ConSol ☀ CM

Administrator Manual

ConSol*CM Administrator Manual (Version 6.10, up to 6.10.2)

Table of Contents

1 Introduction

- ConSol*CM
- ConSol*CM Documents
- This Book's Structure
- Layout Explanations
- Basic Principles of ConSol*CM
  - System Components from a User's Point of View
  - Basic Technical ConSol*CM Principles and Objects
    - The Ticket
    - The Workflow
    - The Queue
    - The Customer
    - The Engineer
    - Resources
    - The CM Action Framework
    - The Task Execution Framework
    - ConSol*CM Dogma
  - ConSol*CM from a System Administrator's Point of View

1.1 ConSol*CM

ConSol*CM is a **customer centric business process management software**. Using ConSol*CM you can control and steer business processes with a strong focus on human communication and interaction, e.g. user help desk, customer service processes, marketing and sales or ordering processes. Basically, every process that is in operation in a company can be modeled and brought to life with ConSol*CM6.

When you read this manual, your company is presumably using ConSol*CM6 as a process management tool and it is your job to administer the system. The book will help you get a quick overview of the most important components of ConSol*CM and will also provide a deeper and more detailed introduction to all aspects of the CM administration.

1.2 ConSol*CM Documents

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.
- **Process Designer Manual**

  A guideline for workflow developers about the graphical user interface of the Process Designer and how to program workflow scripts.
- **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.
- **Set-Up Manual**

  A technical description for CM set-up in different IT environments. For expert CM administrators.
- **User Manual**

  An introduction to the ConSol*CM Web Client for end users.
- **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.

For you as a CM administrator the book you are reading, the **Administrator Manual**, provides the required information to cover all fields of CM administrator work, see the following section for more information about the sections of the book. Please make sure that all system requirements have been taken into consideration.

1.3 This Book's Structure

First, some basic principles of the ConSol*CM6 application will be explained to provide the theoretical background you need to become a CM administrator.

The *Overview* section explains how to get access to the system.

The following four sections explain the features and functionalities of the main administration application, the ConSol*CM Admin Tool. You can decide which section(s) you need:

- In the **Access and Roles Section**, the basic principle and configuration concerning access permissions are explained. For example, you will learn how to define roles, how to assign roles to engineers (the users of the CM Web Client), and how to configure views (the to do lists in the system).
- In the **Ticket Data Model and GUI Design Section**, the set-up of the ticket data model and the placement of the data fields in the GUI are explained. For example, you will learn, how to define the data fields which are required in a certain process and how to build different types of lists.
- In the **Customer Data Model Section**, the set-up of the data models for different customer groups and the respective GUI design are explained. 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.
- In the **Resource Pool 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.
- In the **Global Configuration Section**, some general configurations are explained, For example you will learn a lot about Queue Management, a queue being one of the core components of ConSol*CM. Furthermore, the work with Business Calendars and projects is explained.
- Follows the **Expert Section**. This is the part of the book which is targeted at CM administrators who are responsible for the advanced CM system configuration. You might want to work together with you CM consultant in case you want to change system settings. This section covers topics like
    - preparing the CM system for reporting
    - ConSol*CM services
    - configuration of the search module
    - e-mail configuration
    - import and export of configurations
    - template management
    - the portal CM.Track
    - working with system properties
    - working with Admin Tool scripts

- In the **appendix**, you find lists of all important terms that are used in the book (glossary), of all annotations (important for the GUI design), and properties (important for the CM system management). Please see also the trademarks page.

1.4 Layout Explanations
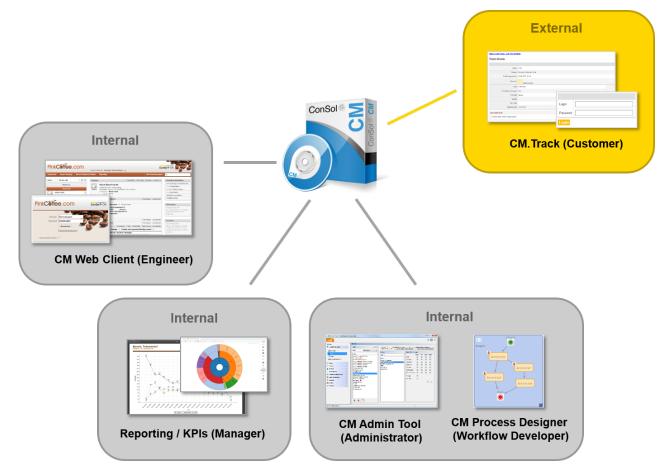
In order to emphasize and/or mark a section, icons are used.

> **Information:**
>
> This is an additional information.

> **Attention:**
>
> This is an important note. Be careful here!

> **Warning:**
>
> This is a warning!

> **Tip:**
>
> This is a recommendation from our everyday consulting life.

1.5 Basic Principles of ConSol*CM

1.5.1 System Components from a User's Point of View

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

Fig. 1: ConSol*CM System Components

- **Web Client**
  The primary access 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. If you have purchased this module, you can offer portal access to the tickets to your customers and/or can offer FAQs via the Web.
- **Admin Tool**
  For all system configuration tasks. As an administrator, this is the tool you will mostly work with. The tool is used to define the system set-up. All settings but workflows are configured using he Admin Tool. The access is restricted to admin users.
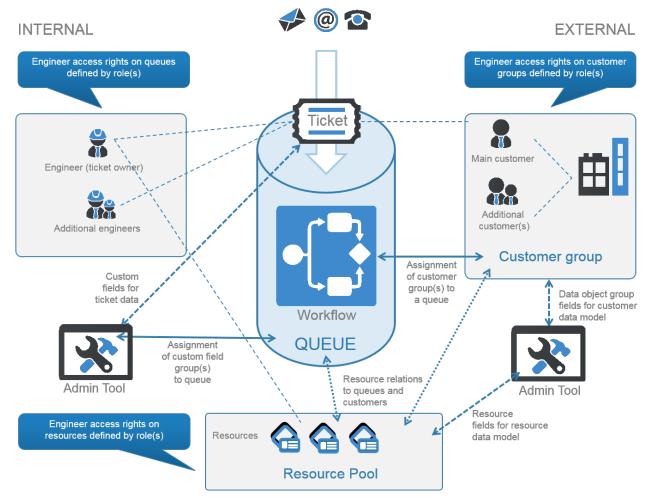
- **Process Designer**
  For the workflow design and implementation. As a workflow developer you will work with the Process Designer. In this tool, all workflows are designed graphically as well as in Groovy code.

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 a stand-alone application but can be easily integrated into your company's IT infrastructure, e.g. using Web Services and/or an Enterprise Service Bus (ESB).

For a detailed explanation of the system components seen from a more technical point of view, please refer to the system administrator's section (System Overview).

1.5.2 Basic Technical ConSol*CM Principles and Objects

Fig. 2: ConSol*CM Basic Principles

The Ticket

ConSol*CM can manage incidents, service cases, and/or other requests of internal and/or external customers. Every request is managed as a ticket that is created in the system, passes through the desired process, and is then (hopefully) solved and closed.

Closed tickets are not *lost*, but they represent a powerful archive and knowledge base. The user can search the tickets using the system search. Furthermore, the system can be configured to provide FAQ (frequently asked questions) functionalities.

Every ticket has an ID that is used internally and cannot be seen by the user.

Every ticket has a name (often called *ticket number*) that is displayed on the GUI to mark a ticket for the user.

The ticket icon in the GUI 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.

However, for every process, a different color-defining value can be used. While the user help desk uses the priority, the marketing and sales department can use the probability for the conclusion of a contract, and a multi-brand service team can use the brand.

Fig. 3: ConSol*CM Web Client - Ticket

The Workflow

Every process that should be managed using ConSol*CM is modeled as a workflow. During its life cycle a ticket runs through several steps, e.g it is opened as a new ticket, has to be processed by several experts and can then be closed. During the process there might also be a pause, usually called *resubmission*. All those steps are modeled as steps in a workflow.
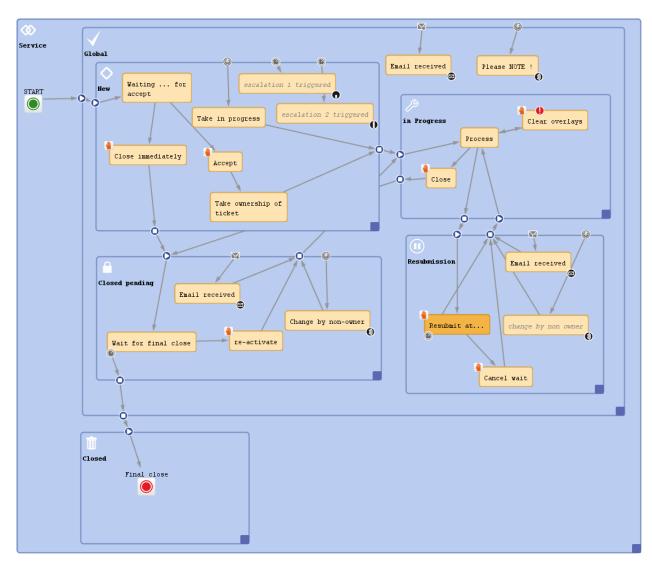
Fig. 4: ConSol*CM Process Designer - Workflow

A business process is modeled in ConSol*CM using the *Process Designer*, an application which is an integral element of a standard ConSol*CM installation. A process can be represented by one or more workflows.

