) *track_reseller* that has the role *TrackReseller*. This role has read/write/append permissions for the queues *FAQs_active*, *Helpdesk_1st_Level*, *Helpdesk_2nd_Level*, *ServiceDesk*, and *SpecialTasks* and has customer group permissions for one customer group.

Please note that

- always queue **and** customer group permissions have to be granted to allow ticket access via CM/Track for customers.
- you must assign matching queue and customer group permissions, i.e., assign queues where the respective customer groups have been assigned (see section Queue Administration).
- read permissions for customer groups will be sufficient in most (standard) cases, since it is not possible to edit customer data using the portal.
- write permissions for the “own” customer group are required for the customer to be allowed to reset his password.

In our example, the role *TrackReseller* has read and write access to the *Reseller* customer group. You reach the following screens by opening the navigation group *Access and Roles*, navigation item *Roles*. In your system, it might be required to create different CM/Track roles with access to different customer groups. For a detailed introduction to role administration, please refer to section Role Administration.

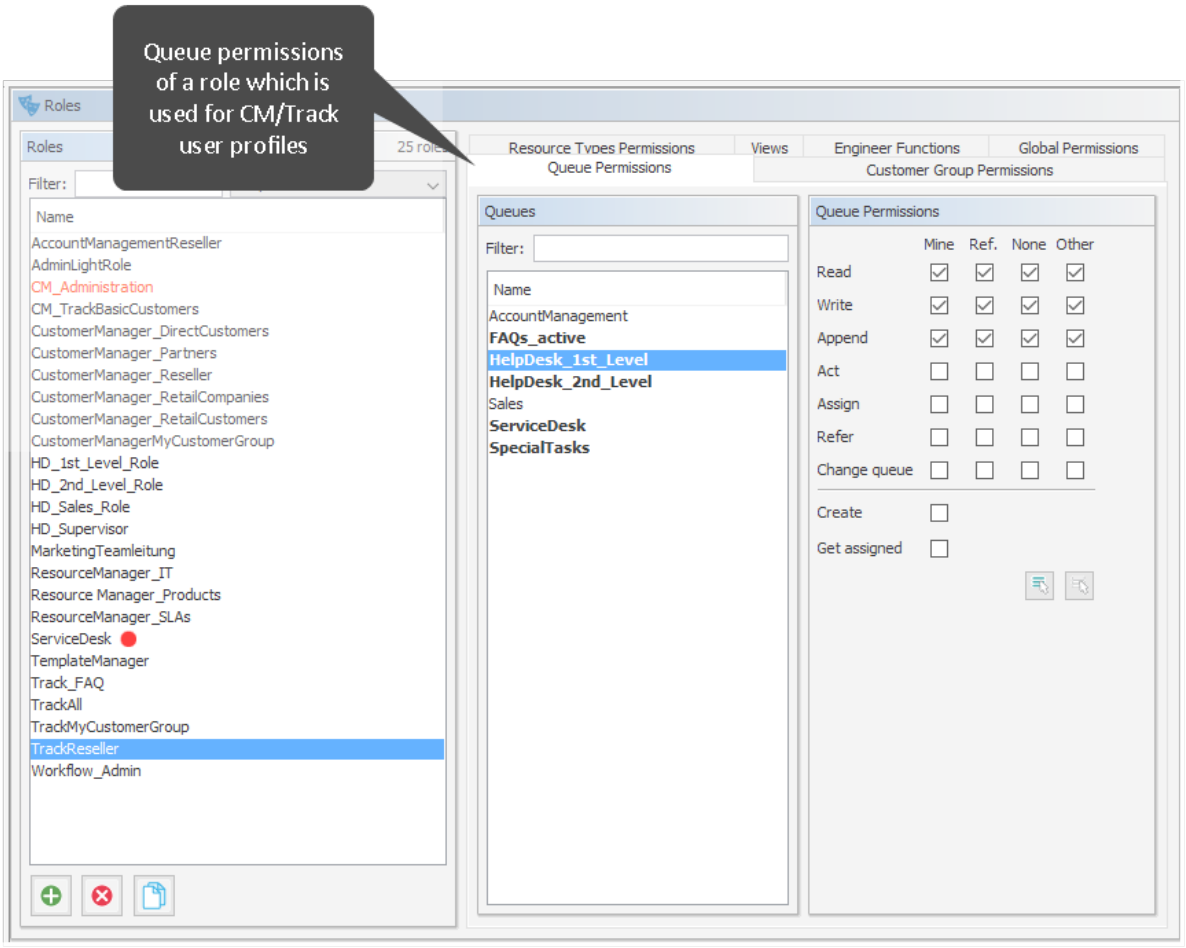


Figure 694: *ConSol CM Admin Tool - Access and Roles, Roles: User profile for CM/Track, queue permissions*

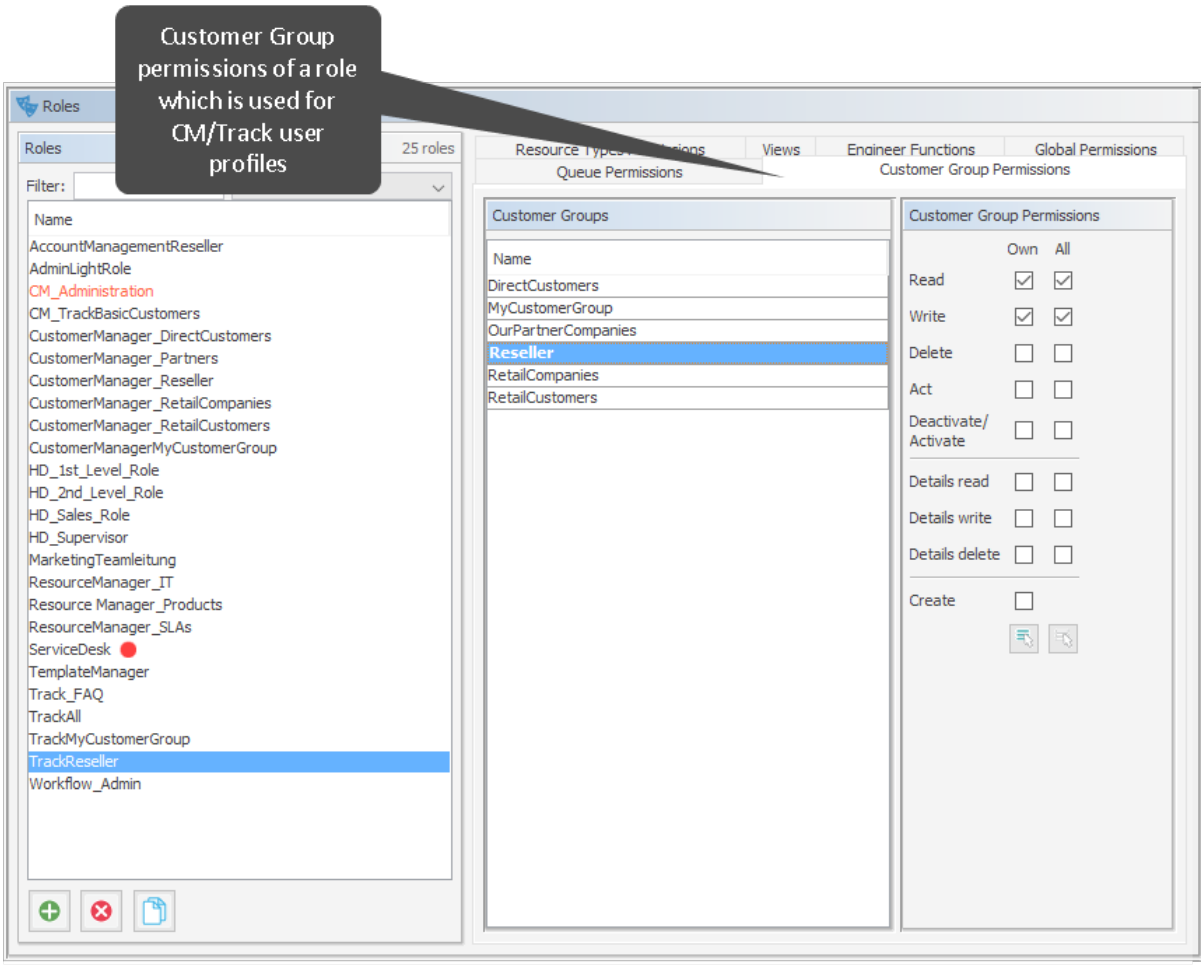


Figure 695: *ConSol CM Admin Tool - Access and Roles, Roles: User profile for CM/Track, customer group permissions*

With this approach, a customer with the CM/Track user profile *TrackReseller* can only see and add comments to tickets from those queues. Another user profile might have access to *Sales* tickets and/or to an *FAQ* queue.

Defining the User Assignment Mode

The user assignment mode is defined for each customer group and defines the system behavior concerning the way of assigning a CM/Track user profile to a contact. Open the customer group edit panel: in the navigation group *Customers*, navigation item *Customer Groups*, select a group and click the *Edit* button. In the last line of the panel, the CM/Track user assignment can be made.

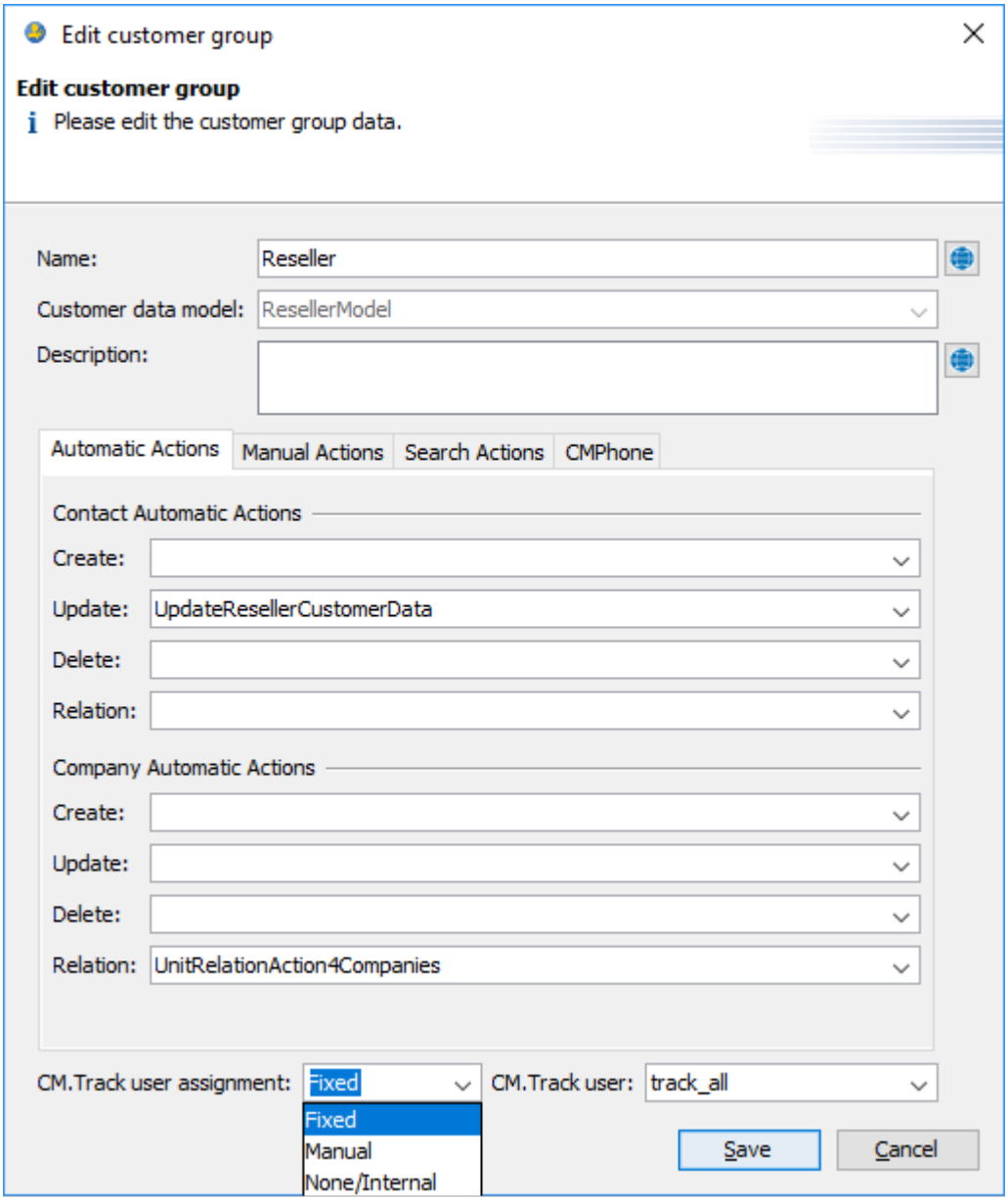


Figure 696: *ConSol CM Admin Tool - Customers, Customer Groups: Defining the CM/Track user profile mode for a customer group*

Three modes are available for the assignment mode of the CM/Track user profiles:

- **Fixed**
  A CM/Track user profile (i.e. a ConSol CM engineer object which has been defined as CM/Track user profile, see section Defining the User Profiles/Access Permissions for CM/Track V2) is selected in the pull-down menu *CM/Track user*. This CM/Track user profile is used for all contacts of this customer group. The data fields which are potentially available in the Web Client to assign a CM/Track user profile to a contact (see section Granting Access to CM/Track V2 for Customers) are not displayed in the Web Client.

> ⚠ Please be aware that the assignment mode “Fixed” cannot be changed anymore as soon as the customer group has contacts!

- **Manual**
  Default. The assignment of a CM/Track user profile is done manually by an engineer in the Web Client as described in section Granting Access to CM/Track V2 for Customers. The user profile can also be set using the REST API.
- **None/Internal**
  The association of a CM/Track user profile is not allowed in any client and can only be done by script, if desired. There will be no choice available in the Web Client, and an attempt to change it via REST API will also return a status message METHOD NOT ALLOWED.

Defining the Customer Fields for CM/Track V2 Login and Password

The fields for login and password for a customer are regular customer fields at the contact level. Please see section Setting Up the Customer Data Model for an introduction to customer field management and GUI configuration for customer data.

Edit the fields which contain the customer data (if there are two levels: **not** the company level, but the contact level!) as demonstrated in the following example. You reach the following screen by opening the navigation group *Customers*, navigation item *Data Models*.

- One field for the login has to be created, annotation `username` = “true”.

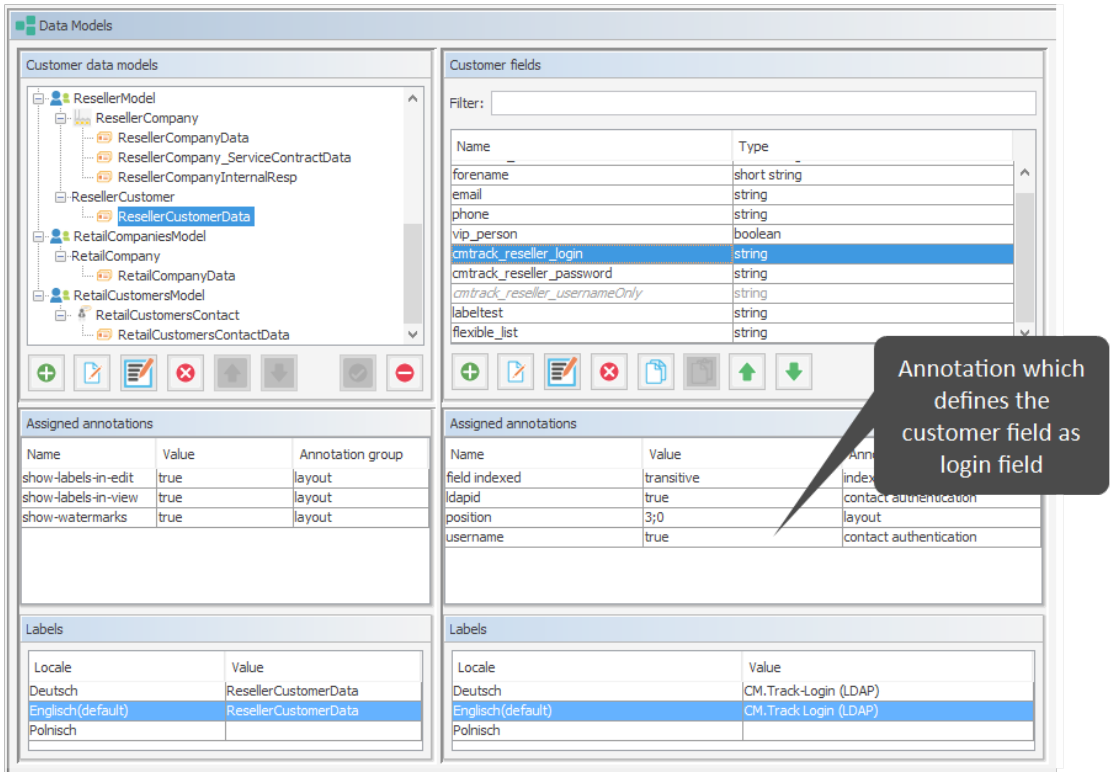


Figure 697: *ConSol CM Admin Tool - Customers, Data Models: CM/Track - Annotation 'username' for login*

> ⚠ Assigning the annotation `username` to a customer field is only possible, if there is no previous assignment of this annotation. Otherwise it will be prohibited. When assigning it, a warning dialog must be confirmed before it is executed, since it can be a longer running operation. Un-assigning the annotation must be confirmed as well, because it cannot be undone: Un-assignment delete the username values unrecoverably from internal storage.

- One field for the password has to be created, annotation `password` = “true”. The annotation `text-type` = “password” guarantees that only stars/dots are displayed in the Web Client, not the clear text password.

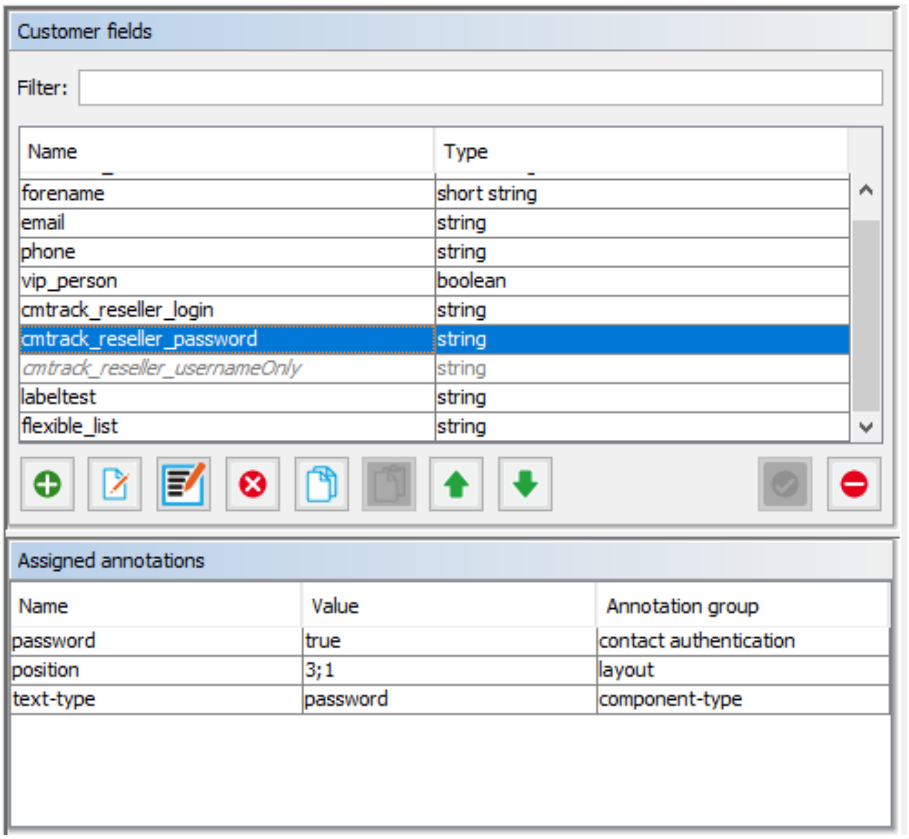


Figure 698: *ConSol CM Admin Tool - Customers, Data Models: CM/Track - Annotation for password*

> ⚠ The annotation `password` requires confirmation when assigned.
>
> In case of an update from CM versions lower than 6.11 to 6.11 and up: When this annotation is set, the system reads the plain text passwords from the original field values, encrypts them and saves the encrypted values to the internal storage. Then the original field values are deleted and thus the plain text value cannot be recovered anymore.
>
> When trying to un-assign the password annotation the operation must be confirmed as well, since the encrypted passwords are deleted from the internal storage. After the annotation unassignment the password information is completely lost and cannot be recovered at all.

When a scenario from a CM version lower than 6.11 is imported into a system with CM 6.11 (or higher), a transformation of user names and passwords is performed automatically. This is described in detail in section Transformation of User Name and Password Fields During Import into CM 6.11.

Granting Access to CM/Track V2 for Customers

For all customer groups where the CM/Track user profile assignment mode is set to “Manual”, the engineer working with the Web Client can then assign a user name, initial password, and a CM/Track user profile to every customer who should have access to the portal CM/Track. The user name has to be unique. This is checked by the system. You cannot enter a name a second time if this has already been assigned to another customer. The password is stored as encrypted string in the CM database. This means that an engineer can set a new password, e.g., when a customer calls and asks for this, but it is never possible to read the password from the system.

> You, as an administrator, can define if the CM/Track user names should be case sensitive. Use the CM system property cmas-core-security, policy.track.username.case.sensitive. This is a boolean variable. When it is set to “true”, the CM/Track user names are case sensitive. Please make sure that the database collation which is in use supports case sensitive strings!

The following example shows the customer data of an example contact in the ConSol CM Web Client. You reach this screen by opening a contact data set in edit mode.

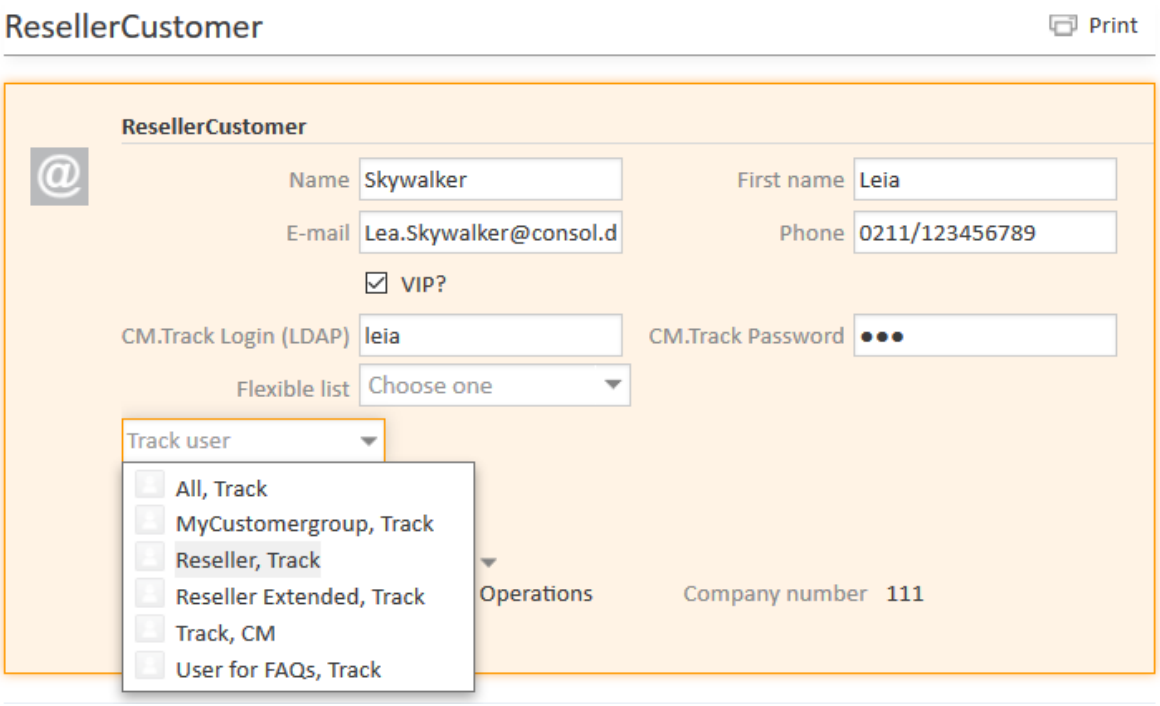


Figure 699: *ConSol CM Web Client - Contact page: CM/Track user data*

Customer Login to the System

Then customers can log in to the system and see their tickets. Please refer to the *ConSol CM User Manual*, section *CM/Track* for a detailed explanation on how to work with ConSol CM as a customer.

There are two mechanisms for performing user authentication:

- simple authentication
- LDAP authentication

Please refer to section Authentication Methods for Customers in CM/Track for details.

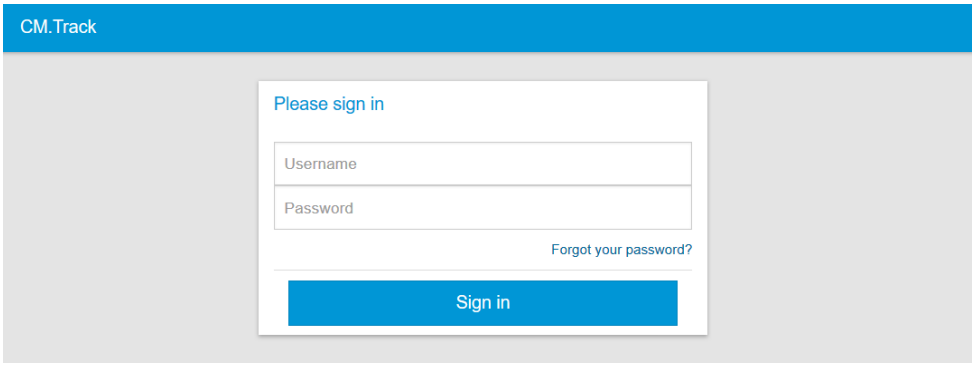


Figure 700: *ConSol CM/Track - Customer login*

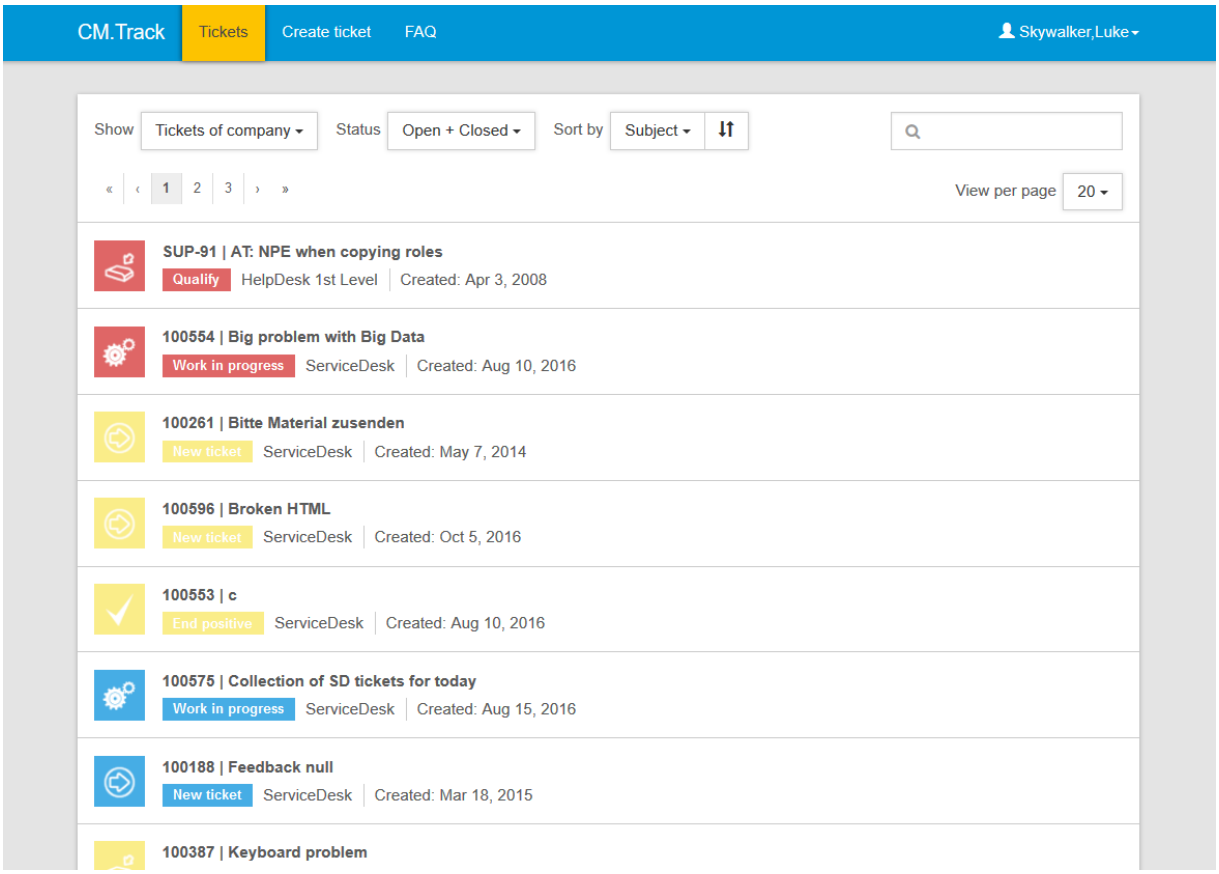


Figure 701: *ConSol CM/Track - Ticket list*

Extended Customer Permissions to See Company Tickets

In some cases it might be required that customers log in to the ConSol CM portal CM/Track and have to have access not only to their personal tickets but to all tickets of their company. In this case, the role for the CM/Track user (user profile) should be assigned the permission *Access tickets of the own company* under *Track User Permissions*. Please refer to the section Role Administration for a detailed explanation.

Configure CM/Track V2 for Password Reset by Customers

Starting with ConSol CM version 6.10.5.4, CM/Track V2 can be configured to offer a hyperlink for customers where a customer can reset his password. This is based on the template `track-password-reset-template`. Please refer to section Password Reset Template for Customers Using CM/Track V2 for a detailed explanation. The password reset in CM/Track is only possible when the DATABASE mode is used. It is not possible when LDAP authentication is in operation. See section Authentication Methods for Customers in CM/Track for the portal for an explanation of all possible authentication modes.

Please note that the `From` address of the email which is sent to a customer who has requested a new password can be set using the CM system property cmas-core-security, password.reset.mail.from, see password.reset.mail.from for details.

G.3.4.2 CM/Track V2: FAQs in CM/Track

Introduction to FAQs in CM/Track V2

If you use CM/Track as a portal where your customers can access their tickets or the tickets of their company, you might consider offering an FAQ (Frequently Asked Questions) search to this clientele. This has proven very helpful in help desk or service desk environments where customers can check if the problem they face has occurred before and if there is a known solution. They only need to contact the service desk and/or open a new ticket if they do not find any help in the FAQ, saving time for both customer and the service team. It might also be employed in other environments where you would like to offer this service.

In ConSol CM, every FAQ is treated as a ticket. Any queues which should be available as FAQ queues via CM/Track have to be defined as special FAQ queues because customers are usually allowed to see only their own tickets or tickets from their company, but FAQ tickets do not belong to any specific customer. They can be accessed by every customer who logs in with a user profile that has access to the FAQ queues. Here, only read access has to be granted.

Configuring the ConSol CM System to Allow FAQ Search in CM/Track V2

As a first step you have to create an FAQ workflow (please see the *ConSol CM Process Designer Manual* for details) and create an FAQ queue that is marked as a queue for frequently asked questions (checkbox *FAQ*). In addition, you have to assign at least one class of text which has the option *Customer readable* selected (see Classes of Text for details) to the queue. The comments which should be visible to the customers in CM/Track must be marked with this class of text in the Web Client.

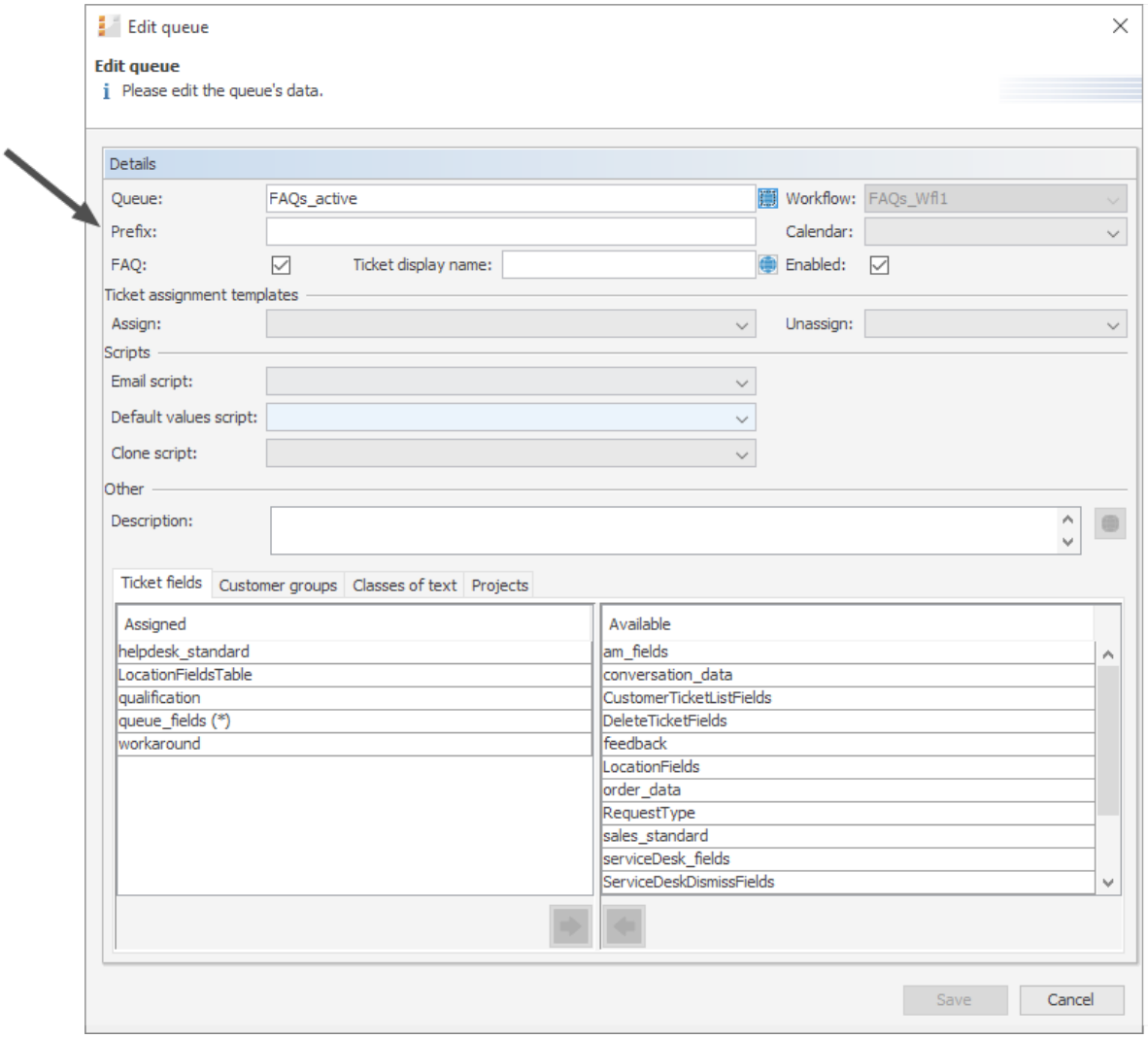


Figure 702: *ConSol CM Admin Tool - Global Configuration, Queues*

Then a role has to be defined which can access the FAQ queue in read-only mode. Please keep in mind that this role also needs read access to the customer group under which you have located the FAQ queue tickets.

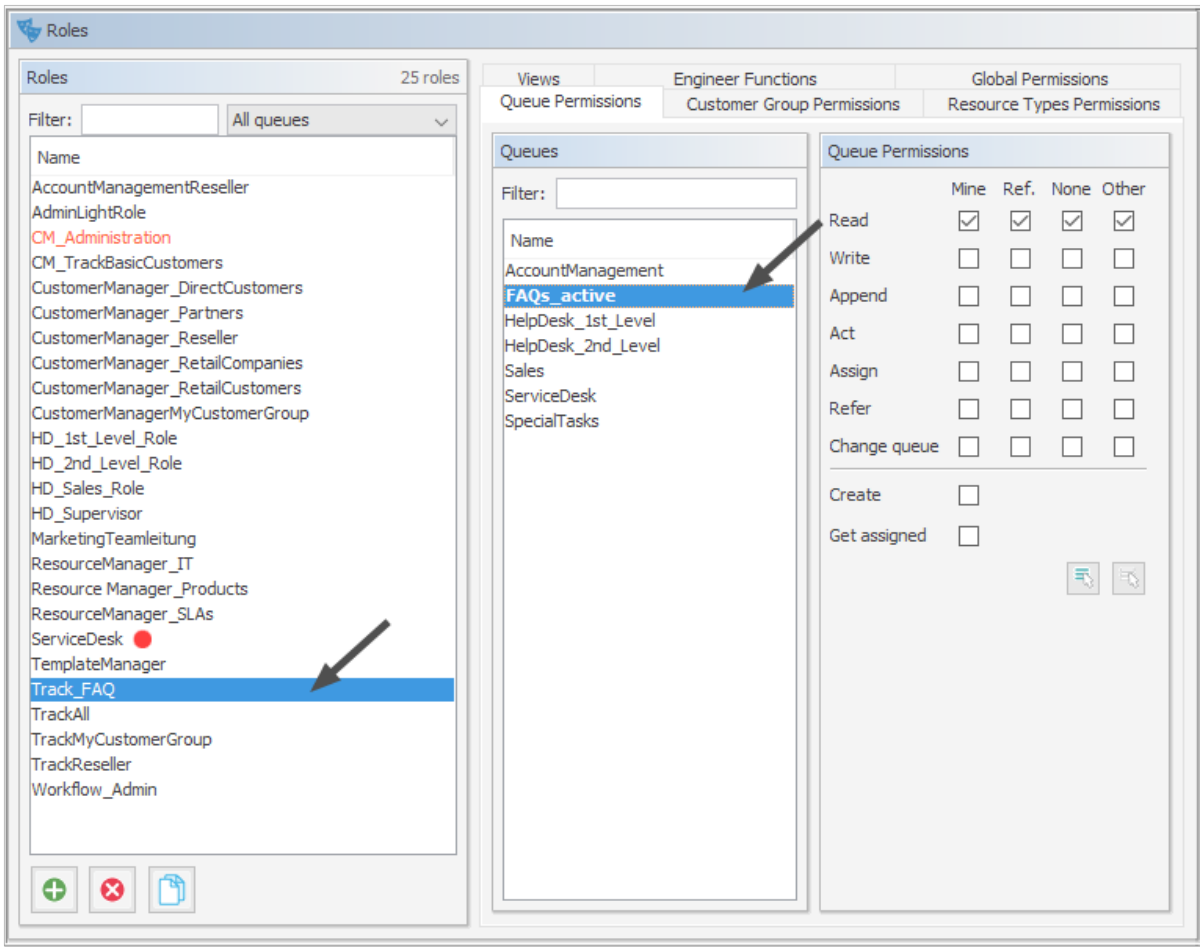


Figure 703: *ConSol CM Admin Tool - Access and Roles, Roles: Role for FAQ*

Then this new role has to be assigned to the user (profile) which is used as CM/Track access user (see section CM/Track V2: System Access for CM/Track Users (Customers)).

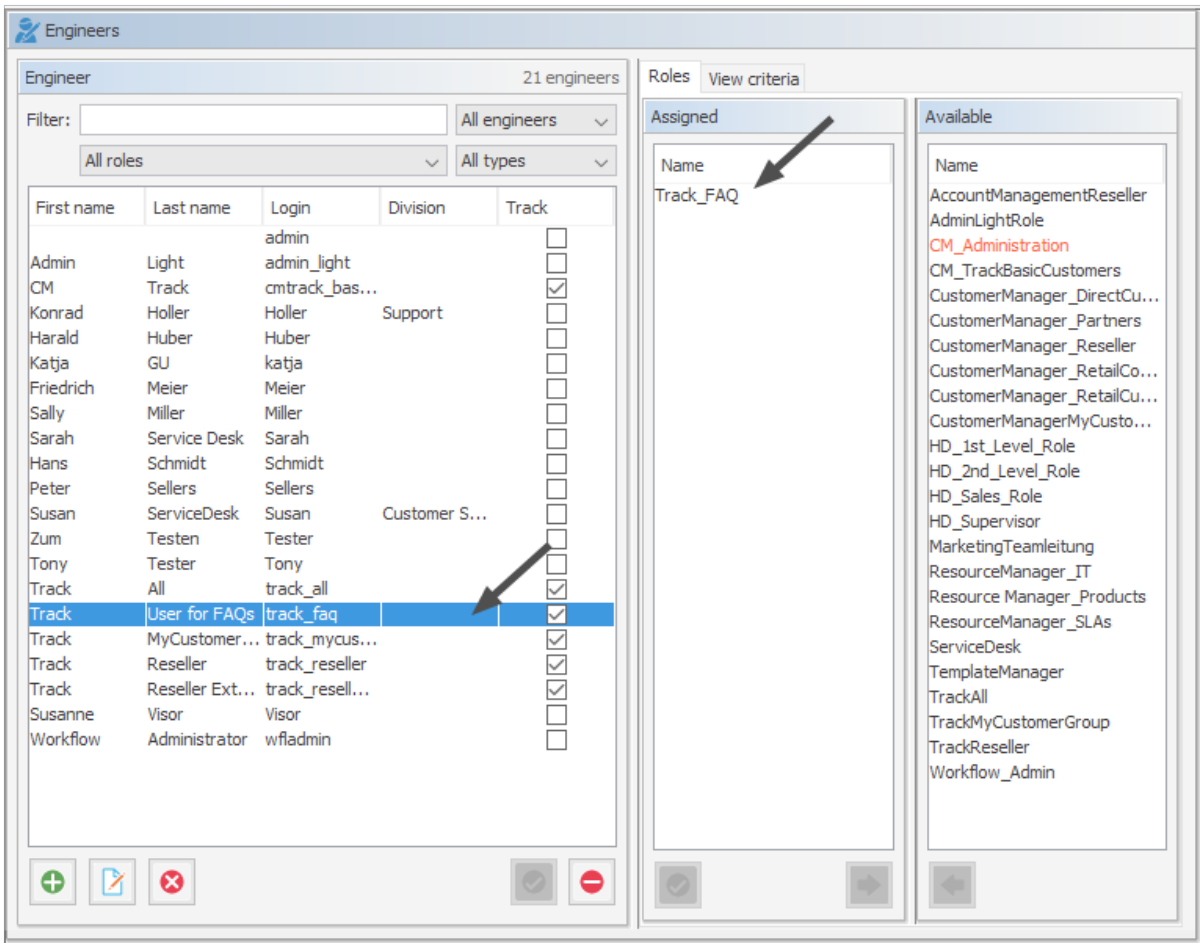


Figure 704: *ConSol CM Admin Tool - Access and Roles, Engineers: Engineer for FAQ*

FAQ Search in CM/Track V2 from a Customer's Point of View

All queues which are marked as *FAQ* are automatically offered as FAQ queues in the portal (see following figure). No further access configuration is required.