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 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, e-mails that are sent automatically, 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 to process modeling using the ConSol*CM Process Designer.

The Queue

The queue is **the** core component of the ConSol*CM administration. It comprises tickets from the same domain and makes sure that all tickets of this domain are treated in the same way. Every queue has exactly one workflow which cannot be changed. All data fields required in a process are assigned to tickets of the process by queue management.

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.

Access permissions are also managed using the queue as basic entity.

The Customer

The customer is the person (**contact**) or **company** who has the question or service request. This person or company is the main customer for the ticket. This represents the *external* side of the CM system.

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, e-mail address, address) and complex, two-level models which contain contact data (e.g. name, phone number, e-mail 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.

The Engineer

The engineer is the *worker* who has a login to the Web Client and who has to manage the tasks defined in the tickets. This represents the *internal* side of the company or service team.

The engineer's access permissions are managed using roles, i.e. the roles are defined using the Admin Tool and engineers are assigned the role(s) they need.

Often, a ticket does not have an engineer when it just has been opened and it is assigned to an engineer in one of the first steps of the process. This engineer is responsible for the ticket, he/she will get the escalation e-mails and will see the ticket in his/her view *My Tickets* (*personal to-do list*).

There can be additional engineers in certain so-called *engineer roles* for a ticket, who also have tasks to do for the case. For example, a ticket has a regular engineer and an additional engineer in the engineer role *supervisor*. In this way, in the run of the process, the ticket can be automatically assigned to the supervisor.

Resources

A resource is an object in the **CM.Resource Pool**. This is a CM module which has to be licensed separately. You as an administrator can define the resource model, i.e. you can (and have to) define the object (resource) hierarchy and the data fields for each resource type. A resource can be each object which has to be managed in the system, e.g. an IT asset like a PC or monitor, or an SLA which has to be taken into consideration when working with a ticket, or a scientific sample, a product, or even a newsletter which should be sent to customers.

For resources, relations are defined. In this way, a resource can have relations to tickets, e.g. when a PC has a problem and the respective resource has to be liked to the incident ticket. A resource can also have relations to other resources to represent certain system configurations, product hierarchies or other types of relations. Or a resource can have relations to customers, .e.g. an SLA can be linked to all companies where you have to work according to this contract.

All aspects of the CM.Resource Pool are covered in the Resource Pool Section, starting with chapter CM Resource Pool - Introduction.

The CM Action Framework

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

- **Customer Actions** (Data Object 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 created.
- **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 e-mail 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.

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.

ConSol*CM Dogma

In ConSol*CM6, there is a main *dogma*:

> ⓘ   A ticket always has a **main customer**. This can be a contact or a company.
>
> A ticket can have no or one **engineer** who has to work on the ticket.

1.5.3 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 e-mail server to retrieve incoming e-mails and sends e-mails 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. If you need to know the supported application servers and relational database systems, please ask for the current *System Requirements* and/or the *Release Notes* of the latest ConSol*CM version.

2 Start of the Admin Tool

- Login
- Troubleshooting: When the Admin Tool Does Not Start
  - Correct Process
  - Process with Errors
- How to Use the Admin Tool

2.1 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.

Fig. 1: 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* section):

Fig. 2: ConSol*CM Admin Tool - Login Window

Enter your login data to get access to the Admin Tool functions. A first 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:

Fig. 3: ConSol*CM Admin Tool - Start Page

2.2 Troubleshooting: When the Admin Tool Does Not Start

2.2.1 Correct Process

To be able to find the problems you should know the correct process.

In case everything is set-up correctly, after clicking on the Admin Tool hyper link, the following steps are performed:

1. In a pop-up window, you are prompted for the decision if you would like to open the *jnlp* file - *Java (TM) Web Start Launcher* should be offered as default application for that - or if you want to store a local copy of the *jnlp* file.
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. In case the Java Web Console is activated, the console is opened and you can follow the download.
5. The Admin Tool GUI is displayed with the login window in the foreground.

Fig. 4: ConSol*CM Admin Tool - Start: Steps 1 and 2

Step3

Step 4 (only if Java console is activated)

Fig. 5: ConSol*CM Admin Tool - Start: Steps 3 and 4

Fig. 6: ConSol*CM Admin Tool - Start: Step 5

2.2.2 Process with Errors

In case the Admin Tool cannot be started, check the following settings:

1. **Problems with step 1:**
   a. Is Java in the correct version 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, you enter your login data. If a connection error occurs then, check the proxy settings.

2.3 How to Use the Admin Tool

You can reach the Admin Tool functions using the navigation tree on the left-hand side. Several navigation groups (see next figure) provide access to the navigation items of each group.
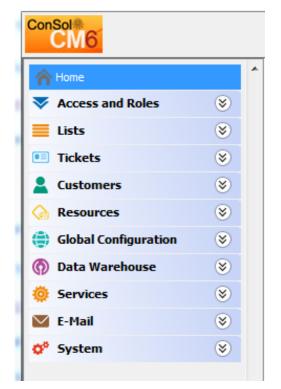
Fig. 7: ConSol*CM Admin Tool - Navigation Tree

The function pages are very similar in structure. The following picture shows the operational concept using the engineer administration as an example:
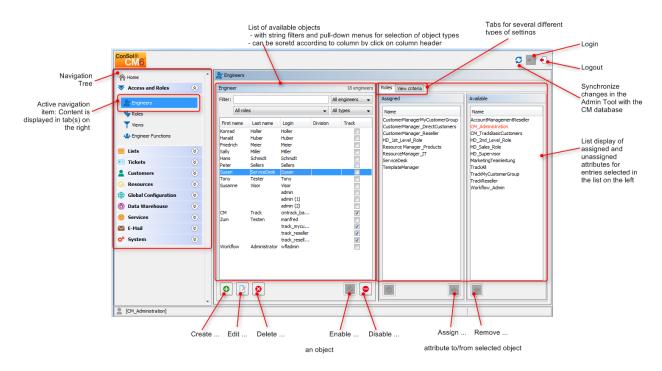
Fig. 8: ConSol*CM - Handling of the Admin Tool

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. You can move them from a list of *available attributes* to a list of *assigned attributes* either via double click or via click on the icon ⬅ (example: *available roles* and *assigned roles*). Attributes can also be assigned via check 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**
  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 will be adopted automatically, only the matching entries are displayed.
  - **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 icons ▲ or ▼ show the sort order.
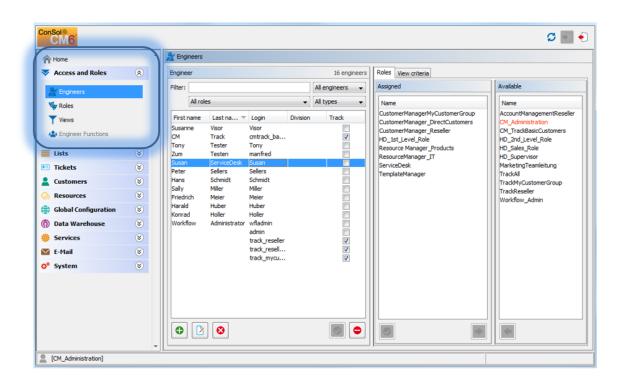
Usually all changes you perform in the Admin Tool are submitted immediately without the need to synchronize the data. However, if changes in another module have been performed and the Admin Tool has to use the new data, it is required to synchronize the data. You can achieve this by clicking on the *Synchronize* button in the icon bar.

One example for this is the deployment of a new workflow using the Process Designer. Before you can assign the new workflow 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. Then this new workflow can be used for further operations like assigning it to a new queue.

3 ConSol CM Administrator Manual 6.10.2 - -- Access and Roles Section--

This section provides some background knowledge about the general principle of 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

4 Engineer Administration

4.1 Introduction to Engineer Administration

An *engineer account* is the basic access object which allows the engineer or the administrator to access the Web Client, the Admin Tool, or the 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 multiple *role(s)* displayed in the tab *Roles*. If you have not created any roles yet, you will see only the administrator role (see tab *Roles*).

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*).

> ⓘ **Information:**
>
> We would recommend that you create at least one role and one view first before you create engineer accounts.

4.2 Engineer Administration Using the Admin Tool

To open the Engineer Administration, open the navigation group *Access and Roles* in the Admin Tool and click on the navigation item *Engineers*.
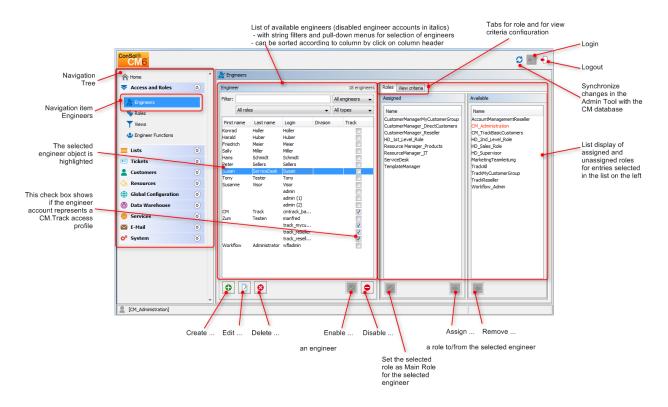
Fig. 1: ConSol*CM Admin Tool - Engineer Administration

4.2.1 Create or Edit an Engineer Account

To create an engineer account click on the icon ⊕ below the account list. Or click on 🗎 if you want to edit the settings of an existing account. The same pop-up window appears:

Fig. 2: ConSol*CM Admin Tool - Edit 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.

- **E-mail:**
  Mandatory. The engineer's e-mail address. Please use only international alphabetic and numeric characters, hyphens, underscores, periods, and the @ sign. The entry of multiple e-mail 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.

> ⚠ **Attention:**
>
> Several fields which contain engineer data (like *Company*, *Division*, or *Phone*) are optional fields. However, when you work with text templates which contain engineer data fields (see section The ConSol*CM Template Manager) the e-mails or comments will not be formed correctly when the data is missing. For example, the field *ticket-engineer, phone* cannot be filled-in in the template when 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.

> ⓘ **Information:**
>
> 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)!

- **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.
- **Password:**
  Mandatory. The engineer's password. The entry of a password is mandatory. Please use only international alphabetic and numeric characters, and punctuation marks, do **not** use any special characters as e.g. umlauts. The password entered will be shown as a string of asterisks. Please see section Configuration of the Password Policy for information about the optional password policy.

  > ℹ **Information:**
  >
  > This field will only appear if the engineer authenticates against the Web Client via the CM6 database, i.e. when LDAP or Kerberos authentication is used, the field is not visible.

- **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. The repetition of the password is mandatory. Please see section Configuration of the Password Policy for information about the optional password policy.

  > ℹ **Information:**
  >
  > This field will only appear if the engineer authenticates against the Web Client via the CM6 database, i.e. when LDAP or Kerberos authentication is used, the field is not visible.

- **Track user**
  This check box 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) will be shown in the Web Client when creating or modifying a customer. So, by ticking this check box, 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: System Access for CM.Track Users (Customers).

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

Configuration of 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*.

- **policy.password.pattern (String)**
  RegEx pattern for the password, example and default value: "^.3,$".

- **policy.password.age (Integer)**
  Maximum validation period, in number of days, example 183 (6 months), default value: 5500 (= 15 years, i.e. no password change enforced).
- **policy.rotation.ratio (Integer)**
  number defines the number of previous passwords which may not be identical, example and default value: 1.
- **policy.username.case.sensitive(Boolean)**
  Defines if 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.

Reset of an Engineer's Password

When an engineer has forgotten his/her password, he/she can request a new password by using the link *Forgot your password* on the initial login page. An e-mail 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 when a valid e-mail account is available for this engineer and when the respective value has been entered as e-mail for the enginner in the engineer data!

The e-mail which is sent to the engineer is based on the template *password-reset-template* which is stored in the *Templates* seection 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 when the standard mode is used. It is not possible when LDAP or Kerberos authentication is in operation. See section Authentication Methods for Engineers in the CM Web Client for an explanation of all possible authentication modes.

4.2.2 Delete an Engineer Account

To delete an engineer account, select the account in the list and click on ⊗ . 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. For all history entries in tickets and in customer pages which were performed by the deleted engineer, his or her name will still be displayed.

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

4.2.3 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 login anymore and other engineers cannot assign any tickets to their accounts.

To disable an engineer account, select the account and click on ⊝ . 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. Just click on ⊘ at the bottom of the page, if you want to enable the account again.

4.2.4 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 on ⬅ 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).

> ⓘ **Information:**
>
> When engineers log in to the system, they will have access permissions from all roles that have been assigned to them. So all permissions are added! There is no way of explicitly denying access to objects in ConSol*CM, you always grant access! The sum of all granted permissions defines the final access permissions for the engineer.

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 on ⊘ below the list. Afterwards the main role is marked with a red dot. Now the views of the main role will always appear at the top of the view list in the Web Client for this engineer account.

4.2.5 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).

> ⓘ **Information:**
>
> This tab will show view criteria only if you have created a view with dynamic criteria and assigned it to the engineer's role.

Select the engineer account on the left and then the desired criterion in the list of *available* view criteria on the right. Click on ⬅ to move it to the list of *assigned* view criteria. You will see the possible values below the criterion in the list. Tick the check boxes 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 will be immediately visible in the engineer's profile page.

Fig. 3: ConSol*CM Admin Tool - Define Engineer-Specific View Criteria

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 into 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.

> ⚠ **Attention:**
>
> 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, 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.

5 Role Administration

- Introduction to Role Administration
- Role Administration Using the Admin Tool
  - Create a Role
    - Tab Queue Permissions
    - Tab Global Permissions
    - Tab Customer Group Permissions
    - Tab Resource Types Permissions
    - Tab Views
  - Tab Engineer Functions
  - Delete a Role
  - Copy a Role
  - Edit a Role

5.1 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) the engineer obtains 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.

> **ⓘ Information:**
>
> 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, but you always grant access!
>
> The sum of all granted permissions defines the final permissions for the engineer.

Roles define:

- **Access permissions to 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.
- **Access permissions to customer data**
  Read, write, modify, and delete permissions for each distinct customer group.
- **Access permissions to resource data**
  Read, write, modify, create permissions, assigned on the basis of resource types.
- **Views**
  To do lists of tickets which will be 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.*
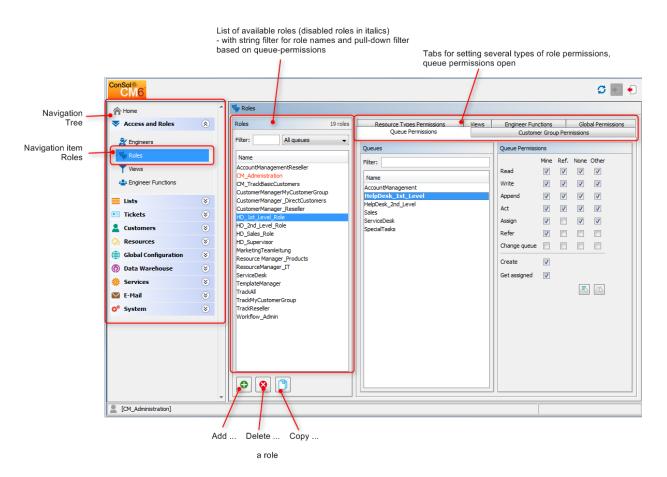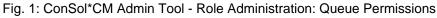
5.2 Role Administration Using the Admin Tool

To open the Role Administration, open the navigation group *Access and Roles* in the Admin Tool and click on the navigation item *Roles*. In the Role Administration, you see the list of all available roles on the left-hand side and the permissions which can be granted on the right-hand side. 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 with the access permissions of the role which has been selected in the list of roles. Only one role can be selected at a time. On the right-hand side, several tabs are available. During role management you switch between those tabs:

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

Fig. 1: ConSol*CM Admin Tool - Role Administration: Queue Permissions

> ⚠ **Attention:**
>
> All changes on the Role Management tabs are immediately effective resp. after clicking the *OK* button. You do not have to click on ⟳ (Refresh data) in the icon bar.
>
> In the Web Client, engineers have to log in again to use their new roles. Views become effective after clicking *F5* (page refresh).

5.2.1 Create a Role

Click on *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 or to act in the system.
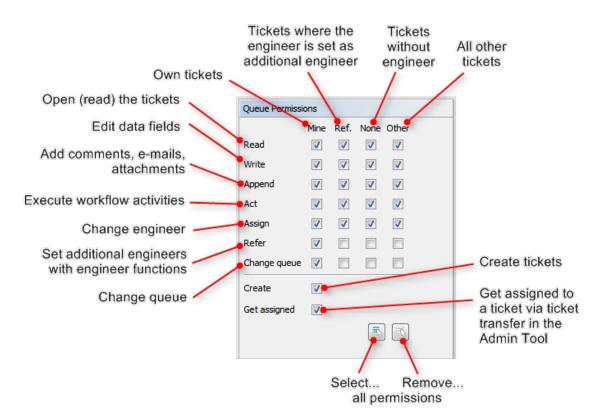
Fig. 2: ConSol*CM Admin Tool - Role Administration: Setting Queue Permissions

The following permissions can be set:

- **Read**
  Read tickets.
- **Write**
  Edit data fields (default fields, custom 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, e-mails, attachments, time booking entries), i.e. add content in the ticket protocol.

- **Act**

  Execute workflow activities, i.e. move the ticket forward in the workflow.
- **Assign**

  Assign tickets to another engineer and/or be assigned as engineer for tickets by other engineers.
- **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 (appropriate permissions for the *initial* and the *target queue* required).

> ⚠ **Attention:**
>
> 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 well.

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

- **Mine**

  Own tickets.
- **Ref**

  Tickets for 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 on the corresponding check box to assign one or more permissions to the desired ticket range.

Two general permissions can also be set:

- **Create**

  An engineer is allowed to create tickets in this queue.
- **Get assigned**

  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 on ▤ below the list. Clicking on ▧ removes all selections.