Initially, the FAQ queue selector is set to *All FAQ Lists*, thus all tickets from all FAQ queues are displayed. A customer can narrow down the FAQ ticket list by using the string filter option or by selecting a particular queue using the queue selector. By opening one ticket from the list, the fields of the tickets are displayed.

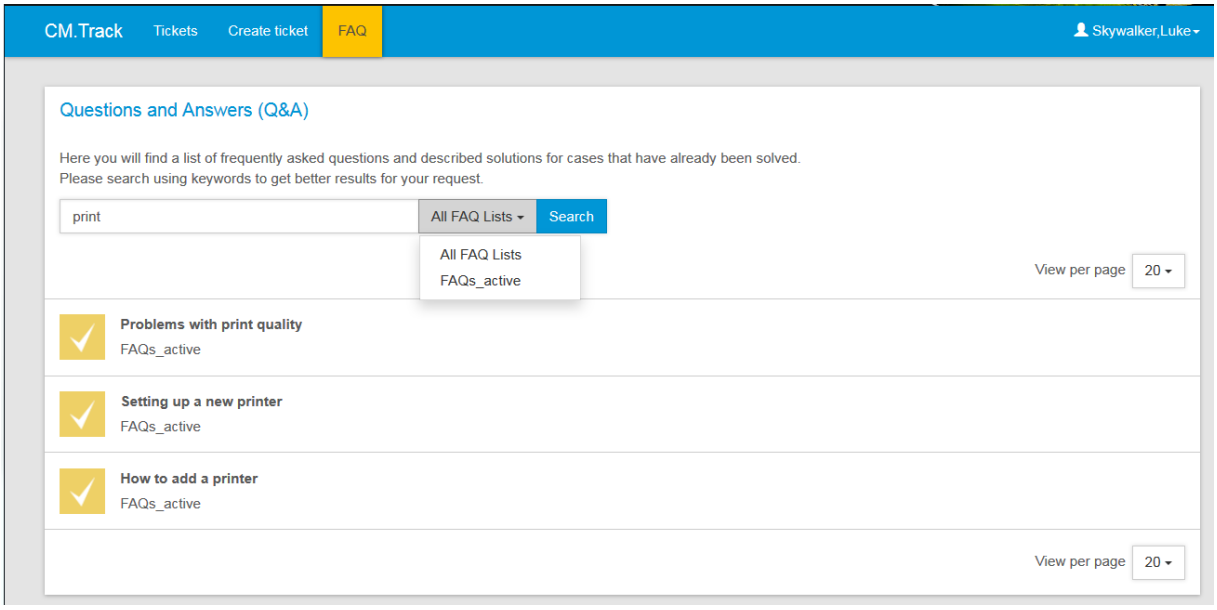


Figure 705: *ConSol CM/Track - Example for FAQ search (1)*

More Complex Solutions for Managing FAQs

Using Two FAQ Queues: FAQ Management and Active FAQs

Instead of using only one FAQ queue, two queues might be used. One can be an FAQ management queue where tickets can be placed manually or be transferred from help or service desk queues. An FAQ manager checks the FAQ and edits the ticket if required. Then the ticket is placed in the queue for active FAQs. Only this active queue is marked as FAQ in the queue management, thus only tickets in this queue can be accessed by customers. After a certain period of time, or when the FAQ manager decides the FAQ should no longer be available, it is transferred back to the FAQ management queue. It may later be re-activated or closed.

Setting Up Two (or More) Parallel FAQ Environments Using Track Users

By creating more than one FAQ queue (or a pair of FAQ queues) and creating the respective CM/Track user profiles, it is possible to provide FAQs for different customer groups. For example, for one customer group technical help desk questions and answers are provided, whereas for the other customer group support and update information is provided. Of course, there can also be a CM/Track user profile which has access to both FAQ environments.

G.3.4.3 CM/Track V2: Data Availability For Customers

Introduction

When your company provides a portal with CM/Track for your customers, it is required that you define a strategy concerning the visibility of ticket data. You might want to keep internal information internal but, on the other hand, inform your customer as well and as detailed as possible about their tickets or service cases.

In this context, you have to think about two topics:

- Which comments (text entries, emails, attachments) should be available to the customer?
- Which data fields (ticket fields) should be visible for the customer?

Both topics will be treated in the following sections.

Which Comments Should Be Available to the Customer?

The visibility of comments, i.e.,

- text entries
- attachments
- emails

is configured using classes of text. All comments which are marked with a class of text which is *customer readable* will be visible for the customer when the detail view of the ticket is displayed. Please see section Classes of Text for details about configuring classes of text.

Which Ticket Fields Should Be Visible for the Customer?

Visibility of Ticket Fields in CM prior to 6.10.5.4

In CM versions prior to 6.10.5.4, all ticket fields (only with content, no empty fields) of the ticket are visible in CM/Track.

Visibility of Ticket Fields in CM 6.10.5.4 and Up

Starting with CM version 6.10.5.4, the visibility of ticket fields in CM/Track can be configured on ticket field group level as well as on single ticket field level.

The control mechanism for the visibility of ticket fields in CM/Track is switched on/off by the CM system property cmas-rest-api, security.fields.customer.exposure.check.enabled, a boolean, default “true”. The values mean:

- **true**
  The security control mechanism for the visibility of ticket fields is switched on. Only fields which have explicitly been annotated to be visible can be seen by the customer in CM/Track.
- **false**
  The security control mechanism for the visibility of ticket fields is switched off. All ticket fields can be seen by the customer in CM/Track. This results in the same behavior as the standard system behavior in CM versions lower than 6.10.5.4.

When the security control mechanism for the visibility of ticket fields is switched on (i.e. when the CM system property cmas-rest-api, security.fields.customer.exposure.check.enabled is set to “true”, the following annotations control the visibility of ticket field data in CM/Track:

- Ticket field group annotation
  - **customer exposure group**, values:
    - **full**
      the field group is available for
      - reading (display)
      - writing (saving).
    - **read**
      the field group is available for
      - reading (display)
- Ticket field annotation
  - **customer exposure**, values:
    - **full**
      the field is available for
      - reading (display)
      - writing (saving).
    - **read**
      the field is available for
      - reading (display)
    - **none**
      the field is not available / visible

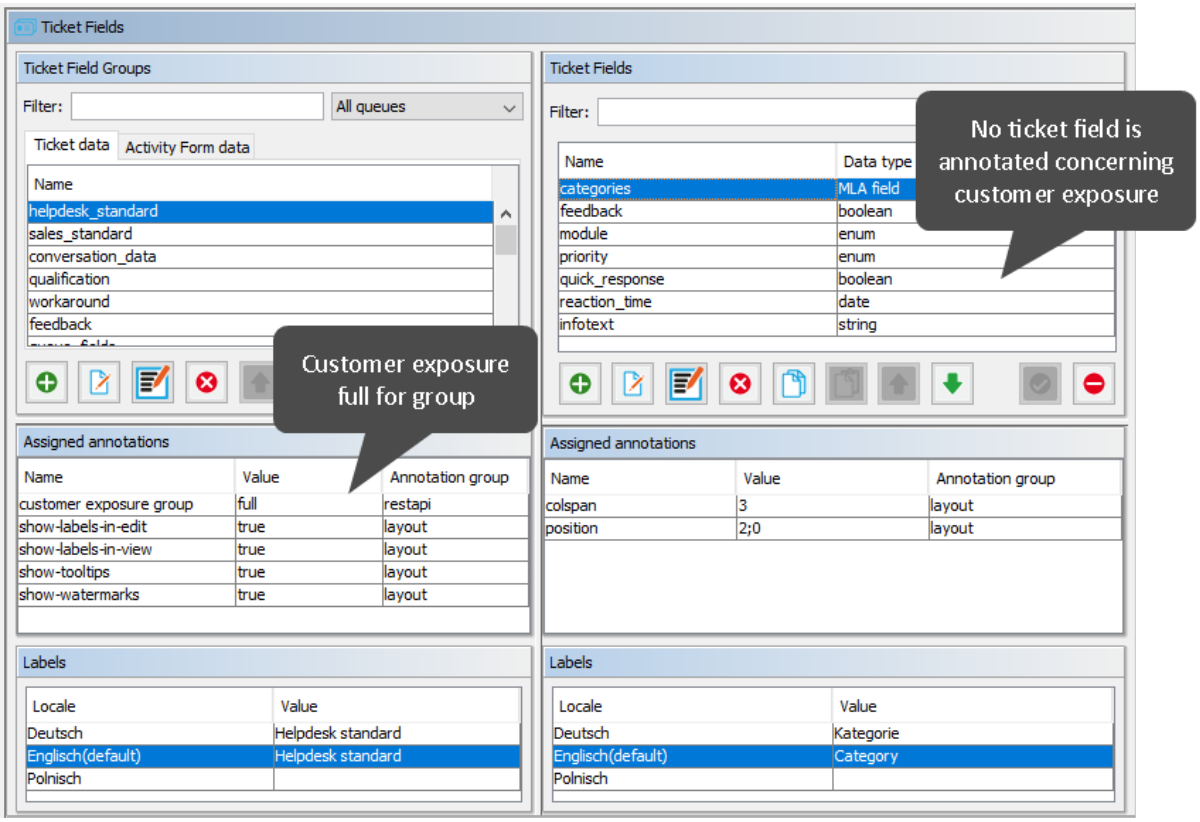
The value of the annotation on field basis overwrites the value of the group annotation. In this way, you can fine-tune the visibility of each single field without having to really annotate each field in case entire groups should be made (un)available.

The annotations `group-visiblity` (for ticket field groups) and `visibility` (for single ticket fields) will work in any case and are stronger than the `customer exposure group` and `customer exposure` annotations, i.e. when a group or field is annotated with `(group-)visibility` = “none”, it will not be displayed, no matter what `customer exposure (group)` annotation might been set.

Please see the following examples for more clarity.

Example #1: Security Check Enabled, Ticket Field Group Custom Exposed

`security.fields.customer.exposure.check.enabled` = “true”



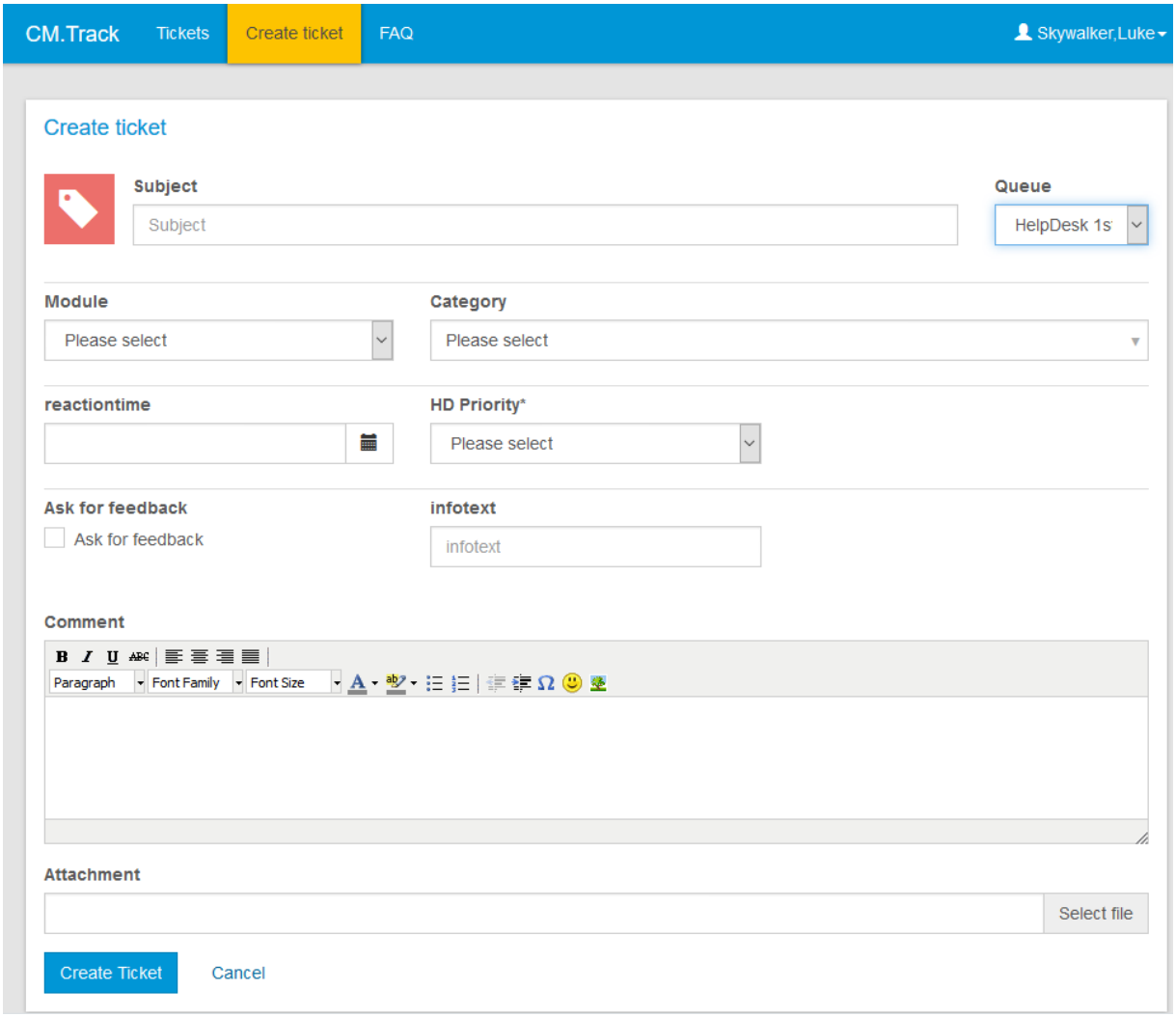


Figure 706: *Display of data in CM/Track with security control mechanism for ticket field display on. Visibility depends on annotation, here: group annotation full visibility.*

Example #2: Security Check Enabled, Ticket Field Group Not Custom Exposed

`security.fields.customer.exposure.check.enabled` = "true"

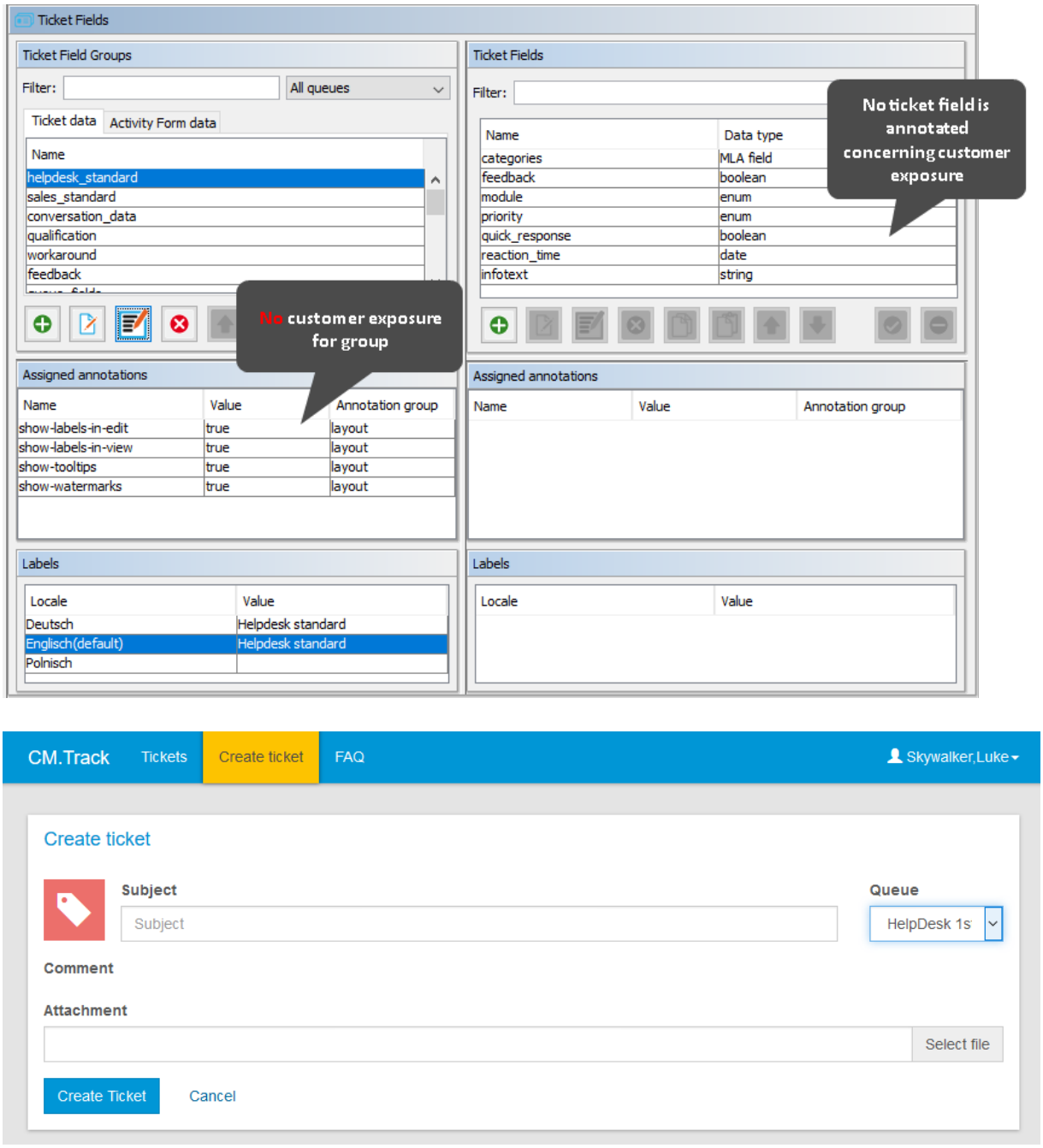
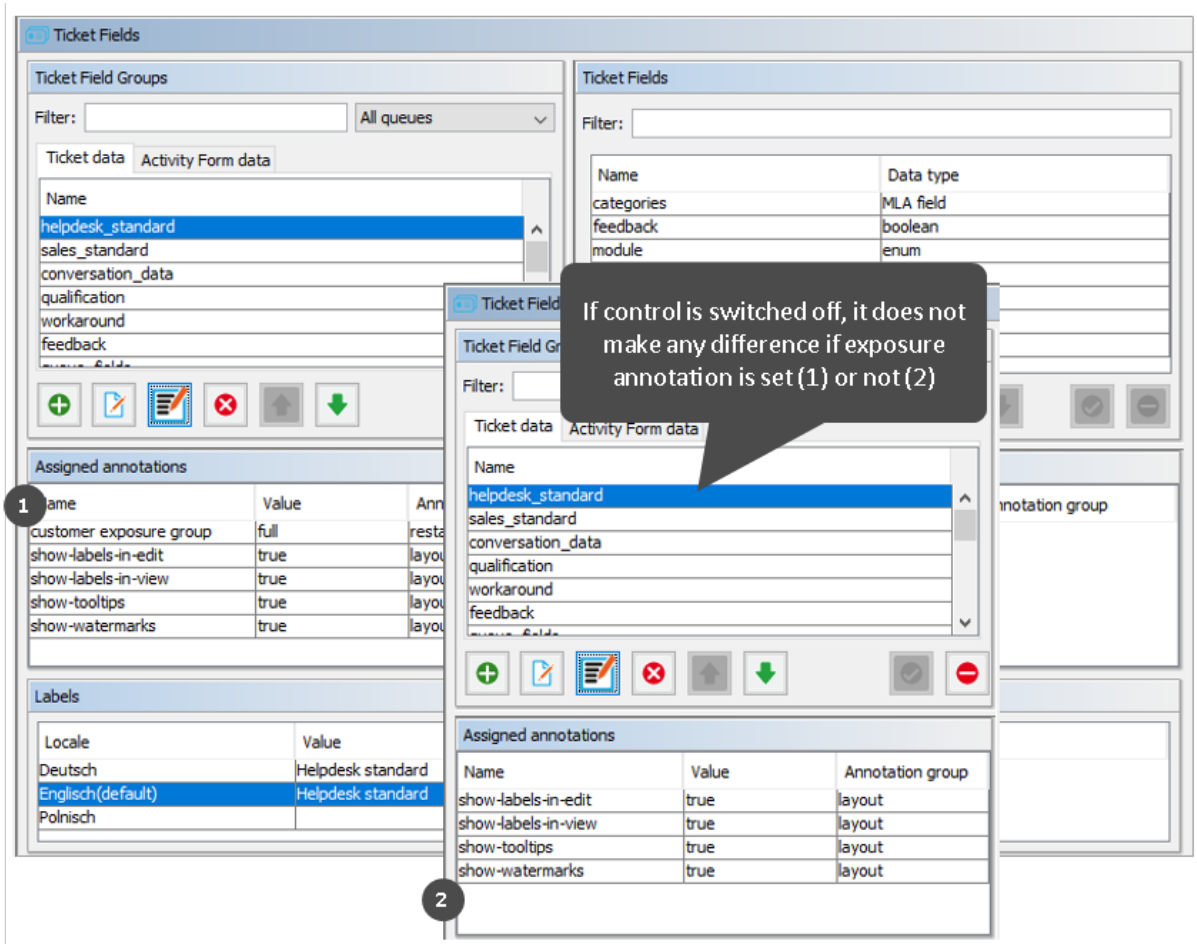


Figure 707: *Display of data in CM/Track with security control mechanism for ticket field display on. Visibility depends on annotation, here: no annotation for group or fields set.*

Example #3: Security Check Disabled, All Ticket Fields Visible

`security.fields.customer.exposure.check.enabled` = “false”



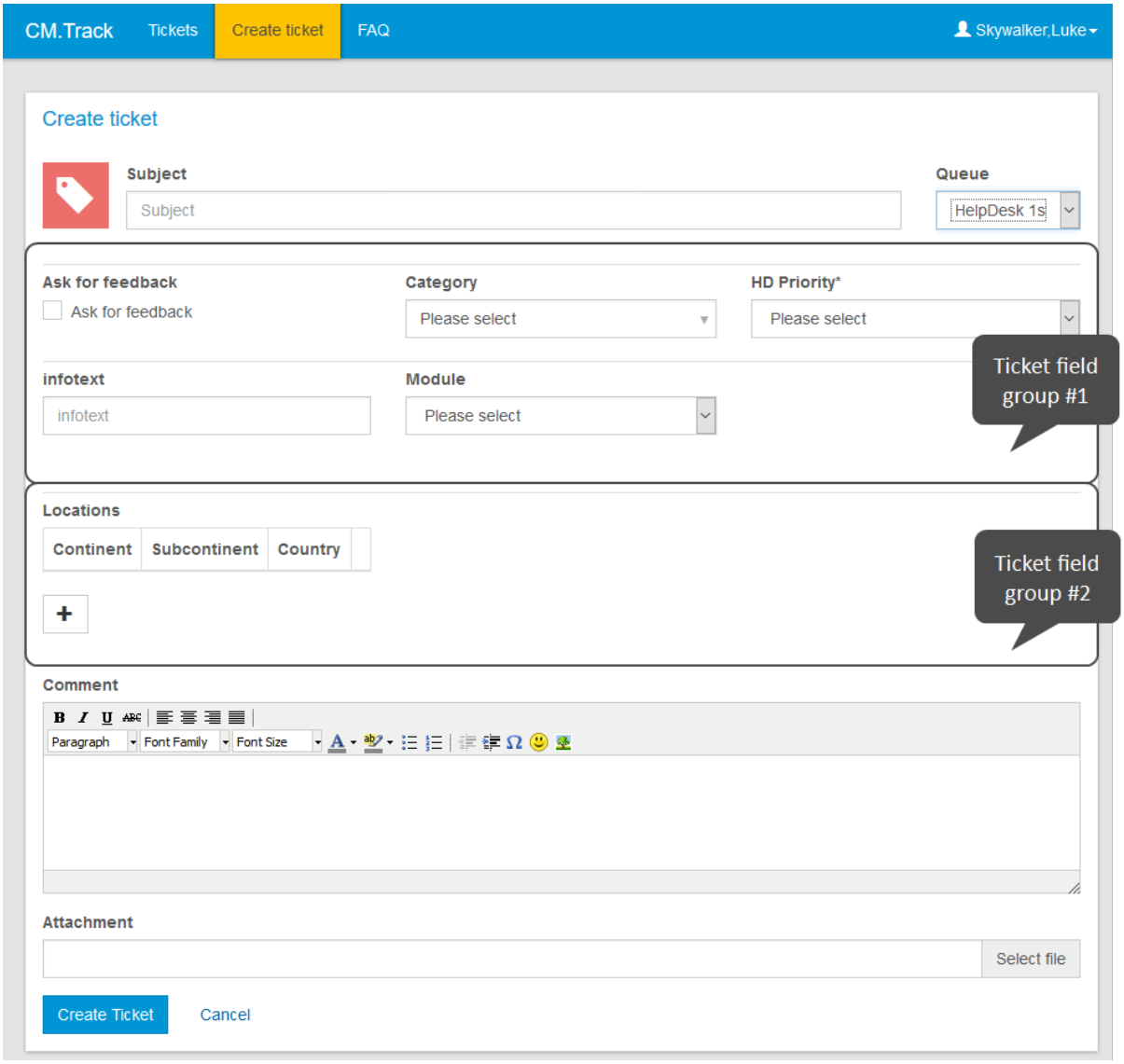


Figure 708: *Display of data in CM/Track with security control mechanism for ticket field display off. Customer exposure annotations do not have any influence on ticket field visibility in CM/Track. All ticket fields of the queue are exposed/displayed.*

G.4 CM/Phone: CTI with ConSol CM

This chapter discusses the following:

G.4.1 Introduction to CM/Phone

CM/Phone is a distinct ConSol CM module which has to be licensed in addition to the core ConSol CM system. For license information, please see section License Management, REST.

CM/Phone is a Windows client application for the integration of telephony systems using the ***TAPI 3*** protocol. TAPI is part of any Windows operating system and provides generic telephony functions. The CM/Phone client has to be installed on each Windows client which should use the CTI (Computer Telephony Integration) functionality with ConSol CM.

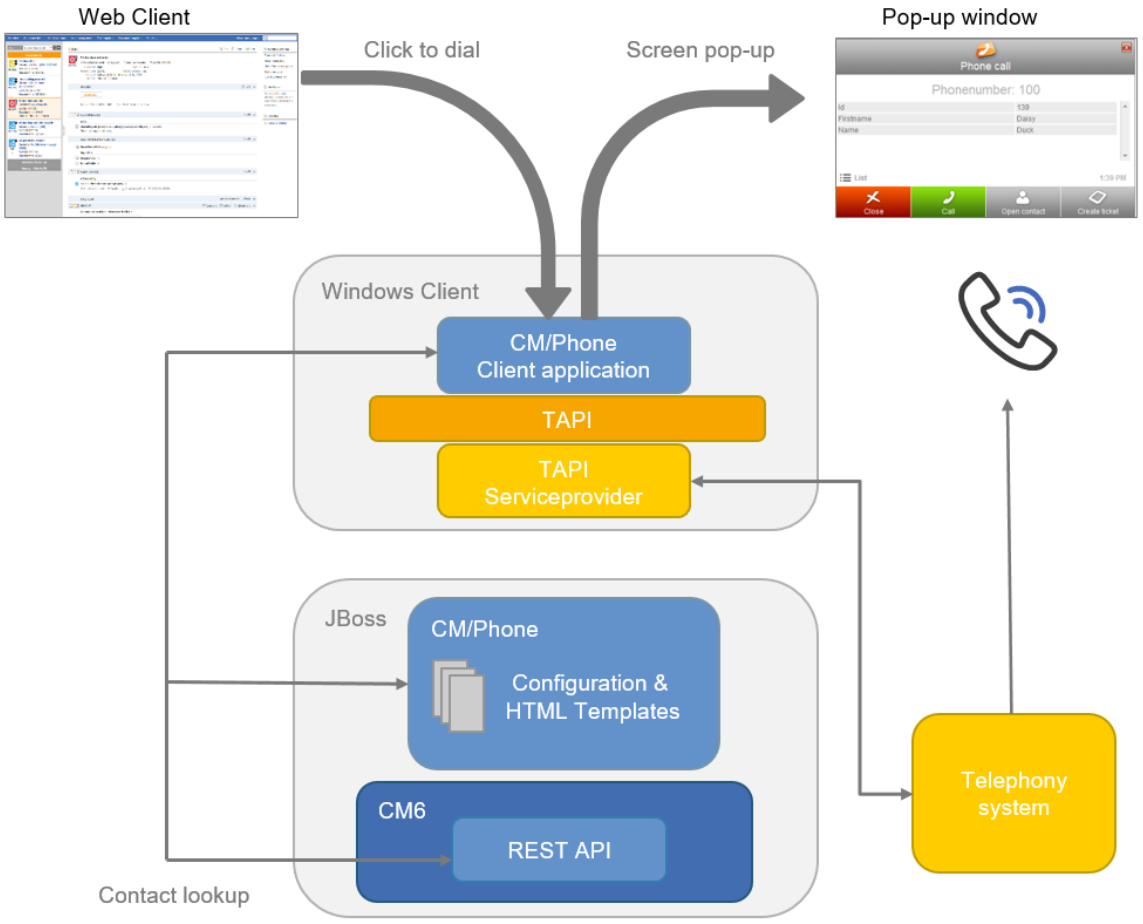


Figure 709: *ConSol CM/Phone - Basic principle*

G.4.1.1 Incoming Calls

The CM/Phone client monitors the telephone handset (i.e., the selected TAPI device, `address` or `line`) for incoming calls. When an incoming call has been registered, a pop-up window is displayed with the phone number of the caller. The ConSol CM customer database is searched for matches for this customer. If one or more matches have been found, a customer list is offered for selection. Engineers can then decide if they want to create a ticket for the customer or if they want to have the customer page displayed. If no corresponding customer data matches the phone number, just the calling number is displayed and the option *Create customer* is offered.

> ⚠ Please note that a user can only see the customer data in the CM/Phone pop-up window which is allowed by the user's permissions. Others will be filtered out and will thus not be visible.

The pop-up window is based on HTML template files which are located in the CM/Phone folder on the ConSol CM server. These templates are loaded by the CM/Phone client application during startup. The information displayed in the pop-up window (customer fields from the customer data model) can be customized by editing the template files (see *ConSol CM Setup Manual*).

The following options can be selected in the pop-up window if exactly one customer matches in the CM database:

- **Open customer**
  Opens the customer page (contact/company) in the Web Client (alternatively *Create customer* will be listed if the caller is unknown in ConSol CM).
- **Create ticket**
  Opens the *Create ticket* page for this found (or new) customer in the Web Client.
- **Call back**
  Will be available in the case of a missed call.
- **Close**
  Closes the CM/Phone pop-up window.

In case the customer is not yet present in the ConSol CM system, the caller's phone number will be used to fill in the phone number field in the customer data (customer fields) annotated as `dialable`. This will be done for new customers and newly created tickets. Should multiple fields be annotated as `dialable`, the first one will be pre-filled. If the user has access to multiple customer groups, the respective `dialable` phone number fields of each customer group will be pre-filled.

G.4.1.2 Outgoing Calls

The engineer can start an outgoing call directly by clicking on a phone number (e.g., in the customer data) in a customer field which has been annotated as `dialable`. The CM/Phone application is started automatically by the browser and the phone number is passed to the telephone system as a command line parameter. The CM/Phone application creates an outgoing call via TAPI and quits immediately.

G.4.2 CM/Phone Setup

Please refer to the *ConSol CM Setup Manual* for a detailed explanation about how to install CM/Phone on the CM server and clients. Here, only the CM/Phone configuration in the Admin Tool will be described.

G.4.3 Configuration of CM/Phone in the Admin Tool

In the Admin Tool you have to perform the following steps to configure CM/Phone:

- Set the annotations for the customer fields which contain phone numbers.
- Configure the Admin Tool templates for customer data for each customer group.
- Configure the phone number format for each customer group.
- Set the system properties.
- Optional: Change the dialing prefix for outgoing calls.

G.4.3.1 Set Annotations for Customer Fields Containing Phone Numbers

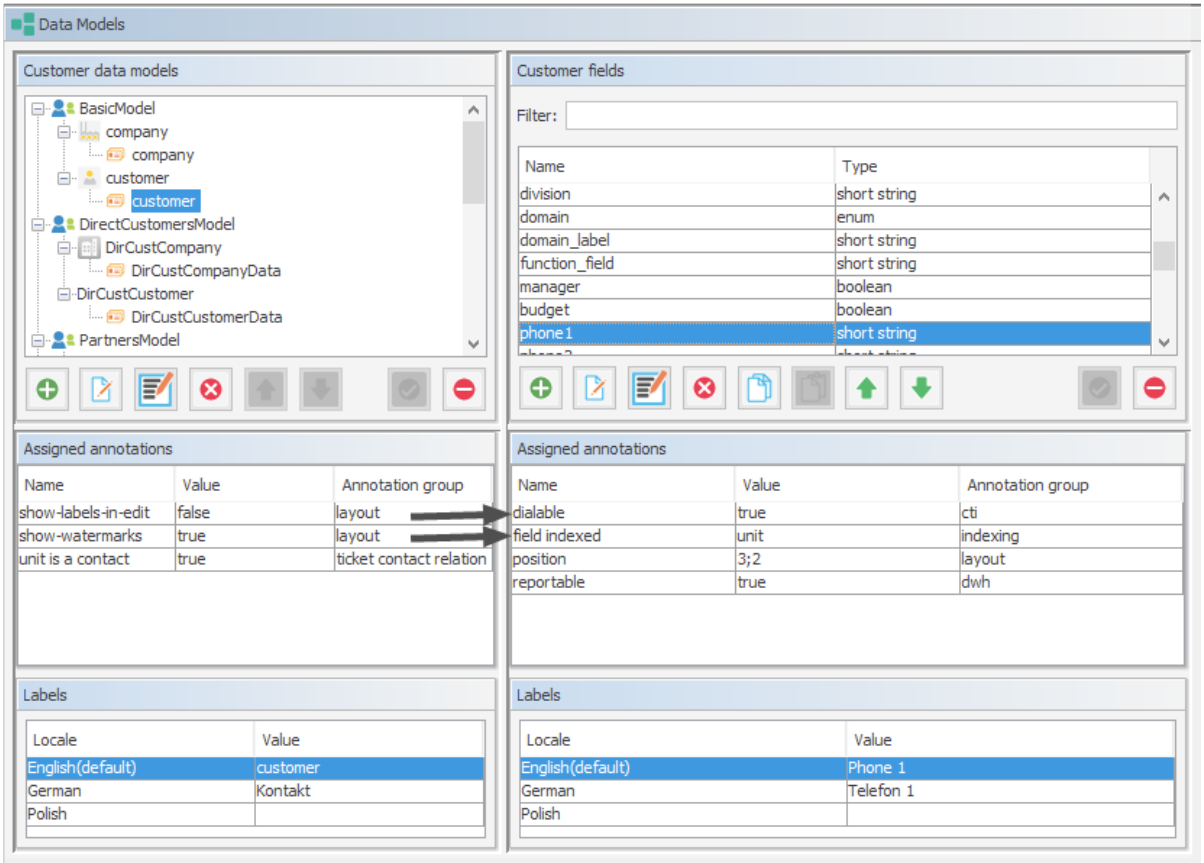


Figure 710: *ConSol CM Admin Tool - Customers, Data Models: Annotations for customer fields with phone numbers*

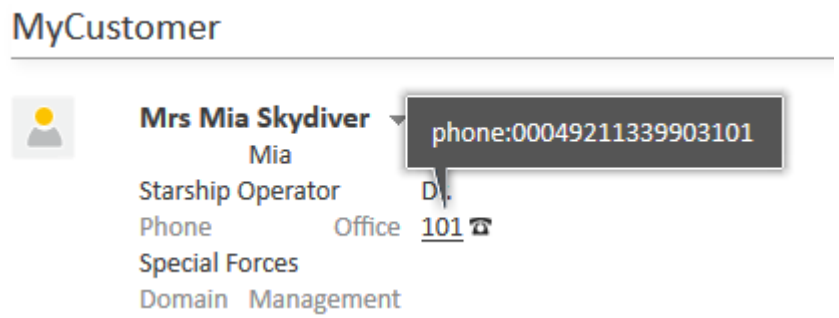


Figure 711: *ConSol CM Web Client - Dialable number when using CM/Phone*

Two annotations are required for customer fields which contain phone numbers:

- **dialable = "true"**
  This configures the phone numbers as dialable links in the Web Client. This is for outgoing calls.
- **field-indexed = "local"**
  This makes the field searchable which is important for the customer look-up. This is for incoming calls.

G.4.3.2 Configure the Admin Tool Templates for Customer Data for Each Customer Group (Used for Incoming Calls)

The customer data model configuration includes two CM/Phone-specific types of templates:

- **CMPhone customer details**
  For the display of customer data for a single customer (contact or company, the definition can be made on contact and/or on company level)
- **CMPhone customer list**
  For the display of a list of customers (contact or company, the definition can be made on contact and/or on company level)

They are used for defining how CM/Phone should render incoming call information. The first one is used for exactly one customer matching the phone number and the second one is used for multiple matches, so that the engineer may select the desired customer.

You have to perform two steps:

1. Write the templates and store them in the *Scripts and Templates* section of the Admin Tool.
2. Assign the templates to customer data models (navigation group *Customers*, navigation item *Data Models)*.

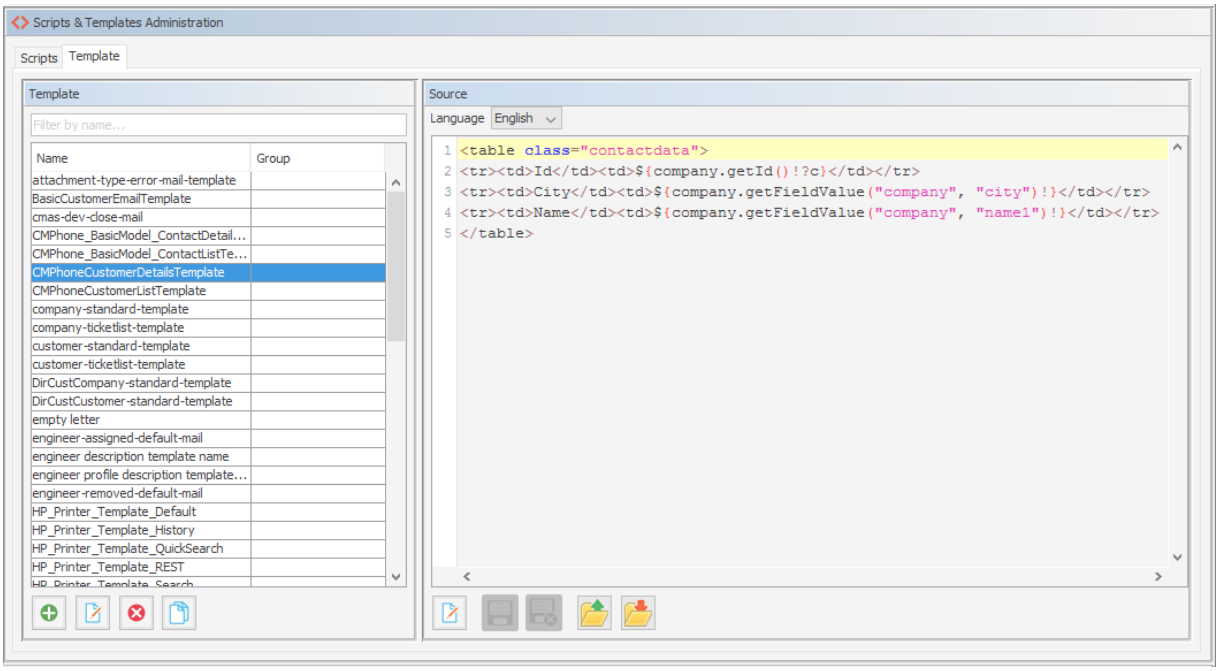


Figure 712: *ConSol CM Admin Tool - System, Scripts and Templates: Example template for rendering customer data for display in CM/Phone*

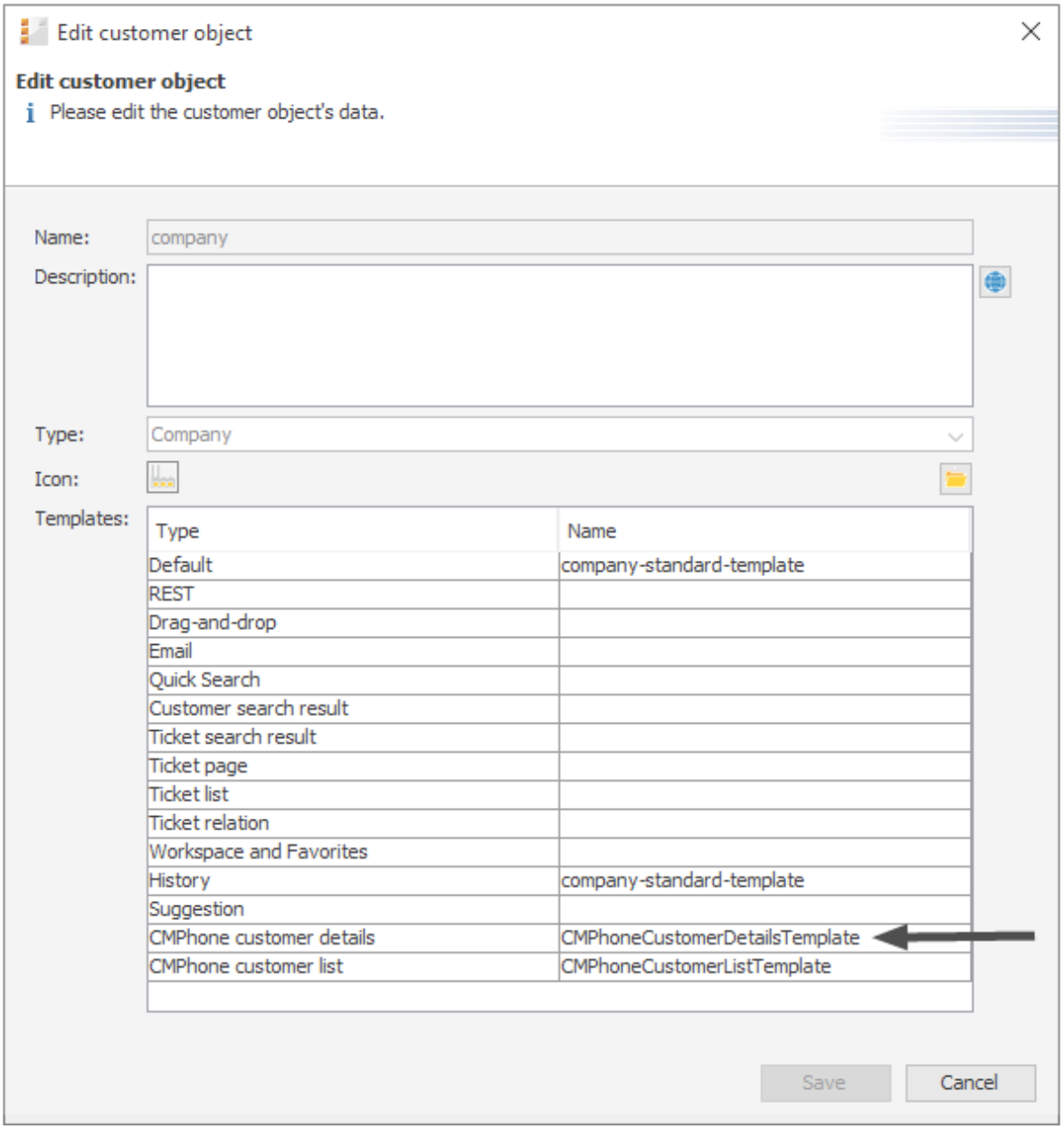


Figure 713: *ConSol CM Admin Tool - Customers, Data Models: Assignment of CM/Phone templates for customer data to customer groups*

G.4.3.3 Configure the Phone Number Format for Each Customer Group (for Incoming and Outgoing Calls)

The format defined here is used to transform phone numbers (from the respective customer group) to a common canonical form. The engineer can enter a phone number in any format with or without prefixes, e.g., as company internal number. To avoid problems with interpreting such numbers there

is a dedicated configuration per customer group which is used when a user submits a phone number for a particular customer. The patterns/elements of the different formats which can be interpreted as a phone number in the fields marked as `dialable` can be defined in detail in the Admin Tool.