Tab Global Permissions

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

Fig. 3: ConSol*CM Admin Tool - Role Administration: Global Permissions

You can specify the following:

- **Global Permissions**
    - **Administrate**
      Provides administrator access to the system, this applies to the Admin Tool, the Process Designer, and admin access to the Web Client.
- **Workflow Permissions**
  Provides permissions concerning workflow design and management. These are
    - **Read**
    - **Write** (modify and store)
    - **Deploy** (install and put in operation).
- **Template Permissions**
    - **Write Template** provides the permission
        - to use the Template Manager which is used to create and edit e-mail and commentary templates, see section The ConSol*CM Template Manager for details.
        - to use the Doc Template Manager which is required to define templates for CM.Doc. Only available if CM.Doc is active in the CM system.

- **Representation Permissions**
  - **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**

In case a representation rule has been configured by an engineer (see *ConSol*CM User Manual* for details), all e-mails which are sent by CM are sent to the original recipient ⚠ and ⚠ his/her current representative. 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).

- **Track User Permissions**
  - **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 user group(s) to one or more roles.

Fig. 4: ConSol* Admin Tool - Assigning Permissions for Customer Groups to a Role

> ### ℹ Bests Practice
>
> A concept which has proven very useful in various customer environment 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 modified compared to previous ConSol*CM versions. New rights have been added which concern a new section of the customer page. The contact page as well as the company page in the Web Client have a new section, the *Details* section.
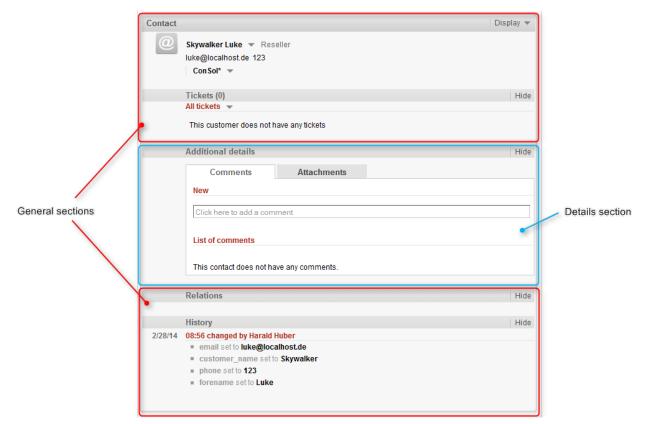
Fig. 5: ConSol*CM Web Client - Details Section of a Contact Page

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**
  - **Read**
    Read the customer data.
  - **Write**
    Write/modify the customer data.
  - **Delete**
    Delete a customer data set. Transfer all tickets associated with a customer of this customer group to another customer.
  - **Act**
    Execute actions for this customer (see section Action Framework - Data Object 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.

- **Details section**
    - **Details read**

        Read customer data in the *Details* section.
    - **Details write**

        Write/modify customer data in the *Details* section.
    - **Details delete**

        Delete customer data in the *Details* 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.
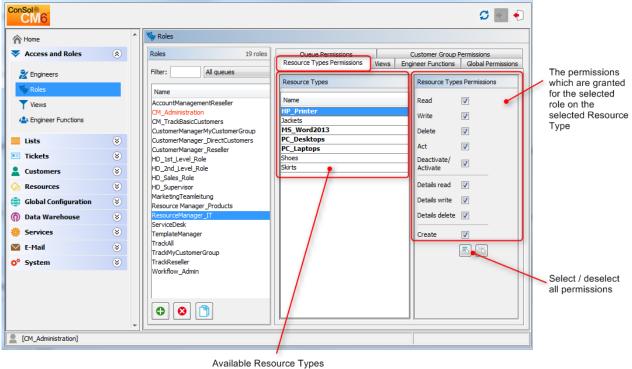
> ⚠️ **Attention:**
>
> Please keep in mind that an engineer must have at least *read* permissions for a customer group to open and/or create tickets for customers in this group!

Tab Resource Types Permissions

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

Fig. 6: ConSol*CM Admin Tool - Resource Types Permissions

The following permissions can be granted:

- **Read**

  Load and display resources of the selected type in the Web Client.
- **Write**

  Change custom field data of this type of resources.
- **Delete**

  Delete resource of the respective type from CM6.
- **Act**

  Execute resource actions defined for this type of resources.
- **Deactivate/Activaze**

  (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.

> ⓘ **Information:**
>
> We recommend to assign at least one view to a role. Otherwise an engineer with this role will see no tickets in the ticket list of the Web Client.
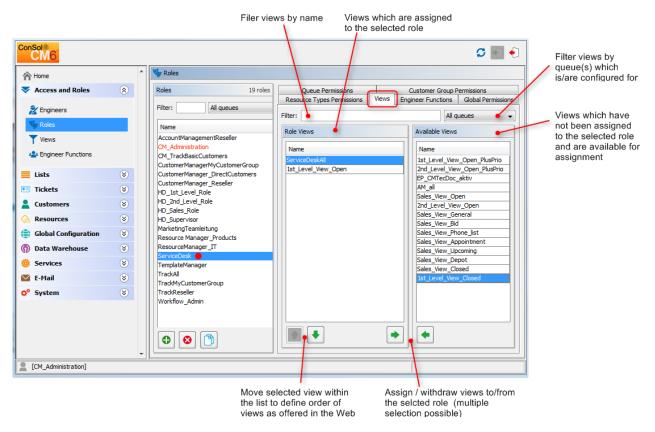
Fig. 7: ConSol*CM Admin Tool - Role Administration: Views

Select a role on the left side of the page first and then the desired view(s) in the list of *available views*. Click on ⬅ 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 on ➡ .

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. Only in case 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 arrow buttons ⬆ and ⬇ .

5.2.2 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.
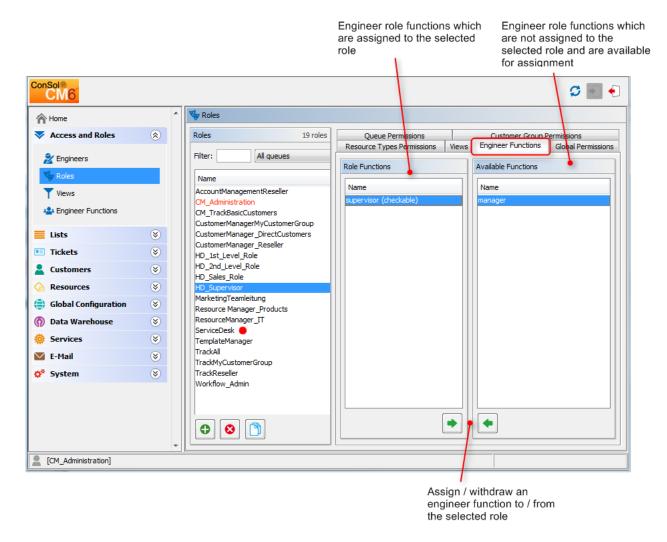
Fig. 8: ConSol*CM Admin Tool - Role Administration: Engineer Functions

Select a role on the left side of the page and then the desired engineer function(s) in the list of *available functions*. Click on ← 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 on → .

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 will obtain the rights of a role immediately after assignment (without an additional update of the system).

5.2.3 Delete a Role

Select the role you want to delete and click on ⊗ below the role list. If you choose *Yes* in the following confirmation dialog, the role will be removed from the list and the system.

> ⚠ **Attention:**
>
> If you delete a role, please consider that engineers with only this role will immediately loose all permissions in the system.

5.2.4 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 on 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.

5.2.5 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.

6 View Administration

- Introduction to View Administration
- View Administration Using the Admin Tool
  - Create a View
    - Queue Filter
    - Scope Filter
    - Static Criterion
    - Dynamic Criterion
  - Edit a View
  - Delete a View
  - Copy a View

6.1 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.

6.2 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.
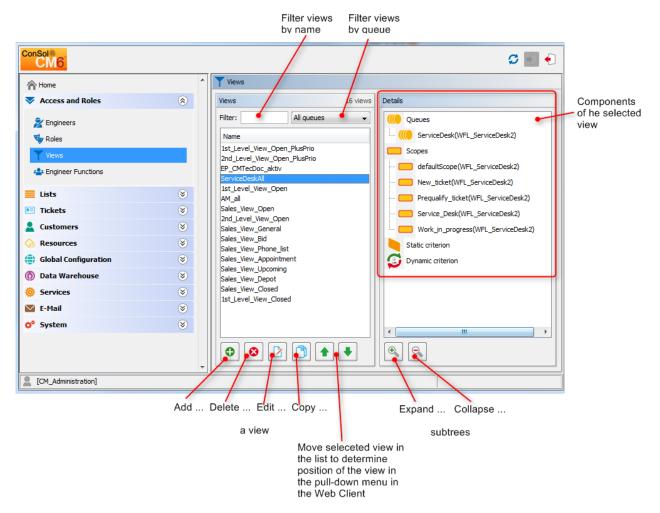
Fig. 1: ConSol*CM Admin Tool - View Administration

6.2.1 Create a View

After clicking on 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 on you can localize view name and description. The pop-up window *Localize* shows the available locales on the left side. Enter the corresponding view name or description in the *Value* field for each additional language on the right. After clicking *Save* the name or description will be displayed in the respective language of the engineer's locale.

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 on ← . To remove an assigned queue, select it and click on → . Continue with the *Next >* button, in order to define scope filters, too.

Fig. 2: ConSol*CM Admin Tool - View Wizard: Queue Filter

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 on ← . To remove assigned scopes, select them and click on → . Continue with the *Next >* button, if you want to define further criteria. Otherwise click on *Finish* to create the view.

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

> ⚠ **Attention:**
>
> 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,
> when 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!