The navigation item *Customer Groups* in navigation group *Customers* has to be selected after logging in to the Admin Tool for this configuration. On the navigation item *Customer groups*, the desired customer group has to be selected for editing. After clicking the *Edit* button below the list of customer groups, the edit dialog opens, containing a new tab titled *CMPhone*.

On the *CMPhone* tab of the *Edit customer group* dialog there are fields in which you can enter phone number prefixes for different scopes and number patterns for several phone number types.

The fields for configuration values are:

- **Country prefix**
  The international country prefix for extending national phone numbers, without prefixes like “0” or “+”. Such a prefix is not allowed here!
  The country prefix is required in order to check whether or not an outgoing call is within the same country. Some phone providers do not handle canonical (so, theoretically, correct) numbers for domestic calls, and the country prefix has to be removed from the number in such cases.
- **Area prefix**
  The local city/area prefix for extending local phone numbers. Please note that this also does not include general prefixes like “0” or “1”, so the entry for Munich in Germany would be “89”, not “089”!
- **Company prefix**
  The phone number of the company as used in (local) calls without extensions. Adding an extension number to this prefix would allow a local call from outside the company to this extension.
- **Subscriber pattern**
  This regular expression (RegEx) describes a number pattern used to identify whether the number provided is a full subscriber number (potentially including an extension) which would allow for a local call.
- **Internal pattern**
  The regular expression (RegEx) in this field defines the pattern to classify extensions if only a phone extension is entered.
- **Mobile pattern**
  This regular expression (RegEx) is used to identify a number entered as a mobile/cell phone number in the country, which would be valid to make a national call to a mobile phone.

For example, for all numbers (12, 33990312, 21133990312) from above points the result should be always the full canonical number: 4921133990312. For mobile numbers, a country prefix will also be added, so the result will be: 49600289906. If the engineer enters a full number starting with “+” or “0” then the configuration is skipped - ConSol CM assumes no number conversion is required.

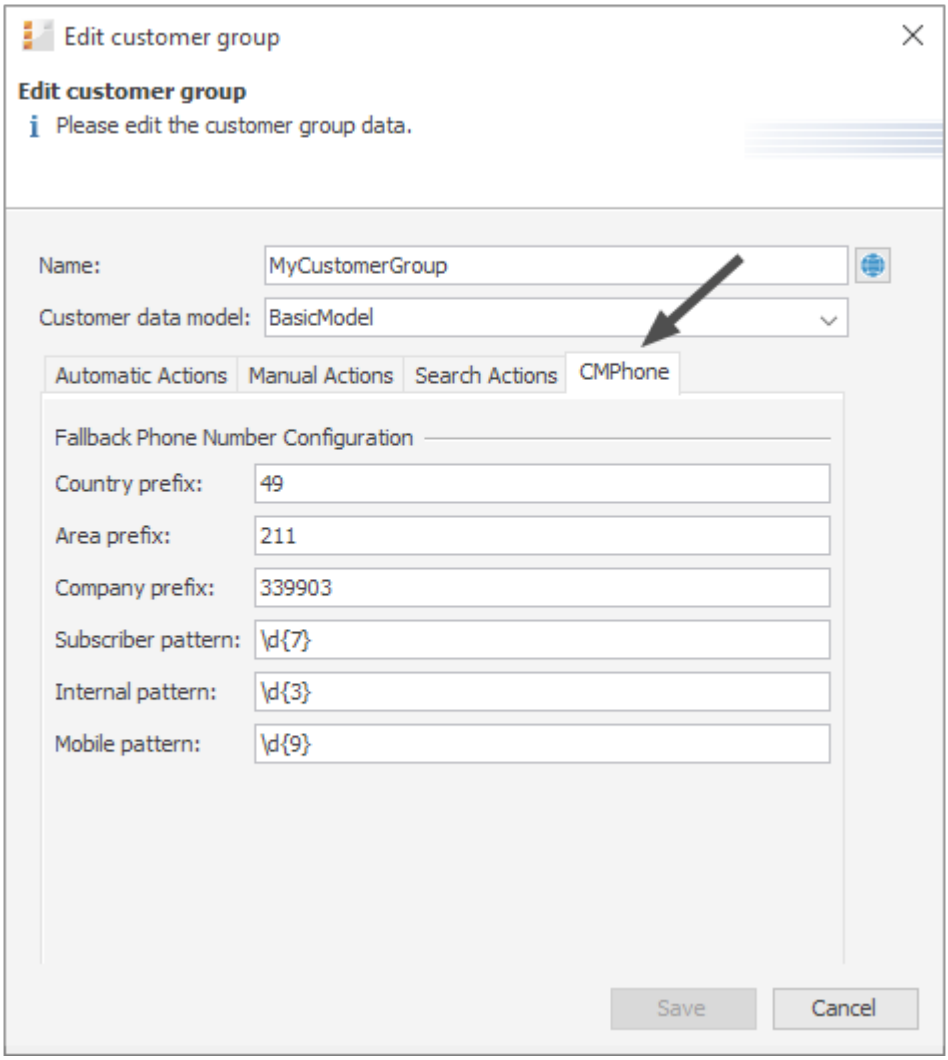


Figure 714: *ConSol CM Admin Tool - Customers, Customer Groups: Configuration of phone number formats for a customer group*

These prefix values are defaults for extending phone numbers which are not fully qualified. They can always be overridden by entering a fully qualified phone number.

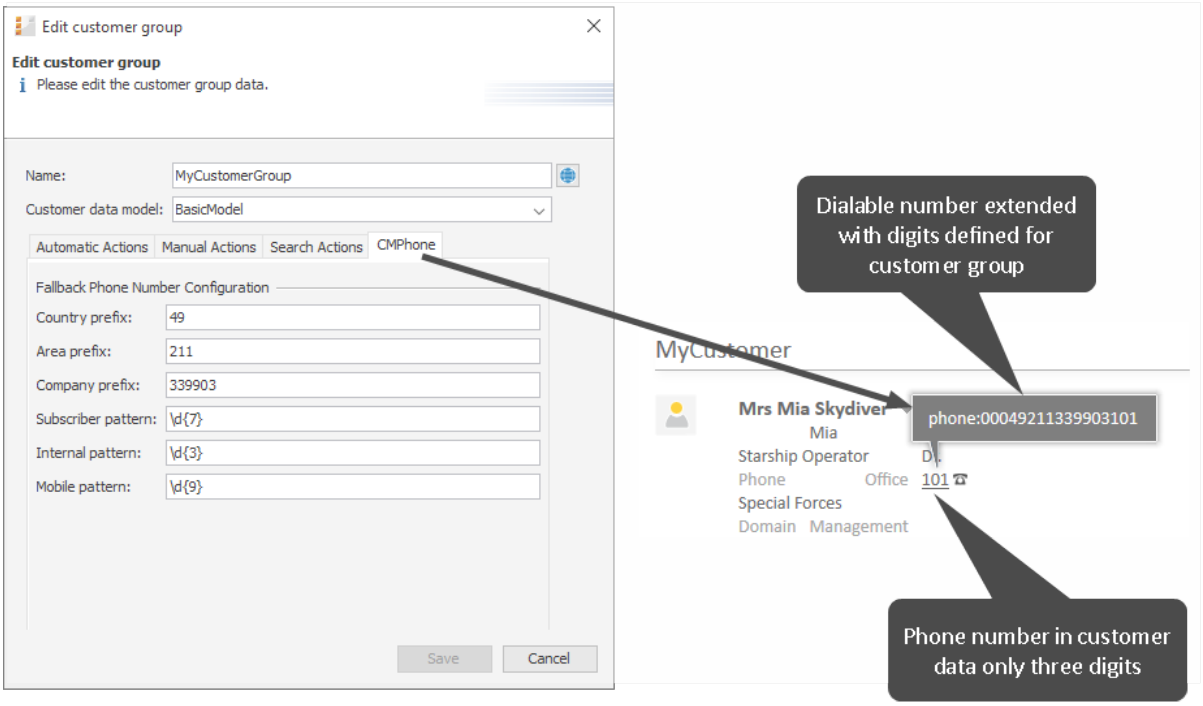


Figure 715: *CM/Phone - Use of customer group-specific number configuration for dialable numbers (outgoing calls)*

The patterns are used to guess the type of a phone number which is not fully qualified. The guessed type determines its use and necessary additions for connecting a call. For this purpose, after removing unnecessary characters, a number is checked to determine whether it is already fully qualified. If not, it is matched against these patterns. For exactly one match, a valid number is constructed and used. If two matches are area code and mobile number, these are combined with the country prefix to create the number to be dialed. In all other cases the supplied original number cannot be used for making a connection.

G.4.3.4 Set the System Properties

There are three new properties in ConSol CM, to be set in the Admin Tool, which are relevant for CM/Phone. The correct configuration for these is essential for proper usage of phone numbers for connection calls. The properties are elements of the module `cmas-core-server`:

- local.country.prefix
  String. This is the local country code. The value is an international country code, like “49” for Germany. Default value is “49”.
- internal.line.access.prefix
  This is a prefix that the company’s telephony system uses for outside lines, if required. So, if a “0” or a “9” needs to be dialed in order to make a call to any number outside the company, this value needs to be configured here. Default value is “0”.

- external.line.access.prefix
  This is the general prefix to dial before an area code to get a long-distance connection in the country. For example, in Germany it is a “0” that needs to be prepended to the area code. Default value is “0”.

These properties are all optional, so they have to be added manually, if needed.

G.4.3.5 Change the Prefix for Outgoing Calls

This step is optional!

Usually the prefix `phone:` is set before the number for outgoing calls for interaction with the TAPI. If another prefix (e.g., `tel:`) is required, this can be configured in the ***Windows Registry***. Please ask your ConSol CM consultant for advice.

H - Appendix

This section contains several appendices:

H.1 Annotations

There are two types of annotations: field annotations and group annotations. Field annotations are applied to a single ticket, customer or resource field. Group annotations are applied to a ticket, customer or resource field group. Please see:

- List of Field Annotations
- List of Group Annotations

H.1.1 List of Field Annotations

This chapter describes the following field annotations grouped by annotation type:

H.1.1.1 anonymize (type)

personal-data

- **Type**: anonymize
- **Description**: Can be assigned to ticket and contact fields. Determines that the field holds personal data. Contact fields with this annotation will be deleted when the contact is anonymized. Ticket fields with this annotation will be deleted when the main customer of the ticket is anonymized. See Example 8: Removing Customer Data for information about how to anonymize a contact.
- **Values**:
  - *true / false*: Set “true” if the field contains personal data.

H.1.1.2 cmweb-common (type)

groupable

- **Type**: cmweb-common
- **Description**: Enables grouping of the ticket list by this field. Please see section Grouping for a detailed explanation.
- **Values**:
  - *true*: Used only with ENUM data fields. Remove the annotation if you want to disable grouping.

sortable

- **Type**: cmweb-common
- **Description**: Used to enable sorting of the ticket list by this field. Please see also the detailed explanation in section List of Field Annotations
- **Values**:
  - *true*: Used for data fields of type DATE or of type ENUM. Remove the annotation if you want to disable sorting.
    For ENUM fields: Works only if order index is set for all values of the ENUM field.

H.1.1.3 common (type)

autocomplete-script

- **Type**: common
- **Description**: Specifies the script (type *Text autocomplete*) used for the scripted autocomplete field.
- **Values**: <name of the script>

Note:

This annotation is part of the way of configuring autocomplete fields which should only be used in ConSol CM versions up to 6.11.1.0. Starting with version 6.11.1.1, a new way of implementing scripted autocomplete lists is available. See section *Scripted Autocomplete Lists* in the *ConSol CM Administrator Manual*.

Existing fields which use the old method will remain functional as of version 6.11.1.1. These fields will not be modified during an update. The previously used method `onEditDisplayEntered` still works for existing fields. Nevertheless, it is not needed anymore and will be removed in a future version of ConSol CM.

leave-trailing-zeros

- **Type**: common
- **Description**: Used for the display of fixed point numbers.
- **Values**:
  - *true / false*: Trailing zeros in the fractional part are not cut off when value is “true”.

readonly

- **Type**: common
- **Description**: Used to indicate that the ticket field cannot be modified.
- **Values**:
  - *true / false*: Field is read-only if value is set to “true”. Lack of value, or any value except “false”, is treated as “true”.

visibility

- **Type**: common
- **Description**: Defines when the field is visible.
- **Values**:
  - *edit*: Field will be displayed in edit mode.
  - *view*: Field will be displayed in view mode.
  - *none*: Field is not visible.
  - If any other, or no value, is set then the field will always be visible.

visualization

- **Type**: common
- **Description**: Specifies the script (type *Field visualization*) used to render the content of the data field. For a detailed explanation, please refer to the section *Scripts of Type Field Visualization* .
- **Values**: <name of the script>

visualize-when-empty

- **Type**: common
- **Description**: Determines if the visualization script of a data field with the annotation `visualization` should be executed even if the field is empty. By default, the visualization script is only executed if the field contains a value.
- **Values**:
  - *true / false*: Set “true” if the field visualization script should be executed even if the field is empty.

This annotation is applied for scripted field visualization both in the Web Client and CM/Track V2.

H.1.1.4 component type (type)

boolean-type

- **Type**: component-type
- **Description**: Definition of the layout of a boolean field.
- **Values**:
  - *checkbox* (default): Field that can be checked (set to “false ”by default).
  - *radio*: 2 radio buttons (yes/no) for selection (only one can be active).
  - *select*: Drop-down field with 2 values (yes/no).

enum-in-search-type

- **Type**: component-type
- **Description**: Defines whether an ENUM field used in a search accepts searching over multiple values.
- **Values**:
  - *single* (default) / *multiple*: Accepts searching over multiple values if value “multiple” is set.

enum-type

- **Type**: component-type
- **Description**: Layout definition of list display
- **Values**:
  - *select* (default): Drop-down list for selection.
  - *radio*: List of radio buttons to select (only one option can be active).
  - *autocomplete*: Drop-down list for selections where the field is an input field used to filter the list.

list-type

- **Type**: component-type
- **Description**: Disables the add and/or delete options for data fields of type LIST or STRUCT.
- **Values**:
  - *fixed-size*: It is not possible to add or delete fields/rows.
  - *non-shrinkable*: It is not possible to delete fields/rows.
  - *non-growable*: It is not possible to add fields/rows.

text-type

- **Type**: component-type
- **Description**: Defines the possible types of a STRING field.
- **Values**:
  - *text* (default): Single-line input field
  - *textarea*: Multi-line input field
  - *password*: Input field for passwords.
    Password will be displayed as ******* in view mode.
  - *label*: Input will be displayed as a label, i.e., the field is displayed only, no input is possible.
  - *autocomplete*: The field will be used as autocomplete list. Please see the detailed explanation in section Scripted Autocomplete Lists.
  - *url*: The input will be displayed as a hyperlink in view mode. If no protocol is provided, *http://* is added automatically in front of the entered string. If a protocol, e.g., “http”, “https”, “mailto”, “file”, or “ftp” is used, the URL is rendered as is. The display text for the URL can be entered after a whitespace.
    Example: “http://consol.de ConSol”
  - *file-url*: Input will be displayed as a link to a file on the file system. The web browser has to allow/support those links! See section Details about String Fields: Use Annotations to Fine-Tune Strings on how to achieve this. The link will also be displayed as tooltip.

    The URL is correctly formed if the following conditions are met:

    It starts with `file:` followed by regular slashes:

    - three slashes “///” for files on the same computer as the browser (alternatively “//localhost/”) or
    - two slashes followed by the server name followed by another slash for files on file servers accessible from the computer running the browser.

    These are followed by the full path to the file ending with the file name. The path on Microsoft Windows systems is also written with forward slashes instead of backslashes.

    The drive letter of a local path on Microsoft Windows systems is noted as usual, for example C:. Paths with spaces and special characters like “{, }, ^, #, ?” need to be percent encoded (“%20” for a space for example) for Microsoft Windows systems.

Example URLs:

- file://file-server/path/to/my/file.ext
- file:///linux/local/file.pdf
- file:///C:/Users/myuser/localfile.doc

See also the explanation in the section Details about String Fields: Use Annotations to Fine-Tune Strings.

H.1.1.5 contact authentication (type)

ldapid

- **Type**: contact authentication
- **Description**: Used in a customer field group of type *contact*, for the customer field which contains the LDAP ID for CM/Track authentication.
- **Values**: Indicates that this field will be used as an LDAP ID in the authentication process. Data type string is required.
  Since the definition is made at the customer group level, the LDAP authentication can be run in mixed mode. I.e., use LDAP for some customer groups and regular authentication for other customer groups.

password

- **Type**: contact authentication
- **Description**: Indicates that this field will be used as a password in the authentication process.
- **Values**:
  - **<string>**: Used for CM/Track.

username

- **Type**: contact authentification
- **Description**: Indicates that this field will be used as a login name in the authentication process.
- **Values**:
  - *true / false*: Used for CM/Track.

H.1.1.6 dwh (type)

dwh-no-history-field

- **Type**: dwh
- **Description**: Annotation used to indicate that field will not be historized in DWH
- **Values**:
  - *true / false*: Since version 6.10.2.0.

reportable

- **Type**: dwh
- **Description**: Indicates that the field is reportable and that it should be transferred to the DWH.
- **Values**:
  - *true / false*: Field is reportable if value is set to “true”.

H.1.1.7 indexing (type)

field indexed

- **Type**: indexing
- **Description**: Indicates that a database index will be created for this field. If it should be possible to sort result tables (in the Web Client) according to a column (by clicking on the column header), the respective field has to be indexed!
- **Values**:
  - *transitive* (default): All data is displayed (ticket data, customer data and resource data).
  - *unit*: Used for customer data. Only the unit and the parent unit (i.e., company) is given as a search result, no tickets are provided.
  - *local*: Used for customer data. Only the unit is given as a search result, no company and no tickets are displayed.
  - *<annotation not set>*: Field is not indexed.

phonetic

- **Type**: indexing
- **Description**: activates the phonetic search for this field. Can only be used for data fields of type String (also long or short string).
- **Values**: *true/false*. Will automatically set to “true” when the annotation is added.

H.1.1.8 layout (type)

colspan

- **Type**: layout
- **Description**: Defines how many columns are reserved for the field in the layout.
- **Values**:
  - **<number>**: Number of columns.

This annotation only works if the annotation `position` is also set for the field.

field-group

- **Type**: layout
- **Description**: Allows grouping of fields in view mode. Annotation is ignored in edit mode.
- **Values**:
  - **<string>**: To group fields the same string value has to be set in the annotation of each field. Two or more data fields are bound when they share the same value for this annotation. The group of coupled data fields is shown only if all of them have values set.
- Removed in ConSol CM version 6.11.0.1.

fieldsize

- **Type**: layout
- **Description**: Displayed field size within the ticket layout.
- **Values**:
  - **<rows>;<cols>**: Displayed field size for textareas

    For STRING fields with `text-type` = `textarea`: rows;cols (corresponds to <textarea rows="" cols="">).
  - **<number>**: Displayed field size for strings and numbers

    For STRING fields with `text-type` other than `textarea` and NUMBER fields: n indicates the number of characters in the fields; for string fields this is the number of monospaced capital M characters.

    For ENUM data fields: Defines how many values are directly visible in the list box. Used only for layout purposes. This annotation is not relevant for ENUM autocomplete fields.

This is only a layout configuration, for validation use `maxlength` of type `validation`.

label-group

- **Type**: layout
- **Description**: Indicates a group of fields along with its descriptive label in view mode. Annotation is ignored in edit mode.
- **Values**:
  - **<string>**: Indicates a group of data fields along with its descriptive label. The annotation is used in view mode, ignored in edit mode. The group can have exactly one label (a data field of type STRING with assigned additional annotation `text-type` with value “label”). The label is shown when at least one data field from its group has a value set. All fields with the same label value are grouped and displayed under this label.
    The annotation `label-group` has to be assigned to the label, too.

label-in-view

- **Type**: layout
- **Description**: Shows data field value as a label in view mode. Annotation is ignored in edit mode.
- **Values**:
  - *true*: Remove the annotation if the label should not be visible in view mode.

order-in-result

- **Type**: layout
- **Description**: Shows field as a column at given position in the search result list.
- **Values**:
  - **<number>**: The columns are sorted in ascending order.
    Since CM version 6.0.1. Please see detailed explanation in info box in section Search Configuration, order-in-result.

position

- **Type**: layout
- **Description**: Defines the position of a field within a grid layout or defines the position of a field within a list (STRUCT).
- **Values**:
  - **<number>;<number>**: Values define row and column (row;column), numbering starts at 0;0. If no values are set, the data field will take the next free grid cell.
  - **0;<number>**: Only the column value is used, the row value is ignored.

rowspan

- **Type**: layout
- **Description**: Indicates how many rows within the layout are occupied by this field.
- **Values**:
  - **<number>**: Number of rows.

⚠ This annotation only works if the annotation `position` is also set for the field.

show-label-in-edit

- **Type**: layout
- **Description**: Whether the data field should be displayed in edit mode with label.
- **Values**:
  - *true / false*: Since version 6.9.4

show-label-in-view

- **Type**: layout
- **Description**: Whether the data field should be displayed in view mode with label.
- **Values**:
  - *true / false*: Since version 6.9.4

show-tooltip

- **Type**: layout
- **Description**: Whether the data field should be displayed with tooltip.
- **Values**:
  - *true / false*: Since version 6.9.4

show-watermark

- **Type**: layout
- **Description**: Whether the data field should be displayed with watermark.
- **Values**:
  - *true / false*: Since version 6.9.4

ticket-list-colspan

- **Type**: layout
- **Description**: Defines how many columns are occupied by the field in the ticket list box.
- **Values**:
  - **<number>**: Number of columns.

This annotation only works if the annotation `ticket-list-position` is also set for the field.

ticket-list-position

- **Type**: layout
- **Description**: Defines the position of the field in the ticket list box.
- **Values**:
  - **<number>;<number>**: Values define row and column (row;column), numbering starts at 0;0.

ticket-list-rowspan

- **Type**: layout
- **Description**: Defines how many rows are occupied by the field in the ticket list box.

- **Values**:
  - **<number>**: Number of rows.

This annotation only works if the annotation `ticket-list-position` is also set for the field.

H.1.1.9 performance (type)

no-history-field

- **Type**: performance
- **Description**: Indicates that a single data field should not be historicized. Overwrites the group annotation `no-history`.
- **Values**:
  - *true / false*: Annotation is active if value is set to “true”. For fields that should be stored but not be visible in history use annotation `visibility configuration`.
    In CM versions up to 6.10.2, the DWH transfer of a field history is also controlled by this annotation.
    Starting with CM version 6.10.2, use the annotation `dwh-no-history-field` for this.

H.1.1.10 phone commander (type)

dialable

- **Type**: phone commander (CM/Phone)
- **Description**: Defines a field with a phone number.
- **Values**:
  - *true*: Used with CM/Phone only. Marks a phone number as automatically dialable for outgoing calls for the CTI system.

H.1.1.11 resource (type)

resource-color

- **Type**: resource
- **Description**:
- **Values**:
  - *true / false*: Should be assigned to a color ENUM. Color of selected enum value should be applied to a resource icon's background.

H.1.1.12 restapi (type)

customer exposure

- **Type**: restapi
- **Description**: Indicates whether a data field should be available to customers using the REST API, e.g. in CM/Track.
- **Values**:
  - *full* (default): The data field is available for reading and writing.
  - *read*: The data field is available for reading only.
  - *none*: The data field is not available. This value can be helpful, for example, when an entire ticket field group has been configured with `customer exposure group` = "full|read" and dedicated single fields should not be available in CM/Track (or over the REST API in general).

H.1.1.13 search result (type)

contact search result column

- **Type**: search result
- **Description**: Identifies whether the field should be presented in the search result by default. **Deprecated! Do not use!** Removed in ConSol CM version 6.11.
- **Values**:
  - *true*:
    Remove the annotation if the field should not be visible by default.
    Since CM version 6.1.3.
    (Replaced by `order-in-result`! `contact search result column` is obsolete!)

H.1.1.14 ticket contact relation type (type)

contains contacts

- **Type**: ticket contact relation type
- **Description**: Only one instance of this annotation is allowed in one data model! Used only for list field definition, indicates that it can hold unit references to units annotated as contacts. Starting with ConSol CM version 6.11, the field annotated with contains contact will not produce history entries. This decreases the loading time for tickets which contain a great number of customers.
- **Values**:
  - *true/false*: Value type is boolean. Specifies whether the list is shown with the contact ("true") or with the ticket ("false").

H.1.1.15 ticket display (type)

enum field with ticket color

- **Type**: ticket display
- **Description**: Defines the background color of the ticket icon for ticket list and ticket.
- **Values**:
  - *true / false*: The field has to exist within Enum Administration where lists, values, and colors are defined.

H.1.1.16 validation (type)

accuracy

- **Type**: validation
- **Description**: For ticket, customer and resource fields of type DATE. To define the level of detail displayed
- **Values**:
  - *date (default)*: Show date without time.
  - *date-time*: Show date with time.
  - *only-time*: Show only time, no date.

email

- **Type**: validation
- **Description**: Used for email addresses to validate that the format is correct, i.e., that it matches <name>@<domain>.
- **Values**:
  - *true*: May be used with STRING data fields. Remove the annotation if the format should not be validated.

format

- **Type**: validation
- **Description**: Used for validating the format of date fields.
- **Values**:
  - *<date format>*: The pattern for the date is based on `SimpleDateFormat`, e.g., dd.MM.yyyy.
    Remember to set the proper colspan when including hours/minutes in the format. See http://docs.oracle.com/javase/6/docs/api/java/text/SimpleDateFormat.html for the format reference.

This annotation is applied to date fields both in the Web Client and in CM/Track V2. If it is not set, the standard date format of the browser locale is used in the Web Client, and the German standard date format (dd.MM.yyyy) is used in CM/Track V2.

matches

- **Type**: validation
- **Description**: Checks if input of STRING data fields matches the given RegEx.
- **Values**:
  - **<string>**: May be used with STRING data fields.

maxLength

- **Type**: validation
- **Description**: Defines the maximum length of input for STRING data fields.
- **Values**:
  - **<number>**: May be used with STRING data fields.

maxValue

- **Type**: validation
- **Description**: Defines the maximum value for number data fields.
- **Values**:
  - **<number>**: May be used with NUMBER data fields, i.e., NUMBER and FIXED-POINT NUMBER.

minLength

- **Type**: validation
- **Description**: Defines the minimum length of input for STRING data fields.
- **Values**:
  - **<number>**: May be used with STRING data fields.

minValue

- **Type**: validation
- **Description**: Defines the minimum value for NUMBER data fields.
- **Values**:
  - **<number>**: May be used with NUMBER data fields, i.e., NUMBER and FIXED-POINT NUMBER.

required

- **Type**: validation
- **Description**: Indicates that this is a required field.
- **Values**:
    - *true / false*: Field is required if value is set to “true”. The user cannot save the ticket without having entered a value in a required field. In the Web Client, required fields are marked by a red asterisk.

H.1.1.17 visibility (type)

visibility configuration

- **Type**: visibility
- **Description**: Indicates the visibility of this field in history.
- **Values**:
    - *on every level*: Field is shown on every level of history.
    - *2nd level and 3rd level*: Field is shown only on the 2nd and the 3rd level of history.
    - *only 3rd level*: Field is shown only on the 3rd level of history.

H.1.2 List of Group Annotations

This chapter describes the following group annotations grouped by annotation type:

H.1.2.1 common (type)

group-visibility

- **Type**: common
- **Description**: Defines the default visibility of a data field group.
- **Values**:
  - *true / false*: The annotation can be overwritten at the field level.

H.1.2.2 dwh (type)

dwh-no-history

- **Type**: dwh
- **Description**: Indicates that all fields in the group will not be historicized in DWH
- **Values**:
  - *true / false*: Since version 6.10.2.0

reportable group

- **Type**: dwh
- **Description**: Indicates that all data fields belonging to this group are reportable and should be transferred to CMRF.
- **Values**:
  - *true / false*: A value has to be set. Annotation is active if value is set to “true”.

H.1.2.3 layout (type)

auto-open-group

- **Type**: layout
- **Description**: The group will be opened initially. More than one value can be entered as a comma- or semicolon-separated list (can be used for the customer annotation).
- **Values**:
  - *ticket:create*: Group is opened initially when a new ticket is created.
  - *ticket:view*: Group is opened initially when a ticket page is opened.
  - *customer:create*: Group is opened initially when a new customer is created.
  - *customer:view*: Group is opened when the customer (contact or company) page is opened.

show-contact-in-ticket-list

- **Type**: layout
- **Description**: Obsolete! Use page customization!

accordionTicketList.mainCustomerDescriptionVisible={true, false}

- **Values**: obsolete

show-in-group-section

- **Type**: layout
- **Description**: Defines that a data field group is displayed in the Details section (as tab).
- **Values**:
  - *true / false*: Without this annotation the group is shown in the non-tabbed ticket, customer or resource section.

show-labels-in-edit

- **Type**: layout
- **Description**: Whether the data fields in this group should be displayed in edit mode with labels.
- **Values**:
  - *true / false*: Since version 6.9.4

show-labels-in-view

- **Type**: layout
- **Description**: Whether the data fields in this group should be displayed in view mode with labels.
- **Values**:
  - *true / false*: Since version 6.9.4

show-tooltips

- **Type**: layout
- **Description**: Whether the data fields in this group should be displayed with tooltips.
- **Values**:
  - *true / false*: Since version 6.9.4

show-watermarks

- **Type**: layout
- **Description**: Whether the data fields in this group should be displayed with watermarks.
- **Values**:
  - *true / false*: Since version 6.9.4

H.1.2.4 performance (type)

no-history

- **Type**: performance
- **Description**: Indicates that all data fields belonging to this group will not be historicized.
- **Values**:
  - *true / false*: Indicates that all data fields that belong to this group should not be historicized. Possible values are “true” if this annotation should be active or “false”, which is the same as removing the annotation. Use this annotation if you want to prevent storing history for all/many fields in a group. If you only want to prevent historization for a single/some field(s), use the annotation `no-history-field` at the field level.
    In CM versions up to 6.10.2, the DWH transfer of a field history is also controlled by this annotation.
    Starting with CM version 6.10.2, use the annotation `dwh-no-history` for this.

H.1.2.5 resource (type)

resource-fields-group-mode

- **Type**: resource
- **Description**: Controls the mode of a resource field group concerning editing via Web Client.
- **Values**:
  - *internal / external*: Possible values: internal, external. A resource field is not editable in the Web Client if value is “external”.
    Since 6.10.4.0
    Removed 6.10.5.0

resource-custom-fields-group-mode

- **Type**: resource
- **Description**: Controls the mode of a resource field group concerning editing via Web Client.
- **Values**:
  - *internal / external*: Possible values: internal, external. A resource field is not editable in the Web Client if value is “external”.
    Since 6.10.5.0

H.1.2.6 restapi (type)

customer group exposure

- **Type**: restapi
- **Description**: Indicates whether a data field group should be available to customers using the REST API, e.g. in CM/Track.

- **Values**:
  - *full* (default): The data field group is available for reading and writing.
  - *read*: The data field group is available for reading only.

H.1.2.7 ticket contact relation (type)

unit is a contact

- **Type**: ticket contact relation
- **Description**: deprecated
- **Values**:
  - *true / false*: Removed in version 6.9.0.

H.2 System Properties

The following chapter provides detailed information about the system properties used in ConSol CM.

H.2.1 Alphabetical List of System Properties

This chapter describes the following properties:

admin.email

- **Module**: cmas-core-security
- **Description**: The email address of the ConSol CM administrator. The value which you entered during system set-up is used initially.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: myuser@consol.de
- **Since**: 6.0

admin.login

- **Module**: cmas-core-security
- **Description**: The name of the ConSol CM administrator. The value which you entered during system set-up is used initially.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: admin
- **Since**: 6.0

admin.tool.consumed.licences.check.interval

- **Module**: cmas-app-admin-tool
- **Description**: Sets the interval (in seconds) to monitor the number of consumed licenses. The default value is 30.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 30
- **Since**: 6.11.0.0

admin.tool.consumed.licences.pool.name

- **Module**: cmas-app-admin-tool
- **Description**: Sets the license pool name to monitor the number of consumed licenses. The default value is “CONCURRENT_USERS”.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: CONCURRENT_USERS
- **Since**: 6.11.0.0

admin.tool.session.check.interval

- **Module**: cmas-app-admin-tool
- **Description**: Admin Tool inactive (ended) sessions check time interval (in seconds)
- **Type**: integer
- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: 30
- **Since**: 6.7.5

attachment.allowed.types

- **Module**: cmas-core-server
- **Description**: Comma-separated list of allowed filename extensions (if no value defined, all file extensions are allowed).
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: txt,zip,doc
- **Since**: 6.5.0

attachment.max.size

- **Module**: cmas-core-server
- **Description**: Maximum attachment size, in MB. This is a validation property of the CM API. It controls the size of attachments at tickets, at units, and at resources. It also controls the size of incoming (not outgoing!) email attachments.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 100
- **Since**: 6.4.0

attachment.upload.timeout

- **Module**: cmweb-server-adapter
- **Description**: Defines the transaction timeout in minutes for adding attachments to a ticket, a resource or a customer. Counts the time for the upload of all attachments of one transaction. When the timeout occurs, all files which have been temporarily stored on the server are deleted. No file is uploaded.
- **Type**: Integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 3
- **Since**: 6.10.5.3

authentication.method

- **Module**: cmas-core-security
- **Description**: User authentication method (internal CM database or LDAP authentication). Allowed values are LDAP or DATABASE.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: DATABASE
- **Since**: 6.0

autocommit.cf.changes

- **Module**: cmas-dwh-server
- **Description**: Defines whether DWH tasks which result from configurational changes on ticket fields are executed automatically without manual interaction in the Admin Tool. Can be also set in the Admin Tool in the navigation item *DWH*. The default and recommended value is “false”.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.7.0

autocomplete.enabled

- **Module**: cmas-app-admin-tool
- **Description**: If the flag is missing or its value is “false”, then the *Autocomplete address* navigation item is hidden in Admin Tool.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: true
- **Since**: 6.9.2.0

automatic.booking.enabled

- **Module**: cmweb-server-adapter
- **Description**: If enabled, time spend on creating comment/email will be measured and automatic time booking will be added.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: true
- **Since**: 6.9.4.2

batch-commit-interval

- **Module**: cmas-dwh-server
- **Description**: Number of objects in a JMS message. Larger values mean better transfer performance at the cost of higher memory usage.
  Starting with ***ConSol CM*** version 6.11, this property is only used if the package size of a DWH operation is not set. This can only happen when the command is directly addressed to the Java MBean consol.cmas.global.dwh.synchronizationService, e.g. using the update() method. When a DWH operation is started using the Admin Tool, there is always a value for the package size. If not explicitly set, the default value of 1000 is used as value for the batch.-commit.interval.
- **Default value**: 1000
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 100
- **Since**: 6.0.0

big.task.minimum.size

- **Module**: cmas-core-index-common
- **Description**: Indicates the minimum size of index task (in parts, each part has 100 entities) to qualify this task as a big one. Big tasks have lower priority than normal tasks.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 15 (default)
- **Since**: 6.8.3

cache-cluster-name

- **Module**: cmas-core-cache
- **Description**: JBoss cache cluster name.
- **Type**: string
- **Restart required**: yes
- **System**: yes
- **Optional**: no

- **Example value**: 635a6de1-629a-4129-8299-2d98633310f0
- **Since**: 6.4.0

calendar.csv.dateFormat

- **Module**: cmas-core-server
- **Description**: Format of the date given in the csv file containing the list of holidays.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: dd/MM/yyyy
- **Since**: 6.9.3.2

calendar.csv.separator

- **Module**: cmas-core-server
- **Description**: Separator used in the csv file containing the list of holidays.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: ,
- **Since**: 6.9.3.2

checkUserOnlineIntervalInSeconds

- **Module**: cmweb-server-adapter
- **Description**: The interval in seconds to check which users are online (default 180sec = 3min).
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 180
- **Since**: 6.0
- **Removed in**: 6.5 / 6.11.0.1

cluster.mode

- **Module**: cmas-core-shared
- **Description**: Specifies whether CMAS is running in cluster.
- **Type**: boolean
- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.1.0

cluster.unicast

- **Module**: cmas-core-shared
- **Description**: Flag to activate jgroups unicast mode for ConSol CM clusters (as opposed to the default multicast mode causing problems in some data center environments). If set to “true” remember to set the JVM start parameters: `jgroups.bind.port`, `jgroups.bind.address` and `jgroups.initial_hosts`.
- **Type**: boolean
- **Restart required**: yes
- **System**: yes
- **Optional**: yes
- **Example value**: false (default)
- **Since**: 6.11.0.0

cmas.dropSchemaBeforeSetup

- **Module**: cmas-setup-hibernate
- **Description**: Flag if schema is to be (was) dropped during setup
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true
- **Since**: 6.0

cmoffice.enabled

- **Module**: cmweb-server-adapter
- **Description**: Flag if CM/Doc (former CM/Office) is enabled.

- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.4.0

cmoffice.oo.path.NUMBER

- **Module**: cmweb-server-adapter
- **Description**: Possible location of the OpenOffice installation. The properties are numbered starting with 0.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: c:\Program Files (x86)\LibreOffice 3.6\program
- **Since**: 6.10.1.0

cmoffice.strict.versioning.enabled

- **Module**: cmweb-server-adapter
- **Description**: Controls if the SAVE operation in Microsoft Word / OpenOffice documents creates a new attachment (“true”) or overwrites the existing attachment (“false”). This concerns the behavior within one session using the text editing program. If the program is stopped, the overwrite mechanism will not work anymore.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true
- **Since**: 6.10.5.4

comment.authors.disabled

- **Module**: cmas-restapi-core
- **Description**: Disables the display of the content's author via REST API. The default value is “false”.
- **Type**: boolean
- **Restart required**: no

- **System**: no
- **Optional**: yes
- **Example value**: false
- **Since**: 6.11.0

commentRequiredForTicketCreation

- **Module**: cmweb-server-adapter
- **Description**: Flag if comment is a required field for ticket creation.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true (default)
- **Since**: 6.2.0

communication.channel

- **Module**: cmas-dwh-server
- **Description**: Communication channel. Only possible value since CM version 6.11.0.0: DIRECT
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: DIRECT
- **Since**: 6.8.5.0
- **Removed in**: 6.11.0.0 (DIRECT mode is the only available mode and is set automatically)

config.data.version

- **Module**: cmas-core-server
- **Description**: The internal version number of the current system configuration. This property is maintained internally, please do not change it unless advised by ConSol.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 11
- **Since**: 6.0

config.import.global.transaction.enabled

- **Module**: cmas-core-server
- **Description**: Flag deciding whether configuration (without localizations) should be imported within single transaction.
- **Type**: Boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true
- **Since**: 6.11.1.0

connection.release.mode

- **Module**: cmas-setup-hibernate
- **Description**: Describes the JEE connection handling strategy for transactions. If set to “AFTER_TRANSACTION”, the connection will be cached during the transaction and released at the end. If set to “AFTER_STATEMENT”, the connection will be released to the pool after each statement execution. Please do not change the default here unless advised by ConSol.
- **Type**: string
- **Restart required**: yes
- **System**: no
- **Optional**: yes
- **Example value**: AFTER_STATEMENT (default for JEE environment)
- **Since**: 6.0

contact.authentication.method

- **Module**: cmas-core-security
- **Description**: Indicates contact authentication method, where possible values are DATABASE or LDAP or LDAP,DATABASE or DATABASE,LDAP.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Since**: 6.9.3.0

contact.inherit.permissions.only.to.own.customer.group

- **Module**: cmas-core-security
- **Description**: Indicates whether authenticated contact inherits all customer group permissions from the representing engineer (false) or only has permissions to his own customer group (true).
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Since**: 6.9.2.3

csrf.domain.white.list

- **Module**: cmweb-server-adapter
- **Description**: The list of domains (separated with '|') which are allowed and will not be checked by CSRF (cross-site request forgery) filter
- **Type**: String
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: example.com | consol.de
- **Since**: 6.10.7.0

csrf.request.filter.enabled

- **Module**: cmweb-server-adapter
- **Description**: It allows to disable CSRF (Cross-site request forgery) request filter
- **Type**: Boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true
- **Since**: 6.10.7.0

customizationVersion

- **Module**: cmweb-server-adapter
- **Description**: UID representing the latest web customization version. Used only internally, please do not change the value.
- **Type**: string

- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: cd58453e-f3cc-4538-8030-d15e8796a4a7
- **Since**: 6.5.0

dao.log.threshold.milliseconds

- **Module**: cmas-core-server
- **Description**: Used to configure database operation times logging. DAO methods whose execution take longer than the time set in this property (in milliseconds) are logged.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 500 (default)
- **Since**: 6.11.1.0

dao.log.username

- **Module**: cmas-core-server
- **Description**: Used to configure database operation times logging. The execution of DAO methods which are related to the user name stated in this property is logged. Only one user name can be provided.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: webadmin (default: empty string)
- **Since**: 6.11.1.0

data.directory

- **Module**: cmas-core-shared
- **Description**: Directory for CMAS data (e.g., index)
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no

- **Example value**: C:\Users\user\cmas
- **Since**: 6.0

data.optimization

- **Module**: cmweb-server-adapter
- **Description**: Defines optimization to be applied on response data. So far, the following values are supported (for setting more than one value, separate values by '|'): MINIFICATION and COMPRESSION. MINIFICATION minifies HTML data by e.g. stripping whitespaces and comments. COMPRESSION applies gzip compression to HTTP response. (Note: If you are running in cluster mode and want to test different configurations in parallel, you can set different values for each cluster node by specifying property `data.optimization.nodeId` to override default property.)
- **Type**: string
- **Restart required**: COMPRESSION can be switched on/off without restart, MINIFICATION requires restart.
- **System**: yes
- **Optional**: yes
- **Example value**: MINIFICATION|COMPRESSION

database.notification.enabled

- **Module**: cmas-core-index-common
- **Description**: Indicates whether index update database notification channel should be used instead of JMS.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.8.4.7

database.notification.redelivery.delay.seconds

- **Module**: cmas-core-index-common
- **Description**: In case of index update database notification channel, indicates notification redelivery delay when an exception occurs.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no

- **Example value**: 60
- **Since**: 6.8.4.7

database.notification.redelivery.max.attempts

- **Module**: cmas-core-index-common
- **Description**: In case of index update database notification channel, indicates maximum redelivery attempts when an exception occurs.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 60
- **Since**: 6.8.4.7

defaultAttachmentEntryClassName

- **Module**: cmweb-server-adapter
- **Description**: The default content entry class used to classify an attachment if no other class was set explicitly.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: DefaultTextElement
- **Since**: 6.9.2.0

defaultCommentClassName

- **Module**: cmas-core-server
- **Description**: Default text class name for comments.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**:
- **Since**: 6.3.0

defaultContentEntryClassName

- **Module**: cmweb-server-adapter
- **Description**: Default text class for new ACIMs.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: default_class
- **Since**: 6.3.0

defaultIncommingMailClassName

- **Module**: cmas-core-server
- **Description**: Default text class name for incoming emails.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Since**: 6.3.0

defaultNumberOfCustomFieldsColumns

- **Module**: cmweb-server-adapter
- **Description**: Default number of columns for ticket fields.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 3
- **Since**: 6.2.0

defaultOutgoingMailClassName

- **Module**: cmas-core-server
- **Description**: Default text class name for outgoing emails.
- **Type**: string
- **Restart required**: no
- **System**: no

- **Optional**: yes
- **Example value**:
- **Since**: 6.3.0

delete.ticket.enabled

- **Module**: cmas-app-admin-tool
- **Description**: Controls if the menu entry *Delete* is displayed in the context menu in the Admin Tool for the ticket list in ticket administration.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true
- **Since**: 6.9.4.0

diffTrackingEnabled

- **Module**: cmweb-server-adapter
- **Description**: Removed in ConSol CM version 6.11.
  Defines if parallel editing of a ticket by different engineers should be possible. Default is “true”.
  “false”: Previous way of handling changes when editing a ticket. If the ticket has been changed in the meantime, the current engineer will not be able to submit his changes without being forced to reload the page before submitting.
  “true”: New changes handling mode. If the ticket has been changed, this will not block the submission of other changes anymore. If the part of the ticket that was changed was exactly the part that is changed by the submitting engineer, then an information message will be displayed, but the ticket change will be persisted/stored anyway.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true (default)
- **Since**: 6.10.1
- **Removed in**: 6.11.0

diffTrackingEnabledForUnitAndResource

- **Module**: cmweb-server-adapter
- **Description**: Enables the prevention of concurrent modifications on units / resources.
- **Type**: boolean

- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 3
- **Since**: 6.11.0.0

diff.tracking.disabled

- **Module**: cmas-restapi-core
- **Description**: Fallback property for disabling diff tracking for CM/Track, which is history-based so it can be heavy.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: false (default)
- **Since**: 6.10.5.6

disable.admin.task.auto.commit

- **Module**: cmas-core-index-common
- **Description**: All tasks created for index update will be automatically executed right after creation.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.6.1

dwh.administration.refresh.interval.seconds

- **Module**: cmas-app-admin-tool
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes

- **Example value**: 10
- **Since**: 6.11.0.1

dwh.mode

- **Module**: cmas-dwh-server
- **Description**: Current mode for DWH data transfer. Possible values are OFF, ADMIN, LIVE
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: OFF
- **Since**: 6.0.1

engineer.description.cache.enabled

- **Module**: cmas-core-server
- **Description**: Defines whether user descriptions are cached. The default value is “true”, please do not change it unless advised by ConSol.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: false
- **Since**: 6.11.0

engineer.description.mode

- **Module**: cmas-core-server
- **Description**: Defines whether user names in the ticket history are taken from the database or dynamically rendered using templates. The default value “DYNAMIC” is a bit more costly from the performance perspective, while “PROTOCOL” is faster but returns historical names which might be outdated. Use “PROTOCOL” if you have lots of history entries from many different users.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: PROTOCOL
- **Since**: 6.11.0

engineer.description.template.name

- **Module**: cmas-core-server
- **Description**: Defines the name of the template which is used to render engineer names for display in the Web Client. The template has to be stored in the *Templates* section of the Admin Tool. Default “engineer description template name”.
- **Type**: String
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Since**: 6.11.0

eviction.event.queue.size

- **Module**: cmas-core-cache
- **Description**: The size of the queue holding cache events. The default value is 200000. It is recommended to increase the value slightly (up to 400000) on systems with high traffic or load.
- **Type**: integer
- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: 200000
- **Since**: 6.4.0

eviction.max.nodes

- **Module**: cmas-core-cache
- **Description**: Sets the maximum size of internal caches. The default value is 100000. Increasing it will lead to higher memory consumption and is not recommended unless explicitly advised by ConSol.
- **Type**: integer
- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: 100000
- **Since**: 6.4.0

eviction.wakeup.interval

- **Module**: cmas-core-cache
- **Description**: Sets the interval (in milliseconds) between two cache queue event processing cycles. The default value is 3000. It is recommended to decrease it (minimum is 1500) on systems with high traffic or load.
- **Type**: integer
- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: 3000
- **Since**: 6.4.0

expert.mode

- **Module**: cmas-core-shared
- **Description**: Switches expert mode on/off thereby unblocking/blocking expert features. E.g., only in expert mode, the CM system property `initialized` will be available.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.0

external.line.access.prefix

- **Module**: cmas-core-server
- **Description**: General prefix to dial before an area code. Set for each customer group separately.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 411
- **Since**: 6.9.3.0

favoritesSizeLimit

- **Module**: cmweb-server-adapter
- **Description**: Maximum number of items in Favorites list.

- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 10
- **Since**: 6.0

fetchLock.interval

- **Module**: cmas-workflow-jbpm
- **Description**:
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 5000
- **Removed in**: 6.8.0

fetchSize.strategy

- **Module**: cmas-core-server
- **Description**: Strategy for selecting the fetch size on JDBC result sets.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: FetchSizePageBasedStrategy, FetchSizeThresholdStrategy, FetchSizeFixedStrategy
- **Since**: 6.8.4.1

fetchSize.strategy.FetchSizeFixedStrategy.value

- **Module**: cmas-core-server
- **Description**: Sets fetch size value if the selected strategy to set the fetch size is `FetchSizeFixedStrategy`.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes

- **Example value**: 150
- **Since**: 6.8.4.1

fetchSize.strategy.FetchSizePageBasedStrategy.limit

- **Module**: cmas-core-server
- **Description**: Sets maximum fetch size value if the selected strategy to set the fetch size is `FetchSizePageBasedStrategy`.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 10000
- **Since**: 6.8.4.1

fetchSize.strategy.FetchSizeThresholdStrategy.value

- **Module**: cmas-core-server
- **Description**: Sets fetch size threshold border values if the selected strategy to set the fetch size is `FetchSizeThresholdStrategy`.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 150,300,600,1000
- **Since**: 6.8.4.1

filesystem.polling.threads.number

- **Module**: cmas-nimh
- **Description**: Number of threads started for db emails' queue polling. Default: 1
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 10
- **Since**: 6.4.0

filesystem.polling.threads.shutdown.timeout.seconds

- **Module**: cmas-nimh
- **Description**: Waiting time after the shutdown signal. When the timeout reached, thread will be terminated. Default: 60
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.4.0

filesystem.polling.threads.watchdog.interval.seconds

- **Module**: cmas-nimh
- **Description**: Watchdog thread interval. Default: 30
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.4.0

filesystem.task.enabled

- **Module**: cmas-nimh
- **Description**: With this property service thread related to given poller can be disabled. Default: true
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true
- **Since**: 6.4.0

filesystem.task.interval.seconds

- **Module**: cmas-nimh
- **Description**: Default interval for polling mailboxes. Default: 60 seconds
- **Type**: integer

- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.4.0

filesystem.task.polling.folder

- **Module**: cmas-nimh
- **Description**: Polling folder location which will be scanned for emails in the format of eml files. Default: "mail" subdir of cmas data directory
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: c://cmas//mail
- **Since**: 6.4.0

filesystem.task.timeout.seconds

- **Module**: cmas-nimh
- **Description**: After this time (of inactivity) the service thread is considered as damaged and automatically restarted. Default: 120 seconds
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.4.0

filesystem.task.transaction.timeout.seconds

- **Module**: cmas-nimh
- **Description**: Default transaction timeout for email fetching transactions. Should be correlated with number of messages fetched at once. Default: 60 seconds
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes

- **Example value**: 60
- **Since**: 6.4.0

forward.mails.to.representatives

- **Module**: cmweb-server-adapter
- **Description**: Determines if emails which are manually sent from the Web Client are also sent to representing engineers. The default value of the property is “false”, meaning that this kind of emails are not forwarded to the representing engineer. Set the property to “true” if you want to restore the previous behavior, i.e., all emails which are sent to the represented engineer are automatically forwarded to the representing engineer. Please take into account that this might not be desired if the same person is an engineer and a customer in the CM system.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: false (default)
- **Since**: 6.11.1.7

This property only configures the handling of manually sent emails. The handling of automatically sent emails depends on the used Java method.

globalSearchResultSizeLimit

- **Module**: cmweb-server-adapter
- **Description**: Maximum number of items in Quick Search result.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 10
- **Since**: 6.0

heartbeat

- **Module**: cmas-core-server
- **Description**: Timestamp that indicates if an instance of the application is connected to the database schema.
- **Type**: integer
- **Restart required**: no

- **System**: yes
- **Optional**: no
- **Example value**: 1234567899
- **Since**: 6.10.5.3

helpFilePath

- **Module**: cmweb-server-adapter
- **Description**: URL for online help. If not empty, Help button is displayed in Web Client.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: http://www.consol.de
- **Since**: 6.2.1

hibernate.dialect

- **Module**: cmas-setup-hibernate
- **Description**: The dialect used by hibernate. Usually set during initial set-up (depending on the database system).
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: org.hibernate.dialect.MySQL5InnoDBDialect
- **Since**: 6.0

hideTicketSubject

- **Module**: cmweb-server-adapter
- **Description**: If set to “true”, ticket subject is hidden.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.2.1

ignore-queues

- **Module**: cmas-dwh-server
- **Description**: A comma-separated list of queue names which are not not transferred to the DWH.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: QueueName1,QueueName2,QueueName3
- **Since**: 6.6.19
- **Removed in**: 6.8.1

index.attachment

- **Module**: cmas-core-index-common
- **Description**: Specifies whether content of attachments is indexed.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true
- **Since**: 6.4.3

index.history

- **Module**: cmas-core-index-common
- **Description**: Specifies whether unit and ticket history are indexed.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.1.0
- **Removed in**: 6.11.0

index.status

- **Module**: cmas-core-index-common
- **Description**: Status of the Indexer, possible values RED, YELLOW, GREEN, will be displayed in the Admin Tool.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: GREEN
- **Since**: 6.6.1

index.task.worker.threads

- **Module**: cmas-core-index-common
- **Description**: How many threads will be used to execute index tasks (synchronization, administrative, and repair tasks).
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 1 (default) (we recommend to use a value not larger than 2)
- **Since**: 6.6.14, 6.7.3. Since 6.8.0 and exclusively in 6.6.21 also normal (live) index updates are affected by this property.

index.version.current

- **Module**: cmas-core-index-common
- **Description**: Holds information about current (possibly old) index version.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 1 (default)
- **Since**: 6.7.0

index.version.newest

- **Module**: cmas-core-index-common
- **Description**: Holds information about which index version is considered newest.

- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 1 (default)
- **Since**: 6.7.0

indexed.assets.per.thread.in.memory

- **Module**: cmas-core-index-common
- **Description**: How many assets should be loaded into memory at once, per thread, during indexing.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 200 (default)
- **Since**: 6.8.0

indexed.engineers.per.thread.in.memory

- **Module**: cmas-core-index-common
- **Description**: How many engineers should be loaded into memory at once, per thread, during indexing.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 300 (default)
- **Since**: 6.6.14, 6.7.3

indexed.resources.per.thread.in.memory

- **Module**: cmas-core-index-common
- **Description**: How many resources should be loaded into memory at once, per thread, during indexing.
- **Type**: integer
- **Restart required**: no
- **System**: yes

- **Optional**: no
- **Example value**: 200 (default)
- **Since**: 6.10.0.0

indexed.tickets.per.thread.in.memory

- **Module**: cmas-core-index-common
- **Description**: How many tickets should be loaded into memory at once, per thread, during indexing.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 100 (default)
- **Since**: 6.6.14, 6.7.3

indexed.units.per.thread.in.memory

- **Module**: cmas-core-index-common
- **Description**: How many units should be loaded into memory at once, per thread, during indexing.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 200 (default)
- **Since**: 6.6.14, 6.7.3

initialized

- **Module**: cmas-setup-manager
- **Description**: Flag if CMAS is initialized. If this value is missing or not “true”, set-up will be performed. Starting with ConSol CM version 6.11, this property is only available in expert.mode.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true
- **Since**: 6.0

> ⚠ Be careful with using this property!!! When you set the value to “false”, the ConSol CM server will perform the system set-up at the next start, i.e. all data of the existing system is lost, including system properties!!!

internal.line.access.prefix

- **Module**: cmas-core-server
- **Description**: Prefix that the company’s telephony system asks for outside lines. Set for each customer group separately.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 199
- **Since**: 6.9.3.0

is.cmrf.alive

- **Module**: cmas-dwh-server
- **Description**: As a starting point, the time the last message was sent to CMRF should be used. If a response from CMRF is not received after value (in seconds), it should create a DWH operation status with an error message indicating that CMRF is down.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 1200
- **Since**: 6.7.0

java.naming.factory.initial

- **Module**: cmas-dwh-server
- **Description**: Factory class for the DWH context factory.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: org.jnp.interfaces.NamingContextFactory

- **Since**: 6.0.1
- **Removed in**: 6.11.0.0

java.naming.factory.url.pkgs

- **Module**: cmas-dwh-server
- **Description**:
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: org.jboss.naming:org.jnp.interfaces
- **Since**: 6.0.1
- **Removed in**: 6.11.0.0

java.naming.provider.url

- **Module**: cmas-dwh-server
- **Description**: URL of naming provider.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: localhost
- **Since**: 6.0.1
- **Removed in**: 6.11.0.0

jobExecutor.adminMail

- **Module**: cmas-workflow-engine
- **Description**: Email address which will get notified about job execution problems (when retry counter is exceeded).
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: admin@consol.de
- **Since**: 6.8.0

jobExecutor.idleInterval

- **Module**: cmas-workflow-jbpm
- **Description**:
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 45000
- **Removed in**: 6.8.0
- **Replaced by**: jobExecutor.idleInterval.seconds

jobExecutor.idleInterval.seconds

- **Module**: cmas-workflow-engine
- **Description**: Determines how often job executor thread will look for new jobs to execute.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 45 (default up to CM version 6.10.5.2. Default CM versions 6.10.5.3 and up is 5)
- **Since**: 6.8.0

jobExecutor.jobExecuteRetryNumber

- **Module**: cmas-workflow-jbpm
- **Description**:
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 5
- **Removed in**: 6.8.0
- **Replaced by**: jobExecutor.jobMaxRetries

jobExecutor.jobMaxRetries

- **Module**: cmas-workflow-engine
- **Description**: Controls the number of retry attempts the job executor will do before declaring a job as failed.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 5 (default)
- **Since**: 6.8.0

jobExecutor.jobMaxRetriesReachedSubject

- **Module**: cmas-workflow-engine
- **Description**: The subject used in the notification mail admins receive about failed job executors.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: Job maximum retries reached. Job was removed!!! (default)
- **Since**: 6.8.0

jobExecutor.lockingLimit

- **Module**: cmas-workflow-engine
- **Description**: Number of jobs locked at once (marked for execution) by job executor thread.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 5 (default since CM version 6.10.5.3)
- **Since**: 6.8.0

jobExecutor.lockTimeout.seconds

- **Module**: cmas-workflow-engine
- **Description**: How long the job can be locked (marked for execution) by job executor.
- **Type**: integer

- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 360 (default)
- **Since**: 6.8.0

jobExecutor.mailFrom

- **Module**: cmas-workflow-engine
- **Description**: Email which will be set as From header during admin notifications.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: jobexecutor@consol.de
- **Since**: 6.8.0

jobExecutor.maxInactivityInterval.minutes

- **Module**: cmas-workflow-engine
- **Description**: Number of minutes of allowed job executor inactivity (e.g. when it is blocked by long timer execution). After this time executors threads are restarted.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes. Default value is set to 30 minutes
- **Example value**: 15 (default)
- **Since**: 6.9.2.0

jobExecutor.threads

- **Module**: cmas-workflow-engine
- **Description**: Number of job execution threads.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 1 (default)
- **Since**: 6.8.0

jobExecutor.timerRetryInterval

- **Module**: cmas-workflow-jbpm
- **Description**:
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 10000
- **Removed in**: 6.8.0
- **Replaced by**: jobExecutor.timerRetryInterval.seconds

jobExecutor.timerRetryInterval.seconds

- **Module**: cmas-workflow-engine
- **Description**: Determines how long job executor thread will wait after job execution error.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 10 (default up to CM version 6.10.5.2. Default CM versions 6.10.5.3 and up is 30)
- **Since**: 6.8.0

jobExecutor.txTimeout.seconds

- **Module**: cmas-workflow-engine
- **Description**: Transaction timeout used for job execution.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 60 (default)
- **Since**: 6.8.0

kerberos.v5.enabled

- **Module**: cmas-core-security
- **Description**: Indicates whether SSO via Kerberos is enabled.
- **Type**: boolean

- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false (default if Kerberos was not enabled during system set-up)
- **Since**: 6.2.0

kerberos.v5.username.regex

- **Module**: cmas-core-security
- **Description**: Regular expression used for mapping Kerberos principals to CM user login names.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: (.*)@.*
- **Since**: 6.2.0

last.config.change

- **Module**: cmas-core-server
- **Description**: Random UUID created during the last configuration change. This is a value maintained internally, please do not change it unless advised by ConSol.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 2573c7b7-2bf5-47ff-b5a2-bad31951a266
- **Since**: 6.1.0, 6.2.1

last.config.change.templates

- **Module**: cmas-core-server
- **Description**: Random UUID created during the last change in templates. This is a value maintained internally, please do not change it unless advised by ConSol.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no

- **Example value**: 2573c7c7-2af5-4eff-b9c2-bad31951a266
- **Since**: 6.10.5.0

last.ping.timestamp

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 32323214
- **Since**: 6.11.0.1

ldap.authentication

- **Module**: cmas-core-security
- **Description**: Authentication method used when using LDAP authentication. Possible values are 'anonymous' and 'simple' (default).
- **Type**: string
- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: simple
- **Since**: 6.0

ldap.basedn

- **Module**: cmas-core-security
- **Description**: Base DN used for looking up LDAP user accounts when using LDAP authentication.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: ou=accounts,dc=consol,dc=de
- **Since**: 6.0

ldap.certificate.basedn

- **Module**: cmas-core-server
- **Description**: Base DN for certificates location in the LDAP tree. If not provided, `cmas-core-security, ldap.basedn` is used.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: ou=accounts,dc=consol,dc=de
- **Since**: 6.8.4

ldap.certificate.content.attribute

- **Module**: cmas-core-server
- **Description**: LDAP attribute name used where certificate data is stored in the LDAP tree. Default value: usercertificate
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: usercertificate
- **Since**: 6.8.4

ldap.certificate.password

- **Module**: cmas-core-server
- **Description**: LDAP Certificates manager password. If not set, `cmas-core-security, ldap.password` is used.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.8.4

ldap.certificate.providerurl

- **Module**: cmas-core-server
- **Description**: LDAP Certificates provider URL. If not set, `cmas-core-security, ldap.providerurl` is used.
- **Type**: string

- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: ldap://ldap.consol.de:389
- **Since**: 6.8.4

ldap.certificate.searchattr

- **Module**: cmas-core-server
- **Description**: LDAP attribute name used to search for certificate in the LDAP tree. Default value: mail
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: mail
- **Since**: 6.8.4

ldap.certificate.userdn

- **Module**: cmas-core-server
- **Description**: LDAP Certificates manager DN. If not set, `cmas-core-security, ldap.userdn` is used.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.8.4

ldap.contact.name.basedn

- **Module**: cmas-core-security
- **Description**: Base path to search for contact DN by LDAP ID (e.g. ou=a-ccounts,dc=consol,dc=de).
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Since**: 6.9.3.0

ldap.contact.name.password

- **Module**: cmas-core-security
- **Description**: Password to look up contact DN by LDAP ID. If not set, the anonymous account is used.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Since**: 6.9.3.0

ldap.contact.name.providerurl

- **Module**: cmas-core-security
- **Description**: Address of the LDAP server (ldap[s]://host:port).
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Since**: 6.9.3.0

ldap.contact.name.searchattr

- **Module**: cmas-core-security
- **Description**: Attribute to search for contact DN by LDAP ID (e.g. uid).
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Since**: 6.9.3.0

ldap.contact.name.userdn

- **Module**: cmas-core-security
- **Description**: User DN to look up contact DN by LDAP ID. If not set, the anonymous account is used.
- **Type**: string
- **Restart required**: no
- **System**: no

- **Optional**: yes
- **Since**: 6.9.3.0

ldap.initialcontextfactory

- **Module**: cmas-core-security
- **Description**: Class name for the initial context factory of the LDAP implementation when using LDAP authentication. If it is not set, `com.sun.jndi.ldap.LdapCtxFactory` is used.
- **Type**: string
- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: com.sun.jndi.ldap.LdapCtxFactory
- **Since**: 6.0

ldap.password

- **Module**: cmas-core-security
- **Description**: Password for connecting to LDAP to look up users when using LDAP authentication. Only needed if look-up cannot be performed anonymously.
- **Type**: password
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.1.2

ldap.providerurl

- **Module**: cmas-core-security
- **Description**: LDAP provider when using LDAP authentication.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: ldap://myserver.consol.de:389
- **Since**: 6.0

ldap.searchattr

- **Module**: cmas-core-security
- **Description**: Search attribute for looking up LDAP entry associated with a CM login.

- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: uid
- **Since**: 6.0

ldap.userdn

- **Module**: cmas-core-security
- **Description**: LDAP user for connecting to LDAP to look up users when using LDAP authentication. Only needed if look-up cannot be performed anonymously.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.1.2

live.start

- **Module**: cmas-dwh-server
- **Description**: When the DWH synchronization mode is set to LIVE using the Admin Tool (navigation group *Data Warehouse*, navigation item *Administration*, *Configuration* button), this property is created and set to the current date.
  If LIVE mode is not enabled and there is no data in `cmas_dwh_ser_sync_object`, the property `live.start` is deleted.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes (automatically added in DWH “LIVE” mode)
- **Example value**: 15028802377645
- **Since**: 6.7.0

local.country.prefix

- **Module**: cmas-core-server
- **Description**: Prefix of the local country code. Set for each customer group separately.
- **Type**: integer
- **Restart required**: no
- **System**: no

- **Optional**: yes
- **Example value**: 48
- **Since**: 6.9.3.0

mail.attachments.validation.info.sender

- **Module**: cmas-nimh-extension
- **Description**: Sets From header of attachments type *error notification mail*
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: admin@mail.com
- **Since**: 6.7.5

mail.attachments.validation.info.subject

- **Module**: cmas-nimh-extension
- **Description**: Sets subject of attachments type error notification mail.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: Email was not processed because its attachments were rejected!
- **Since**: 6.7.5

mail.db.archive

- **Module**: cmas-nimh-extension
- **Description**: If property is set to “true”, incoming emails are archived in the database.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: false (default)
- **Since**: 6.8.5.5

mail.encryption

- **Module**: cmas-core-server
- **Description**: If property is set to “true”, the encrypt checkbox in the Ticket Email Editor is checked by default.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true (default = false)
- **Since**: 6.8.4.0

mail.error.from.address

- **Module**: cmas-nimh-extension
- **Description**: From address for error emails from NIMH
- **Type**: email
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: myuser@consol.de
- **Since**: 6.4.0

mail.error.to.address

- **Module**: cmas-nimh-extension
- **Description**: To address for error emails from NIMH. As a default the email address of the administrator which you have entered during system setup is used.
- **Type**: email
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: myuser@consol.de
- **Since**: 6.4.0

mail.from

- **Module**: cmweb-server-adapter
- **Description**: Use this address if set instead of engineer email address during email conversation.

- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.1.2

mail.notification.engineerChange

- **Module**: cmas-core-server
- **Description**: Whether notification emails should be sent when the engineer of a ticket is changed.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true
- **Since**: 6.1.0

mail.notification.sender

- **Module**: cmas-core-server
- **Description**: From address for notification emails when the engineer of a ticket is changed. If not set, `cmas-core-security, admin.email` is used instead.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: cm6notification@cm6installation
- **Since**: 6.6.3

mail.on.error

- **Module**: cmas-nimh-extension
- **Description**: If set to “true ”an error email is sent to the above configured address in case the email message could not be processed. Default: true
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes

- **Example value**: false
- **Since**: 6.4.0

mail.process.error

- **Module**: cmas-nimh-extension
- **Description**: To address for error emails from Mule.
- **Type**: email
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: myuser@consol.de
- **Since**: 6.4.0

mail.redelivery.retry.count

- **Module**: cmas-core-server
- **Description**: Number of redelivery attempts of an outgoing email.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 3
- **Since**: 6.1.0

mail.reply.to

- **Module**: cmweb-server-adapter
- **Description**: When set, Web Client will display Reply-To field on email send, prefilled with this value.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.0.1

> ⚠ Please read the detailed information about ConSol CM Reply-To addresses in section Scripts of Type Email.

mail.sender.address

- **Module**: cmas-workflow-jbpm
- **Description**: From address for emails from the workflow engine.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: myuser@consol.de
- **Removed in**: 6.8.0
- **Replaced by**: jobExecutor.mailFrom

mail.smtp.email

- **Module**: cmas-core-server
- **Description**: SMTP email URL for outgoing emails
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: smtp://mail.mydomain.com:25
- **Since**: 6.0

mail.smtp.envelopesender

- **Module**: cmas-core-server
- **Description**: Email address used as sender in SMTP envelope. If not set, the From address of the email is used.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: mysender@mydomain.com
- **Since**: 6.5.7

mail.smtp.tls.enabled

- **Module**: cmas-core-server
- **Description**: Activates SMTP via SSL/TLS (SMTPS) for sending emails from the Web Client and scripts. The default value is “false”. If it is set to “true”, SMTPS is activated for sending emails.

- **Type**: boolean
- **Restart required**: yes
- **System**: no
- **Optional**: yes
- **Example value**: true
- **Since**: 6.11.1.6

mail.ticketname.pattern

- **Module**: cmas-nimh-extension
- **Description**: Regular expression pattern used to identify the ticket name in the subject of incoming mails.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: .*?Ticket\s+\((\S+)\).*
- **Since**: 6.4.0

mailbox.1.connection.host

- **Module**: cmas-nimh
- **Description**: Host (server) for first configured mailbox. Will overwrite the default parameter `mailbox.default.connection.host`.

mailbox.1.connection.password

- **Module**: cmas-nimh
- **Description**: Password for first configured mailbox. Will overwrite the default parameter `mailbox.default.connection.password`.

mailbox.1.connection.port

- **Module**: cmas-nimh
- **Description**: Port for first configured mailbox. Will overwrite the default parameter `mailbox.default.connection.port`.

mailbox.1.connection.protocol

- **Module**: cmas-nimh
- **Description**: Protocol (e.g., IMAP or POP3) for first configured mailbox. Will overwrite the default parameter `mailbox.default.connection.protocol`.

mailbox.1.connection.username

- **Module**: cmas-nimh
- **Description**: User name for first configured mailbox. Will overwrite the default parameter `mailbox.default.connection.username`.

mailbox.2.connection.host

- **Module**: cmas-nimh
- **Description**: Host (server) for second configured mailbox. Will overwrite the default parameter `mailbox.default.connection.host`.

mailbox.2.connection.password

- **Module**: cmas-nimh
- **Description**: Password for second configured mailbox. Will overwrite the default parameter `mailbox.default.connection.password`.

mailbox.2.connection.port

- **Module**: cmas-nimh
- **Description**: Port for second configured mailbox. Will overwrite the default parameter `mailbox.default.connection.port`.

mailbox.2.connection.protocol

- **Module**: cmas-nimh
- **Description**: Protocol (e.g., IMAP or POP3) for second configured mailbox. Will overwrite the default parameter `mailbox.default.connection.protocol`.

mailbox.2.connection.username

- **Module**: cmas-nimh
- **Description**: User name for second configured mailbox. Will overwrite the default parameter `mailbox.default.connection.username`.

For all NIMH-related mailbox properties, the following principle is used: a default property is defined (e.g. `mailbox.default.connection.port`). If no mailbox-specific value is configured, this default value will be used.

mailbox.default.connection.host

- **Module**: cmas-nimh
- **Description**: Host (server name) of a given mailbox from which the poller reads emails.
- **Type**: string
- **Restart required**: no

- **System**: no
- **Optional**: yes
- **Example value**: 10.10.1.157
- **Since**: 6.4.0

mailbox.default.connection.password

- **Module**: cmas-nimh
- **Description**: Password for given mailbox from which the poller reads emails.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: consol
- **Since**: 6.4.0

mailbox.default.connection.port

- **Module**: cmas-nimh
- **Description**: Port for a given mailbox from which the poller reads emails.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 143
- **Since**: 6.4.0

mailbox.default.connection.protocol

- **Module**: cmas-nimh
- **Description**: Poller's protocol e.g., IMAP or POP3. No default value
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: imap
- **Since**: 6.4.0

mailbox.default.connection.username

- **Module**: cmas-nimh
- **Description**: User name for a given mailbox from which the poller reads emails.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: username
- **Since**: 6.4.0

mailbox.default.session.mail.debug

- **Module**: cmas-nimh
- **Description**: Example javax.mail property - allows for more detailed javax.mail session debugging
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true
- **Since**: 6.4.0

mailbox.default.session.mail.imap.timeout

- **Module**: cmas-nimh
- **Description**: Example javax.mail property
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 120
- **Since**: 6.4.0

mailbox.default.session.mail.mime.address.strict

- **Module**: cmas-nimh
- **Description**: Example javax.mail property - counterpart of the old `mule mail.mime.strict`, allows to set not so strict email header parsing
- **Type**: boolean

- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true
- **Since**: 6.4.0

mailbox.default.session.mail.pop3.timeout

- **Module**: cmas-nimh
- **Description**: Example javax.mail property.
- **Type**:
- **Restart required**:
- **System**:
- **Optional**:
- **Example value**:
- **Since**: 6.4.0

mailbox.default.session.mail.PROTOCOL.fetchsize

- **Module**: cmas-nimh
- **Description**: Sets java mail property for partialfetch size in bytes for the indicated protocol.
  For IMAP systems: in CM versions 6.10.7.0 and up, the value of `mailbox.default.session.mail.imap.fetchsize` is set to 1048576 (equals 1 MB) during the initial setup of a ConSol CM system. During an update of an existing ConSol CM system, the value of the property is left unchanged, if the property is already present. In case the property is not yet present, it is added with the default value.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 1048576
- **Since**: 6.9.4.0

mailbox.default.session.mail.PROTOCOL.partialfetch

- **Module**: cmas-nimh
- **Description**: Sets java mail property for partialfetch i.e. controls whether the protocol partial-fetch capability should be used.
  For IMAP systems: in CM versions 6.10.7.0 and up, the value of `mailbox.default.session.mail.imap.partialfetch` is set to “false” during

the initial setup of a ConSol CM system. During an update of an existing ConSol CM system, the value of the property is left unchanged, if the property is already present. In case the property is not yet present, it is added with the default value.

- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**:
- **Since**: 6.9.4.0

mailbox.default.task.delete.read.messages

- **Module**: cmas-nimh
- **Description**: This defines whether messages should be removed from the mailbox after processing. For IMAP protocol messages are marked as SEEN by default. For POP3 protocol, when flag is set to true the message is removed, otherwise remains on server and will result in infinite reads. Default: false.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: false
- **Since**: 6.4.0

mailbox.default.task.enabled

- **Module**: cmas-nimh
- **Description**: With this property service thread related to given poller can be disabled. Default: true
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: false
- **Since**: 6.4.0

mailbox.default.task.interval.seconds

- **Module**: cmas-nimh
- **Description**: Default interval for polling mailboxes. Default: 60 seconds

- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.4.0

mailbox.default.task.max.message.size

- **Module**: cmas-nimh
- **Description**: Maximum size of email messages (i.e., email plus attachment). Emails exceeding the size limit will not be automatically processed by NIMH but will be stored in the database (table `cmas_nimh_archived_mail`) and will therefore appear in the email backups in the Admin Tool (see section Email Backups). From there they can be resent, downloaded to the file system, or deleted. For those operations the message size is not relevant. Default is set to 10MB: 10485760
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 10485760
- **Since**: 6.4.0

mailbox.default.task.max.messages.per.run

- **Module**: cmas-nimh
- **Description**: Number of messages fetched at once from mailbox. Must be correlated with transaction timeout. Default set to: 20
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.4.0

mailbox.default.task.timeout.seconds

- **Module**: cmas-nimh
- **Description**: After this time (of inactivity) the service thread is considered as damaged and automatically restarted. Default: 120 seconds
- **Type**: integer

- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.4.0

mailbox.default.task.transaction.timeout.seconds

- **Module**: cmas-nimh
- **Description**: Default transaction timeout for email fetching transactions. Should be correlated with number of messages fetched at once. Default: 60 seconds
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.4.0

mailbox.polling.threads.mail.log.enabled

- **Module**: cmas-nimh
- **Description**: Enables email logging which is especially crucial in cluster environment (used as semaphore there)
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true (default)
- **Since**: 6.9.4.1

mailbox.polling.threads.number

- **Module**: cmas-nimh
- **Description**: Number of threads for accessing mailboxes. Default: 1
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes

- **Example value**: 1
- **Since**: 6.4.0

mailTemplateAboveQuotedText

- **Module**: cmweb-server-adapter
- **Description**: Indicates behavior of email template in the Ticket Email Editor when another email is quoted, i.e. forwarded or replied to. Often used to place the signature correctly.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.2.4

max.licences.perUser

- **Module**: cmas-core-server
- **Description**: Sets maximum licenses single user can use (e.g., logging in from different browsers). By default this value is not restricted.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 10
- **Since**: 6.8.4.5

maxSizePerPagemapInMegaBytes

- **Module**: cmweb-server-adapter
- **Description**: The parameter defines the size (in MB) if the file which is created by the Wicket framework per user session. i.e. for each engineer which is currently logged in. The file is used to save pages during the running session. When the defined size limit has been reached and new entries are added, the oldest entries are removed. In the Web Client, due to this behavior, an engineer who works with an "old" page will be redirected to the *Overview*/Start page (usually the dashboard page) when the "old" page is removed from the file.
  So in case engineers who work with a great number of open tabs in ConSol CM and complain about being redirected to the *Overview* page, it might be useful to increase this parameter. In large systems, you could use e.g. a value of 45 or 50. Since this is the size of the file which is saved on disk, the maximum value depends on the available disk space, however, a value which is too large is not recommended either.
- **Type**: integer

- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: 15
- **Since**: 6.3.5

monitoring.engineer.login

- **Module**: cmas-core-server
- **Description**: Login of monitoring engineer.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: nagios
- **Since**: 6.9.3.0

monitoring.unit.login

- **Module**: cmas-core-server
- **Description**: Login of monitoring unit.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: nagios
- **Since**: 6.9.3.0

nimh.enabled

- **Module**: cmas-core-server
- **Description**: Enables NIMH service. Must be suffixed with the cluster node ID, e.g., `nimh.enabled.NODEID` = “true”.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: false
- **Since**: 6.9.4.0

notification.error.description

- **Module**: cmas-dwh-server
- **Description**: Text for error emails from the DWH.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: Error occurred
- **Since**: 6.0.1

notification.error.from

- **Module**: cmas-dwh-server
- **Description**: From address for error emails from the DWH
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.0.1

notification.error.subject

- **Module**: cmas-dwh-server
- **Description**: Subject for error emails from the DWH
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: Error occurred
- **Since**: 6.0.1

notification.error.to

- **Module**: cmas-dwh-server
- **Description**: To address for error emails from the DWH
- **Type**: string
- **Restart required**: no
- **System**: yes

- **Optional**: no
- **Example value**: myuser@consol.de
- **Since**: 6.0.1

notification.finished_successfully.description

- **Module**: cmas-dwh-server
- **Description**: Text for emails from the DWH when a transfer finishes successfully.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: Transfer finished successfully
- **Since**: 6.0.1

notification.finished_successfully.from

- **Module**: cmas-dwh-server
- **Description**: From address for emails from the DWH when a transfer finishes successfully.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.0.1

notification.finished_successfully.subject

- **Module**: cmas-dwh-server
- **Description**: Subject for emails from the DWH when a transfer finishes successfully.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: Transfer finished successfully
- **Since**: 6.0.1

notification.finished_successfully.to

- **Module**: cmas-dwh-server
- **Description**: To address for emails from the DWH when a transfer finishes successfully.

- **Type**: string
- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: myuser@consol.de
- **Since**: 6.0.1

notification.finished_unsuccessfully.description

- **Module**: cmas-dwh-server
- **Description**: Text for emails from the DWH when a transfer finishes unsuccessfully.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: Transfer finished unsuccessfully
- **Since**: 6.0.1

notification.finished_unsuccessfully.from

- **Module**: cmas-dwh-server
- **Description**: From address for emails from the DWH when a transfer finishes unsuccessfully.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.0.1

notification.finished_unsuccessfully.subject

- **Module**: cmas-dwh-server
- **Description**: Subject for emails from the DWH when a transfer finishes unsuccessfully.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: Transfer finished unsuccessfully
- **Since**: 6.0.1

notification.finished_unsuccessfully.to

- **Module**: cmas-dwh-server
- **Description**: To address for emails from the DWH when a transfer finishes unsuccessfully.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: myuser@consol.de
- **Since**: 6.0.1

notification.host

- **Module**: cmas-dwh-server
- **Description**: Email (SMTP) server hostname for sending DWH emails.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: myserver.consol.de
- **Since**: 6.0.1

notification.password

- **Module**: cmas-dwh-server
- **Description**: Password for sending DWH emails (optional).
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.0.1

notification.port

- **Module**: cmas-dwh-server
- **Description**: SMTP port for sending DWH emails.
- **Type**: string
- **Restart required**: no
- **System**: yes

- **Optional**: yes
- **Example value**: 25
- **Since**: 6.0.1

notification.protocol

- **Module**: cmas-dwh-server
- **Description**: The protocol used for sending emails from the DWH.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: pop3\

notification.tls.enabled

- **Module**: cmas-dwh-server
- **Description**: Activates SMTP via SSL/TLS (SMTPS) for sending notification emails from the DWH. The default value is “false”. If it is set to “true”, SMTPS is activated for sending notifications from the DWH.
- **Type**: string
- **Restart required**: yes
- **System**: no
- **Optional**: yes
- **Example value**: false
- **Since**: 6.11.1.6

notification.username

- **Module**: cmas-dwh-server
- **Description**: (SMTP) User name for sending DWH emails.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: myuser
- **Since**: 6.0.1

number.of.tasks

- **Module**: cmas-core-server
- **Description**: Number of threads to use by the Task Execution Framework (TEF).
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 1
- **Since**: 6.9.4.0

outdated.lock.age

- **Module**: cmas-workflow-jbpm
- **Description**:
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 60000
- **Removed in**: 6.8.0
- **Replaced by**: cmas-workflow-engine, jobExecutor.lockTimeout.seconds

pagemapLockDurationInSeconds

- **Module**: cmweb-server-adapter
- **Description**: Number of seconds to pass before pagemap is considered to be locked for too long.
- **Type**: integer
- **Restart required**: yes
- **System**: yes
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.7.3

password.reset.mail.from

- **Module**: cmas-core-security
- **Description**: The `From` address for the email which is sent to a customer who requests a new password (using the *Forgot your password?* link) in CM/Track and to an engineer who requests a new password (using the *Forgot your password?* link) in the Web Client.
- **Type**: String
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: mypwreset@consol.de
- **Since**: 6.11.0.1

policy.password.age

- **Module**: cmas-core-security
- **Description**: Maximum validity period, in number of days, example 183 (6 months), default value: 5500 (= 15 years, i.e. no password change enforced). In case you would like to have the engineer change his/her password asap, use one of the two following values:
  - **0**
    The engineer will be forced to change his/her password on the next login.
  - **1**
    The engineer will be forced to change his/her password the next day.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 5500 (15 years, default)
- **Since**: 6.10.1.0

policy.password.pattern

- **Module**: cmas-core-security
- **Description**: Defines password pattern.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: "^.3,$" (default)
- **Since**: 6.10.1.0

policy.rotation.ratio

- **Module**: cmas-core-security
- **Description**: Defines how often password may repeat. E.g., setting the value to X means that the new password cannot be present among the user's X previous passwords.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 1 (default)
- **Since**: 6.10.1.0

policy.track.username.case.sensitive

- **Module**: cmas-core-security
- **Description**: Defines whether customer (user) names in CM/Track are treated case-sensitive on login.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true (default)
- **Since**: 6.11.0.0

policy.username.case.sensitive

- **Module**: cmas-core-security
- **Description**: Defines whether user names are case-sensitive.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true (default)
- **Since**: 6.10.1.0

postActivityExecutionScriptName

- **Module**: cmweb-server-adapter
- **Description**: Defines the name for the script which should be executed after every workflow activity, see section PostActivityExecutionScript. If no script should be executed, leave the value empty.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: postActivityExecutionHandler
- **Since**: 6.2.0

queue.polling.threads.number

- **Module**: cmas-nimh
- **Description**: Number of threads started for emails' queue polling. Default: 1
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 1
- **Since**: 6.4.0

queue.polling.threads.shutdown.timeout.seconds

- **Module**: cmas-nimh
- **Description**: Waiting time after the shutdown signal. When the timeout is reached, the thread will be terminated. Default: 60
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.4.0

queue.polling.threads.watchdog.interval.seconds

- **Module**: cmas-nimh
- **Description**: Watchdog thread interval. Default: 30

- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 30
- **Since**: 6.4.0

queue.task.error.pause.seconds

- **Module**: cmas-nimh
- **Description**: Maximum number of seconds, the queue poller waits after infrastructure (e.g. database) error. Default 180 seconds
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 180
- **Since**: 6.4.0

queue.task.interval.seconds

- **Module**: cmas-nimh
- **Description**: Main emails' queue polling thread interval. Default: 15
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 15
- **Since**: 6.4.0

queue.task.max.retries

- **Module**: cmas-nimh
- **Description**: Maximum number of email processing retries after an exception. When reached, the email is moved to the email archive. This email can be rescheduled again using NIMH API (or the Admin Tool).
- **Type**: integer
- **Restart required**: no
- **System**: no

- **Optional**: yes
- **Example value**: 10
- **Since**: 6.4.0

queue.task.timeout.seconds

- **Module**: cmas-nimh
- **Description**: After this time (of inactivity) the service thread is considered as damaged and automatically restarted. Default: 600 seconds
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 600
- **Since**: 6.4.0

queue.task.transaction.timeout.seconds

- **Module**: cmas-nimh
- **Description**: Transaction timeout for email processing in the pipe. Default: 60
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.4.0

queuesExcludedFromGS

- **Module**: cmweb-server-adapter
- **Description**: Comma-separated list of queue names which are excluded from Quick Search.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.0

recent.items.cleanup.cluster.node.id

- **Module**: cmas-core-server
- **Description**: Value of a `-Dcmas.clusternode.id` designating the node which will clean up recent items.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 1 (assuming the cluster node started with `-Dcmas.clusternode.id=1` parameter)
- **Since**: 6.11.0.1

recent.items.cleanup.interval.minutes

- **Module**: cmas-core-server
- **Description**: Controls the time interval (in minutes) in which recent items should be checked for removal.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 60
- **Since**: 6.11.0.1

recent.items.max.per.engineer

- **Module**: cmas-core-server
- **Description**: Maximum number of preserved recent items per engineer while cleaning up (older recent items will be deleted).
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 500
- **Since**: 6.11.0.1

recent.items.persistence.enabled

- **Module**: cmas-core-server
- **Description**: Enables persistence of recent items, if false - prevents storing new recent items.

- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true
- **Since**: 6.11.1.0

recoverable.exceptions

- **Module**: cmas-dwh-server
- **Description**: Comma-separated list of exception definitions: CLASS[+][:REGEX]. The exceptions included in the list do not stop CM from sending to the CMRF process, but force it to try again. If optional '+' after CLASS is present, classes which extend CLASS are matched.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: java.sql.SQLRecoverableException,java.lang.RuntimeException+:.*T.1\,2T.*
- **Since**: 6.8.4.6

refreshTimeInCaseOfConcurrentRememberMeRequests

- **Module**: cmas-workflow-jbpm
- **Description**: It sets the refresh time (in seconds) after which page will be reloaded in case of concurrent remember me requests. This feature prevents one user from occupying many licenses. Please increase that time if sessions are still occupying.
- **Type**: integer
- **Restart required**: yes
- **System**: yes
- **Optional**: yes
- **Example value**: 5
- **Since**: 6.8.2

rememberMeLifetimeInMinutes

- **Module**: cmweb-server-adapter
- **Description**: Lifetime for *remember me* in minutes.
- **Type**: integer
- **Restart required**: yes
- **System**: yes

- **Optional**: no
- **Example value**: 1440
- **Since**: 6.0

request.scope.transaction

- **Module**: cmweb-server-adapter
- **Description**: It allows to disable request scope transaction. By default one transaction is used per request. Setting this property to “false” there will cause one transaction per service method invocation.
- **Type**: boolean
- **Restart required**: yes
- **System**: yes
- **Optional**: yes
- **Example value**: true
- **Since**: 6.8.1

resetCode.expiriationPeriod

- **Module**: cmas-core-security
- **Description**: Defines the expiration period for the link when resetting the password in CM/Track.
- **Type**: Integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 86400000 (default, 24 hours)
- **Since**: 6.10.1

resource.replace.batchSize

- **Module**: cmas-core-server
- **Description**: Defines the number of objects to be processed in a resource replace action.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 5
- **Since**: 6.10.0.0

resource.replace.timeout

- **Module**: cmas-core-server
- **Description**: Transaction timeout (in seconds) of a resource replacement action step.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 120
- **Since**: 6.10.0.0

scene

- **Module**: cmas-setup-scene
- **Description**: Scene file which was imported during set-up (can be empty).
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: vfszip:/P:/dist/target/jboss/server/cmas/deploy/cm-dist-6.5.1-SNAPSHOT.ear/APP-INF/lib/dist-scene-6.5.1-SNAPSHOT.jar/META-INF/cmas/scenes/helpdesk-sales_scene.jar/
- **Since**: 6.0

script.evict.unused.after.hours

- **Module**: cmas-core-server
- **Description**: Determines the number of hours for which unused scripts remain in the cache. After this time, the compiled class of the script is removed. The ConSol CM server checks for scripts to evict every hour.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 24 (default)
- **Since**: 6.11.1.14

script.logging.threshold.seconds

- **Module**: cmas-core-server
- **Description**: When this time, in seconds, is exceeded during script execution, a warning is emitted in the logs.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 10 (default)
- **Since**: 6.10.1.0

script.validation.interval.seconds

- **Module**: cmas-app-admin-tool
- **Description**: Interval in seconds between two code checks in the Admin Tool or the Process Designer code editor
- **Type**: Integer
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: 1 (default)
- **Since**: 6.11.0.1

searchPageSize

- **Module**: cmweb-server-adapter
- **Description**: Default page size for search results.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 20
- **Since**: 6.0

searchPageSizeOptions

- **Module**: cmweb-server-adapter
- **Description**: Options for page size for search results.
- **Type**: string

- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 10|20|30|40|50|75|100
- **Since**: 6.0

security.fields.customer.exposure.check.enabled

- **Module**: cmas-restapi-core
- **Description**: Enables customer exposure annotation checks for ticket fields.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true (default)
- **Since**: 6.10.5.4

security.restrict.unit.access.to.own.data

- **Module**: cmas-restapi-core
- **Description**: If set to “true”, an additional check is performed when a user logs in as a customer using the REST API, e.g. CM/Track. When requesting customer data, only the company of the user or other contacts of the user’s company are returned. If set to “false”, no additional security check is performed and the former security rules apply.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true (default)
- **Since**: 6.9.2.14

serial.mods.tracking.enabled

- **Module**: cmas-core-server
- **Description**: Low level technical flag deciding whether serial diff tracking for entities is enabled. If enabled, there will be no `StackOverflow Error` in case a dependency between two entities (for example engineer and ticket) causes an infinite loop first and then as a result, the StackOverflow. The property must be added to the configuration manually. It will not be added to a system configuration during setup or update.

> ⚠ Please enable the restricted ticket change behavior described in this section only when advised by a ConSol representative! It is a low level technical flag with intricate consequences for system behavior and thus should not be used without thorough scrutiny.

- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: false (default)
- **Since**: 6.10.7.0, 6.11.0.5

server.session.archive.reaper.interval

- **Module**: cmas-core-server
- **Description**: Server archived sessions reaper interval (in seconds).
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.7.1

server.session.archive.timeout

- **Module**: cmas-core-server
- **Description**: Server sessions archive validity timeout (in days). After this time session info is removed from the DB.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 31
- **Since**: 6.7.1

server.session.reaper.interval

- **Module**: cmas-core-server
- **Description**: Server inactive (ended) sessions reaper interval (in seconds).

- **Type**: integer
- **Restart required**: only Session Service
- **System**: yes
- **Optional**: no
- **Example value**: 60
- **Since**: 6.6.1, 6.7.1

server.session.timeout

- **Module**: cmas-core-server
- **Description**: Server session timeout (in seconds) for connected clients. Each client can overwrite this timeout with custom value using its ID (ADMIN_TOOL, WEB_CLIENT, WORKFLOW_EDITOR, TRACK (before 6.8, please use PORTER), ETL, REST) appended to property name, e.g., `server.session.timeout.ADMIN_TOOL`.
  Please see also the Page Customization attributes *updateTimeServerSessionActivityEnabled* and *updateTimeServerSessionActivity*, both of type *cmApplicationCustomization*.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 1800
- **Since**: 6.6.1, 6.7.1

Detailed explanation for the Admin Tool:

- server.session.timeout.ADMIN_TOOL
  Defines the time interval how long the server considers a session valid while there is no activity from the Admin Tool holding the session. The Admin Tool is not aware of this value, it only suffers having an invalid session, if the last activity has been longer in the past.
- admin.tool.session.check.interval
  Defines the time between two checks done by the Admin Tool, if the server still considers its session valid.

For example, if admin.tool.session.check.interval = 60 the Admin Tool queries the server every minute if its session is still active/valid. In case server.session.timeout.ADMIN_TOOL = 600 the Admin Tool will get the response that the session is now invalid after ten minutes of inactivity.

serverPoolingInterval

- **Module**: cmweb-server-adapter
- **Description**: Defines the time in seconds for pooling server to invalidate caches on the web layer.
- **Type**: integer
- **Restart required**: no

- **System**: yes
- **Optional**: no
- **Example value**: 5
- **Since**: 6.1.0

skip-ticket

- **Module**: cmas-dwh-server
- **Description**: Tickets are not transferred during transfer/update.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.6.19
- **Removed in**: 6.8.1

skip-ticket-history

- **Module**: cmas-dwh-server
- **Description**: History of ticket is not transferred during transfer/update.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.6.19
- **Removed in**: 6.8.1

skip-unit

- **Module**: cmas-dwh-server
- **Description**: Units are not transferred during transfer/update.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false

- **Since**: 6.6.19
- **Removed in**: 6.8.1

skip-unit-history

- **Module**: cmas-dwh-server
- **Description**: History of unit is not transferred during transfer/update.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.6.19
- **Removed in**: 6.8.1

skip.wfl.transfer.cleanup

- **Module**: cmas-core-server
- **Description**: If set to “true”, skips workflow cleanup after transfer.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: false (default)
- **Since**: 6.9.4.1

skip.wfl.transfer.translations.cleanup

- **Module**: cmas-core-server
- **Description**: Enables skipping the cleanup of localized properties of removed workflow elements.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: false (default)
- **Since**: 6.10.5.5

split.history

- **Module**: cmas-dwh-server
- **Description**: Changes the SQL that fetches the history for the tickets during DWH transfer not to all tickets at once but only for one ticket per SQL.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: false
- **Since**: 6.8.0

start.groovy.task.enabled

- **Module**: cmas-app-admin-tool
- **Description**: For being able to run Admin Tool scripts of type *Task* in the Admin Tool (navigation group *Services*, navigation item *Task Execution*). It is required to enable the *Start task* button, which is hidden by default. This is done by setting this system property to “true”.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true
- **Since**: 6.9.4.0

statistics.calendar

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.client.group

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.

- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.contact.role

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.content.entry

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.content.entry.class

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.content.entry.history

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.customer.definition

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.engineer

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.enum.group

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no

- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.field.definition

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.group.definition

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.locale

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.localized.property

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.mla

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.project

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.queue

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no

- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.resource

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.resource.group

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.resource.history

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.resource.relation.definition

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.resource.type

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.ticket

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.ticket.function

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no

- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.ticket.history

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.time.booking

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.timestamp

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.unit

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.unit.history

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.unit.relation.definition

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.workflow

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no

- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

strict.utf.bmp.enabled

- **Module**: cmas-core-server
- **Description**: In ConSol CM versions lower than 6.10.6, incoming emails with a subject line containing four-byte UTF8 characters could not be handled by some installations using the MySQL database engine. The reason is the encoding/collation configuration of the database using a two-byte BMP (Basic Multilingual Plane) 0 plane which cannot be changed in some installations for technical reasons. Other database engines were unaffected. Emails with this encoding could not be imported into the system at all in CM versions lower than 6.10.6. In order to accommodate this issue this system property for configuration is available.
  Setting it to “true” will filter out all four-byte UTF8 characters before any database interaction, so the problems mentioned above will not occur.
  The property value is “true” by default for MySQL databases, and “false” for any other database where it should not be necessary at all. Change it for a MySQL database only, if the settings positively will support four-byte characters.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 100
- **Since**: 6.10.6.0

supportEmail

- **Module**: cmweb-server-adapter
- **Description**:
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.0
- **Removed in**: 6.11.0.1

synchronize.master.address

- **Module**: cmas-core-index-common
- **Description**: Value of `-Dcmas.http.host.port` specifying how to connect to the indexing master server. Default null. Since 6.6.17 this value is configurable in set-up to designate the initial indexing master server. Please note that changing this value is only allowed when all cluster nodes' index change receivers are stopped.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 127.0.0.1:80
- **Since**: 6.6.0

synchronize.master.security.token

- **Module**: cmas-core-index-common
- **Description**: The password for accessing the index snapshot via URL, e.g., for index synchronization or for backups.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: token
- **Since**: 6.6.0

synchronize.master.security.user

- **Module**: cmas-core-index-common
- **Description**: The user name for accessing the index snapshot via URL, e.g., for index synchronization or for backups.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: user
- **Since**: 6.6.0

synchronize.master.timeout.minutes

- **Module**: cmas-core-index-common
- **Description**: How long the master server may continually fail until a new master gets elected. Default 5. Since 6.6.17 this value is configurable in set-up, where zero means that master server will never change (failover is disabled).
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 5
- **Since**: 6.6.0

synchronize.megabits.per.second

- **Module**: cmas-core-index-common
- **Description**: How much bandwidth the master server may consume when transferring index changes to all slave servers. Default 85. Please do not use all available bandwidth to transfer index changes between hosts, as doing so will most probably partition the cluster due to some subsystems being unable to communicate.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 85
- **Since**: 6.6.0

synchronize.sleep.millis

- **Module**: cmas-core-index-common
- **Description**: How often each slave server polls the master server for index changes. Default 1000.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 1000
- **Since**: 6.6.0

task.execution.interval.seconds

- **Module**: cmas-core-server
- **Description**: Time in seconds between the end of an accomplished task in the TEF (Task Execution Framework) and the start of the next task.
- **Type**: Integer
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: 5
- **Since**: 6

task.execution.node.id

- **Module**: cmas-core-server
- **Description**: Only relevant in clustered environments. The ID of the node where scripts of the TEF (Task Execution Framework) will be executed. This applies to both scripts called from the workflow and scripts called manually using the Admin Tool. The Admin Tool can be started from any node.
- **Type**: Integer
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: 2
- **Since**: 6.11.0.1

task.panel.refresh.interval.seconds

- **Module**: cmas-app-admin-tool
- **Description**: Time in seconds after which the task list (in the Admin Tool) of the Task Execution Framework is refreshed.
- **Type**: Integer
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: 10
- **Since**: 6.10.5.3 (not added automatically during update from versions prior to 6.10.5.3!)

themeOverlay

- **Module**: cmweb-server-adapter
- **Description**: Name of used theme overlay
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: consolINT
- **Since**: 6.0

ticket.delete.timeout

- **Module**: cmas-core-server
- **Description**: Transaction timeout (in seconds) for deleting tickets.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 60
- **Since**: 6.1.3

ticket.from.incoming.message.accepted.links

- **Module**: cmas-core-server
- **Description**: List of domains to which links in incoming emails and links in comments added via REST API are clickable in the ticket history. Regular expressions can be used to specify the allowed URLs. It is possible to add several URLs by using a whitespace as delimiter. The URL must start with one of the allowed protocols (http, https, ftp, ftps, file, mailto). All other links are removed, i.e., the link is displayed in the ticket history as text but it cannot be clicked. If the property is left empty, all links are removed. The regular expression `.+` can be used to allow all domains.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: https://.*\.consol\.de (allows links to “https://<any>.consol.de”)
- **Since**: 6.11.1.7

⚠ Please note that whitelisting domains might make ConSol CM vulnerable to cross-site scripting and other attacks. Choose the domains you whitelist carefully!

ticketListRefreshIntervalInSeconds

- **Module**: cmweb-server-adapter
- **Description**: Refresh interval for ticket list (in seconds).
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 180
- **Since**: 6.0

ticketListSizeLimit

- **Module**: cmweb-server-adapter
- **Description**: Maximum number of tickets in ticket list.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 100
- **Since**: 6.0

tickets.delete.size

- **Module**: cmas-core-server
- **Description**: Defines a number of tickets deleted per transaction. By default it is set to 10.
- **Type**: integer
- **Restart required**: only Session Service
- **System**: yes
- **Optional**: no
- **Example value**: 10
- **Since**: 6.8.1

time.buffer

- **Module**: cmas-dwh-server
- **Description**: Number of minutes to extend date of start live mode.

- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 5
- **Since**: 6.8.1.11

transaction.timeout.minutes

- **Module**: cmas-core-server
- **Description**: Sets the transaction timeout for the task execution service, i.e., one run of a task must finish before this timeout is reached. The changes are visible only for new tasks, the execution of which started after the configuration change.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 10*3600 (10 hours - default)
- **Since**: 6.10

tx.read.only.mode.enabled

- **Module**: cmweb-server-adapter
- **Description**: Enables read-only transactions for faster page loading. This transactional behavior was introduced in 6.11.0, and this property acts as a safety guard to restore the old behaviors. Do not change this value unless facing tx problems and advised by ConSol.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true (default)
- **Since**: 6.11

unit.description.mode

- **Module**: cmas-core-server
- **Description**: Defines whether unit (contact) descriptions in the ticket history are taken from the database or dynamically rendered using templates. The value, “DYNAMIC”, is a bit more costly from the performance perspective, while “PROTOCOL” is faster but returns historical names

which might be outdated. Use “PROTOCOL” if you have lots of history entries from many different units. This is also the default value in CM versions 6.11.1.1 and up. In CM versions up to 6.11.1.0, “DYNAMIC” is the default.

- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: PROTOCOL
- **Since**: 6.11.0

unit.replace.batchSize

- **Module**: cmas-core-server
- **Description**: Defines the number of objects to be processed in a unit replace action.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 5
- **Since**: 6.8.2

unit.replace.timeout

- **Module**: cmas-core-server
- **Description**: Transaction timeout (seconds) of a unit replacement action step.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 120
- **Since**: 6.8.2

unit.transfer.order

- **Module**: cmas-dwh-server
- **Description**: Define in which order customer field groups should be transferred to the DWH.
- **Type**: string
- **Restart required**: no
- **System**: yes

- **Optional**: yes
- **Example value**: company;customer
- **Since**: 6.6.19
- **Removed in**: 6.8.1

unitIndexSearchResultSizeLimit

- **Module**: cmweb-server-adapter
- **Description**: Maximum number of units in unit search result (e.g. when searching for contact).
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 5
- **Since**: 6.0

unused.content.remover.cluster.node.id

- **Module**: cmas-core-server
- **Description**: Value of a `cmas.clusternode.id` designating which node will remove unused ticket attachments and unit content entries.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 1 (assuming cluster node started with the parameter `-Dcmas.clusternode.id=1`)
- **Since**: 6.9.0.0

unused.content.remover.enabled

- **Module**: cmas-core-server
- **Description**: Specifies whether removal of unused ticket attachments and unit content entries should take place.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no

- **Example value**: true
- **Since**: 6.9.0.0

unused.content.remover.polling.minutes

- **Module**: cmas-core-server
- **Description**: How often unused ticket attachments and unit content entries should be checked for removal.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 15
- **Since**: 6.9.0.0

unused.content.remover.ttl.minutes

- **Module**: cmas-core-server
- **Description**: Minimum interval, in minutes, after which unused ticket attachments and unit content entries can be removed.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 1440
- **Since**: 6.9.0.0

update.6.11.0.0.sleep

- **Module**: cmas-setup-hibernate
- **Description**: Helper property for the update preparation scripts introduced in context of CM database refactoring in version 6.11. This is an optional setting allowing a delay (in milliseconds) after each loop iteration of the preparation scripts. Setting the delay should lower the database load, for example during working hours. This property may be removed after the update preparation tasks finish.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes

- **Example value**: 0
- **Since**: 6.11.0.0, for use in 6.10.5.x

update.6.11.0.0.timezone

- **Module**: cmas-setup-hibernate
- **Description**: Helper property for the ticket history migration (the new way of counting history groups). Since 6.11.0.0 the groups are constant (2h time span), but before 6.11.0.0 groups were not constant and depended on the customer's time zone. Migration scripts use an old algorithm to calculate groups and therefore need information about the time zone. The property should be set to the timezone which is most commonly used by the customers. If the property is not set, the default server time zone is used (`TimeZone.getDefault()`). The property should be set before updating to 6.11.0.0 and will be removed automatically after migration. The list of accepted timezones can be found for example here: http://joda-time.sourceforge.net/timezones.html.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: Europe/Berlin
- **Since**: 6.11.0.0, for use before updating to this version

urlLogoutPath

- **Module**: cmweb-server-adapter
- **Description**: URL which is used when user logs out. (If no value is set, logout leads to login-mask.)
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: http://intranet.consol.de
- **Since**: 6.3.1

voCacheEnabled

- **Module**: cmweb-server-adapter
- **Description**: This property enables additional caching for the Web Client, voCaching, of complete objects, thus improving performance.
- **Type**: boolean
- **Restart required**: no

- **System**: yes
- **Optional**: yes
- **Example value**: true
- **Since**: 6.11.1.0

Notes:

- Since ConSol CM version 6.11.1.1, the default value is “true” for non-clustered environments. The value is set to “true” automatically during the setup or update of ConSol CM 6.11.1.1.
- When voCaching is enabled and lazy loading is used for folding ticket history entries, once the engineer unfolded an entry, he cannot fold it again by reloading the page or opening the ticket from the workspace.
- When using the dynamic mode for displaying engineer and customer names in the ticket history (as configured in the system properties cmas-core-server, engineer.description.mode and cmas-core-server, unit.description.mode), the new version of the engineer and/or customer name is only displayed after the ticket has been changed.
- This system property is ignored for clustered environments (environments with cmas-core-shared, cluster.mode set to “true”). In clustered environments, voCaching is always disabled to avoid problems that changes made to objects on one node are not visible on the other nodes.

warmup.executor.enabled

- **Module**: cmas-core-server
- **Description**: Specifies whether the server should asynchronously warm up during startup (e.g., fill some of the internal caches).
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true
- **Since**: 6.9.4.2

webSessionTimeoutInMinutes

- **Module**: cmweb-server-adapter
- **Description**: Session timeout in minutes.
- **Type**: integer
- **Restart required**: yes

- **System**: yes
- **Optional**: no
- **Example value**: 180
- **Removed in**: 6.7.1
- **Replaced by**: cmas-core-server, server.session.timeout

wfl.sticky.transfer.disabled

- **Module**: cmas-core-server
- **Description**: Enables using preserved original names of workflow elements.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: false
- **Since**: 6.10.1.0

wicketAjaxRequestHeaderFilterEnabled

- **Module**: cmweb-server-adapter
- **Description**: This enables filter for Wicket AJAX requests, coming from stale pages with Wicket 1.4 scripting (CM pre-6.8.0), after update to CM6 post-6.8.0.
- **Type**: boolean
- **Restart required**: yes
- **System**: yes
- **Optional**: yes
- **Example value**: false
- **Since**: 6.8.1

X-Frame-Options

- **Module**: cmweb-server-http-headers
- **Description**: Example property to illustrate the configuration of HTTP headers. In this case the delivered HTTP header contains the field *X-Frame-Options* with the value “SAMEORIGIN”.

  Each property in the module `cmweb-server-http-headers` represents one header field. The property name/key identifies the response header field and the value of the property is the field value sent in this header.

> ⚠ Please be aware that additional HTTP response headers must be correctly defined with the exact spelling as officially specified! Please note also that the correct interpretation and application of these headers is fully in the realm and responsibility of the client browser which requested the page!

- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: SAMEORIGIN
- **Since**: 6.10.8

H.2.2 List of System Properties by Area

This chapter lists the system properties which are relevant for the following areas.

- CMRF & DWH Configuration
- Indexer and Search Configuration
- LDAP Configuration
- Email Configuration
- Activity Interval Configuration
- Administrator Email Addresses
- HTTP Header Configuration

H.2.2.1 CMRF & DWH Configuration

autocommit.cf.changes

- **Module**: cmas-dwh-server
- **Description**: Defines whether DWH tasks which result from configurational changes on ticket fields are executed automatically without manual interaction in the Admin Tool. Can be also set in the Admin Tool in the navigation item *DWH*. The default and recommended value is “false”.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.7.0

batch-commit-interval

- **Module**: cmas-dwh-server
- **Description**: Number of objects in a JMS message. Larger values mean better transfer performance at the cost of higher memory usage.
  Starting with ***ConSol CM*** version 6.11, this property is only used if the package size of a DWH operation is not set. This can only happen when the command is directly addressed to the Java MBean consol.cmas.global.dwh.synchronizationService, e.g. using the update() method. When a DWH operation is started using the Admin Tool, there is always a value for the package size. If not explicitly set, the default value of 1000 is used as value for the batch.-commit.interval.
- **Default value**: 1000
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 100
- **Since**: 6.0.0

communication.channel

- **Module**: cmas-dwh-server
- **Description**: Communication channel. Only possible value since CM version 6.11.0.0: DIRECT
- **Type**: string
- **Restart required**: no
- **System**: yes

- **Optional**: no
- **Example value**: DIRECT
- **Since**: 6.8.5.0
- **Removed in**: 6.11.0.0 (DIRECT mode is the only available mode and is set automatically)

dwh.mode

- **Module**: cmas-dwh-server
- **Description**: Current mode for DWH data transfer. Possible values are OFF, ADMIN, LIVE
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: OFF
- **Since**: 6.0.1

ignore-queues

- **Module**: cmas-dwh-server
- **Description**: A comma-separated list of queue names which are not not transferred to the DWH.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: QueueName1,QueueName2,QueueName3
- **Since**: 6.6.19
- **Removed in**: 6.8.1

is.cmrf.alive

- **Module**: cmas-dwh-server
- **Description**: As a starting point, the time the last message was sent to CMRF should be used. If a response from CMRF is not received after value (in seconds), it should create a DWH operation status with an error message indicating that CMRF is down.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no

- **Example value**: 1200
- **Since**: 6.7.0

java.naming.factory.initial

- **Module**: cmas-dwh-server
- **Description**: Factory class for the DWH context factory.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: org.jnp.interfaces.NamingContextFactory
- **Since**: 6.0.1
- **Removed in**: 6.11.0.0

java.naming.factory.url.pkgs

- **Module**: cmas-dwh-server
- **Description**:
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: org.jboss.naming:org.jnp.interfaces
- **Since**: 6.0.1
- **Removed in**: 6.11.0.0

java.naming.provider.url

- **Module**: cmas-dwh-server
- **Description**: URL of naming provider.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: localhost
- **Since**: 6.0.1
- **Removed in**: 6.11.0.0

notification.error.description

- **Module**: cmas-dwh-server
- **Description**: Text for error emails from the DWH.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: Error occurred
- **Since**: 6.0.1

notification.error.from

- **Module**: cmas-dwh-server
- **Description**: From address for error emails from the DWH
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.0.1

notification.error.subject

- **Module**: cmas-dwh-server
- **Description**: Subject for error emails from the DWH
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: Error occurred
- **Since**: 6.0.1

notification.error.to

- **Module**: cmas-dwh-server
- **Description**: To address for error emails from the DWH
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no

- **Example value**: myuser@consol.de
- **Since**: 6.0.1

notification.finished_successfully.description

- **Module**: cmas-dwh-server
- **Description**: Text for emails from the DWH when a transfer finishes successfully.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: Transfer finished successfully
- **Since**: 6.0.1

notification.finished_successfully.from

- **Module**: cmas-dwh-server
- **Description**: From address for emails from the DWH when a transfer finishes successfully.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.0.1

notification.finished_successfully.subject

- **Module**: cmas-dwh-server
- **Description**: Subject for emails from the DWH when a transfer finishes successfully.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: Transfer finished successfully
- **Since**: 6.0.1

notification.finished_successfully.to

- **Module**: cmas-dwh-server
- **Description**: To address for emails from the DWH when a transfer finishes successfully.
- **Type**: string
- **Restart required**: yes

- **System**: yes
- **Optional**: no
- **Example value**: myuser@consol.de
- **Since**: 6.0.1

notification.finished_unsuccessfully.description

- **Module**: cmas-dwh-server
- **Description**: Text for emails from the DWH when a transfer finishes unsuccessfully.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: Transfer finished unsuccessfully
- **Since**: 6.0.1

notification.finished_unsuccessfully.from

- **Module**: cmas-dwh-server
- **Description**: From address for emails from the DWH when a transfer finishes unsuccessfully.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.0.1

notification.finished_unsuccessfully.subject

- **Module**: cmas-dwh-server
- **Description**: Subject for emails from the DWH when a transfer finishes unsuccessfully.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: Transfer finished unsuccessfully
- **Since**: 6.0.1

notification.finished_unsuccessfully.to

- **Module**: cmas-dwh-server
- **Description**: To address for emails from the DWH when a transfer finishes unsuccessfully.

- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: myuser@consol.de
- **Since**: 6.0.1

notification.host

- **Module**: cmas-dwh-server
- **Description**: Email (SMTP) server hostname for sending DWH emails.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: myserver.consol.de
- **Since**: 6.0.1

notification.password

- **Module**: cmas-dwh-server
- **Description**: Password for sending DWH emails (optional).
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.0.1

notification.port

- **Module**: cmas-dwh-server
- **Description**: SMTP port for sending DWH emails.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 25
- **Since**: 6.0.1

notification.protocol

- **Module**: cmas-dwh-server
- **Description**: The protocol used for sending emails from the DWH.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: pop3\

notification.tls.enabled

- **Module**: cmas-dwh-server
- **Description**: Activates SMTP via SSL/TLS (SMTPS) for sending notification emails from the DWH. The default value is “false”. If it is set to “true”, SMTPS is activated for sending notifications from the DWH.
- **Type**: string
- **Restart required**: yes
- **System**: no
- **Optional**: yes
- **Example value**: false
- **Since**: 6.11.1.6

notification.username

- **Module**: cmas-dwh-server
- **Description**: (SMTP) User name for sending DWH emails.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: myuser
- **Since**: 6.0.1

skip-ticket

- **Module**: cmas-dwh-server
- **Description**: Tickets are not transferred during transfer/update.
- **Type**: boolean
- **Restart required**: no
- **System**: yes

- **Optional**: no
- **Example value**: false
- **Since**: 6.6.19
- **Removed in**: 6.8.1

skip-ticket-history

- **Module**: cmas-dwh-server
- **Description**: History of ticket is not transferred during transfer/update.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.6.19
- **Removed in**: 6.8.1

skip-unit

- **Module**: cmas-dwh-server
- **Description**: Units are not transferred during transfer/update.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.6.19
- **Removed in**: 6.8.1

skip-unit-history

- **Module**: cmas-dwh-server
- **Description**: History of unit is not transferred during transfer/update.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false

- **Since**: 6.6.19
- **Removed in**: 6.8.1

split.history

- **Module**: cmas-dwh-server
- **Description**: Changes the SQL that fetches the history for the tickets during DWH transfer not to all tickets at once but only for one ticket per SQL.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: false
- **Since**: 6.8.0

unit.transfer.order

- **Module**: cmas-dwh-server
- **Description**: Define in which order customer field groups should be transferred to the DWH.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: company;customer
- **Since**: 6.6.19
- **Removed in**: 6.8.1

H.2.2.2 Indexer and Search Configuration

Indexer

big.task.minimum.size

- **Module**: cmas-core-index-common
- **Description**: Indicates the minimum size of index task (in parts, each part has 100 entities) to qualify this task as a big one. Big tasks have lower priority than normal tasks.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no

- **Example value**: 15 (default)
- **Since**: 6.8.3

database.notification.enabled

- **Module**: cmas-core-index-common
- **Description**: Indicates whether index update database notification channel should be used instead of JMS.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.8.4.7

database.notification.redelivery.delay.seconds

- **Module**: cmas-core-index-common
- **Description**: In case of index update database notification channel, indicates notification redelivery delay when an exception occurs.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 60
- **Since**: 6.8.4.7

database.notification.redelivery.max.attempts

- **Module**: cmas-core-index-common
- **Description**: In case of index update database notification channel, indicates maximum redelivery attempts when an exception occurs.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 60
- **Since**: 6.8.4.7

disable.admin.task.auto.commit

- **Module**: cmas-core-index-common
- **Description**: All tasks created for index update will be automatically executed right after creation.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.6.1

index.attachment

- **Module**: cmas-core-index-common
- **Description**: Specifies whether content of attachments is indexed.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true
- **Since**: 6.4.3

index.history

- **Module**: cmas-core-index-common
- **Description**: Specifies whether unit and ticket history are indexed.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.1.0
- **Removed in**: 6.11.0

index.status

- **Module**: cmas-core-index-common
- **Description**: Status of the Indexer, possible values RED, YELLOW, GREEN, will be displayed in the Admin Tool.
- **Type**: string

- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: GREEN
- **Since**: 6.6.1

index.task.worker.threads

- **Module**: cmas-core-index-common
- **Description**: How many threads will be used to execute index tasks (synchronization, administrative, and repair tasks).
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 1 (default) (we recommend to use a value not larger than 2)
- **Since**: 6.6.14, 6.7.3. Since 6.8.0 and exclusively in 6.6.21 also normal (live) index updates are affected by this property.

index.version.current

- **Module**: cmas-core-index-common
- **Description**: Holds information about current (possibly old) index version.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 1 (default)
- **Since**: 6.7.0

index.version.newest

- **Module**: cmas-core-index-common
- **Description**: Holds information about which index version is considered newest.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 1 (default)
- **Since**: 6.7.0

indexed.assets.per.thread.in.memory

- **Module**: cmas-core-index-common
- **Description**: How many assets should be loaded into memory at once, per thread, during indexing.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 200 (default)
- **Since**: 6.8.0

indexed.engineers.per.thread.in.memory

- **Module**: cmas-core-index-common
- **Description**: How many engineers should be loaded into memory at once, per thread, during indexing.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 300 (default)
- **Since**: 6.6.14, 6.7.3

indexed.resources.per.thread.in.memory

- **Module**: cmas-core-index-common
- **Description**: How many resources should be loaded into memory at once, per thread, during indexing.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 200 (default)
- **Since**: 6.10.0.0

indexed.tickets.per.thread.in.memory

- **Module**: cmas-core-index-common
- **Description**: How many tickets should be loaded into memory at once, per thread, during indexing.

- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 100 (default)
- **Since**: 6.6.14, 6.7.3

indexed.units.per.thread.in.memory

- **Module**: cmas-core-index-common
- **Description**: How many units should be loaded into memory at once, per thread, during indexing.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 200 (default)
- **Since**: 6.6.14, 6.7.3

synchronize.master.address

- **Module**: cmas-core-index-common
- **Description**: Value of `-Dcmas.http.host.port` specifying how to connect to the indexing master server. Default null. Since 6.6.17 this value is configurable in set-up to designate the initial indexing master server. Please note that changing this value is only allowed when all cluster nodes' index change receivers are stopped.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 127.0.0.1:80
- **Since**: 6.6.0

synchronize.master.security.token

- **Module**: cmas-core-index-common
- **Description**: The password for accessing the index snapshot via URL, e.g., for index synchronization or for backups.
- **Type**: string
- **Restart required**: no

- **System**: yes
- **Optional**: yes
- **Example value**: token
- **Since**: 6.6.0

synchronize.master.security.user

- **Module**: cmas-core-index-common
- **Description**: The user name for accessing the index snapshot via URL, e.g., for index synchronization or for backups.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: user
- **Since**: 6.6.0

synchronize.master.timeout.minutes

- **Module**: cmas-core-index-common
- **Description**: How long the master server may continually fail until a new master gets elected. Default 5. Since 6.6.17 this value is configurable in set-up, where zero means that master server will never change (failover is disabled).
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 5
- **Since**: 6.6.0

synchronize.megabits.per.second

- **Module**: cmas-core-index-common
- **Description**: How much bandwidth the master server may consume when transferring index changes to all slave servers. Default 85. Please do not use all available bandwidth to transfer index changes between hosts, as doing so will most probably partition the cluster due to some subsystems being unable to communicate.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no

- **Example value**: 85
- **Since**: 6.6.0

synchronize.sleep.millis

- **Module**: cmas-core-index-common
- **Description**: How often each slave server polls the master server for index changes. Default 1000.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 1000
- **Since**: 6.6.0

Search Results

globalSearchResultSizeLimit

- **Module**: cmweb-server-adapter
- **Description**: Maximum number of items in Quick Search result.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 10
- **Since**: 6.0

searchPageSize

- **Module**: cmweb-server-adapter
- **Description**: Default page size for search results.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 20
- **Since**: 6.0

searchPageSizeOptions

- **Module**: cmweb-server-adapter
- **Description**: Options for page size for search results.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 10|20|30|40|50|75|100
- **Since**: 6.0

unitIndexSearchResultSizeLimit

- **Module**: cmweb-server-adapter
- **Description**: Maximum number of units in unit search result (e.g. when searching for contact).
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 5
- **Since**: 6.0

H.2.2.3 LDAP Configuration

LDAP Configuration (if LDAP is Used as Authentication Mode in the CM Web Client)

LDAP parameters apply only if the authentication mode for the CM Web Client has been set to "LDAP":

authentication.method

- **Module**: cmas-core-security
- **Description**: User authentication method (internal CM database or LDAP authentication). Allowed values are LDAP or DATABASE.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: DATABASE
- **Since**: 6.0

ldap.authentication

- **Module**: cmas-core-security
- **Description**: Authentication method used when using LDAP authentication. Possible values are 'anonymous' and 'simple' (default).
- **Type**: string
- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: simple
- **Since**: 6.0

ldap.basedn

- **Module**: cmas-core-security
- **Description**: Base DN used for looking up LDAP user accounts when using LDAP authentication.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: ou=accounts,dc=consol,dc=de
- **Since**: 6.0

ldap.initialcontextfactory

- **Module**: cmas-core-security
- **Description**: Class name for the initial context factory of the LDAP implementation when using LDAP authentication. If it is not set, `com.sun.jndi.ldap.LdapCtxFactory` is used.
- **Type**: string
- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: com.sun.jndi.ldap.LdapCtxFactory
- **Since**: 6.0

ldap.password

- **Module**: cmas-core-security
- **Description**: Password for connecting to LDAP to look up users when using LDAP authentication. Only needed if look-up cannot be performed anonymously.
- **Type**: password

- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.1.2

ldap.providerurl

- **Module**: cmas-core-security
- **Description**: LDAP provider when using LDAP authentication.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: ldap://myserver.consol.de:389
- **Since**: 6.0

ldap.searchattr

- **Module**: cmas-core-security
- **Description**: Search attribute for looking up LDAP entry associated with a CM login.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: uid
- **Since**: 6.0

ldap.userdn

- **Module**: cmas-core-security
- **Description**: LDAP user for connecting to LDAP to look up users when using LDAP authentication. Only needed if look-up cannot be performed anonymously.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.1.2

LDAP Configuration (if LDAP is Used as Authentication Mode in CM/Track)

LDAP parameters apply only if the authentication mode for CM/Track has been set to “LDAP”:

contact.authentication.method

- **Module**: cmas-core-security
- **Description**: Indicates contact authentication method, where possible values are DATABASE or LDAP or LDAP,DATABASE or DATABASE,LDAP.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Since**: 6.9.3.0

ldap.contact.name.basedn

- **Module**: cmas-core-security
- **Description**: Base path to search for contact DN by LDAP ID (e.g. ou=a-ccounts,dc=consol,dc=de).
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Since**: 6.9.3.0

ldap.contact.name.password

- **Module**: cmas-core-security
- **Description**: Password to look up contact DN by LDAP ID. If not set, the anonymous account is used.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Since**: 6.9.3.0

ldap.contact.name.providerurl

- **Module**: cmas-core-security
- **Description**: Address of the LDAP server (ldap[s]://host:port).
- **Type**: string
- **Restart required**: no
- **System**: no

- **Optional**: yes
- **Since**: 6.9.3.0

ldap.contact.name.searchattr

- **Module**: cmas-core-security
- **Description**: Attribute to search for contact DN by LDAP ID (e.g. uid).
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Since**: 6.9.3.0

ldap.contact.name.userdn

- **Module**: cmas-core-security
- **Description**: User DN to look up contact DN by LDAP ID. If not set, the anonymous account is used.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Since**: 6.9.3.0

ldap.initialcontextfactory

- **Module**: cmas-core-security
- **Description**: Class name for the initial context factory of the LDAP implementation when using LDAP authentication. If it is not set, `com.sun.jndi.ldap.LdapCtxFactory` is used.
- **Type**: string
- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: com.sun.jndi.ldap.LdapCtxFactory
- **Since**: 6.0

H.2.2.4 Email Configuration

Outgoing Email

mail.smtp.email

- **Module**: cmas-core-server
- **Description**: SMTP email URL for outgoing emails
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: smtp://mail.mydomain.com:25
- **Since**: 6.0

mail.smtp.envelopesender

- **Module**: cmas-core-server
- **Description**: Email address used as sender in SMTP envelope. If not set, the From address of the email is used.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: mysender@mydomain.com
- **Since**: 6.5.7

mail.from

- **Module**: cmweb-server-adapter
- **Description**: Use this address if set instead of engineer email address during email conversation.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.1.2

mail.reply.to

- **Module**: cmweb-server-adapter
- **Description**: When set, Web Client will display Reply-To field on email send, prefilled with this value.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.0.1

> ⚠ Please read the detailed information about ConSol CM Reply-To addresses in section Scripts of Type Email.

mailTemplateAboveQuotedText

- **Module**: cmweb-server-adapter
- **Description**: Indicates behavior of email template in the Ticket Email Editor when another email is quoted, i.e. forwarded or replied to. Often used to place the signature correctly.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.2.4

mail.sender.address

- **Module**: cmas-workflow-jbpm
- **Description**: From address for emails from the workflow engine.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: myuser@consol.de
- **Removed in**: 6.8.0
- **Replaced by**: jobExecutor.mailFrom

mail.smtp.tls.enabled

- **Module**: cmas-core-server
- **Description**: Activates SMTP via SSL/TLS (SMTPS) for sending emails from the Web Client and scripts. The default value is “false”. If it is set to “true”, SMTPS is activated for sending emails.
- **Type**: boolean
- **Restart required**: yes
- **System**: no
- **Optional**: yes
- **Example value**: true
- **Since**: 6.11.1.6

Incoming Email

nimh.enabled

- **Module**: cmas-core-server
- **Description**: Enables NIMH service. Must be suffixed with the cluster node ID, e.g., `nimh.enabled.NODEID` = “true”.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: false
- **Since**: 6.9.4.0

filesystem.polling.threads.number

- **Module**: cmas-nimh
- **Description**: Number of threads started for db emails' queue polling. Default: 1
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 10
- **Since**: 6.4.0

filesystem.polling.threads.shutdown.timeout.seconds

- **Module**: cmas-nimh
- **Description**: Waiting time after the shutdown signal. When the timeout reached, thread will be terminated. Default: 60
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.4.0

filesystem.polling.threads.watchdog.interval.seconds

- **Module**: cmas-nimh
- **Description**: Watchdog thread interval. Default: 30
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.4.0

filesystem.task.enabled

- **Module**: cmas-nimh
- **Description**: With this property service thread related to given poller can be disabled. Default: true
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true
- **Since**: 6.4.0

filesystem.task.interval.seconds

- **Module**: cmas-nimh
- **Description**: Default interval for polling mailboxes. Default: 60 seconds
- **Type**: integer
- **Restart required**: no

- **System**: no
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.4.0

filesystem.task.polling.folder

- **Module**: cmas-nimh
- **Description**: Polling folder location which will be scanned for emails in the format of eml files. Default: "mail" subdir of cmas data directory
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: c://cmas//mail
- **Since**: 6.4.0

filesystem.task.timeout.seconds

- **Module**: cmas-nimh
- **Description**: After this time (of inactivity) the service thread is considered as damaged and automatically restarted. Default: 120 seconds
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.4.0

filesystem.task.transaction.timeout.seconds

- **Module**: cmas-nimh
- **Description**: Default transaction timeout for email fetching transactions. Should be correlated with number of messages fetched at once. Default: 60 seconds
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.4.0

mailbox.1.connection.host

- **Module**: cmas-nimh
- **Description**: Host (server) for first configured mailbox. Will overwrite the default parameter `mailbox.default.connection.host`.

mailbox.1.connection.password

- **Module**: cmas-nimh
- **Description**: Password for first configured mailbox. Will overwrite the default parameter `mailbox.default.connection.password`.

mailbox.1.connection.port

- **Module**: cmas-nimh
- **Description**: Port for first configured mailbox. Will overwrite the default parameter `mailbox.default.connection.port`.

mailbox.1.connection.protocol

- **Module**: cmas-nimh
- **Description**: Protocol (e.g., IMAP or POP3) for first configured mailbox. Will overwrite the default parameter `mailbox.default.connection.protocol`.

mailbox.1.connection.username

- **Module**: cmas-nimh
- **Description**: User name for first configured mailbox. Will overwrite the default parameter `mailbox.default.connection.username`.

mailbox.2.connection.host

- **Module**: cmas-nimh
- **Description**: Host (server) for second configured mailbox. Will overwrite the default parameter `mailbox.default.connection.host`.

mailbox.2.connection.password

- **Module**: cmas-nimh
- **Description**: Password for second configured mailbox. Will overwrite the default parameter `mailbox.default.connection.password`.

mailbox.2.connection.port

- **Module**: cmas-nimh
- **Description**: Port for second configured mailbox. Will overwrite the default parameter `mailbox.default.connection.port`.

mailbox.2.connection.protocol

- **Module**: cmas-nimh
- **Description**: Protocol (e.g., IMAP or POP3) for second configured mailbox. Will overwrite the default parameter `mailbox.default.connection.protocol`.

mailbox.2.connection.username

- **Module**: cmas-nimh
- **Description**: User name for second configured mailbox. Will overwrite the default parameter `mailbox.default.connection.username`.