Fig. 3: ConSol*CM Admin Tool - View Wizard: Scope Filter

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.

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, too. Otherwise click on
Finish to create the view.

Fig. 4: ConSol*CM Admin Tool - View Wizard: 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 parameter themselves. This can be done in the Web Client by editing the *User 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.

Fig. 5: ConSol*CM Admin Tool - View Wizard: Dynamic Criterion

Click on *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.
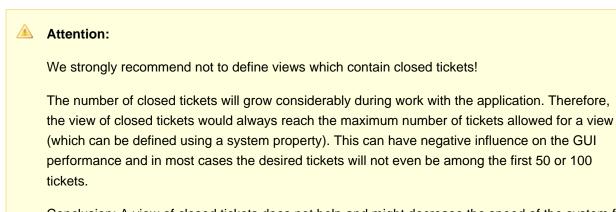
> ⚠ **Attention:**
>
> Please note that in a view with a dynamic criterion, only the tickets are displayed which match this criterion. So if an engineer has not selected any criteria in his/her engineer profile or if the administrator has removed all selections using the Admin Tool (*View criteria* in 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.

Fig. 6: ConSol*CM Admin Tool - View Administration: View Details

You can expand or collapse all details by clicking on 🔍 or 🔍 below the list.

> ⚠ **Attention:**
>
> 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 GUI 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.

6.2.2 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 on a filter criterion with the right mouse button. The following drop-down menu appears:
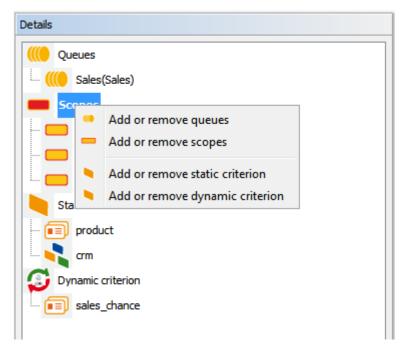
Fig. 7: ConSol*CM Admin Tool - View Administration: Edit a View

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 on 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*.

> **ⓘ Information:**
>
> You cannot edit view criteria by clicking on 🖉 . Here you can only modify name and description of a view.

6.2.3 Delete a View

Click on ⊗ below the view list to delete the selected view. A pop-up window appears where you are asked whether you really want to delete the view. If you choose *Yes*, the view will not be available for any engineer. Engineer permissions are not affected by this operation.

6.2.4 Copy a View

The icon 🗋 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 on the 🖉 icon.

7 Engineer Functions

- Introduction
  - Create or Edit an Engineer Function
  - Delete an Engineer Function
  - Disable or Enable an Engineer Function
- Engineer Permissions Concerning Engineer Functions

7.1 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 the 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*.

Fig. 1: ConSol*CM Admin Tool - User Attributes: Engineer Functions
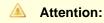
The corresponding activities for such a process have to be created in the workflow. Engineer functions are assigned to engineer roles which in turn need to be assigned to the respective engineers. In the Web Client you can choose a function and an appropriate engineer when adding an engineer to the ticket.

Fig. 2: ConSol*CM Web Client - Assigning an (Additional) Engineer with an Engineer Function

7.1.1 Create or Edit an Engineer Function

An engineer function is defined by its name. By clicking on ⊕ a pop-up window appears where you can enter the name. You will get the same window when you click on ⊡ in order to edit an engineer function. The check box *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.

> ⚠ **Attention:**
>
> After creation of an engineer function the check box *Checkable* cannot be de-selected anymore.

Fig. 3: ConSol*CM Admin Tool - User Attributes: Create or Edit an Engineer Function

You can also localize the name of an engineer function in this window. The available locales are shown on the left side of the table. Enter the corresponding engineer function name in the *Value* field for each additional language on the right. After clicking *OK* the engineer function is created and the name will be displayed in the respective language of the engineer's locale.

7.1.2 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 on ⊗ . After choosing *Yes* in the confirmation dialog the engineer function will be removed from the list and the system.

7.1.3 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 on ⊖ . The entry in the list is shown in italics afterwards. The engineer function cannot be assigned anymore. Just click on ✓ at the bottom of the page, if you want to enable the function again.

7.2 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):
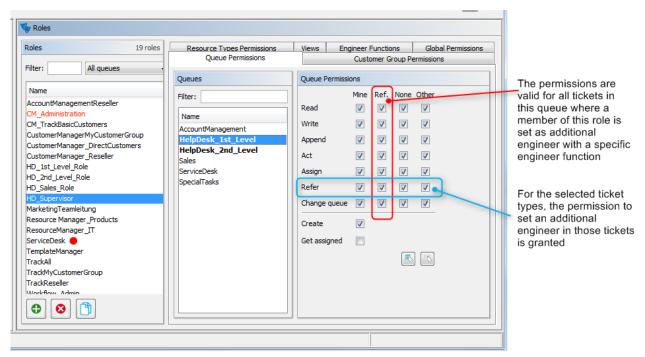
Fig. 4: ConSol*CM Admin Tool - Permissions Concerning Engineer Functions

8 Ticket Data Model and GUI Design Section

Ticket Data Model and GUI Design

In this section, you learn how to define the data model for ticket data, thus this is about Custom Field definition and the positioning of the custom fields on the web Client GUI. However, some of the data structures (enums, MLAs) might also be used for data object group fields, see section Data Object Group Field Management and GUI Design. 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:

- Custom Field Administration
- Managing Sorted Lists: Enum Administration
- MLA Administration
- Configuration of the Ticket History
- Configuration of the Web Client Dashboard
- Page Customization

9 Custom Field Administration (Setting Up the Ticket Data Model)

9.1 Introduction to Custom Field Administration

Custom Fields are fields that contain ticket data (e.g. *priority*, *software module*, *reaction time*, or *sales potential*) of the ConSol*CM system.

> ⚠ **Attention:**
>
> Custom fields (CFs) are defined as distinct fields, but the system configuration concerning Custom Fields is always based on Custom Field Groups (CF groups), never on single CFs.

A **Custom Field Group** (CF group) ...

- can be assigned to a queue, e.g. the CF group *helpdesk_data_fields* is assigned to the queue *HelpDesk*.
- can be faded in and out in the GUI during the process; only the whole group, not single fields (CFs).
- can be displayed as tab or in the *Groups* section of a ticket. The title (and mouse-over) of the tab is the (localized) name of the CF group.
- is configured by the group annotations. Annotations are used to define special parameters and characteristics for a CF, e.g. the position in the user interface. Please see section Appendix A for a list of all available annotations.
- is placed on the GUI based on its position in the CF group list (defines e.g. the order of tabs).

A **Custom Field** (CF) ...

- is always defined within a Custom Field Group.
- can be assigned to a queue only as part of its Custom 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 used to define special parameters and characteristics for a CF, e.g. the position in the user interface. Please see section Appendix A for a list of all available annotations.
- is placed on the GUI based on the value of its *position* annotation or based on its creation time ("first CFs first in the GUI") if *position* is not set.

When you want to **define new Custom Fields** for a queue (e.g. you need some data about hardware details like order number, IP address, and brand), you have to perform the following steps:

1. Define a Custom Field Group, set the respective annotations.
2. Define all sorted lists (enums) in the enum administration which you will need in the Custom Fields (e. g. when you need a CF brand which should contain a sorted list, you have to define this enum first).
3. Define all Custom Fields within the new CF group.
4. Assign the new CF group to all queues where the 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 press F5 to refresh the page.

All those steps will be 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.

9.2 Custom Field Administration Using the Admin Tool

In the Admin Tool, Custom Fields are managed using the navigation item *Custom Fields* in the navigation group *Tickets*.
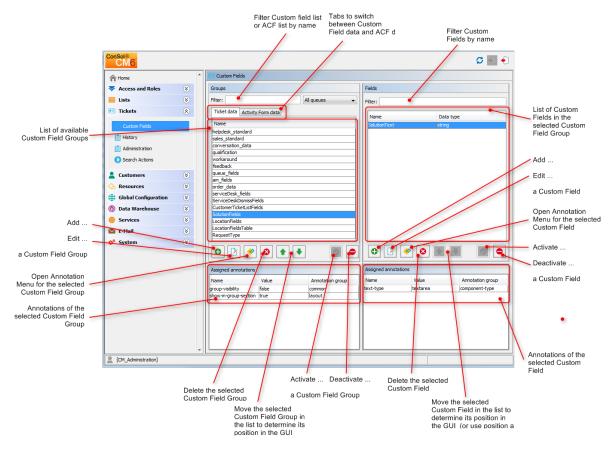
Fig. 1: ConSol*CM Admin Tool - Custom Field Administration

9.2.1 Tab Ticket Data

On this tab, you can define groups and fields for *ticket* data. The tab Activity Form data will be explained in one of the subsequent sections.

Create a Custom Field Group

To create a new Custom Field Group just click on the ⊕ icon below the list on the left side of the page. The following pop-up window appears.