> For all NIMH-related mailbox properties, the following principle is used: a default property is defined (e.g. `mailbox.default.connection.port`). If no mailbox-specific value is configured, this default value will be used.

mailbox.default.connection.host

- **Module**: cmas-nimh
- **Description**: Host (server name) of a given mailbox from which the poller reads emails.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 10.10.1.157
- **Since**: 6.4.0

mailbox.default.connection.password

- **Module**: cmas-nimh
- **Description**: Password for given mailbox from which the poller reads emails.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: consol
- **Since**: 6.4.0

mailbox.default.connection.port

- **Module**: cmas-nimh
- **Description**: Port for a given mailbox from which the poller reads emails.
- **Type**: string

- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 143
- **Since**: 6.4.0

mailbox.default.connection.protocol

- **Module**: cmas-nimh
- **Description**: Poller's protocol e.g., IMAP or POP3. No default value
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: imap
- **Since**: 6.4.0

mailbox.default.connection.username

- **Module**: cmas-nimh
- **Description**: User name for a given mailbox from which the poller reads emails.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: username
- **Since**: 6.4.0

mailbox.default.session.mail.debug

- **Module**: cmas-nimh
- **Description**: Example javax.mail property - allows for more detailed javax.mail session debug-ging
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true
- **Since**: 6.4.0

mailbox.default.session.mail.imap.timeout

- **Module**: cmas-nimh
- **Description**: Example javax.mail property
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 120
- **Since**: 6.4.0

mailbox.default.session.mail.mime.address.strict

- **Module**: cmas-nimh
- **Description**: Example javax.mail property - counterpart of the old `mule mail.mime.strict`, allows to set not so strict email header parsing
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true
- **Since**: 6.4.0

mailbox.default.session.mail.pop3.timeout

- **Module**: cmas-nimh
- **Description**: Example javax.mail property.
- **Type**:
- **Restart required**:
- **System**:
- **Optional**:
- **Example value**:
- **Since**: 6.4.0

mailbox.default.session.mail.<protocol>.partialfetch

- **Module**: cmas-nimh
- **Description**: Sets java mail property for partialfetch i.e. controls whether the protocol partial-fetch capability should be used.
  For IMAP systems: in CM versions 6.10.7.0 and up, the value of `mailbox.default.session.mail.imap.partialfetch` is set to “false” during

the initial setup of a ConSol CM system. During an update of an existing ConSol CM system, the value of the property is left unchanged, if the property is already present. In case the property is not yet present, it is added with the default value.

- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**:
- **Since**: 6.9.4.0

mailbox.default.task.delete.read.messages

- **Module**: cmas-nimh
- **Description**: This defines whether messages should be removed from the mailbox after processing. For IMAP protocol messages are marked as SEEN by default. For POP3 protocol, when flag is set to true the message is removed, otherwise remains on server and will result in infinite reads. Default: false.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: false
- **Since**: 6.4.0

mailbox.default.task.enabled

- **Module**: cmas-nimh
- **Description**: With this property service thread related to given poller can be disabled. Default: true
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: false
- **Since**: 6.4.0

mailbox.default.task.interval.seconds

- **Module**: cmas-nimh
- **Description**: Default interval for polling mailboxes. Default: 60 seconds
- **Type**: integer

- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.4.0

mailbox.default.task.max.message.size

- **Module**: cmas-nimh
- **Description**: Maximum size of email messages (i.e., email plus attachment). Emails exceeding the size limit will not be automatically processed by NIMH but will be stored in the database (table `cmas_nimh_archived_mail`) and will therefore appear in the email backups in the Admin Tool (see section Email Backups). From there they can be resent, downloaded to the file system, or deleted. For those operations the message size is not relevant. Default is set to 10MB: 10485760
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 10485760
- **Since**: 6.4.0

mailbox.default.task.max.messages.per.run

- **Module**: cmas-nimh
- **Description**: Number of messages fetched at once from mailbox. Must be correlated with transaction timeout. Default set to: 20
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.4.0

mailbox.default.task.timeout.seconds

- **Module**: cmas-nimh
- **Description**: After this time (of inactivity) the service thread is considered as damaged and automatically restarted. Default: 120 seconds
- **Type**: integer
- **Restart required**: no

- **System**: no
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.4.0

mailbox.default.task.transaction.timeout.seconds

- **Module**: cmas-nimh
- **Description**: Default transaction timeout for email fetching transactions. Should be correlated with number of messages fetched at once. Default: 60 seconds
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.4.0

mailbox.polling.threads.mail.log.enabled

- **Module**: cmas-nimh
- **Description**: Enables email logging which is especially crucial in cluster environment (used as semaphore there)
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true (default)
- **Since**: 6.9.4.1

mailbox.polling.threads.number

- **Module**: cmas-nimh
- **Description**: Number of threads for accessing mailboxes. Default: 1
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 1
- **Since**: 6.4.0

queue.polling.threads.number

- **Module**: cmas-nimh
- **Description**: Number of threads started for emails' queue polling. Default: 1
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 1
- **Since**: 6.4.0

queue.polling.threads.shutdown.timeout.seconds

- **Module**: cmas-nimh
- **Description**: Waiting time after the shutdown signal. When the timeout is reached, the thread will be terminated. Default: 60
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.4.0

queue.polling.threads.watchdog.interval.seconds

- **Module**: cmas-nimh
- **Description**: Watchdog thread interval. Default: 30
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 30
- **Since**: 6.4.0

queue.task.error.pause.seconds

- **Module**: cmas-nimh
- **Description**: Maximum number of seconds, the queue poller waits after infrastructure (e.g. database) error. Default 180 seconds
- **Type**: integer
- **Restart required**: no

- **System**: no
- **Optional**: yes
- **Example value**: 180
- **Since**: 6.4.0

queue.task.interval.seconds

- **Module**: cmas-nimh
- **Description**: Main emails' queue polling thread interval. Default: 15
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 15
- **Since**: 6.4.0

queue.task.max.retries

- **Module**: cmas-nimh
- **Description**: Maximum number of email processing retries after an exception. When reached, the email is moved to the email archive. This email can be rescheduled again using NIMH API (or the Admin Tool).
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 10
- **Since**: 6.4.0

queue.task.timeout.seconds

- **Module**: cmas-nimh
- **Description**: After this time (of inactivity) the service thread is considered as damaged and automatically restarted. Default: 600 seconds
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 600
- **Since**: 6.4.0

queue.task.transaction.timeout.seconds

- **Module**: cmas-nimh
- **Description**: Transaction timeout for email processing in the pipe. Default: 60
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.4.0

mail.attachments.validation.info.sender

- **Module**: cmas-nimh-extension
- **Description**: Sets From header of attachments type *error notification mail*
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: admin@mail.com
- **Since**: 6.7.5

mail.attachments.validation.info.subject

- **Module**: cmas-nimh-extension
- **Description**: Sets subject of attachments type error notification mail.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: Email was not processed because its attachments were rejected!
- **Since**: 6.7.5

mail.db.archive

- **Module**: cmas-nimh-extension
- **Description**: If property is set to “true”, incoming emails are archived in the database.
- **Type**: boolean
- **Restart required**: no
- **System**: yes

- **Optional**: yes
- **Example value**: false (default)
- **Since**: 6.8.5.5

mail.error.from.address

- **Module**: cmas-nimh-extension
- **Description**: From address for error emails from NIMH
- **Type**: email
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: myuser@consol.de
- **Since**: 6.4.0

mail.error.to.address

- **Module**: cmas-nimh-extension
- **Description**: To address for error emails from NIMH. As a default the email address of the administrator which you have entered during system setup is used.
- **Type**: email
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: myuser@consol.de
- **Since**: 6.4.0

mail.on.error

- **Module**: cmas-nimh-extension
- **Description**: If set to “true ”an error email is sent to the above configured address in case the email message could not be processed. Default: true
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: false
- **Since**: 6.4.0

mail.process.error

- **Module**: cmas-nimh-extension
- **Description**: To address for error emails from Mule.
- **Type**: email
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: myuser@consol.de
- **Since**: 6.4.0

mail.ticketname.pattern

- **Module**: cmas-nimh-extension
- **Description**: Regular expression pattern used to identify the ticket name in the subject of incoming mails.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: .*?Ticket\s+\((\S+)\).*
- **Since**: 6.4.0

Attachments for Incoming Emails

attachment.allowed.types

- **Module**: cmas-core-server
- **Description**: Comma-separated list of allowed filename extensions (if no value defined, all file extensions are allowed).
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: txt,zip,doc
- **Since**: 6.5.0

attachment.max.size

- **Module**: cmas-core-server
- **Description**: Maximum attachment size, in MB. This is a validation property of the CM API. It controls the size of attachments at tickets, at units, and at resources. It also controls the size of incoming (not outgoing!) email attachments.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 100
- **Since**: 6.4.0

Email Encryption (Outgoing and Incoming)

These settings only apply if email encryption is active (true).

mail.encryption

- **Module**: cmas-core-server
- **Description**: If property is set to “true”, the encrypt checkbox in the Ticket Email Editor is checked by default.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true (default = false)
- **Since**: 6.8.4.0

In case certificates are stored in an LDAP directory, the following settings have to be made:

ldap.certificate.basedn

- **Module**: cmas-core-server
- **Description**: Base DN for certificates location in the LDAP tree. If not provided, `cmas-core-security, ldap.basedn` is used.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: ou=accounts,dc=consol,dc=de
- **Since**: 6.8.4

ldap.certificate.content.attribute

- **Module**: cmas-core-server
- **Description**: LDAP attribute name used where certificate data is stored in the LDAP tree. Default value: usercertificate
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: usercertificate
- **Since**: 6.8.4

ldap.certificate.password

- **Module**: cmas-core-server
- **Description**: LDAP Certificates manager password. If not set, `cmas-core-security, ldap.password` is used.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.8.4

ldap.certificate.providerurl

- **Module**: cmas-core-server
- **Description**: LDAP Certificates provider URL. If not set, `cmas-core-security, ldap.providerurl` is used.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: ldap://ldap.consol.de:389
- **Since**: 6.8.4

ldap.certificate.searchattr

- **Module**: cmas-core-server
- **Description**: LDAP attribute name used to search for certificate in the LDAP tree. Default value: mail
- **Type**: string

- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: mail
- **Since**: 6.8.4

ldap.certificate.userdn

- **Module**: cmas-core-server
- **Description**: LDAP Certificates manager DN. If not set, `cmas-core-security, ldap.userdn` is used.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.8.4

H.2.2.5 Activity Interval Configuration

admin.tool.session.check.interval

- **Module**: cmas-app-admin-tool
- **Description**: Admin Tool inactive (ended) sessions check time interval (in seconds)
- **Type**: integer
- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: 30
- **Since**: 6.7.5

server.session.timeout

- **Module**: cmas-core-server
- **Description**: Server session timeout (in seconds) for connected clients. Each client can overwrite this timeout with custom value using its ID (ADMIN_TOOL, WEB_CLIENT, WORKFLOW_EDITOR, TRACK (before 6.8, please use PORTER), ETL, REST) appended to property name, e.g., `server.session.timeout.ADMIN_TOOL`.
  Please see also the Page Customization attributes *updateTimeServerSessionActivityEnabled* and *updateTimeServerSessionActivity*, both of type *cmApplicationCustomization*.
- **Type**: integer
- **Restart required**: no

- **System**: yes
- **Optional**: no
- **Example value**: 1800
- **Since**: 6.6.1, 6.7.1

Detailed explanation for the Admin Tool:

- server.session.timeout.ADMIN_TOOL
  Defines the time interval how long the server considers a session valid while there is no activity from the Admin Tool holding the session. The Admin Tool is not aware of this value, it only suffers having an invalid session, if the last activity has been longer in the past.
- admin.tool.session.check.interval
  Defines the time between two checks done by the Admin Tool, if the server still considers its session valid.

For example, if admin.tool.session.check.interval = 60 the Admin Tool queries the server every minute if its session is still active/valid. In case server.session.timeout.ADMIN_TOOL = 600 the Admin Tool will get the response that the session is now invalid after ten minutes of inactivity.

H.2.2.6 Administrator Email Addresses

ConSol CM can use different administrator email addresses, depending on the subsystem. Please see Administrator and Notification Email Addresses for detailed explanations concerning admin email addresses. If no specific admin email addresses are configured, the global admin email address (that you have defined during system set-up) is used.

H.2.2.7 HTTP Header Configuration

It is possible to configure the HTTP response header returned with the web page in the Web Client. This allows, for example, to define security-related response headers according to the requirements dictated by policy or environment. These properties are managed in the module `cmweb-server-http-headers`. The name of the property is the field of the response header and the value of the property is the field value sent in the header.

> Both name and value must match the exact spelling of the HTTP specification for the desired response header field. The correct interpretation and handling of the header lies in the responsibility of the client browser.

Example Property: X-Frame-Options

- **Module**: cmweb-server-http-headers
- **Description**: Example property to illustrate the configuration of HTTP headers. In this case the delivered HTTP header contains the field *X-Frame-Options* with the value “SAMEORIGIN”.

  Each property in the module `cmweb-server-http-headers` represents one header field. The property name/key identifies the response header field and the value of the property is the field value sent in this header.

Please be aware that additional HTTP response headers must be correctly defined with the exact spelling as officially specified! Please note also that the correct interpretation and application of these headers is fully in the realm and responsibility of the client browser which requested the page!

- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: SAMEORIGIN
- **Since**: 6.10.8

H.2.3 List of System Properties by Module

This chapter lists the system properties included in the following modules.

- cmas-app-admin-tool (module)
- cmas-core-cache (module)
- cmas-core-index-common (module)
- cmas-core-security (module)
- cmas-core-server (module)
- cmas-core-shared (module)
- cmas-dwh-server (module)
- cmas-nimh (module)
- cmas-nimh-extension (module)
- cmas-restapi-core (module)
- cmas-setup-hibernate (module)
- cmas-setup-manager (module)
- cmas-setup-scene (module)
- cmas-workflow-engine (module)
- cmas-workflow-jbpm (module)
- cmweb-server-http-headers (module)
- cmweb-server-adapter (module)

H.2.3.1 cmas-app-admin-tool (module)

admin.tool.consumed.licences.check.interval

- **Module**: cmas-app-admin-tool
- **Description**: Sets the interval (in seconds) to monitor the number of consumed licenses. The default value is 30.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 30
- **Since**: 6.11.0.0

admin.tool.consumed.licences.pool.name

- **Module**: cmas-app-admin-tool
- **Description**: Sets the license pool name to monitor the number of consumed licenses. The default value is “CONCURRENT_USERS”.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: CONCURRENT_USERS
- **Since**: 6.11.0.0

admin.tool.session.check.interval

- **Module**: cmas-app-admin-tool
- **Description**: Admin Tool inactive (ended) sessions check time interval (in seconds)
- **Type**: integer
- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: 30
- **Since**: 6.7.5

autocomplete.enabled

- **Module**: cmas-app-admin-tool
- **Description**: If the flag is missing or its value is “false”, then the *Autocomplete address* navigation item is hidden in Admin Tool.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: true
- **Since**: 6.9.2.0

delete.ticket.enabled

- **Module**: cmas-app-admin-tool
- **Description**: Controls if the menu entry *Delete* is displayed in the context menu in the Admin Tool for the ticket list in ticket administration.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true
- **Since**: 6.9.4.0

dwh.administration.refresh.interval.seconds

- **Module**: cmas-app-admin-tool
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 10
- **Since**: 6.11.0.1

script.validation.interval.seconds

- **Module**: cmas-app-admin-tool
- **Description**: Interval in seconds between two code checks in the Admin Tool or the Process Designer code editor
- **Type**: Integer

- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: 1 (default)
- **Since**: 6.11.0.1

start.groovy.task.enabled

- **Module**: cmas-app-admin-tool
- **Description**: For being able to run Admin Tool scripts of type *Task* in the Admin Tool (navigation group *Services*, navigation item *Task Execution*). It is required to enable the *Start task* button, which is hidden by default. This is done by setting this system property to "true".
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true
- **Since**: 6.9.4.0

task.panel.refresh.interval.seconds

- **Module**: cmas-app-admin-tool
- **Description**: Time in seconds after which the task list (in the Admin Tool) of the Task Execution Framework is refreshed.
- **Type**: Integer
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: 10
- **Since**: 6.10.5.3 (not added automatically during update from versions prior to 6.10.5.3!)

H.2.3.2 cmas-core-cache (module)

cache-cluster-name

- **Module**: cmas-core-cache
- **Description**: JBoss cache cluster name.
- **Type**: string
- **Restart required**: yes
- **System**: yes

- **Optional**: no
- **Example value**: 635a6de1-629a-4129-8299-2d98633310f0
- **Since**: 6.4.0

eviction.event.queue.size

- **Module**: cmas-core-cache
- **Description**: The size of the queue holding cache events. The default value is 200000. It is recommended to increase the value slightly (up to 400000) on systems with high traffic or load.
- **Type**: integer
- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: 200000
- **Since**: 6.4.0

eviction.max.nodes

- **Module**: cmas-core-cache
- **Description**: Sets the maximum size of internal caches. The default value is 100000. Increasing it will lead to higher memory consumption and is not recommended unless explicitly advised by ConSol.
- **Type**: integer
- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: 100000
- **Since**: 6.4.0

eviction.wakeup.interval

- **Module**: cmas-core-cache
- **Description**: Sets the interval (in milliseconds) between two cache queue event processing cycles. The default value is 3000. It is recommended to decrease it (minimum is 1500) on systems with high traffic or load.
- **Type**: integer
- **Restart required**: yes
- **System**: yes
- **Optional**: no

- **Example value**: 3000
- **Since**: 6.4.0

H.2.3.3 cmas-core-index-common (module)

big.task.minimum.size

- **Module**: cmas-core-index-common
- **Description**: Indicates the minimum size of index task (in parts, each part has 100 entities) to qualify this task as a big one. Big tasks have lower priority than normal tasks.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 15 (default)
- **Since**: 6.8.3

database.notification.enabled

- **Module**: cmas-core-index-common
- **Description**: Indicates whether index update database notification channel should be used instead of JMS.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.8.4.7

database.notification.redelivery.delay.seconds

- **Module**: cmas-core-index-common
- **Description**: In case of index update database notification channel, indicates notification redelivery delay when an exception occurs.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 60
- **Since**: 6.8.4.7

database.notification.redelivery.max.attempts

- **Module**: cmas-core-index-common
- **Description**: In case of index update database notification channel, indicates maximum redelivery attempts when an exception occurs.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 60
- **Since**: 6.8.4.7

disable.admin.task.auto.commit

- **Module**: cmas-core-index-common
- **Description**: All tasks created for index update will be automatically executed right after creation.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.6.1

index.attachment

- **Module**: cmas-core-index-common
- **Description**: Specifies whether content of attachments is indexed.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true
- **Since**: 6.4.3

index.history

- **Module**: cmas-core-index-common
- **Description**: Specifies whether unit and ticket history are indexed.
- **Type**: boolean
- **Restart required**: no

- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.1.0
- **Removed in**: 6.11.0

index.status

- **Module**: cmas-core-index-common
- **Description**: Status of the Indexer, possible values RED, YELLOW, GREEN, will be displayed in the Admin Tool.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: GREEN
- **Since**: 6.6.1

index.task.worker.threads

- **Module**: cmas-core-index-common
- **Description**: How many threads will be used to execute index tasks (synchronization, administrative, and repair tasks).
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 1 (default) (we recommend to use a value not larger than 2)
- **Since**: 6.6.14, 6.7.3. Since 6.8.0 and exclusively in 6.6.21 also normal (live) index updates are affected by this property.

index.version.current

- **Module**: cmas-core-index-common
- **Description**: Holds information about current (possibly old) index version.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no

- **Example value**: 1 (default)
- **Since**: 6.7.0

index.version.newest

- **Module**: cmas-core-index-common
- **Description**: Holds information about which index version is considered newest.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 1 (default)
- **Since**: 6.7.0

indexed.assets.per.thread.in.memory

- **Module**: cmas-core-index-common
- **Description**: How many assets should be loaded into memory at once, per thread, during indexing.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 200 (default)
- **Since**: 6.8.0

indexed.engineers.per.thread.in.memory

- **Module**: cmas-core-index-common
- **Description**: How many engineers should be loaded into memory at once, per thread, during indexing.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 300 (default)
- **Since**: 6.6.14, 6.7.3

indexed.resources.per.thread.in.memory

- **Module**: cmas-core-index-common
- **Description**: How many resources should be loaded into memory at once, per thread, during indexing.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 200 (default)
- **Since**: 6.10.0.0

indexed.tickets.per.thread.in.memory

- **Module**: cmas-core-index-common
- **Description**: How many tickets should be loaded into memory at once, per thread, during indexing.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 100 (default)
- **Since**: 6.6.14, 6.7.3

indexed.units.per.thread.in.memory

- **Module**: cmas-core-index-common
- **Description**: How many units should be loaded into memory at once, per thread, during indexing.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 200 (default)
- **Since**: 6.6.14, 6.7.3

synchronize.master.address

- **Module**: cmas-core-index-common
- **Description**: Value of `-Dcmas.http.host.port` specifying how to connect to the indexing master server. Default null. Since 6.6.17 this value is configurable in set-up to designate the initial indexing master server. Please note that changing this value is only allowed when all cluster nodes' index change receivers are stopped.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 127.0.0.1:80
- **Since**: 6.6.0

synchronize.master.security.token

- **Module**: cmas-core-index-common
- **Description**: The password for accessing the index snapshot via URL, e.g., for index synchronization or for backups.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: token
- **Since**: 6.6.0

synchronize.master.security.user

- **Module**: cmas-core-index-common
- **Description**: The user name for accessing the index snapshot via URL, e.g., for index synchronization or for backups.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: user
- **Since**: 6.6.0

synchronize.master.timeout.minutes

- **Module**: cmas-core-index-common
- **Description**: How long the master server may continually fail until a new master gets elected. Default 5. Since 6.6.17 this value is configurable in set-up, where zero means that master server will never change (failover is disabled).
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 5
- **Since**: 6.6.0

synchronize.megabits.per.second

- **Module**: cmas-core-index-common
- **Description**: How much bandwidth the master server may consume when transferring index changes to all slave servers. Default 85. Please do not use all available bandwidth to transfer index changes between hosts, as doing so will most probably partition the cluster due to some subsystems being unable to communicate.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 85
- **Since**: 6.6.0

synchronize.sleep.millis

- **Module**: cmas-core-index-common
- **Description**: How often each slave server polls the master server for index changes. Default 1000.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 1000
- **Since**: 6.6.0

H.2.3.4 cmas-core-security (module)

admin.email

- **Module**: cmas-core-security
- **Description**: The email address of the ConSol CM administrator. The value which you entered during system set-up is used initially.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: myuser@consol.de
- **Since**: 6.0

admin.login

- **Module**: cmas-core-security
- **Description**: The name of the ConSol CM administrator. The value which you entered during system set-up is used initially.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: admin
- **Since**: 6.0

authentication.method

- **Module**: cmas-core-security
- **Description**: User authentication method (internal CM database or LDAP authentication). Allowed values are LDAP or DATABASE.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: DATABASE
- **Since**: 6.0

contact.authentication.method

- **Module**: cmas-core-security
- **Description**: Indicates contact authentication method, where possible values are DATABASE or LDAP or LDAP,DATABASE or DATABASE,LDAP.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Since**: 6.9.3.0

contact.inherit.permissions.only.to.own.customer.group

- **Module**: cmas-core-security
- **Description**: Indicates whether authenticated contact inherits all customer group permissions from the representing engineer (false) or only has permissions to his own customer group (true).
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Since**: 6.9.2.3

kerberos.v5.enabled

- **Module**: cmas-core-security
- **Description**: Indicates whether SSO via Kerberos is enabled.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false (default if Kerberos was not enabled during system set-up)
- **Since**: 6.2.0

kerberos.v5.username.regex

- **Module**: cmas-core-security
- **Description**: Regular expression used for mapping Kerberos principals to CM user login names.
- **Type**: string
- **Restart required**: no
- **System**: yes

- **Optional**: no
- **Example value**: (.*)@.*
- **Since**: 6.2.0

ldap.authentication

- **Module**: cmas-core-security
- **Description**: Authentication method used when using LDAP authentication. Possible values are 'anonymous' and 'simple' (default).
- **Type**: string
- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: simple
- **Since**: 6.0

ldap.basedn

- **Module**: cmas-core-security
- **Description**: Base DN used for looking up LDAP user accounts when using LDAP authentication.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: ou=accounts,dc=consol,dc=de
- **Since**: 6.0

ldap.contact.name.basedn

- **Module**: cmas-core-security
- **Description**: Base path to search for contact DN by LDAP ID (e.g. ou=a-ccounts,dc=consol,dc=de).
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Since**: 6.9.3.0

ldap.contact.name.password

- **Module**: cmas-core-security
- **Description**: Password to look up contact DN by LDAP ID. If not set, the anonymous account is used.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Since**: 6.9.3.0

ldap.contact.name.providerurl

- **Module**: cmas-core-security
- **Description**: Address of the LDAP server (ldap[s]://host:port).
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Since**: 6.9.3.0

ldap.contact.name.searchattr

- **Module**: cmas-core-security
- **Description**: Attribute to search for contact DN by LDAP ID (e.g. uid).
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Since**: 6.9.3.0

ldap.contact.name.userdn

- **Module**: cmas-core-security
- **Description**: User DN to look up contact DN by LDAP ID. If not set, the anonymous account is used.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Since**: 6.9.3.0

ldap.initialcontextfactory

- **Module**: cmas-core-security
- **Description**: Class name for the initial context factory of the LDAP implementation when using LDAP authentication. If it is not set, `com.sun.jndi.ldap.LdapCtxFactory` is used.
- **Type**: string
- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: com.sun.jndi.ldap.LdapCtxFactory
- **Since**: 6.0

ldap.password

- **Module**: cmas-core-security
- **Description**: Password for connecting to LDAP to look up users when using LDAP authentication. Only needed if look-up cannot be performed anonymously.
- **Type**: password
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.1.2

ldap.providerurl

- **Module**: cmas-core-security
- **Description**: LDAP provider when using LDAP authentication.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: ldap://myserver.consol.de:389
- **Since**: 6.0

ldap.searchattr

- **Module**: cmas-core-security
- **Description**: Search attribute for looking up LDAP entry associated with a CM login.
- **Type**: string
- **Restart required**: no
- **System**: yes

- **Optional**: no
- **Example value**: uid
- **Since**: 6.0

ldap.userdn

- **Module**: cmas-core-security
- **Description**: LDAP user for connecting to LDAP to look up users when using LDAP authentication. Only needed if look-up cannot be performed anonymously.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.1.2

password.reset.mail.from

- **Module**: cmas-core-security
- **Description**: The `From` address for the email which is sent to a customer who requests a new password (using the *Forgot your password?* link) in CM/Track and to an engineer who requests a new password (using the *Forgot your password?* link) in the Web Client.
- **Type**: String
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: mypwreset@consol.de
- **Since**: 6.11.0.1

policy.password.age

- **Module**: cmas-core-security
- **Description**: Maximum validity period, in number of days, example 183 (6 months), default value: 5500 (= 15 years, i.e. no password change enforced). In case you would like to have the engineer change his/her password asap, use one of the two following values:
  - **0**
    The engineer will be forced to change his/her password on the next login.
  - **1**
    The engineer will be forced to change his/her password the next day.
- **Type**: integer
- **Restart required**: no
- **System**: no

- **Optional**: yes
- **Example value**: 5500 (15 years, default)
- **Since**: 6.10.1.0

policy.password.pattern

- **Module**: cmas-core-security
- **Description**: Defines password pattern.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: "^.3,$" (default)
- **Since**: 6.10.1.0

policy.rotation.ratio

- **Module**: cmas-core-security
- **Description**: Defines how often password may repeat. E.g., setting the value to X means that the new password cannot be present among the user's X previous passwords.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 1 (default)
- **Since**: 6.10.1.0

policy.username.case.sensitive

- **Module**: cmas-core-security
- **Description**: Defines whether user names are case-sensitive.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true (default)
- **Since**: 6.10.1.0

policy.track.username.case.sensitive

- **Module**: cmas-core-security
- **Description**: Defines whether customer (user) names in CM/Track are treated case-sensitive on login.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true (default)
- **Since**: 6.11.0.0

resetCode.expiriationPeriod

- **Module**: cmas-core-security
- **Description**: Defines the expiration period for the link when resetting the password in CM/Track.
- **Type**: Integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 86400000 (default, 24 hours)
- **Since**: 6.10.1

H.2.3.5 cmas-core-server (module)

attachment.allowed.types

- **Module**: cmas-core-server
- **Description**: Comma-separated list of allowed filename extensions (if no value defined, all file extensions are allowed).
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: txt,zip,doc
- **Since**: 6.5.0

attachment.max.size

- **Module**: cmas-core-server
- **Description**: Maximum attachment size, in MB. This is a validation property of the CM API. It controls the size of attachments at tickets, at units, and at resources. It also controls the size of incoming (not outgoing!) email attachments.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 100
- **Since**: 6.4.0

calendar.csv.dateFormat

- **Module**: cmas-core-server
- **Description**: Format of the date given in the csv file containing the list of holidays.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: dd/MM/yyyy
- **Since**: 6.9.3.2

calendar.csv.separator

- **Module**: cmas-core-server
- **Description**: Separator used in the csv file containing the list of holidays.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: ,
- **Since**: 6.9.3.2

config.data.version

- **Module**: cmas-core-server
- **Description**: The internal version number of the current system configuration. This property is maintained internally, please do not change it unless advised by ConSol.
- **Type**: integer

- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 11
- **Since**: 6.0

config.import.global.transaction.enabled

- **Module**: cmas-core-server
- **Description**: Flag deciding whether configuration (without localizations) should be imported within single transaction.
- **Type**: Boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true
- **Since**: 6.11.1.0

dao.log.threshold.milliseconds

- **Module**: cmas-core-server
- **Description**: Used to configure database operation times logging. DAO methods whose execution take longer than the time set in this property (in milliseconds) are logged.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 500 (default)
- **Since**: 6.11.1.0

dao.log.username

- **Module**: cmas-core-server
- **Description**: Used to configure database operation times logging. The execution of DAO methods which are related to the user name stated in this property is logged. Only one user name can be provided.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes

- **Example value**: webadmin (default: empty string)
- **Since**: 6.11.1.0

defaultCommentClassName

- **Module**: cmas-core-server
- **Description**: Default text class name for comments.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**:
- **Since**: 6.3.0

defaultIncommingMailClassName

- **Module**: cmas-core-server
- **Description**: Default text class name for incoming emails.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Since**: 6.3.0

defaultOutgoingMailClassName

- **Module**: cmas-core-server
- **Description**: Default text class name for outgoing emails.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**:
- **Since**: 6.3.0

engineer.description.cache.enabled

- **Module**: cmas-core-server
- **Description**: Defines whether user descriptions are cached. The default value is “true”, please do not change it unless advised by ConSol.
- **Type**: boolean

- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: false
- **Since**: 6.11.0

engineer.description.mode

- **Module**: cmas-core-server
- **Description**: Defines whether user names in the ticket history are taken from the database or dynamically rendered using templates. The default value “DYNAMIC” is a bit more costly from the performance perspective, while “PROTOCOL” is faster but returns historical names which might be outdated. Use “PROTOCOL” if you have lots of history entries from many different users.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: PROTOCOL
- **Since**: 6.11.0

engineer.description.template.name

- **Module**: cmas-core-server
- **Description**: Defines the name of the template which is used to render engineer names for display in the Web Client. The template has to be stored in the *Templates* section of the Admin Tool. Default “engineer description template name”.
- **Type**: String
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Since**: 6.11.0

external.line.access.prefix

- **Module**: cmas-core-server
- **Description**: General prefix to dial before an area code. Set for each customer group separately.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes

- **Example value**: 411
- **Since**: 6.9.3.0

fetchSize.strategy

- **Module**: cmas-core-server
- **Description**: Strategy for selecting the fetch size on JDBC result sets.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: FetchSizePageBasedStrategy, FetchSizeThresholdStrategy, FetchSizeFixedStrategy
- **Since**: 6.8.4.1

fetchSize.strategy.FetchSizeFixedStrategy.value

- **Module**: cmas-core-server
- **Description**: Sets fetch size value if the selected strategy to set the fetch size is `FetchSizeFixedStrategy`.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 150
- **Since**: 6.8.4.1

fetchSize.strategy.FetchSizePageBasedStrategy.limit

- **Module**: cmas-core-server
- **Description**: Sets maximum fetch size value if the selected strategy to set the fetch size is `FetchSizePageBasedStrategy`.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 10000
- **Since**: 6.8.4.1

fetchSize.strategy.FetchSizeThresholdStrategy.value

- **Module**: cmas-core-server
- **Description**: Sets fetch size threshold border values if the selected strategy to set the fetch size is `FetchSizeThresholdStrategy`.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 150,300,600,1000
- **Since**: 6.8.4.1

heartbeat

- **Module**: cmas-core-server
- **Description**: Timestamp that indicates if an instance of the application is connected to the database schema.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 1234567899
- **Since**: 6.10.5.3

internal.line.access.prefix

- **Module**: cmas-core-server
- **Description**: Prefix that the company's telephony system asks for outside lines. Set for each customer group separately.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 199
- **Since**: 6.9.3.0

last.config.change

- **Module**: cmas-core-server
- **Description**: Random UUID created during the last configuration change. This is a value maintained internally, please do not change it unless advised by ConSol.

- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 2573c7b7-2bf5-47ff-b5a2-bad31951a266
- **Since**: 6.1.0, 6.2.1

last.config.change.templates

- **Module**: cmas-core-server
- **Description**: Random UUID created during the last change in templates. This is a value maintained internally, please do not change it unless advised by ConSol.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 2573c7c7-2af5-4eff-b9c2-bad31951a266
- **Since**: 6.10.5.0

ldap.certificate.basedn

- **Module**: cmas-core-server
- **Description**: Base DN for certificates location in the LDAP tree. If not provided, `cmas-core-security, ldap.basedn` is used.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: ou=accounts,dc=consol,dc=de
- **Since**: 6.8.4

ldap.certificate.content.attribute

- **Module**: cmas-core-server
- **Description**: LDAP attribute name used where certificate data is stored in the LDAP tree. Default value: usercertificate
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes

- **Example value**: usercertificate
- **Since**: 6.8.4

ldap.certificate.password

- **Module**: cmas-core-server
- **Description**: LDAP Certificates manager password. If not set, `cmas-core-security, ldap.password` is used.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.8.4

ldap.certificate.providerurl

- **Module**: cmas-core-server
- **Description**: LDAP Certificates provider URL. If not set, `cmas-core-security, ldap.providerurl` is used.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: ldap://ldap.consol.de:389
- **Since**: 6.8.4

ldap.certificate.searchattr

- **Module**: cmas-core-server
- **Description**: LDAP attribute name used to search for certificate in the LDAP tree. Default value: mail
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: mail
- **Since**: 6.8.4

ldap.certificate.userdn

- **Module**: cmas-core-server
- **Description**: LDAP Certificates manager DN. If not set, `cmas-core-security, ldap.userdn` is used.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.8.4

local.country.prefix

- **Module**: cmas-core-server
- **Description**: Prefix of the local country code. Set for each customer group separately.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 48
- **Since**: 6.9.3.0

mail.encryption

- **Module**: cmas-core-server
- **Description**: If property is set to “true”, the encrypt checkbox in the Ticket Email Editor is checked by default.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true (default = false)
- **Since**: 6.8.4.0

mail.notification.engineerChange

- **Module**: cmas-core-server
- **Description**: Whether notification emails should be sent when the engineer of a ticket is changed.
- **Type**: boolean
- **Restart required**: no

- **System**: yes
- **Optional**: no
- **Example value**: true
- **Since**: 6.1.0

mail.notification.sender

- **Module**: cmas-core-server
- **Description**: From address for notification emails when the engineer of a ticket is changed. If not set, `cmas-core-security, admin.email` is used instead.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: cm6notification@cm6installation
- **Since**: 6.6.3

mail.redelivery.retry.count

- **Module**: cmas-core-server
- **Description**: Number of redelivery attempts of an outgoing email.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 3
- **Since**: 6.1.0

mail.smtp.email

- **Module**: cmas-core-server
- **Description**: SMTP email URL for outgoing emails
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: smtp://mail.mydomain.com:25
- **Since**: 6.0

mail.smtp.envelopesender

- **Module**: cmas-core-server
- **Description**: Email address used as sender in SMTP envelope. If not set, the From address of the email is used.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: mysender@mydomain.com
- **Since**: 6.5.7

mail.smtp.tls.enabled

- **Module**: cmas-core-server
- **Description**: Activates SMTP via SSL/TLS (SMTPS) for sending emails from the Web Client and scripts. The default value is “false”. If it is set to “true”, SMTPS is activated for sending emails.
- **Type**: boolean
- **Restart required**: yes
- **System**: no
- **Optional**: yes
- **Example value**: true
- **Since**: 6.11.1.6

max.licences.perUser

- **Module**: cmas-core-server
- **Description**: Sets maximum licenses single user can use (e.g., logging in from different browsers). By default this value is not restricted.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 10
- **Since**: 6.8.4.5

monitoring.engineer.login

- **Module**: cmas-core-server
- **Description**: Login of monitoring engineer.
- **Type**: string

- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: nagios
- **Since**: 6.9.3.0

monitoring.unit.login

- **Module**: cmas-core-server
- **Description**: Login of monitoring unit.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: nagios
- **Since**: 6.9.3.0

nimh.enabled

- **Module**: cmas-core-server
- **Description**: Enables NIMH service. Must be suffixed with the cluster node ID, e.g., `nimh.enabled.NODEID` = “true”.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: false
- **Since**: 6.9.4.0

number.of.tasks

- **Module**: cmas-core-server
- **Description**: Number of threads to use by the Task Execution Framework (TEF).
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 1
- **Since**: 6.9.4.0

recent.items.cleanup.cluster.node.id

- **Module**: cmas-core-server
- **Description**: Value of a `-Dcmas.clusternode.id` designating the node which will clean up recent items.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 1 (assuming the cluster node started with `-Dcmas.clusternode.id=1` parameter)
- **Since**: 6.11.0.1

recent.items.cleanup.interval.minutes

- **Module**: cmas-core-server
- **Description**: Controls the time interval (in minutes) in which recent items should be checked for removal.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 60
- **Since**: 6.11.0.1

recent.items.max.per.engineer

- **Module**: cmas-core-server
- **Description**: Maximum number of preserved recent items per engineer while cleaning up (older recent items will be deleted).
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 500
- **Since**: 6.11.0.1

recent.items.persistence.enabled

- **Module**: cmas-core-server
- **Description**: Enables persistence of recent items, if false - prevents storing new recent items.

- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true
- **Since**: 6.11.1.0

resource.replace.batchSize

- **Module**: cmas-core-server
- **Description**: Defines the number of objects to be processed in a resource replace action.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 5
- **Since**: 6.10.0.0

resource.replace.timeout

- **Module**: cmas-core-server
- **Description**: Transaction timeout (in seconds) of a resource replacement action step.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 120
- **Since**: 6.10.0.0

script.evict.unused.after.hours

- **Module**: cmas-core-server
- **Description**: Determines the number of hours for which unused scripts remain in the cache. After this time, the compiled class of the script is removed. The ConSol CM server checks for scripts to evict every hour.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no

- **Example value**: 24 (default)
- **Since**: 6.11.1.14

script.logging.threshold.seconds

- **Module**: cmas-core-server
- **Description**: When this time, in seconds, is exceeded during script execution, a warning is emitted in the logs.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 10 (default)
- **Since**: 6.10.1.0

serial.mods.tracking.enabled

- **Module**: cmas-core-server
- **Description**: Low level technical flag deciding whether serial diff tracking for entities is enabled. If enabled, there will be no `StackOverflow Error` in case a dependency between two entities (for example engineer and ticket) causes an infinite loop first and then as a result, the StackOverflow. The property must be added to the configuration manually. It will not be added to a system configuration during setup or update.

  > ⚠ Please enable the restricted ticket change behavior described in this section only when advised by a ConSol representative! It is a low level technical flag with intricate consequences for system behavior and thus should not be used without thorough scrutiny.

- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: false (default)
- **Since**: 6.10.7.0, 6.11.0.5

server.session.archive.reaper.interval

- **Module**: cmas-core-server
- **Description**: Server archived sessions reaper interval (in seconds).
- **Type**: integer
- **Restart required**: no

- **System**: yes
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.7.1

server.session.archive.timeout

- **Module**: cmas-core-server
- **Description**: Server sessions archive validity timeout (in days). After this time session info is removed from the DB.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 31
- **Since**: 6.7.1

server.session.reaper.interval

- **Module**: cmas-core-server
- **Description**: Server inactive (ended) sessions reaper interval (in seconds).
- **Type**: integer
- **Restart required**: only Session Service
- **System**: yes
- **Optional**: no
- **Example value**: 60
- **Since**: 6.6.1, 6.7.1

server.session.timeout

- **Module**: cmas-core-server
- **Description**: Server session timeout (in seconds) for connected clients. Each client can overwrite this timeout with custom value using its ID (ADMIN_TOOL, WEB_CLIENT, WORKFLOW_EDITOR, TRACK (before 6.8, please use PORTER), ETL, REST) appended to property name, e.g., `server.session.timeout.ADMIN_TOOL`.
  Please see also the Page Customization attributes *updateTimeServerSessionActivityEnabled* and *updateTimeServerSessionActivity*, both of type *cmApplicationCustomization*.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no

- **Example value**: 1800
- **Since**: 6.6.1, 6.7.1

Detailed explanation for the Admin Tool:

- server.session.timeout.ADMIN_TOOL
  Defines the time interval how long the server considers a session valid while there is no activity from the Admin Tool holding the session. The Admin Tool is not aware of this value, it only suffers having an invalid session, if the last activity has been longer in the past.
- admin.tool.session.check.interval
  Defines the time between two checks done by the Admin Tool, if the server still considers its session valid.

For example, if admin.tool.session.check.interval = 60 the Admin Tool queries the server every minute if its session is still active/valid. In case server.session.timeout.ADMIN_TOOL = 600 the Admin Tool will get the response that the session is now invalid after ten minutes of inactivity.

skip.wfl.transfer.cleanup

- **Module**: cmas-core-server
- **Description**: If set to “true”, skips workflow cleanup after transfer.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: false (default)
- **Since**: 6.9.4.1

skip.wfl.transfer.translations.cleanup

- **Module**: cmas-core-server
- **Description**: Enables skipping the cleanup of localized properties of removed workflow elements.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: false (default)
- **Since**: 6.10.5.5

strict.utf.bmp.enabled

- **Module**: cmas-core-server
- **Description**: In ConSol CM versions lower than 6.10.6, incoming emails with a subject line containing four-byte UTF8 characters could not be handled by some installations using the MySQL database engine. The reason is the encoding/collation configuration of the database using a two-byte BMP (Basic Multilingual Plane) 0 plane which cannot be changed in some installations for technical reasons. Other database engines were unaffected. Emails with this encoding could not be imported into the system at all in CM versions lower than 6.10.6. In order to accommodate this issue this system property for configuration is available.
  Setting it to “true” will filter out all four-byte UTF8 characters before any database interaction, so the problems mentioned above will not occur.
  The property value is “true” by default for MySQL databases, and “false” for any other database where it should not be necessary at all. Change it for a MySQL database only, if the settings positively will support four-byte characters.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 100
- **Since**: 6.10.6.0

task.execution.interval.seconds

- **Module**: cmas-core-server
- **Description**: Time in seconds between the end of an accomplished task in the TEF (Task Execution Framework) and the start of the next task.
- **Type**: Integer
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: 5
- **Since**: 6

task.execution.node.id

- **Module**: cmas-core-server
- **Description**: Only relevant in clustered environments. The ID of the node where scripts of the TEF (Task Execution Framework) will be executed. This applies to both scripts called from the workflow and scripts called manually using the Admin Tool. The Admin Tool can be started from any node.
- **Type**: Integer
- **Restart required**: no

- **System**: no
- **Optional**: no
- **Example value**: 2
- **Since**: 6.11.0.1

tickets.delete.size

- **Module**: cmas-core-server
- **Description**: Defines a number of tickets deleted per transaction. By default it is set to 10.
- **Type**: integer
- **Restart required**: only Session Service
- **System**: yes
- **Optional**: no
- **Example value**: 10
- **Since**: 6.8.1

ticket.delete.timeout

- **Module**: cmas-core-server
- **Description**: Transaction timeout (in seconds) for deleting tickets.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 60
- **Since**: 6.1.3

ticket.from.incoming.message.accepted.links

- **Module**: cmas-core-server
- **Description**: List of domains to which links in incoming emails and links in comments added via REST API are clickable in the ticket history. Regular expressions can be used to specify the allowed URLs. It is possible to add several URLs by using a whitespace as delimiter. The URL must start with one of the allowed protocols (http, https, ftp, ftps, file, mailto). All other links are removed, i.e., the link is displayed in the ticket history as text but it cannot be clicked. If the property is left empty, all links are removed. The regular expression `.+` can be used to allow all domains.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: no

- **Example value**: https://.*\.consol\.de (allows links to “https://<any>.consol.de”)
- **Since**: 6.11.1.7

⚠ Please note that whitelisting domains might make ConSol CM vulnerable to cross-site scripting and other attacks. Choose the domains you whitelist carefully!

transaction.timeout.minutes

- **Module**: cmas-core-server
- **Description**: Sets the transaction timeout for the task execution service, i.e., one run of a task must finish before this timeout is reached. The changes are visible only for new tasks, the execution of which started after the configuration change.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 10*3600 (10 hours - default)
- **Since**: 6.10

unit.description.mode

- **Module**: cmas-core-server
- **Description**: Defines whether unit (contact) descriptions in the ticket history are taken from the database or dynamically rendered using templates. The value, “DYNAMIC”, is a bit more costly from the performance perspective, while “PROTOCOL” is faster but returns historical names which might be outdated. Use “PROTOCOL” if you have lots of history entries from many different units. This is also the default value in CM versions 6.11.1.1 and up. In CM versions up to 6.11.1.0, “DYNAMIC” is the default.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: PROTOCOL
- **Since**: 6.11.0

unit.replace.batchSize

- **Module**: cmas-core-server
- **Description**: Defines the number of objects to be processed in a unit replace action.
- **Type**: integer
- **Restart required**: no

- **System**: yes
- **Optional**: no
- **Example value**: 5
- **Since**: 6.8.2

unit.replace.timeout

- **Module**: cmas-core-server
- **Description**: Transaction timeout (seconds) of a unit replacement action step.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 120
- **Since**: 6.8.2

unused.content.remover.cluster.node.id

- **Module**: cmas-core-server
- **Description**: Value of a `cmas.clusternode.id` designating which node will remove unused ticket attachments and unit content entries.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 1 (assuming cluster node started with the parameter –`Dcmas.clusternode.id=1`)
- **Since**: 6.9.0.0

unused.content.remover.enabled

- **Module**: cmas-core-server
- **Description**: Specifies whether removal of unused ticket attachments and unit content entries should take place.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true
- **Since**: 6.9.0.0

unused.content.remover.polling.minutes

- **Module**: cmas-core-server
- **Description**: How often unused ticket attachments and unit content entries should be checked for removal.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 15
- **Since**: 6.9.0.0

unused.content.remover.ttl.minutes

- **Module**: cmas-core-server
- **Description**: Minimum interval, in minutes, after which unused ticket attachments and unit content entries can be removed.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 1440
- **Since**: 6.9.0.0

warmup.executor.enabled

- **Module**: cmas-core-server
- **Description**: Specifies whether the server should asynchronously warm up during startup (e.g., fill some of the internal caches).
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true
- **Since**: 6.9.4.2

wfl.sticky.transfer.disabled

- **Module**: cmas-core-server
- **Description**: Enables using preserved original names of workflow elements.
- **Type**: boolean

- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: false
- **Since**: 6.10.1.0

H.2.3.6 cmas-core-shared (module)

cluster.mode

- **Module**: cmas-core-shared
- **Description**: Specifies whether CMAS is running in cluster.
- **Type**: boolean
- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.1.0

cluster.unicast

- **Module**: cmas-core-shared
- **Description**: Flag to activate jgroups unicast mode for ConSol CM clusters (as opposed to the default multicast mode causing problems in some data center environments). If set to “true” remember to set the JVM start parameters: `jgroups.bind.port`, `jgroups.bind.address` and `jgroups.initial_hosts`.
- **Type**: boolean
- **Restart required**: yes
- **System**: yes
- **Optional**: yes
- **Example value**: false (default)
- **Since**: 6.11.0.0

data.directory

- **Module**: cmas-core-shared
- **Description**: Directory for CMAS data (e.g., index)
- **Type**: string
- **Restart required**: no
- **System**: yes

- **Optional**: no
- **Example value**: C:\Users\user\cmas
- **Since**: 6.0

expert.mode

- **Module**: cmas-core-shared
- **Description**: Switches expert mode on/off thereby unblocking/blocking expert features. E.g., only in expert mode, the CM system property `initialized` will be available.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.0

H.2.3.7 cmas-dwh-server (module)

autocommit.cf.changes

- **Module**: cmas-dwh-server
- **Description**: Defines whether DWH tasks which result from configurational changes on ticket fields are executed automatically without manual interaction in the Admin Tool. Can be also set in the Admin Tool in the navigation item *DWH*. The default and recommended value is “false”.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.7.0

batch-commit-interval

- **Module**: cmas-dwh-server
- **Description**: Number of objects in a JMS message. Larger values mean better transfer performance at the cost of higher memory usage.
  Starting with ***ConSol CM*** version 6.11, this property is only used if the package size of a DWH operation is not set. This can only happen when the command is directly addressed to the Java MBean consol.cmas.global.dwh.synchronizationService, e.g. using the update() method. When a DWH operation is started using the Admin Tool, there is always a value for the package size. If not explicitly set, the default value of 1000 is used as value for the batch.-commit.interval.

- **Default value**: 1000
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 100
- **Since**: 6.0.0

communication.channel

- **Module**: cmas-dwh-server
- **Description**: Communication channel. Only possible value since CM version 6.11.0.0: DIRECT
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: DIRECT
- **Since**: 6.8.5.0
- **Removed in**: 6.11.0.0 (DIRECT mode is the only available mode and is set automatically)

dwh.mode

- **Module**: cmas-dwh-server
- **Description**: Current mode for DWH data transfer. Possible values are OFF, ADMIN, LIVE
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: OFF
- **Since**: 6.0.1

ignore-queues

- **Module**: cmas-dwh-server
- **Description**: A comma-separated list of queue names which are not not transferred to the DWH.
- **Type**: string
- **Restart required**: no
- **System**: yes

- **Optional**: yes
- **Example value**: QueueName1,QueueName2,QueueName3
- **Since**: 6.6.19
- **Removed in**: 6.8.1

is.cmrf.alive

- **Module**: cmas-dwh-server
- **Description**: As a starting point, the time the last message was sent to CMRF should be used. If a response from CMRF is not received after value (in seconds), it should create a DWH operation status with an error message indicating that CMRF is down.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 1200
- **Since**: 6.7.0

java.naming.factory.initial

- **Module**: cmas-dwh-server
- **Description**: Factory class for the DWH context factory.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: org.jnp.interfaces.NamingContextFactory
- **Since**: 6.0.1
- **Removed in**: 6.11.0.0

java.naming.factory.url.pkgs

- **Module**: cmas-dwh-server
- **Description**:
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: org.jboss.naming:org.jnp.interfaces

- **Since**: 6.0.1
- **Removed in**: 6.11.0.0

java.naming.provider.url

- **Module**: cmas-dwh-server
- **Description**: URL of naming provider.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: localhost
- **Since**: 6.0.1
- **Removed in**: 6.11.0.0

last.ping.timestamp

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 32323214
- **Since**: 6.11.0.1

live.start

- **Module**: cmas-dwh-server
- **Description**: When the DWH synchronization mode is set to LIVE using the Admin Tool (navigation group *Data Warehouse*, navigation item *Administration*, *Configuration* button), this property is created and set to the current date.
  If LIVE mode is not enabled and there is no data in `cmas_dwh_ser_sync_object`, the property `live.start` is deleted.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes (automatically added in DWH “LIVE” mode)
- **Example value**: 15028802377645
- **Since**: 6.7.0

notification.error.description

- **Module**: cmas-dwh-server
- **Description**: Text for error emails from the DWH.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: Error occurred
- **Since**: 6.0.1

notification.error.from

- **Module**: cmas-dwh-server
- **Description**: From address for error emails from the DWH
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.0.1

notification.error.subject

- **Module**: cmas-dwh-server
- **Description**: Subject for error emails from the DWH
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: Error occurred
- **Since**: 6.0.1

notification.error.to

- **Module**: cmas-dwh-server
- **Description**: To address for error emails from the DWH
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no

- **Example value**: myuser@consol.de
- **Since**: 6.0.1

notification.finished_successfully.description

- **Module**: cmas-dwh-server
- **Description**: Text for emails from the DWH when a transfer finishes successfully.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: Transfer finished successfully
- **Since**: 6.0.1

notification.finished_successfully.from

- **Module**: cmas-dwh-server
- **Description**: From address for emails from the DWH when a transfer finishes successfully.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.0.1

notification.finished_successfully.subject

- **Module**: cmas-dwh-server
- **Description**: Subject for emails from the DWH when a transfer finishes successfully.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: Transfer finished successfully
- **Since**: 6.0.1

notification.finished_successfully.to

- **Module**: cmas-dwh-server
- **Description**: To address for emails from the DWH when a transfer finishes successfully.
- **Type**: string
- **Restart required**: yes

- **System**: yes
- **Optional**: no
- **Example value**: myuser@consol.de
- **Since**: 6.0.1

notification.finished_unsuccessfully.description

- **Module**: cmas-dwh-server
- **Description**: Text for emails from the DWH when a transfer finishes unsuccessfully.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: Transfer finished unsuccessfully
- **Since**: 6.0.1

notification.finished_unsuccessfully.from

- **Module**: cmas-dwh-server
- **Description**: From address for emails from the DWH when a transfer finishes unsuccessfully.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.0.1

notification.finished_unsuccessfully.subject

- **Module**: cmas-dwh-server
- **Description**: Subject for emails from the DWH when a transfer finishes unsuccessfully.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: Transfer finished unsuccessfully
- **Since**: 6.0.1

notification.finished_unsuccessfully.to

- **Module**: cmas-dwh-server
- **Description**: To address for emails from the DWH when a transfer finishes unsuccessfully.

- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: myuser@consol.de
- **Since**: 6.0.1

notification.host

- **Module**: cmas-dwh-server
- **Description**: Email (SMTP) server hostname for sending DWH emails.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: myserver.consol.de
- **Since**: 6.0.1

notification.password

- **Module**: cmas-dwh-server
- **Description**: Password for sending DWH emails (optional).
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.0.1

notification.port

- **Module**: cmas-dwh-server
- **Description**: SMTP port for sending DWH emails.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 25
- **Since**: 6.0.1

notification.protocol

- **Module**: cmas-dwh-server
- **Description**: The protocol used for sending emails from the DWH.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: pop3\

notification.tls.enabled

- **Module**: cmas-dwh-server
- **Description**: Activates SMTP via SSL/TLS (SMTPS) for sending notification emails from the DWH. The default value is “false”. If it is set to “true”, SMTPS is activated for sending notifications from the DWH.
- **Type**: string
- **Restart required**: yes
- **System**: no
- **Optional**: yes
- **Example value**: false
- **Since**: 6.11.1.6

notification.username

- **Module**: cmas-dwh-server
- **Description**: (SMTP) User name for sending DWH emails.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: myuser
- **Since**: 6.0.1

recoverable.exceptions

- **Module**: cmas-dwh-server
- **Description**: Comma-separated list of exception definitions: CLASS[+][:REGEX]. The exceptions included in the list do not stop CM from sending to the CMRF process, but force it to try again. If optional '+' after CLASS is present, classes which extend CLASS are matched.
- **Type**: string

- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: java.sql.SQLRecoverableException,java.lang.RuntimeException+:.*T.1\,2T.*
- **Since**: 6.8.4.6

skip-ticket

- **Module**: cmas-dwh-server
- **Description**: Tickets are not transferred during transfer/update.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.6.19
- **Removed in**: 6.8.1

skip-ticket-history

- **Module**: cmas-dwh-server
- **Description**: History of ticket is not transferred during transfer/update.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.6.19
- **Removed in**: 6.8.1

skip-unit

- **Module**: cmas-dwh-server
- **Description**: Units are not transferred during transfer/update.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false

- **Since**: 6.6.19
- **Removed in**: 6.8.1

skip-unit-history

- **Module**: cmas-dwh-server
- **Description**: History of unit is not transferred during transfer/update.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.6.19
- **Removed in**: 6.8.1

split.history

- **Module**: cmas-dwh-server
- **Description**: Changes the SQL that fetches the history for the tickets during DWH transfer not to all tickets at once but only for one ticket per SQL.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: false
- **Since**: 6.8.0

statistics.calendar

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.client.group

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.contact.role

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.content.entry

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.content.entry.class

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes

- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.content.entry.history

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.customer.definition

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.engineer

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.enum.group

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.

- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.field.definition

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.group.definition

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.locale

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.localized.property

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.mla

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.project

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.queue

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes

- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.resource

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.resource.group

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.resource.history

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.resource.relation.definition

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.

- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.resource.type

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.ticket

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.ticket.function

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.ticket.history

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.time.booking

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.timestamp

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.unit

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes

- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.unit.history

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.unit.relation.definition

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.workflow

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

time.buffer

- **Module**: cmas-dwh-server
- **Description**: Number of minutes to extend date of start live mode.

- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 5
- **Since**: 6.8.1.11

unit.transfer.order

- **Module**: cmas-dwh-server
- **Description**: Define in which order customer field groups should be transferred to the DWH.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: company;customer
- **Since**: 6.6.19
- **Removed in**: 6.8.1

H.2.3.8 cmas-nimh (module)

filesystem.polling.threads.number

- **Module**: cmas-nimh
- **Description**: Number of threads started for db emails' queue polling. Default: 1
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 10
- **Since**: 6.4.0

filesystem.polling.threads.shutdown.timeout.seconds

- **Module**: cmas-nimh
- **Description**: Waiting time after the shutdown signal. When the timeout reached, thread will be terminated. Default: 60
- **Type**: integer
- **Restart required**: no
- **System**: no

- **Optional**: yes
- **Example value**: 60
- **Since**: 6.4.0

filesystem.polling.threads.watchdog.interval.seconds

- **Module**: cmas-nimh
- **Description**: Watchdog thread interval. Default: 30
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.4.0

filesystem.task.enabled

- **Module**: cmas-nimh
- **Description**: With this property service thread related to given poller can be disabled. Default: true
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true
- **Since**: 6.4.0

filesystem.task.interval.seconds

- **Module**: cmas-nimh
- **Description**: Default interval for polling mailboxes. Default: 60 seconds
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.4.0

filesystem.task.polling.folder

- **Module**: cmas-nimh
- **Description**: Polling folder location which will be scanned for emails in the format of eml files. Default: "mail" subdir of cmas data directory
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: c://cmas//mail
- **Since**: 6.4.0

filesystem.task.timeout.seconds

- **Module**: cmas-nimh
- **Description**: After this time (of inactivity) the service thread is considered as damaged and automatically restarted. Default: 120 seconds
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.4.0

filesystem.task.transaction.timeout.seconds

- **Module**: cmas-nimh
- **Description**: Default transaction timeout for email fetching transactions. Should be correlated with number of messages fetched at once. Default: 60 seconds
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.4.0

mailbox.1.connection.host

- **Module**: cmas-nimh
- **Description**: Host (server) for first configured mailbox. Will overwrite the default parameter `mailbox.default.connection.host`.

mailbox.1.connection.password

- **Module**: cmas-nimh
- **Description**: Password for first configured mailbox. Will overwrite the default parameter `mailbox.default.connection.password`.

mailbox.1.connection.port

- **Module**: cmas-nimh
- **Description**: Port for first configured mailbox. Will overwrite the default parameter `mailbox.default.connection.port`.

mailbox.1.connection.protocol

- **Module**: cmas-nimh
- **Description**: Protocol (e.g., IMAP or POP3) for first configured mailbox. Will overwrite the default parameter `mailbox.default.connection.protocol`.

mailbox.1.connection.username

- **Module**: cmas-nimh
- **Description**: User name for first configured mailbox. Will overwrite the default parameter `mailbox.default.connection.username`.

mailbox.2.connection.host

- **Module**: cmas-nimh
- **Description**: Host (server) for second configured mailbox. Will overwrite the default parameter `mailbox.default.connection.host`.

mailbox.2.connection.password

- **Module**: cmas-nimh
- **Description**: Password for second configured mailbox. Will overwrite the default parameter `mailbox.default.connection.password`.

mailbox.2.connection.port

- **Module**: cmas-nimh
- **Description**: Port for second configured mailbox. Will overwrite the default parameter `mailbox.default.connection.port`.

mailbox.2.connection.protocol

- **Module**: cmas-nimh
- **Description**: Protocol (e.g., IMAP or POP3) for second configured mailbox. Will overwrite the default parameter `mailbox.default.connection.protocol`.

mailbox.2.connection.username

- **Module**: cmas-nimh
- **Description**: User name for second configured mailbox. Will overwrite the default parameter `mailbox.default.connection.username`.

> For all NIMH-related mailbox properties, the following principle is used: a default property is defined (e.g. `mailbox.default.connection.port`). If no mailbox-specific value is configured, this default value will be used.

mailbox.default.connection.host

- **Module**: cmas-nimh
- **Description**: Host (server name) of a given mailbox from which the poller reads emails.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 10.10.1.157
- **Since**: 6.4.0

mailbox.default.connection.password

- **Module**: cmas-nimh
- **Description**: Password for given mailbox from which the poller reads emails.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: consol
- **Since**: 6.4.0

mailbox.default.connection.port

- **Module**: cmas-nimh
- **Description**: Port for a given mailbox from which the poller reads emails.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes

- **Example value**: 143
- **Since**: 6.4.0

mailbox.default.connection.protocol

- **Module**: cmas-nimh
- **Description**: Poller's protocol e.g., IMAP or POP3. No default value
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: imap
- **Since**: 6.4.0

mailbox.default.connection.username

- **Module**: cmas-nimh
- **Description**: User name for a given mailbox from which the poller reads emails.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: username
- **Since**: 6.4.0

mailbox.default.session.mail.debug

- **Module**: cmas-nimh
- **Description**: Example javax.mail property - allows for more detailed javax.mail session debug-ging
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true
- **Since**: 6.4.0

mailbox.default.session.mail.imap.timeout

- **Module**: cmas-nimh
- **Description**: Example javax.mail property

- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 120
- **Since**: 6.4.0

mailbox.default.session.mail.mime.address.strict

- **Module**: cmas-nimh
- **Description**: Example javax.mail property - counterpart of the old `mule mail.mime.strict`, allows to set not so strict email header parsing
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true
- **Since**: 6.4.0

mailbox.default.session.mail.pop3.timeout

- **Module**: cmas-nimh
- **Description**: Example javax.mail property.
- **Type**:
- **Restart required**:
- **System**:
- **Optional**:
- **Example value**:
- **Since**: 6.4.0

mailbox.default.session.mail.<protocol>.fetchsize

- **Module**: cmas-nimh
- **Description**: Sets java mail property for partialfetch size in bytes for the indicated protocol. For IMAP systems: in CM versions 6.10.7.0 and up, the value of `mailbox.default.session.mail.imap.fetchsize` is set to 1048576 (equals 1 MB) during the initial setup of a ConSol CM system. During an update of an existing ConSol CM system, the value of the property is left unchanged, if the property is already present. In case the property is not yet present, it is added with the default value.
- **Type**: integer
- **Restart required**: no

- **System**: yes
- **Optional**: yes
- **Example value**: 1048576
- **Since**: 6.9.4.0

mailbox.default.session.mail.<protocol>.partialfetch

- **Module**: cmas-nimh
- **Description**: Sets java mail property for partialfetch i.e. controls whether the protocol partial-fetch capability should be used.
  For IMAP systems: in CM versions 6.10.7.0 and up, the value of `mailbox.default.session.mail.imap.partialfetch` is set to “false” during the initial setup of a ConSol CM system. During an update of an existing ConSol CM system, the value of the property is left unchanged, if the property is already present. In case the property is not yet present, it is added with the default value.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**:
- **Since**: 6.9.4.0

mailbox.default.task.delete.read.messages

- **Module**: cmas-nimh
- **Description**: This defines whether messages should be removed from the mailbox after processing. For IMAP protocol messages are marked as SEEN by default. For POP3 protocol, when flag is set to true the message is removed, otherwise remains on server and will result in infinite reads. Default: false.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: false
- **Since**: 6.4.0

mailbox.default.task.enabled

- **Module**: cmas-nimh
- **Description**: With this property service thread related to given poller can be disabled. Default: true
- **Type**: boolean

- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: false
- **Since**: 6.4.0

mailbox.default.task.interval.seconds

- **Module**: cmas-nimh
- **Description**: Default interval for polling mailboxes. Default: 60 seconds
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.4.0

mailbox.default.task.max.message.size

- **Module**: cmas-nimh
- **Description**: Maximum size of email messages (i.e., email plus attachment). Emails exceeding the size limit will not be automatically processed by NIMH but will be stored in the database (table `cmas_nimh_archived_mail`) and will therefore appear in the email backups in the Admin Tool (see section Email Backups). From there they can be resent, downloaded to the file system, or deleted. For those operations the message size is not relevant. Default is set to 10MB: 10485760
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 10485760
- **Since**: 6.4.0

mailbox.default.task.max.messages.per.run

- **Module**: cmas-nimh
- **Description**: Number of messages fetched at once from mailbox. Must be correlated with transaction timeout. Default set to: 20
- **Type**: integer
- **Restart required**: no
- **System**: no

- **Optional**: yes
- **Example value**: 60
- **Since**: 6.4.0

mailbox.default.task.timeout.seconds

- **Module**: cmas-nimh
- **Description**: After this time (of inactivity) the service thread is considered as damaged and automatically restarted. Default: 120 seconds
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.4.0

mailbox.default.task.transaction.timeout.seconds

- **Module**: cmas-nimh
- **Description**: Default transaction timeout for email fetching transactions. Should be correlated with number of messages fetched at once. Default: 60 seconds
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.4.0

mailbox.polling.threads.mail.log.enabled

- **Module**: cmas-nimh
- **Description**: Enables email logging which is especially crucial in cluster environment (used as semaphore there)
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true (default)
- **Since**: 6.9.4.1

mailbox.polling.threads.number

- **Module**: cmas-nimh
- **Description**: Number of threads for accessing mailboxes. Default: 1
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 1
- **Since**: 6.4.0

queue.polling.threads.number

- **Module**: cmas-nimh
- **Description**: Number of threads started for emails' queue polling. Default: 1
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 1
- **Since**: 6.4.0

queue.polling.threads.shutdown.timeout.seconds

- **Module**: cmas-nimh
- **Description**: Waiting time after the shutdown signal. When the timeout is reached, the thread will be terminated. Default: 60
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.4.0

queue.polling.threads.watchdog.interval.seconds

- **Module**: cmas-nimh
- **Description**: Watchdog thread interval. Default: 30
- **Type**: integer
- **Restart required**: no

- **System**: no
- **Optional**: yes
- **Example value**: 30
- **Since**: 6.4.0

queue.task.error.pause.seconds

- **Module**: cmas-nimh
- **Description**: Maximum number of seconds, the queue poller waits after infrastructure (e.g. database) error. Default 180 seconds
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 180
- **Since**: 6.4.0

queue.task.interval.seconds

- **Module**: cmas-nimh
- **Description**: Main emails' queue polling thread interval. Default: 15
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 15
- **Since**: 6.4.0

queue.task.max.retries

- **Module**: cmas-nimh
- **Description**: Maximum number of email processing retries after an exception. When reached, the email is moved to the email archive. This email can be rescheduled again using NIMH API (or the Admin Tool).
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 10
- **Since**: 6.4.0

queue.task.timeout.seconds

- **Module**: cmas-nimh
- **Description**: After this time (of inactivity) the service thread is considered as damaged and automatically restarted. Default: 600 seconds
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 600
- **Since**: 6.4.0

queue.task.transaction.timeout.seconds

- **Module**: cmas-nimh
- **Description**: Transaction timeout for email processing in the pipe. Default: 60
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.4.0

H.2.3.9 cmas-nimh-extension (module)

mail.attachments.validation.info.sender

- **Module**: cmas-nimh-extension
- **Description**: Sets From header of attachments type *error notification mail*
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: admin@mail.com
- **Since**: 6.7.5

mail.attachments.validation.info.subject

- **Module**: cmas-nimh-extension
- **Description**: Sets subject of attachments type error notification mail.
- **Type**: string

- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: Email was not processed because its attachments were rejected!
- **Since**: 6.7.5

mail.db.archive

- **Module**: cmas-nimh-extension
- **Description**: If property is set to “true”, incoming emails are archived in the database.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: false (default)
- **Since**: 6.8.5.5

mail.error.from.address

- **Module**: cmas-nimh-extension
- **Description**: From address for error emails from NIMH
- **Type**: email
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: myuser@consol.de
- **Since**: 6.4.0

mail.error.to.address

- **Module**: cmas-nimh-extension
- **Description**: To address for error emails from NIMH. As a default the email address of the administrator which you have entered during system setup is used.
- **Type**: email
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: myuser@consol.de
- **Since**: 6.4.0

mail.on.error

- **Module**: cmas-nimh-extension
- **Description**: If set to “true ”an error email is sent to the above configured address in case the email message could not be processed. Default: true
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: false
- **Since**: 6.4.0

mail.process.error

- **Module**: cmas-nimh-extension
- **Description**: To address for error emails from Mule.
- **Type**: email
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: myuser@consol.de
- **Since**: 6.4.0

mail.ticketname.pattern

- **Module**: cmas-nimh-extension
- **Description**: Regular expression pattern used to identify the ticket name in the subject of incoming mails.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: .*?Ticket\s+\((\S+)\).*
- **Since**: 6.4.0

H.2.3.10 cmas-restapi-core (module)

comment.authors.disabled

- **Module**: cmas-restapi-core
- **Description**: Disables the display of the content's author via REST API. The default value is “false”.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: false
- **Since**: 6.11.0

diff.tracking.disabled

- **Module**: cmas-restapi-core
- **Description**: Fallback property for disabling diff tracking for CM/Track, which is history-based so it can be heavy.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: false (default)
- **Since**: 6.10.5.6

security.fields.customer.exposure.check.enabled

- **Module**: cmas-restapi-core
- **Description**: Enables customer exposure annotation checks for ticket fields.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true (default)
- **Since**: 6.10.5.4

security.restrict.unit.access.to.own.data

- **Module**: cmas-restapi-core
- **Description**: If set to “true”, an additional check is performed when a user logs in as a customer using the REST API, e.g. CM/Track. When requesting customer data, only the company of the user or other contacts of the user’s company are returned. If set to “false”, no additional security check is performed and the former security rules apply.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true (default)
- **Since**: 6.9.2.14

H.2.3.11 cmas-setup-hibernate (module)

cmas.dropSchemaBeforeSetup

- **Module**: cmas-setup-hibernate
- **Description**: Flag if schema is to be (was) dropped during setup
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true
- **Since**: 6.0

connection.release.mode

- **Module**: cmas-setup-hibernate
- **Description**: Describes the JEE connection handling strategy for transactions. If set to “AFTER_TRANSACTION”, the connection will be cached during the transaction and released at the end. If set to “AFTER_STATEMENT”, the connection will be released to the pool after each statement execution. Please do not change the default here unless advised by ConSol.
- **Type**: string
- **Restart required**: yes
- **System**: no
- **Optional**: yes
- **Example value**: AFTER_STATEMENT (default for JEE environment)
- **Since**: 6.0

hibernate.dialect

- **Module**: cmas-setup-hibernate
- **Description**: The dialect used by hibernate. Usually set during initial set-up (depending on the database system).
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: org.hibernate.dialect.MySQL5InnoDBDialect
- **Since**: 6.0

update.6.11.0.0.sleep

- **Module**: cmas-setup-hibernate
- **Description**: Helper property for the update preparation scripts introduced in context of CM database refactoring in version 6.11. This is an optional setting allowing a delay (in milliseconds) after each loop iteration of the preparation scripts. Setting the delay should lower the database load, for example during working hours. This property may be removed after the update preparation tasks finish.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.0, for use in 6.10.5.x

update.6.11.0.0.timezone

- **Module**: cmas-setup-hibernate
- **Description**: Helper property for the ticket history migration (the new way of counting history groups). Since 6.11.0.0 the groups are constant (2h time span), but before 6.11.0.0 groups were not constant and depended on the customer's time zone. Migration scripts use an old algorithm to calculate groups and therefore need information about the time zone. The property should be set to the timezone which is most commonly used by the customers. If the property is not set, the default server time zone is used (`TimeZone.getDefault()`). The property should be set before updating to 6.11.0.0 and will be removed automatically after migration. The list of accepted timezones can be found for example here: http://joda-time.sourceforge.net/timezones.html.
- **Type**: string
- **Restart required**: no
- **System**: no

- **Optional**: yes
- **Example value**: Europe/Berlin
- **Since**: 6.11.0.0, for use before updating to this version

H.2.3.12 cmas-setup-manager (module)

initialized

- **Module**: cmas-setup-manager
- **Description**: Flag if CMAS is initialized. If this value is missing or not “true”, set-up will be performed. Starting with ConSol CM version 6.11, this property is only available in expert.mode.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true
- **Since**: 6.0

> ⚠ Be careful with using this property!!! When you set the value to “false”, the ConSol CM server will perform the system set-up at the next start, i.e. all data of the existing system is lost, including system properties!!!

H.2.3.13 cmas-setup-scene (module)

scene

- **Module**: cmas-setup-scene
- **Description**: Scene file which was imported during set-up (can be empty).
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: vfszip:/P:/dist/target/jboss/server/cmas/deploy/cm-dist-6.5.1-SNAPSHOT.ear/APP-INF/lib/dist-scene-6.5.1-SNAPSHOT.jar/META-INF/cmas/scenes/helpdesk-sales_scene.jar/
- **Since**: 6.0

H.2.3.14 cmas-workflow-engine (module)

jobExecutor.adminMail

- **Module**: cmas-workflow-engine
- **Description**: Email address which will get notified about job execution problems (when retry counter is exceeded).
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: admin@consol.de
- **Since**: 6.8.0

jobExecutor.idleInterval.seconds

- **Module**: cmas-workflow-engine
- **Description**: Determines how often job executor thread will look for new jobs to execute.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 45 (default up to CM version 6.10.5.2. Default CM versions 6.10.5.3 and up is 5)
- **Since**: 6.8.0

jobExecutor.jobMaxRetries

- **Module**: cmas-workflow-engine
- **Description**: Controls the number of retry attempts the job executor will do before declaring a job as failed.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 5 (default)
- **Since**: 6.8.0

jobExecutor.jobMaxRetriesReachedSubject

- **Module**: cmas-workflow-engine
- **Description**: The subject used in the notification mail admins receive about failed job executors.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: Job maximum retries reached. Job was removed!!! (default)
- **Since**: 6.8.0

jobExecutor.lockingLimit

- **Module**: cmas-workflow-engine
- **Description**: Number of jobs locked at once (marked for execution) by job executor thread.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 5 (default since CM version 6.10.5.3)
- **Since**: 6.8.0

jobExecutor.lockTimeout.seconds

- **Module**: cmas-workflow-engine
- **Description**: How long the job can be locked (marked for execution) by job executor.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 360 (default)
- **Since**: 6.8.0

jobExecutor.mailFrom

- **Module**: cmas-workflow-engine
- **Description**: Email which will be set as From header during admin notifications.
- **Type**: string
- **Restart required**: no

- **System**: yes
- **Optional**: yes
- **Example value**: jobexecutor@consol.de
- **Since**: 6.8.0

jobExecutor.maxInactivityInterval.minutes

- **Module**: cmas-workflow-engine
- **Description**: Number of minutes of allowed job executor inactivity (e.g. when it is blocked by long timer execution). After this time executors threads are restarted.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes. Default value is set to 30 minutes
- **Example value**: 15 (default)
- **Since**: 6.9.2.0

jobExecutor.threads

- **Module**: cmas-workflow-engine
- **Description**: Number of job execution threads.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 1 (default)
- **Since**: 6.8.0

jobExecutor.timerRetryInterval.seconds

- **Module**: cmas-workflow-engine
- **Description**: Determines how long job executor thread will wait after job execution error.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 10 (default up to CM version 6.10.5.2. Default CM versions 6.10.5.3 and up is 30)
- **Since**: 6.8.0

jobExecutor.txTimeout.seconds

- **Module**: cmas-workflow-engine
- **Description**: Transaction timeout used for job execution.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 60 (default)
- **Since**: 6.8.0

H.2.3.15 cmas-workflow-jbpm (module)

fetchLock.interval

- **Module**: cmas-workflow-jbpm
- **Description**:
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 5000
- **Removed in**: 6.8.0

jobExecutor.idleInterval

- **Module**: cmas-workflow-jbpm
- **Description**:
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 45000
- **Removed in**: 6.8.0
- **Replaced by**: jobExecutor.idleInterval.seconds

jobExecutor.jobExecuteRetryNumber

- **Module**: cmas-workflow-jbpm
- **Description**:
- **Type**: integer

- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 5
- **Removed in**: 6.8.0
- **Replaced by**: jobExecutor.jobMaxRetries

jobExecutor.timerRetryInterval

- **Module**: cmas-workflow-jbpm
- **Description**:
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 10000
- **Removed in**: 6.8.0
- **Replaced by**: jobExecutor.timerRetryInterval.seconds

mail.sender.address

- **Module**: cmas-workflow-jbpm
- **Description**: From address for emails from the workflow engine.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: myuser@consol.de
- **Removed in**: 6.8.0
- **Replaced by**: jobExecutor.mailFrom

outdated.lock.age

- **Module**: cmas-workflow-jbpm
- **Description**:
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no

- **Example value**: 60000
- **Removed in**: 6.8.0
- **Replaced by**: cmas-workflow-engine, jobExecutor.lockTimeout.seconds

refreshTimeInCaseOfConcurrentRememberMeRequests

- **Module**: cmas-workflow-jbpm
- **Description**: It sets the refresh time (in seconds) after which page will be reloaded in case of concurrent remember me requests. This feature prevents one user from occupying many licenses. Please increase that time if sessions are still occupying.
- **Type**: integer
- **Restart required**: yes
- **System**: yes
- **Optional**: yes
- **Example value**: 5
- **Since**: 6.8.2

H.2.3.16 cmweb-server-http-headers (module)

X-Frame-Options

- **Module**: cmweb-server-http-headers
- **Description**: Example property to illustrate the configuration of HTTP headers. In this case the delivered HTTP header contains the field *X-Frame-Options* with the value “SAMEORIGIN”.

  Each property in the module `cmweb-server-http-headers` represents one header field. The property name/key identifies the response header field and the value of the property is the field value sent in this header.

  > ⚠ Please be aware that additional HTTP response headers must be correctly defined with the exact spelling as officially specified! Please note also that the correct interpretation and application of these headers is fully in the realm and responsibility of the client browser which requested the page!

- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: SAMEORIGIN
- **Since**: 6.10.8

H.2.3.17 cmweb-server-adapter (module)

attachment.upload.timeout

- **Module**: cmweb-server-adapter
- **Description**: Defines the transaction timeout in minutes for adding attachments to a ticket, a resource or a customer. Counts the time for the upload of all attachments of one transaction. When the timeout occurs, all files which have been temporarily stored on the server are deleted. No file is uploaded.
- **Type**: Integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 3
- **Since**: 6.10.5.3

automatic.booking.enabled

- **Module**: cmweb-server-adapter
- **Description**: If enabled, time spend on creating comment/email will be measured and automatic time booking will be added.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: true
- **Since**: 6.9.4.2

checkUserOnlineIntervalInSeconds

- **Module**: cmweb-server-adapter
- **Description**: The interval in seconds to check which users are online (default 180sec = 3min).
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 180
- **Since**: 6.0
- **Removed in**: 6.5 / 6.11.0.1

cmoffice.enabled

- **Module**: cmweb-server-adapter
- **Description**: Flag if CM/Doc (former CM/Office) is enabled.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.4.0

cmoffice.oo.path.NUMBER

- **Module**: cmweb-server-adapter
- **Description**: Possible location of the OpenOffice installation. The properties are numbered starting with 0.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: c:\Program Files (x86)\LibreOffice 3.6\program
- **Since**: 6.10.1.0

cmoffice.strict.versioning.enabled

- **Module**: cmweb-server-adapter
- **Description**: Controls if the SAVE operation in Microsoft Word / OpenOffice documents creates a new attachment (“true”) or overwrites the existing attachment (“false”). This concerns the behavior within one session using the text editing program. If the program is stopped, the over-write mechanism will not work anymore.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true
- **Since**: 6.10.5.4

commentRequiredForTicketCreation

- **Module**: cmweb-server-adapter
- **Description**: Flag if comment is a required field for ticket creation.

- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true (default)
- **Since**: 6.2.0

csrf.domain.white.list

- **Module**: cmweb-server-adapter
- **Description**: The list of domains (separated with '|') which are allowed and will not be checked by CSRF (cross-site request forgery) filter
- **Type**: String
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: example.com | consol.de
- **Since**: 6.10.7.0

csrf.request.filter.enabled

- **Module**: cmweb-server-adapter
- **Description**: It allows to disable CSRF (Cross-site request forgery) request filter
- **Type**: Boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true
- **Since**: 6.10.7.0

customizationVersion

- **Module**: cmweb-server-adapter
- **Description**: UID representing the latest web customization version. Used only internally, please do not change the value.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no

- **Example value**: cd58453e-f3cc-4538-8030-d15e8796a4a7
- **Since**: 6.5.0

data.optimization

- **Module**: cmweb-server-adapter
- **Description**: Defines optimization to be applied on response data. So far, the following values are supported (for setting more than one value, separate values by '|'): MINIFICATION and COMPRESSION. MINIFICATION minifies HTML data by e.g. stripping whitespaces and comments. COMPRESSION applies gzip compression to HTTP response. (Note: If you are running in cluster mode and want to test different configurations in parallel, you can set different values for each cluster node by specifying property `data.optimization.nodeId` to override default property.)
- **Type**: string
- **Restart required**: COMPRESSION can be switched on/off without restart, MINIFICATION requires restart.
- **System**: yes
- **Optional**: yes
- **Example value**: MINIFICATION|COMPRESSION

defaultAttachmentEntryClassName

- **Module**: cmweb-server-adapter
- **Description**: The default content entry class used to classify an attachment if no other class was set explicitly.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: DefaultTextElement
- **Since**: 6.9.2.0

defaultContentEntryClassName

- **Module**: cmweb-server-adapter
- **Description**: Default text class for new ACIMs.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no

- **Example value**: default_class
- **Since**: 6.3.0

defaultNumberOfCustomFieldsColumns

- **Module**: cmweb-server-adapter
- **Description**: Default number of columns for ticket fields.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 3
- **Since**: 6.2.0

diffTrackingEnabled

- **Module**: cmweb-server-adapter
- **Description**: Removed in ConSol CM version 6.11.
  Defines if parallel editing of a ticket by different engineers should be possible. Default is “true”.
  “false”: Previous way of handling changes when editing a ticket. If the ticket has been changed in the meantime, the current engineer will not be able to submit his changes without being forced to reload the page before submitting.
  “true”: New changes handling mode. If the ticket has been changed, this will not block the submission of other changes anymore. If the part of the ticket that was changed was exactly the part that is changed by the submitting engineer, then an information message will be displayed, but the ticket change will be persisted/stored anyway.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true (default)
- **Since**: 6.10.1
- **Removed in**: 6.11.0

diffTrackingEnabledForUnitAndResource

- **Module**: cmweb-server-adapter
- **Description**: Enables the prevention of concurrent modifications on units / resources.
- **Type**: boolean
- **Restart required**: no
- **System**: no

- **Optional**: yes
- **Example value**: 3
- **Since**: 6.11.0.0

favoritesSizeLimit

- **Module**: cmweb-server-adapter
- **Description**: Maximum number of items in Favorites list.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 10
- **Since**: 6.0

forward.mails.to.representatives

- **Module**: cmweb-server-adapter
- **Description**: Determines if emails which are manually sent from the Web Client are also sent to representing engineers. The default value of the property is “false”, meaning that this kind of emails are not forwarded to the representing engineer. Set the property to “true” if you want to restore the previous behavior, i.e., all emails which are sent to the represented engineer are automatically forwarded to the representing engineer. Please take into account that this might not be desired if the same person is an engineer and a customer in the CM system.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: false (default)
- **Since**: 6.11.1.7

This property only configures the handling of manually sent emails. The handling of automatically sent emails depends on the used Java method.

globalSearchResultSizeLimit

- **Module**: cmweb-server-adapter
- **Description**: Maximum number of items in Quick Search result.
- **Type**: integer
- **Restart required**: no
- **System**: yes

- **Optional**: no
- **Example value**: 10
- **Since**: 6.0

helpFilePath

- **Module**: cmweb-server-adapter
- **Description**: URL for online help. If not empty, Help button is displayed in Web Client.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: http://www.consol.de
- **Since**: 6.2.1

hideTicketSubject

- **Module**: cmweb-server-adapter
- **Description**: If set to “true”, ticket subject is hidden.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.2.1

mail.from

- **Module**: cmweb-server-adapter
- **Description**: Use this address if set instead of engineer email address during email conversation.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.1.2

mail.reply.to

- **Module**: cmweb-server-adapter
- **Description**: When set, Web Client will display Reply-To field on email send, prefilled with this value.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.0.1

> ⚠ Please read the detailed information about ConSol CM Reply-To addresses in section Scripts of Type Email.

mailTemplateAboveQuotedText

- **Module**: cmweb-server-adapter
- **Description**: Indicates behavior of email template in the Ticket Email Editor when another email is quoted, i.e. forwarded or replied to. Often used to place the signature correctly.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.2.4

maxSizePerPagemapInMegaBytes

- **Module**: cmweb-server-adapter
- **Description**: The parameter defines the size (in MB) if the file which is created by the Wicket framework per user session. i.e. for each engineer which is currently logged in. The file is used to save pages during the running session. When the defined size limit has been reached and new entries are added, the oldest entries are removed. In the Web Client, due to this behavior, an engineer who works with an "old" page will be redirected to the *Overview*/Start page (usually the dashboard page) when the "old" page is removed from the file.
  So in case engineers who work with a great number of open tabs in ConSol CM and complain about being redirected to the *Overview* page, it might be useful to increase this parameter. In large systems, you could use e.g. a value of 45 or 50. Since this is the size of the file which is saved on disk, the maximum value depends on the available disk space, however, a value which is too large is not recommended either.
- **Type**: integer
- **Restart required**: yes

- **System**: yes
- **Optional**: no
- **Example value**: 15
- **Since**: 6.3.5

pagemapLockDurationInSeconds

- **Module**: cmweb-server-adapter
- **Description**: Number of seconds to pass before pagemap is considered to be locked for too long.
- **Type**: integer
- **Restart required**: yes
- **System**: yes
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.7.3

postActivityExecutionScriptName

- **Module**: cmweb-server-adapter
- **Description**: Defines the name for the script which should be executed after every workflow activity, see section PostActivityExecutionScript. If no script should be executed, leave the value empty.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: postActivityExecutionHandler
- **Since**: 6.2.0

queuesExcludedFromGS

- **Module**: cmweb-server-adapter
- **Description**: Comma-separated list of queue names which are excluded from Quick Search.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.0

rememberMeLifetimeInMinutes

- **Module**: cmweb-server-adapter
- **Description**: Lifetime for *remember me* in minutes.
- **Type**: integer
- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: 1440
- **Since**: 6.0

request.scope.transaction

- **Module**: cmweb-server-adapter
- **Description**: It allows to disable request scope transaction. By default one transaction is used per request. Setting this property to “false” there will cause one transaction per service method invocation.
- **Type**: boolean
- **Restart required**: yes
- **System**: yes
- **Optional**: yes
- **Example value**: true
- **Since**: 6.8.1

searchPageSize

- **Module**: cmweb-server-adapter
- **Description**: Default page size for search results.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 20
- **Since**: 6.0

searchPageSizeOptions

- **Module**: cmweb-server-adapter
- **Description**: Options for page size for search results.
- **Type**: string
- **Restart required**: no

- **System**: yes
- **Optional**: no
- **Example value**: 10|20|30|40|50|75|100
- **Since**: 6.0

serverPoolingInterval

- **Module**: cmweb-server-adapter
- **Description**: Defines the time in seconds for pooling server to invalidate caches on the web layer.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 5
- **Since**: 6.1.0

supportEmail

- **Module**: cmweb-server-adapter
- **Description**:
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.0
- **Removed in**: 6.11.0.1

themeOverlay

- **Module**: cmweb-server-adapter
- **Description**: Name of used theme overlay
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: consolINT
- **Since**: 6.0

ticketListRefreshIntervalInSeconds

- **Module**: cmweb-server-adapter
- **Description**: Refresh interval for ticket list (in seconds).
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 180
- **Since**: 6.0

ticketListSizeLimit

- **Module**: cmweb-server-adapter
- **Description**: Maximum number of tickets in ticket list.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 100
- **Since**: 6.0

tx.read.only.mode.enabled

- **Module**: cmweb-server-adapter
- **Description**: Enables read-only transactions for faster page loading. This transactional behavior was introduced in 6.11.0, and this property acts as a safety guard to restore the old behaviors. Do not change this value unless facing tx problems and advised by ConSol.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true (default)
- **Since**: 6.11

unitIndexSearchResultSizeLimit

- **Module**: cmweb-server-adapter
- **Description**: Maximum number of units in unit search result (e.g. when searching for contact).
- **Type**: integer
- **Restart required**: no

- **System**: yes
- **Optional**: no
- **Example value**: 5
- **Since**: 6.0

urlLogoutPath

- **Module**: cmweb-server-adapter
- **Description**: URL which is used when user logs out. (If no value is set, logout leads to login-mask.)
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: http://intranet.consol.de
- **Since**: 6.3.1

voCacheEnabled

- **Module**: cmweb-server-adapter
- **Description**: This property enables additional caching for the Web Client, voCaching, of complete objects, thus improving performance.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: true
- **Since**: 6.11.1.0

Notes:

- Since ConSol CM version 6.11.1.1, the default value is “true” for non-clustered environments. The value is set to “true” automatically during the setup or update of ConSol CM 6.11.1.1.
- When voCaching is enabled and lazy loading is used for folding ticket history entries, once the engineer unfolded an entry, he cannot fold it again by reloading the page or opening the ticket from the workspace.
- When using the dynamic mode for displaying engineer and customer names in the ticket history (as configured in the system properties cmas-core-server, engineer.description.mode and cmas-core-server, unit.description.mode), the new version of the engineer and/or customer name is only displayed after the ticket has been changed.
- This system property is ignored for clustered environments (environments with cmas-core-shared, cluster.mode set to “true”). In clustered environments, voCaching is always disabled to avoid problems that changes made to objects on one node are not visible on the other nodes.

webSessionTimeoutInMinutes

- **Module**: cmweb-server-adapter
- **Description**: Session timeout in minutes.
- **Type**: integer
- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: 180
- **Removed in**: 6.7.1
- **Replaced by**: cmas-core-server, server.session.timeout

wicketAjaxRequestHeaderFilterEnabled

- **Module**: cmweb-server-adapter
- **Description**: This enables filter for Wicket AJAX requests, coming from stale pages with Wicket 1.4 scripting (CM pre-6.8.0), after update to CM6 post-6.8.0.
- **Type**: boolean
- **Restart required**: yes
- **System**: yes
- **Optional**: yes
- **Example value**: false
- **Since**: 6.8.1

H.3 Administrator and Notification Email Addresses

This chapter discusses the following:

H.3.1 Introduction

In ConSol CM, several administrator email addresses (or notification email addresses respectively) can be configured. Here, an overview of all those addresses is provided.

H.3.2 Default Configuration

When you set-up a ConSol CM system, you have to enter one global admin email address.

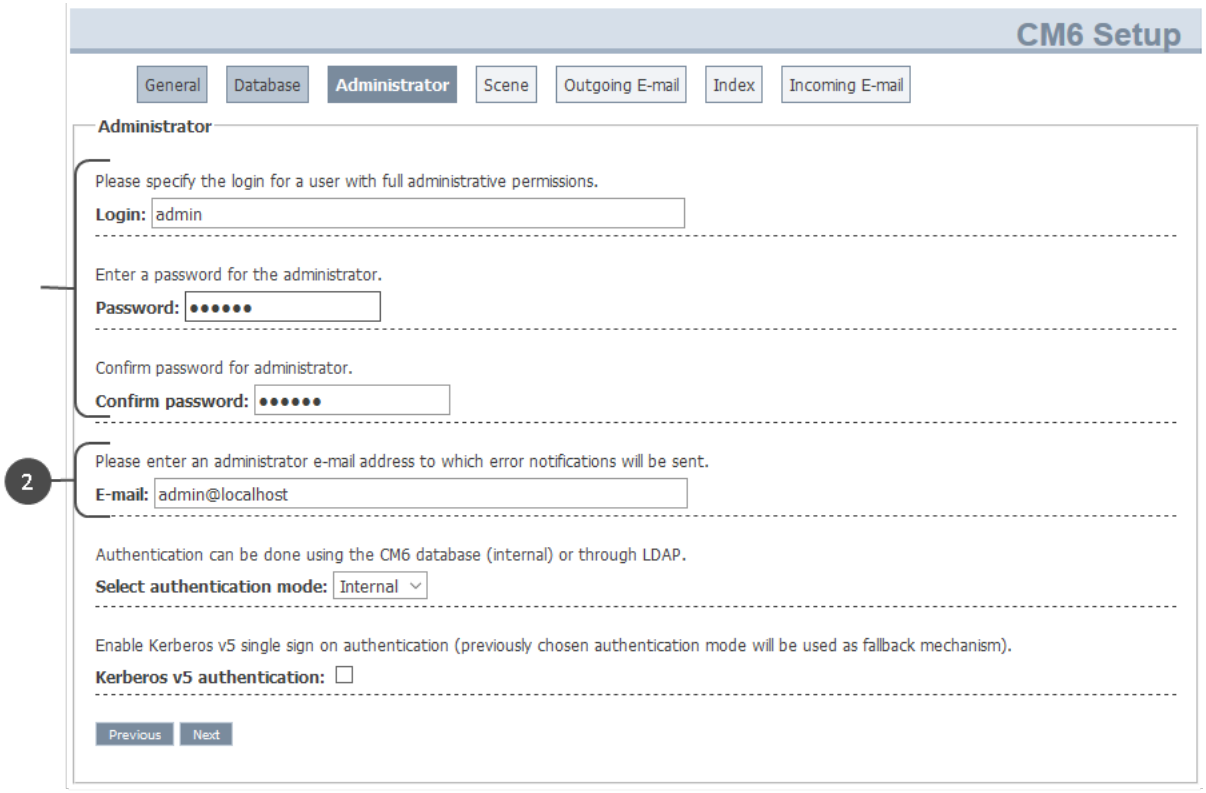


Figure 716: *Admin email address configuration during system set-up*

- Login for the CM admin user (1)
- Global email address of the CM administrator (2)
  This global email address (system property cmas-core-security, admin.email) will be used for all notifications and will be entered automatically for all subsystem-specific email addresses. This

means that this admin email address will initially be set automatically for all system properties which contain admin or notification email addresses. The properties can then be changed using the Admin Tool GUI to configure the specific subsystem, e.g., you can configure a specific notification address for DWH operations in the *DWH Configuration* section of the Admin Tool.

By configuring different admin/notification email addresses for different subsystems, you can spread responsibilities according to responsibilities and roles within your company. For some notifications, even the From address, text, and the subject can be configured.

H.3.2.1 Special Cases Where the Admin Email Address Is Used

Queue-Specific Emails When an Engineer Has Been (De-)Assigned

The emails which are sent when ticket assignment templates are set for a queue (see section *Queue Administration*), use the `admin.email` address as

- `From` address
- `ReplyTo` address

The queue-specific email scripts are not used for those automatic emails, thus it is not possible to change the `From` or `ReplyTo` address for (un-)assignment emails using this script!

Please note: If the CM system property cmas-core-server, mail.notification.sender is set, the value of this property will be used as `From` address and as `ReplyTo` address for (un-)assignment emails.

H.3.3 Subsystem-Specific Notification Email Addresses

H.3.3.1 DWH (Data Warehouse) - Specific Notification Email Addresses and Email Configurations

System Properties

- **Error**
  - cmas-dwh-server, notification.error.to
    An email will be sent to this address when a DWH operation has failed. If the property is not set, no email will be sent.
  - cmas-dwh-server, notification.error.from
    An email will be sent with this From address when a DWH operation has failed.
  - cmas-dwh-server, notification.error.subject
    Subject for error emails from the DWH
  - cmas-dwh-server, notification.error.description
    Text for error emails from the DWH.
- **Successful**
  - cmas-dwh-server, notification.finished_successfully.to
    An email will be sent to this address when a DWH transfer has been completed successfully, e.g., when the transfer has been completed without errors. If the property is not set, no email will be sent.
  - cmas-dwh-server, notification.finished_successfully.from
    From address for emails from the DWH when a transfer finishes successfully.
  - cmas-dwh-server, notification.finished_successfully.subject
    Subject for emails from the DWH when a transfer finishes successfully.
  - cmas-dwh-server, notification.finished_successfully.description
    Text for emails from the DWH when a transfer finishes successfully.
- **Unsuccessful**
  - cmas-dwh-server, notification.finished_unsuccessfully.to
    An email will be sent to this address when a DWH transfer has been completed, but not successfully, e.g., when the transfer has been completed with errors. If the property is not set, no email will be sent.
  - cmas-dwh-server, notification.finished_unsuccessfully.from
    From address for emails from the DWH when a transfer finishes unsuccessfully.
  - cmas-dwh-server, notification.finished_unsuccessfully.subject
    Subject for emails from the DWH when a transfer finishes unsuccessfully.
  - cmas-dwh-server, notification.finished_unsuccessfully.description
    Text for emails from the DWH when a transfer finishes unsuccessfully.
  - cmas-dwh-server, cmas-dwh-server, notification.error.description
    The text for error emails from the DWH

For an overview of all system properties which can be set for DWH notifications, please refer to section CMRF & DWH Configuration of the List of System Properties by Area chapter. All properties which are relevant in this context start with *notification*.

Graphical Configuration

You reach this screen by opening the navigation item *Administration*, navigation group *Data Warehouse*, and clicking the tool icon.

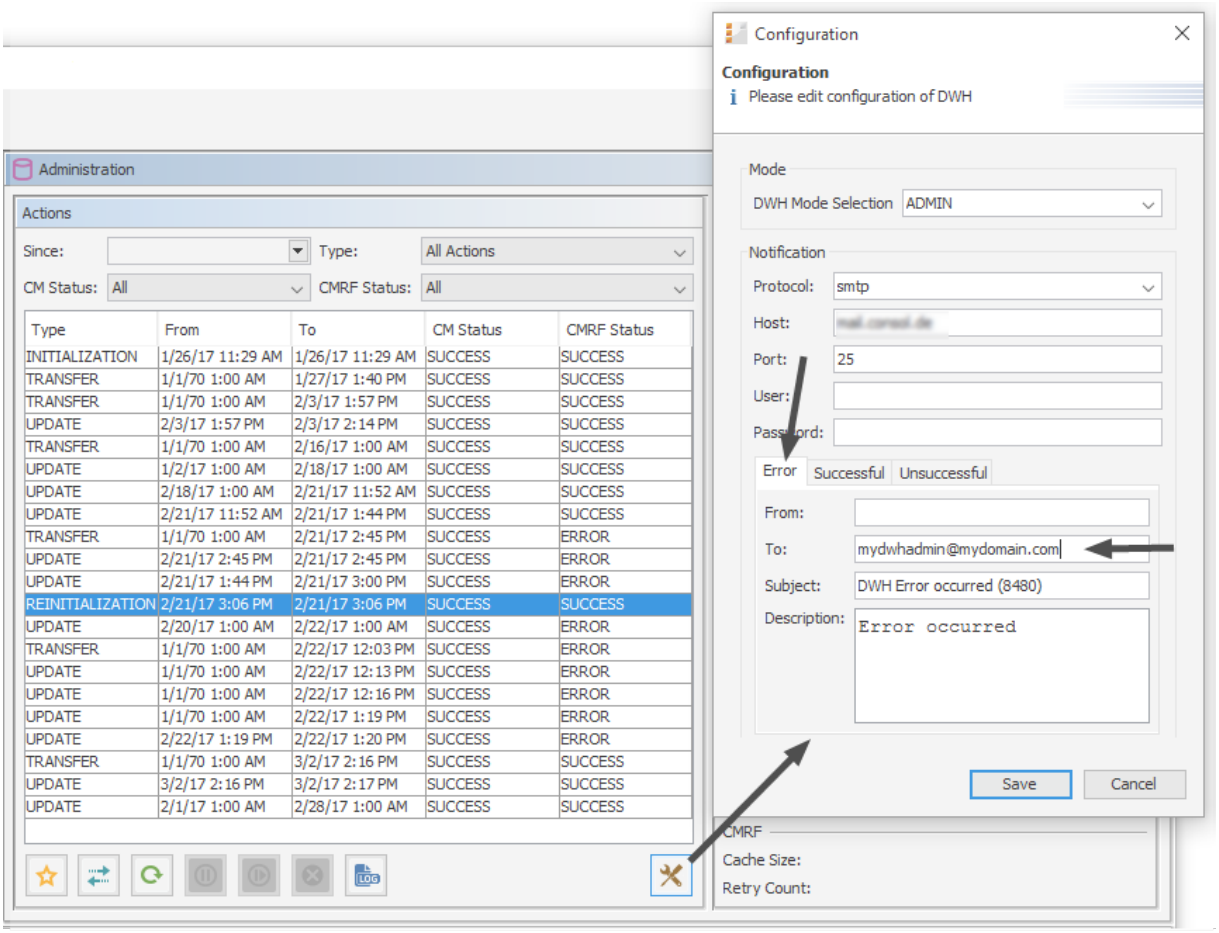


Figure 717: *ConSol CM Admin Tool - Data Warehouse, Administration: Notification address for DWH errors*

The email addresses are checked when you click *Save*. If an email address is not valid, a message is displayed. No mails will be sent to this address.

H.3.3.2 Email-Specific Notification Email Addresses

- cmas-nimh-extension, mail.error.to.address
  An error email is sent to the address in case an email message could not be processed. Starting with CM version 6.10.5.1, an email to this address is also sent when a mail timeout occurs.
  If the value for this property is not set, no email will be sent and an exception will be written

into the log file.

- cmas-nimh-extension, mail.attachments.validation.info.sender
  Sets the From header of attachments type *error notification email*.

Please note that the sending is controlled by the system property cmas-nimh-extension, mail.on.error. Only if this property is set to “true”, an error email is sent to the above configured address in case an email message could not be processed.

Graphical Configuration

The value which is entered here (*Error email address*) in the graphical user interface is set as general email notification address.

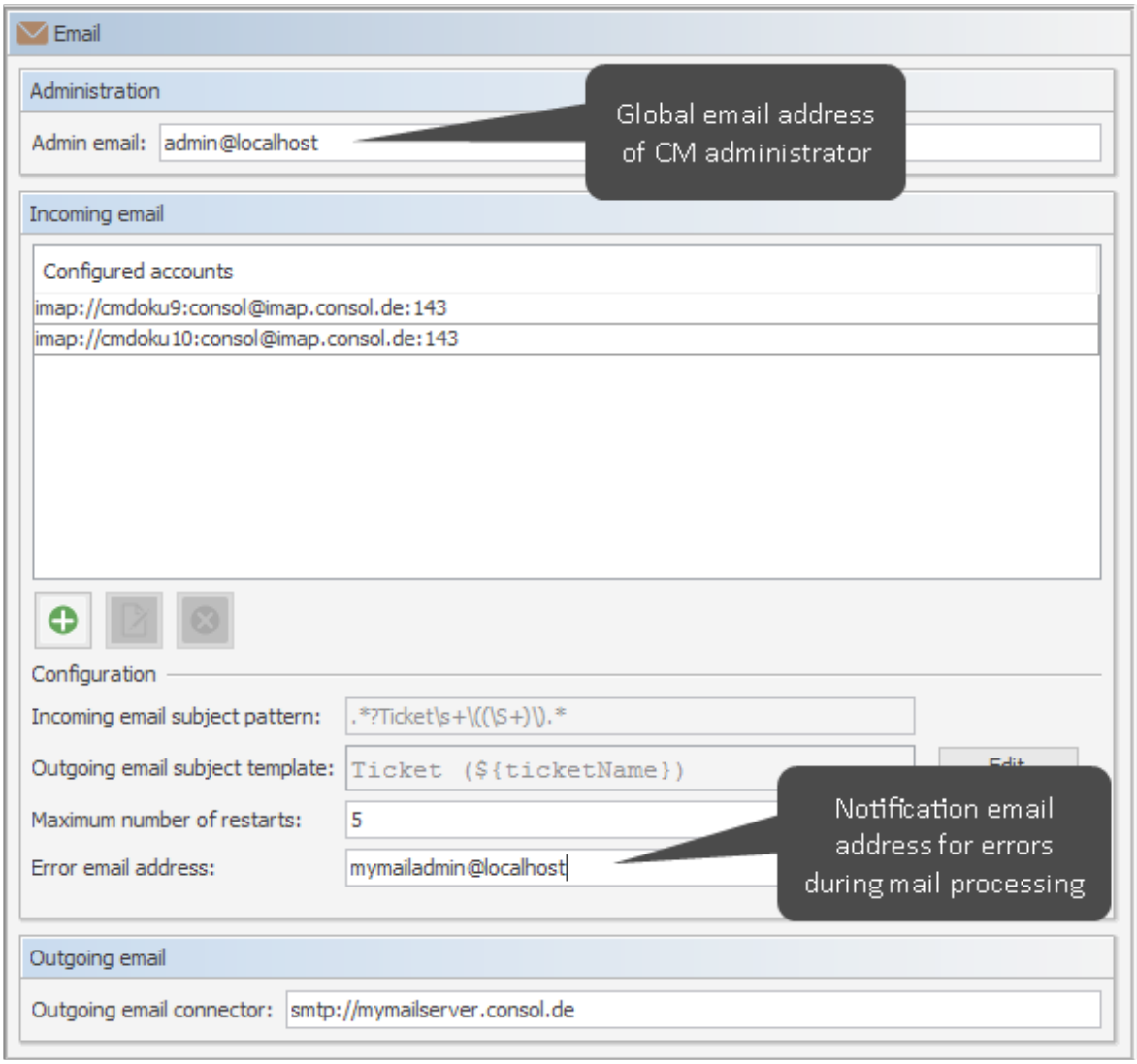


Figure 718: *ConSol CM Admin Tool - Email, Email: Configuration of global admin email address and of notification email address for email processing problems*

Do not mix up the two email addresses which can be configured on the tab of the navigation item *Email*:

- **Admin email**
  is the global administrator email (which has been set during system set-up).
- **Error email address**
  is the address of an email administrator or another person who should receive the error messages if the processing of emails (incoming or outgoing) does not work correctly.

H.3.3.3 Workflow Engine - Specific Notification Email Addresses

- cmas-workflow-engine, jobExecutor.adminMail
  Email address which will get notified about job execution problems (when retry counter is exceeded). If the property is not set, no email will be sent.
- cmas-workflow-engine, jobExecutor.mailFrom
  Email address which will be set as `From` header during admin notifications.

H.4 Default Java Imports

In order to have all required classes and methods available in ConSol CM scripts, you might have to import Java/Groovy classes and packages. This is relevant for the Admin Tool and for the Process Designer. However, starting with ConSol CM version 6.11.1, you will only have to import classes and packages which are not used in most standard cases. All packages which contain classes which are used rather frequently are imported implicitly by the Admin Tool and by the Process Designer. In this way, the code validation during writing the scripts as well as the execution of the code during runtime is based on this implicit imports which makes coding CM scripts rather comfortable.

The following classes are implicitly imported in the Admin Tool:

- com.consol.cmas.common.model.*
- com.consol.cmas.common.model.calendar.*
- com.consol.cmas.common.model.configuration.*
- com.consol.cmas.common.model.content.*
- com.consol.cmas.common.model.content.unit.*
- com.consol.cmas.common.model.customer.*
- com.consol.cmas.common.model.customfield.*
- com.consol.cmas.common.model.customfield.cfel.*
- com.consol.cmas.common.model.customfield.enums.*
- com.consol.cmas.common.model.customfield.meta.*
- com.consol.cmas.common.model.event.*
- com.consol.cmas.common.model.event.configuration.*
- com.consol.cmas.common.model.event.content.*
- com.consol.cmas.common.model.event.content.support.*
- com.consol.cmas.common.model.event.customfield.*
- com.consol.cmas.common.model.event.delete.*
- com.consol.cmas.common.model.event.engineer.*
- com.consol.cmas.common.model.event.engineer.support.*
- com.consol.cmas.common.model.event.localization.*
- com.consol.cmas.common.model.event.localization.support.*
- com.consol.cmas.common.model.event.ticket.*
- com.consol.cmas.common.model.event.ticket.support.*
- com.consol.cmas.common.model.event.unit.*
- com.consol.cmas.common.model.event.unit.support.*
- com.consol.cmas.common.model.event.util.*
- com.consol.cmas.common.model.event.workflow.*

- com.consol.cmas.common.model.history.*
- com.consol.cmas.common.model.history.custom.*
- com.consol.cmas.common.model.history.ticket.*
- com.consol.cmas.common.model.inventory.*
- com.consol.cmas.common.model.inventory.history.*
- com.consol.cmas.common.model.inventory.meta.*
- com.consol.cmas.common.model.localization.*
- com.consol.cmas.common.model.mail.*
- com.consol.cmas.common.model.resource.*
- com.consol.cmas.common.model.resource.history.*
- com.consol.cmas.common.model.resource.meta.*
- com.consol.cmas.common.model.scripting.OperationMessage
- com.consol.cmas.common.model.scripting.OperationResponse
- com.consol.cmas.common.model.scripting.OperationType
- com.consol.cmas.common.model.search.*
- com.consol.cmas.common.model.task.*
- com.consol.cmas.common.model.ticket.*
- com.consol.cmas.common.model.ticket.user.*
- com.consol.cmas.common.model.ticket.user.function.*
- com.consol.cmas.common.model.ticket.view.*
- com.consol.cmas.common.security.authentication.UserType
- com.consol.cmas.workflow.common.model.*
- java.io.*
- java.util.*
- javax.activation.DataSource
- com.consol.cmas.common.model.form.*

The following classes are implicitly imported in the Process Designer:

- com.consol.cmas.common.model.*
- com.consol.cmas.common.model.calendar.*
- com.consol.cmas.common.model.configuration.*
- com.consol.cmas.common.model.content.*
- com.consol.cmas.common.model.content.unit.*
- com.consol.cmas.common.model.customer.*
- com.consol.cmas.common.model.customfield.*

- com.consol.cmas.common.model.customfield.cfel.*
- com.consol.cmas.common.model.customfield.enums.*
- com.consol.cmas.common.model.customfield.meta.*
- com.consol.cmas.common.model.event.*
- com.consol.cmas.common.model.event.configuration.*
- com.consol.cmas.common.model.event.content.*
- com.consol.cmas.common.model.event.content.support.*
- com.consol.cmas.common.model.event.customfield.*
- com.consol.cmas.common.model.event.delete.*
- com.consol.cmas.common.model.event.engineer.*
- com.consol.cmas.common.model.event.engineer.support.*
- com.consol.cmas.common.model.event.localization.*
- com.consol.cmas.common.model.event.localization.support.*
- com.consol.cmas.common.model.event.ticket.*
- com.consol.cmas.common.model.event.ticket.support.*
- com.consol.cmas.common.model.event.unit.*
- com.consol.cmas.common.model.event.unit.support.*
- com.consol.cmas.common.model.event.util.*
- com.consol.cmas.common.model.event.workflow.*
- com.consol.cmas.common.model.history.*
- com.consol.cmas.common.model.history.custom.*
- com.consol.cmas.common.model.history.ticket.*
- com.consol.cmas.common.model.inventory.*
- com.consol.cmas.common.model.inventory.history.*
- com.consol.cmas.common.model.inventory.meta.*
- com.consol.cmas.common.model.localization.*
- com.consol.cmas.common.model.mail.*
- com.consol.cmas.common.model.resource.*
- com.consol.cmas.common.model.resource.history.*
- com.consol.cmas.common.model.resource.meta.*
- com.consol.cmas.common.model.scripting.OperationMessage
- com.consol.cmas.common.model.scripting.OperationResponse
- com.consol.cmas.common.model.scripting.OperationType
- com.consol.cmas.common.model.search.*

- com.consol.cmas.common.model.task.*
- com.consol.cmas.common.model.ticket.*
- com.consol.cmas.common.model.ticket.user.*
- com.consol.cmas.common.model.ticket.user.function.*
- com.consol.cmas.common.model.ticket.view.*
- com.consol.cmas.common.security.authentication.UserType
- com.consol.cmas.workflow.common.model.*
- java.io.*
- java.util.*
- javax.activation.DataSource
- com.consol.cmas.common.model.permission.*
- com.consol.cmas.common.model.scripting.*
- com.consol.cmas.common.model.util.*

H.5 List of Code Examples

In this section, you can find a list of the code examples from this manual.

H.6 Trademarks

- The Apache Commons Codec ™ library is a trademark of the Apache Software Foundation. See Apache Commons Codec web page.
- Apache OpenOffice™ – Apache and the Apache feather logos are trademarks of The Apache Software Foundation. OpenOffice.org and the seagull logo are registered trademarks of The Apache Software Foundation. See Apache OpenOffice Trademarks web page.
- Google Maps™ – Google Maps is a trademark of Google Inc. See Google trademark web page for details.
- HAProxy – HAProxy is copyright of Willy Tarreau. See HAProxy website.
- Microsoft® – Microsoft and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. See Microsoft trademark web page.
- Microsoft® Active Directory® – Microsoft and Microsoft Active Directory are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. See Microsoft trademark web page.
- Microsoft® Exchange Server – Microsoft and Microsoft Exchange Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. See Microsoft trademark web page.
- Microsoft® Office – Microsoft and Microsoft Office are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. See Microsoft trademark web page.
- Windows® operating system – Microsoft and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. See Microsoft trademark web page.
- Microsoft® SQL Server® – Microsoft and Microsoft SQL Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. See Microsoft trademark web page.
- Microsoft® Word® – Microsoft and Microsoft Word are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. See Microsoft trademark web page.
- NGiNX – NGiNX is copyright of Igor Sysoev and Nginx, Inc. See NGiNX license page.
- OpenStreetMap - OpenStreetMap® is open data, licensed under the Open Data Commons Open Database License (ODbL) by the OpenStreetMap Foundation (OSMF). See OpenStreetMap Copyright and License page.
- Oracle® – Oracle is a registered trademark of Oracle Corporation and/or its affiliates. See Oracle trademarks web page.
- Oracle® WebLogic – Oracle is a registered trademark of Oracle Corporation and/or its affiliates. See Oracle trademarks web page.
- Pentaho® – Pentaho and the Pentaho logo are registered trademarks of Pentaho Inc. See Pentaho trademark web page.

- Vis.js – Vis.js is copyright of Almende B.V. See Vis.js license page.
- Wicket™ - Apache Wicket and Wicket, Apache, the Apache feather logo, and the Apache Wicket project logo are trademarks of The Apache Software Foundation. See, for example, the hint at the bottom of the Wicket home page

Glossary

A

ACF

ACF is the abbreviation of Activity Control Form. ACFs can be used in workflow activities to force the engineer to fill out certain fields before proceeding.

ACIM

Activity item - entry in the history section of a ticket (e.g., comment, email, attachment, time booking entry).

AD

Microsoft Active Directory - an LDAP-based directory service for Microsoft Windows domain networks.

additional customer

Additional customers are customers (companies or contacts) who are interested in the ticket. They are optional and usually have a role indicating the reason why they were added.

additional engineer

Additional engineers are engineers who have a specific purpose, which depends on your business process. Usually, they have to carry out certain tasks within the process.

Admin Tool

ConSol CM component, graphical application to configure and manage a ConSol CM system. Uses Java Web Start.

AJP

Apache JServ Protocol, see, for example https://en.wikipedia.org/wiki/Apache_JServ_Protocol

B

BI

Business Intelligence - methods, technologies, and architectures to transform data into useful information for business purposes.

C

CFEL

Custom Field Expression Language - Java classes and methods of the ConSol CM API to access data in ticket fields, customer fields and resource fields.

CIDR

Classless Inter-Domain Routing, notation for IP address ranges

CM/Doc

A standard module of ConSol CM which enables the engineer via ConSol CM Web Client to work with Microsoft Word or OpenOffice documents prefilled with ConSol CM ticket or customer parameters.

CM/Phone

The ConSol CM module which provides CTI for CM.

CM/Resource Pool

CM/Resource Pool is an optional add-on which allows to store different kinds of objects as resources in ConSol CM.

CM/Track

CM/Track is the portal of ConSol CM. Customers can access their tickets through CM/Track.

CMDB

ConSol CM database - the working database of the CM system.

CMRF

ConSol CM Reporting Framework - a JEE application which synchronizes data between the ConSol CM database and the DWH.

company

The company is the upper hierarchical level of a two-level customer model. A company can have several contacts.

contact

The contact is the lower hierarchical level of a two-level customer model. A contact can only belong to one company.

CRM

Customer Relationship Management. Approach to manage a company's customers, e.g., to collect data from different sources and integrate the data to generate information which allows, e.g., to optimize the services for the customers.

CTI

Computer Telephony Integration - a denomination for any technology that facilitates interaction between a telephone and a computer.

customer

The customer represents the external side of a ticket. It designates the person or object that gave the reason for creating a ticket. A customer can either be a company or a contact.

customer action

Part of the Action Framework. An action which is performed for a customer object, i.e., a contact or company object.

customer data model

The customer data model is the definition of the customers. It determines the available data fields and possible relations.

customer field

A field where data for customers (contacts or companies) can be stored. Similar to ticket fields for ticket data. Previously called Data Object Group Field.

customer field group

A group of fields where data for customers (contacts or companies) can be stored. Similar to ticket field group for ticket data. Previously called Data Object Group.

customer group

The customer group determines which customer data model is used for its customers and which actions are available.

customer object

A customer (a contact or a company). Formerly called Data Object. The term Unit is used in the programming context.

D

Dashboard

A type of report which integrates data from different sources providing an overall perspective of a certain topic. Often times graphical representation is used.

DWH

Data Warehouse - A database used for reporting and data analysis. In a standard ConSol CM distribution, a DWH is included and only has to be installed and configured.

E

engineer

Engineers are the users who work on the tickets in the Web Client

ERP system

Enterprise Resource Planning - often used for this type of enterprise management software.

ESB

Enterprise Service Bus - a software architecture used for communication between mutually interacting software applications in a service-oriented architecture (SOA).

ETL

Extract Transform Load - extracts data from one source (a database or other source), transforms it, and loads it into a database, e.g., a data warehouse.

F

FlexCDM

Flexible Customer Data Model - the customer data model introduced in ConSol CM in version 6.9. For each customer group, a specific customer data model can be defined.

G

GUI

Graphical User Interface

H

history

The history contains all changes which were carried out for the ticket, customer, or resource.

HMAC

Hash-based Message Authentication Code, message authentication function using hashes

I

IMAP

Internet Message Access Protocol - Internet standard protocol to access email on a remote email server. Can be used as plain IMAP or as secure IMAP (IMAPs). In the latter case, proper certificates are required.

J

Java EE

Java Enterprise Edition

JMS

Java Message Service - Java EE component used to send messages between JMS clients.

JRE

Java Runtime Environment. Provides a Java Virtual Machine for Clients.

K

Kerberos

A network authentication protocol based on (Kerberos) tickets which requires a special infrastructure.

KPI

Key Performance Indicator - parameter used for performance measurement for companies, projects, etc.

L

LDAP

LDAP is the abbreviation of Lightweight Directory Access Protocol. It is a protocol used to manage login information for several applications.

LDAPS

LDAP over SSL

M

mailbox

Destination to which email messages are delivered. Mailboxes are managed on an email server. ConSol CM can access one or more mailboxes to retrieve emails.

main customer

The main customer is the customer who gave the reason for creating the ticket. The main customer is mandatory for a ticket.

Mule

An open source Java-based Enterprise Service Bus (ESB).

N

NIMH

New Incoming Mail Handler - module for retrieving incoming emails, new in version 6.9.4.

P

PCDS

Page Customization Definition Section

Pentaho

PentahoTM is a business intelligence (BI) suite which is available in open source and as enterprise editions.

permission

Permissions determine which tickets an engineer can see in the Web Client and which actions he is allowed to perform. Permissions are always granted via roles, i.e., they are not assigned to a single user but to a group of users sharing a common role. Usually these users belong to the same team and/or have similar functions in the company.

POP

Post Office Protocol - Internet standard protocol to retrieve emails from a remote server via TCP/IP. Can be used as plain POP or as secure POP (POPs). In the latter case, proper certificates are required.

portal

CM/Track - provides customer access to ConSol CM.

Process Designer

ConSol CM component used to design, develop, and deploy workflows.

Q

queue

The queue contains thematically related tickets which should be handled in the same way and follow the same business process (workflow). Permissions and other parameters are also defined based on queues.

R

RDBMS

Relational Database Management System - e.g. Oracle ®, MS SQL Server ®, MySQL.

relation

Relations are connections between different data objects in ConSol CM. This can be a relation between two objects of the same type, e.g., between tickets, customers, and resources, or a relation between objects of different types, e.g., between a ticket and a resource or a customer and a resource.

resource

Resources are objects managed in CM/Resource Pool.

resource action

Part of the Action Framework. An action performed for a resource object.

resource field

A field where resource data can be stored.

resource field group

A group of fields where data for resources can be stored. Similar to ticket field group for ticket data.

resource type

The resource type is the definition of the resources. It determines the available data fields and possible relations and actions.

REST

Representational State Transfer - conventions for transferring data over HTTP connections.

role

Roles are assigned to engineers. They define the engineers' access permissions and views.

S

script

Program written for a specific run-time environment that can interpret and automate the execution of tasks. In ConSol CM, scripts are stored in the Admin Tool and are stored as scripts for activities in workflows.

search action

Part of the Action Framework. An action performed for the result set of a search.

SMTP

Simple Message Transfer Protocol - standard protocol for sending emails.

T

TAPI

Telephony Application Programming Interface - a Microsoft Windows API which provides computer/telephony integration and enables PCs running Microsoft Windows to use telephone services.

TEF

Task Execution Framework - a ConSol CM module which can execute tasks asynchronously. A new feature as of version 6.9.4.

template

Templates contain predefined and preformatted text. They can be used for comments, emails, and documents.

ticket

The ticket is the request of the customer which the engineer works on. It is the object which runs through the business process defined by the workflow.

ticket field

A field where ticket data can be stored. Previously called Custom Field

ticket field group

A group of ticket fields where ticket data can be stored. Previously called

Custom Field Group.

time booking

Time bookings allow the engineers to register the time they worked on a ticket or project.

U

Unit

Java class which represents a customer object. i.e. a contact is an object of class Unit and a company is also an object of class Unit.

V

view

Views limit the tickets which are shown in the ticket list in the ConSol CM Web Client to those tickets matching specific criteria (scopes from one or more workflows). Views are assigned to roles.

W

Web Client

The Web Client is the primary access to the system for the engineers.

Wicket

Apache Wicket is an open source, component oriented, serverside, Java web application framework. See https://wicket.apache.org/ for details information.

workflow

The workflow is the implementation of the business process managed in ConSol CM. It contains a series of steps which are carried out by the engineers.

Index

V

W