Fig. 2: ConSol*CM Admin Tool - Custom Field Administration: Create a Custom Field Group

- **Name**
  Enter a name for the Custom Field Group. The name must be unique.
- **For all queues**
  If this check box is activated, this group's Custom Fields are visible in all queues. Usually Custom Field Groups for ticket data are only valid in specific queues (see Queue Administration).
- **Dependent Enum Scripts**
  Dependent enum scripts define the structure of *dependent enums* (hierarchical multi-level lists) used in Custom Fields of this Custom 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 Custom Fields have to be created within the 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 Custom Field Group select the desired script(s) in the list *Available scripts* and move them to the list *Assigned scripts* by clicking on ◄ .
- **Localized values**
  Enter the corresponding group name in the *Value* field for each additional language. In the Web Client's user interface the name will be displayed in the language of the engineer's web browser locale. If you do not make an entry here the object name, i.e. the content of the *Name* field, will be taken instead.

Edit a Custom Field Group

If you want to edit a Custom Field Group, select it in the list and click on [icon] . The same window as described above for creating a Custom Field Group will appear. You can modify all fields and save your changes by clicking *OK*.

Annotate a Custom Field Group

Custom 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. if a group is visible in the Web Client (annotation *group-visibility*) or if it is shown in the *Groups* section of the Web Client (annotation *show-in-group-section*). To assign annotations select a group and click on [icon] . The following pop-up window appears:
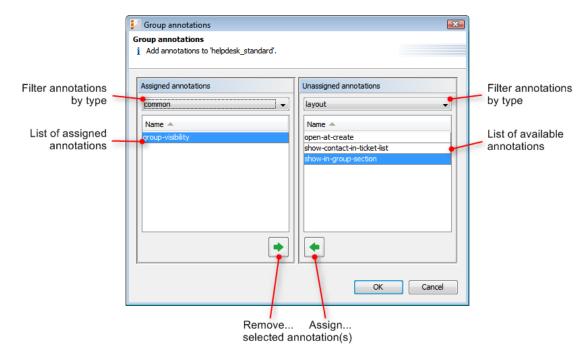
Fig. 3: ConSol*CM Admin Tool - Custom Field Administration: Assign Custom Field Group Annotations

The right part of the window contains the available annotations. Using the selection field above the list you can filter the display according to annotation type (e.g. *common* or *layout*). Select the desired annotations and move them to the *Assigned annotations* list on the left by clicking on [icon] . This list can also be filtered according to annotation type. Click on *OK* to assign the annotations to the Custom Field Group and to close the window. See Appendix A (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.

Custom Field Groups will appear in the Web Client as they are ordered in the list. Select a group and use the icons [icon] and [icon] if you want to change the position of this group in the list.

Delete a Custom Field Group

A Custom 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 Custom Field Group select it in the list and click on ⊗ . If you confirm the following dialog with *Yes*, the group with its corresponding fields will be removed from the list and the system.

Enable or Disable a Custom Field Group

If you cannot delete a Custom 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 on ⊖ . The entry in the list is shown in italics afterwards. Disabled Custom Field Groups are not displayed in the Web Client. Just click on ✓ below the group list, if you want to enable the group again.

Create a Custom Field

Custom Fields contain the data for tickets, e.g. priority, service level, deadline, or hardware module. The fields of a Custom Field Group are created in the right part of the page. Select the desired group first on the left and then click on the ⊕ icon below the Custom Field area on the right. The following pop-up window appears:

Fig. 4: ConSol*CM Admin Tool - Custom Field Administration: Create a Custom Field for Ticket Data

- **Name:**
  Enter a name for the Custom Field. The name must be unique.
- **Data type:**
  Choose one of the following data types for the new Custom Field:
  - **boolean**
    Values: true/false. Depending on the annotation *boolean-type*, the value is displayed as check box, radio buttons, or drop-down list.

    > ⓘ   When 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 check boxes, i.e. with annotation *boolean-type* = *checkbox* (default) is different depending on the CM version!
    >   - In CM versions prior to 6.9.4.0:
    >   When 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:
    >   When 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 already have been set 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*) have always shown and will show in future the behavior as described for versions 6.9.4.0 and up, i.e. with *NULL* value when untouched.

  - **date**
    Format and accuracy can be set by annotations.
  - **enum**
    For sorted lists. The engineer can choose one of the enum values on the Web Client. Enums and values have to be created previously within the Enum Administration. Select the desired *Enum type* and *group* in the fields below.
  - **list**
    A Custom Field of this data type is the first step to create 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 Custom 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 Custom Field of the type *struct* first (see below) before you can add the Custom 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 Custom 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 CF) always belongs to a *struct*, a struct always belongs to a *list*.

- **struct**

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

Fig. 5: Scheme: List of Structs

- **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.

- **long string**

  For large objects, unrestricted length.

- **short string**

  For up to 255 alphanumeric characters.

- **contact data reference**

  Special data type used internally for referencing the contacts associated with a ticket.

- **MLA field**

  This data type is used for Custom Fields that contain hierarchical lists with a tree structure called *MLA* (Multi Level Attributes). The name of the Custom Field will be the name of the new MLA that has to be defined within the MLA Administration. The group of the Custom Field has to be referenced when the MLA is created.

- **Belongs to:**

  This field shows the available Custom 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 Custom Field belongs (if applicable).

- **Localized values:**

  Enter the corresponding Custom Field name in the *Value* field for each additional language. In the Web Client, the name will be displayed in the respective language of the user's web browser locale. If you do not make an entry here the object name, i.e. the content of the *Name* field, will be taken instead.

> ⚠️ **Attention:**
>
> The data type you choose on creating a Custom Field cannot be changed afterwards!

Edit a Custom Field

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

Annotate a Custom Field

Just like Custom Field Groups Custom Fields are annotated to define the properties of the field, e.g. is it read-only, should it be indexed, where should it be displayed on the GUI (please see section Appendix A for a list of all available annotations). Select a field and click on 🖉 below the list. The following pop-up window appears:
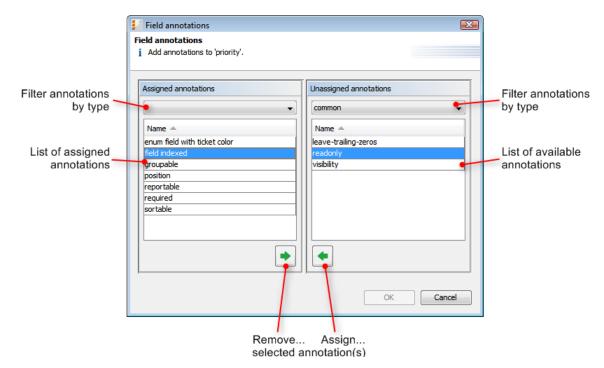
Fig. 6: ConSol*CM Admin Tool - Custom Field Administration: Assign Custom Field Annotations

The right part of the window contains the available annotations. Using the selection field above the list you can filter the display according to annotation type. Select the desired annotations and move them to the *Assigned annotations* list on the left by clicking on ◀ . This list can also be filtered according to annotation type. Click on *OK* to assign the annotations to the Custom 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 into the corresponding *Value* field and typing the desired value. Press the *Enter* key afterwards.

Custom 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 icons ⬆ and ⬇ below.

> ⓘ   For the layout of the ticket data, please note:
>
>   You can define three columns of data fields on each line, i.e. the position annotation (see table below) can start with 0, 1 or 2.
>
> | 0;0 | 0;1 | 0;2 |
> |-----|-----|-----|
> | 1;0 | 1;1 | 1;2 |

Delete a Custom Field

A Custom 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 Custom Field select it in the list and click on ⊗ . If you confirm the following dialog with *Yes*, the Custom Field will be removed from the list and the system.

Enable or Disable a Custom Field

If you cannot delete a Custom 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 on ⊖ . The entry in the list is shown in italics afterwards. A disabled Custom Field is not displayed in the Web Client. Just click on ⊘ below the Custom Field list, if you want to enable the field again.

9.2.2 Tab Activity Form Data

Short Introduction to Activity Control Forms (ACFs)

An Activity Control Form is a web form which is displayed during the process, i.e. when the engineer works on the ticket. When he/she clicks on a workflow activity, the ACF is displayed first. When the engineer has filled in and saved the required data, the process can continue. i.e. the workflow activity is executed.
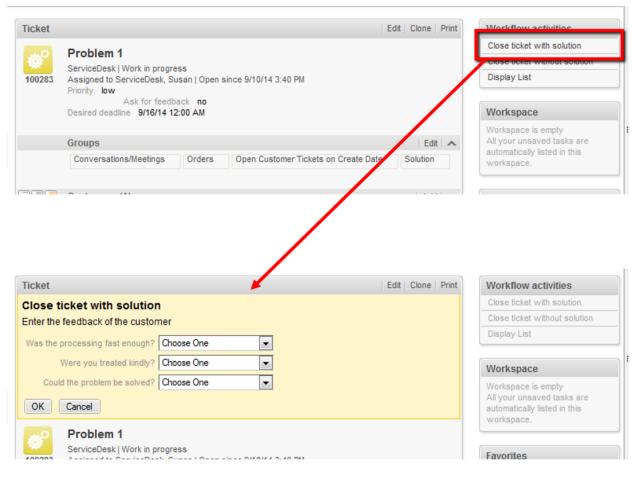
Fig. 7: Example ACF

The Custom Fields which are used in an ACF have to be defined first, as regular CFs. They can belong to one or more CF groups. The ACFs is integrated into the process (workflow) using the ConSol*CM Process Designer. Here, you can also define which CFs 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 when the customer has *Gold* or *Platin* status. ACF dependencies are configured using scripts. This is explained in detail in the *ConSol*CM Process Designer Manual*.

Definition of Activity Control Forms (ACFs)

On 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 when 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 Custom Fields already created on the tab *Ticket data*. An ACF can contain Custom Fields of more than one Custom Field Group. However, all Custom Fields have to be allowed in 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.
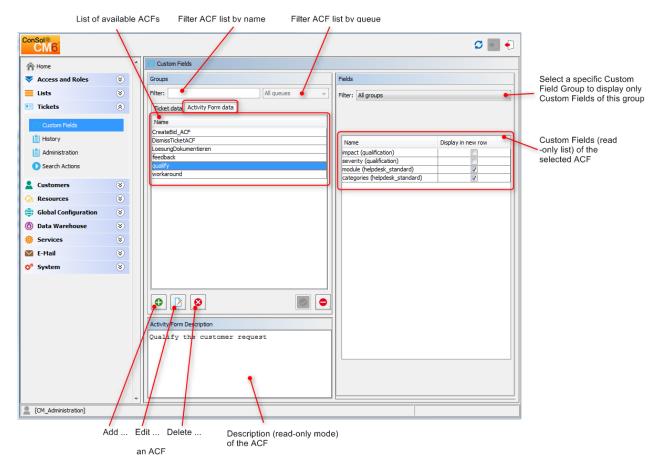
Fig. 8: ConSol*CM Admin Tool - Custom Field Administration: Activity Form Data

Create an Activity Control Form

To create an ACF just click on the ⊕ icon below the list on the left side of the page. The following pop-up window appears.
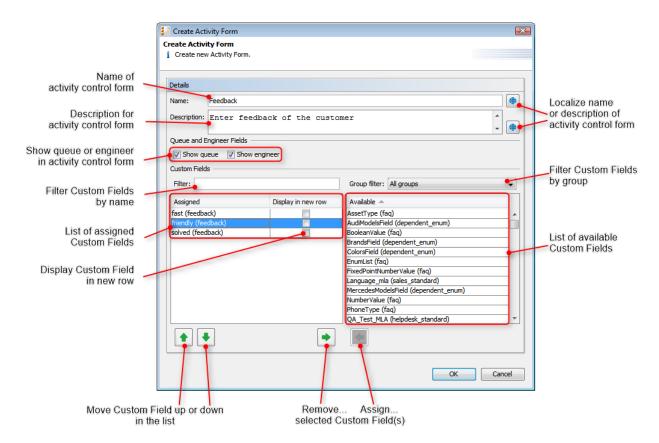
Fig. 9: ConSol*CM Admin Tool - Custom Field Administration: Create an Activity Control Form

Please enter or select the following data:

- **Name**

  Enter the name of the ACF in this field. You can localize the name by clicking on 🌐 (see below).

- **Description**

  Enter a description for the ACF in this field. The description will be shown as the title of the ACF in the Web Client. You can localize the description by clicking on 🌐 (see below).

- **Show queue**

  Mark this check box if the queue of the ticket shall be displayed with the ACF in the Web Client.

- **Show Engineer**

  Mark this check box if the engineer of the ticket shall be displayed with the ACF in the Web Client.

- **Filter**

  You can enter a string of characters into this field to filter the assigned Custom Fields by name.

- **Group filter**

  Select a group of Custom Fields from this list if you want to display only Custom Fields belonging to this group in the list of available Custom Fields below.

- **Custom Field lists**
  The list on the right shows the available Custom Fields with their respective Custom Field Group. You can sort the entries in ascending or descending order by clicking into the title field of the list. The icons ▲ or ▼ show the sort order. Select the Custom Fields for the ACF in this list and move them to the list *Assigned* on the left by clicking on ◀ .
  For each assigned Custom Field you can define if it shall be displayed in a new row by marking the corresponding check box. The assigned Custom 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 on ⬆ or ⬇ . To remove assigned Custom Fields, select them and click on ➡ .

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

Edit an Activity Control Form

If you want to edit an ACF, select it in the list and click on 🖉 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*.

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 you can only disable this ACF (see below). In order to delete an ACF select it in the list and click on ❌ . If you confirm the following dialog with *Yes*, the ACF will be removed from the list and the system.

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 on ⊖ . 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 on ✅ below the ACF list, if you want to enable the form again.

Localize an Activity Control Form

By clicking on the respective globe icon 🌐 in the create or edit window you can localize name and description of an ACF. The pop-up window *Localize* shows the available locales on the left side. Enter the corresponding name or description in the *Value* field for each additional language on the right. After clicking *Save* the name or description will be displayed in the respective language of the engineer's locale.

9.2.3 Frequently Used Annotations

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

Annotation	Annotation Type	Description	Value	Comment
groupable	cmweb-common	Enables grouping of fields in the ticket list.		Used only with *enum* Custom Fields. No value is needed.
sortable	cmweb-common	Used to enable sorting of the ticket list.		Used with Custom Fields of type *enum* or of type *Date*. No value is needed.
readonly	common	Used to indicate that the Custom Field cannot be modified.	true / false	Field is read only if value is set to *true*. Lack of value or any value except *false* is also treated as *true*.
visibility	common	Defines when the field is visible.	edit	Field will be displayed in *edit* mode.
			view	Field will be displayed in *view* mode.
			none	Field is not visible.
text-type	component-type	Defines the possible types of a string field.	text (default)	Single-line input field.
			textarea	Multi-line input field.
			password	Input field for passwords. Password will be displayed as ******* in *view* mode.

Annotation	Annotation Type	Description	Value	Comment
			label	Input will be displayed as a label, i.e. the field is displayed only, no input is possible.
			url	Input will be displayed as a hyperlink in *view* mode. String has to match a specific URL pattern.
reportable	dwh	Indicates that the field is reportable and that it should be transferred to the DWH.	true / false	Field is reportable if value is set to *true*.
field indexed	indexing	Used to indicate that a database index will be created for this field.	transitive (default)	All data is displayed (ticket and customer).
			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.
			not indexed	Field is not indexed.

Annotation	Annotation Type	Description	Value	Comment
position	layout	Defines the position of a field within a grid layout.	<number>; <number>	Values define *row* and *column* (row; column), numbering starts at 0;0. If no values are set, the Custom Field will take the next free grid cell.
		Defines the position of a field within a list (struct).	0;<number>	Only the *column* value is used, the *row* value is ignored.
colspan	layout	Defines how many columns are reserved for the field in the layout.	<number>	Number of columns.
rowspan	layout	Indicates how many rows within the layout are occupied by this field.	<number>	Number of rows.
field-group	layout	Allows grouping of fields in *view* mode. Annotation is ignored in *edit* mode.	<string>	To group fields the same string value has to be set in the annotation of each field.

Annotation	Annotation Type	Description	Value	Comment
fieldsize	layout	Displayed field size within ticket layout.	<rows>;<cols>	For *string* Custom Fields with annotation *text-type* and value *text area*. <rows> defines the number of displayed rows and <cols> defines the number of characters displayed per row. Used only for layout purposes.
			<number>	For *enum* Custom Fields. Defines how many values are directly visible in the list box. Used only for layout purposes.
enum field with ticket color	ticket display	Defines the background color of the ticket icon for ticket list and ticket.	true / false	The field has to exist within enum administration where lists, values, and colors are defined.
accuracy	validation	Used for date fields, to define the level of detail displayed.	date (default)	Show date without time.
			date-time	Show date with time.
			only-time	Show only time, no date.
format	validation	Used to check the correct format of *date* fields.	<date format>	The pattern for the date is based on *SimpleDateFormat*, e.g. dd.mm.yyyy.

Annotation	Annotation Type	Description	Value	Comment
maxLength	validation	Defines the maximum length of input for *string* Custom Fields.	<number>	May be used with *string* Custom Fields.
minLength	validation	Defines the minimum length of input for *string* Custom Fields.	<number>	May be used with *string* Custom Fields.
required	validation	Indicates that this is a required field.	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.

10 Managing Sorted Lists: Enum Administration

10.1 Introduction to Enum Administration

In the *Enum Administration* you can configure enums = sorted lists. They are part of the ConSol*CM data concept. You define an enum once and can use it multiple times:

- as a selection list (in drop-down menus) for custom fields or data object group fields of type *enum*
- as hierarchical lists for custom fields or data object group 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 Dependent Enum Scripts

> ⚠️ **Attention:**
>
> 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 custom field values or data object group field values), you have to complete one of the following steps:
>
> - create a custom field of type *enum* and assign the respective enum there
> - create a data object group field of type *enum* and assign the respective enum there
> - create an MLA which is linked automatically as custom field to the custom field group or as data object group field to the data object group that has been indicated during MLA set-up
> - create a *dependent enum script* and assign this to a custom field group

Examples:

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

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

- *enum-type* not set or *enum-type* = *select*:
  drop-down menu (default, see example in the picture below)
- *enum-type* = *radio*:
  radio buttons
- *enum-type* = *autocomplete*:
  self-completing (autocomplete) list

Fig. 1: ConSol*CM Web Client - Enum for Priority (Localized Enum Values Displayed as List Items)

10.2 Enum Administration Using the Admin Tool

Enums are organized in three levels:

- **Type**
  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**
  The *group* represents a group of enum values, i.e. the list.
- **Value**
  The *value* represents one value within a list.
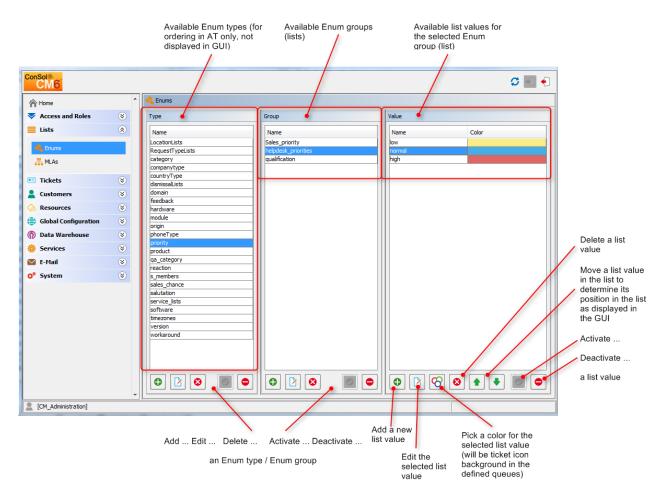
Fig. 2: ConSol*CM Admin Tool - Enum Administration

10.2.1 Enum Types

Create an Enum Type

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

Fig. 3: ConSol*CM Admin Tool - Enum Administration: Create 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. If you do not make an entry here the object name, i.e. the content of the *Name* field, will be taken instead.

Click on *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 on 🖊 . 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 on ❎ . 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 on ⊖ . The entry in the list is shown in italics afterwards. Just click on ⊘ below the type list, if you want to enable the type again.

10.2.2 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 on ⊕ below the *Group* area. The following pop-up window appears:

Fig. 4: ConSol*CM Admin Tool - Enum Administration: Create 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. If you select *user-defined* you
  can determine the sort order by means of arrow icons below the value list. If you choose *name* the
  values will be ordered alphabetically when you create them.
- **Localized values:**
  Enter the corresponding group name in the *Value* field for each additional language. If you do not
  make an entry here the object name, i.e. the content of the *Name* field, will be taken instead.

Click on *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 on ![icon] . 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 on ![icon] . 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

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 on ![icon] . The entry in the list is shown in italics afterwards.
Just click on ![icon] below the group list, if you want to enable the group again.

> ℹ️ **Information:**
>
> An enum group cannot be disabled if it is still used in an MLA.

10.2.3 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 on ![icon] below the *Value* area. The following pop-up window appears.

Fig. 5: ConSol*CM Admin Tool - Enum Administration: Create an Enum Value

- **Name:**
  Enter a value which shall 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. In the Web Client's user interface the name will be displayed in the language of the user's web browser locale. If you do not make an entry here, the technical object name, i.e. the content of the *Name* field, will be displayed.

Click on *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 on ⬚ . 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, if you click on ⬚ . This can be used for example for priorities. The priority of a ticket in the Web Client can be recognized immediately by its background color. This will be effective when the respective annotation is set, see the following info box.

> **ⓘ Information:**
>
> 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 custom field of type *enum* in the
> Custom Field Administration. For example, you can use the custom field *priority* to determine the
> ticket icon color in the helpdesk queue and the custom 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 on the desired color to set it for the marked list value. You can check the selected color in the *Preview*
area. Click on *OK* to save your choice. Click on *Reset* if you want to return to the last saved color.
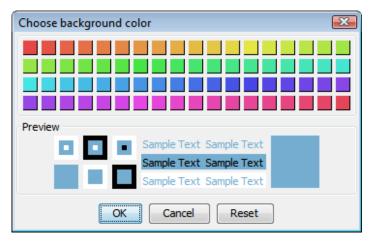
Fig. 6: ConSol*CM Admin Tool - Enum Administration: Set a Background Color

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
on ⊗ . If you confirm the following dialog with *Yes*, the value will be removed from the list and the system.

> **⚠ Attention:**
>
> 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 on ⬆ to move the selected value one line up resp. click on ⬇ 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

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 on 🛇 . The entry in the list is shown in italics afterwards and the value is not available in the Web Client any longer. Just click on ☑ below the value list, if you want to enable the value again.

> ⓘ **Information:**
>
> An enum value cannot be disabled if it is still used in an MLA.

10.2.4 Placing an Enum in the Data Model

Enums can be used in custom fields (i.e. for ticket data) and for data object group fields (i.e. for customer 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 custom field of type *enum* has to be defined. Please see section Custom Field Administration for a detailed explanation of the work with custom 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 data object group field of type *enum* has to be defined. Please see section Data Object Group Field Management and GUI Design for Customer Data for a detailed explanation of the work with data object group fields.

Fig. 7: ConSol*CM Admin Tool - Definition of a Custom Field of Type enum

11 MLA Administration

- Introduction to MLA Administration
- MLA Administration Using the Admin Tool
    - Create an MLA
        - Create an MLA Level
        - Edit a Level Value
        - Delete a Level
        - Enable or Disable a Level
    - Edit an MLA
    - Delete an MLA
    - Enable or Disable an MLA

11.1 Introduction to MLA Administration

MLA means *Multi Level Attributes*. 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 or for customer data.
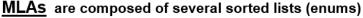
Fig. 1: ConSol*CM Admin Tool - MLA Construction Principle

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 *Mainboard* for item *Hardware* and so on. The picture below shows such an MLA in the Web Client.

Fig. 2: 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 Enum Administration and are just referenced when a new MLA level is defined.
- can be completely created within the MLA Administration during set-up of a new MLA.

11.2 MLA Administration Using the Admin Tool
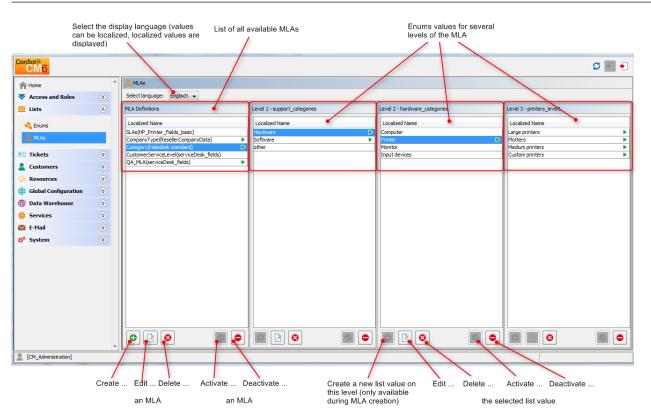
Fig. 3: ConSol*CM Admin Tool - MLA Administration

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 above the list.

11.2.1 Create an MLA

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

Fig. 4: ConSol*CM Admin Tool - MLA Administration: Create 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 custom field.
  - **Data object**
    MLA will be used in customer data, i.e. in a data object group field.
- **Group:**
  Choose the required custom field group (ticket data) or data object group (customer data) in the list box. For the new MLA a custom field or data object group field of type *MLA field* will be created automatically in this group. This is necessary to display the MLA on the Web Client. The custom field or data object group field can be annotated as described in sections Custom Field Administration and Data Object Group Field Management and GUI Design for Customer Data.
- **Localized values:**
  Enter the corresponding MLA name in the *Value* field for each additional language. On the Web Client the name will be displayed in the respective language of the user's browser locale. If you do not make an entry here the object name, i.e. the content of the *Name* field, will be taken instead.

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

> ⓘ **Information:**
>
> You can also create the custom field or data object group field for the MLA first. In this case you will find the localized name of the custom field or data object group field already in the list of available MLAs.

Create an MLA Level

Having created a name and a custom field or data object group field for the MLA you can go on with the definition of levels. Select the MLA in the list and click on ⊕ 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.

Fig. 5: ConSol*CM Admin Tool - MLA Administration: Create 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 the enum administration first) or create a new enum type directly here in the MLA administration. The new type will also be visible in the enum administration afterwards.
- **Select enum group:**
  Choose the desired enum group for this level from the lists in the 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 the 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 the respective enum values in the enum administration. If you have created a new enum group you also have to create one or more enum values for the new group.

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

You can either create all enum types, groups, and values 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 on ⊕ next to the respective fields in the window. By clicking on 🖉 or ❌ 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 already used in another MLA.

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

> ℹ️ **Information:**
>
> 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 the Enum Administration.

Edit a Level Value

If you want to edit a value of the level, select it in the list and click on 🖉 . 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 on ❌ 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 on ➖ below the respective level. The level values (including the values of dependent levels) are shown in italics afterwards. Click on ✅ if you want to enable the level again.

11.2.2 Edit an MLA

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

11.2.3 Delete an MLA

You can only delete an MLA if it is not in use. Otherwise 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 on ⊗ . If you confirm the following dialog with *Yes*, the MLA (and the custom field within *Custom Field Administration* or the data object group field within *User attributes*) will be removed from the list and the system.

11.2.4 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 on ⊖ . The entry in the list is shown in italics afterwards. A disabled MLA will not be displayed in the Web Client. Just click on ⊘ below the MLA list, if you want to enable the MLA again.

12 Ticket History

- Introduction
- Display Modes of Ticket History in the Web Client
- General Information about the Visibility of Ticket History Entries
    - The Display Mode
    - The Visibility Level
- Ticket History Storage and Transfer to the Data Warehouse (DWH)

12.1 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 e-mails as well as automatic and system actions like the change of the responsible engineer or the check if a time trigger has to fire. For those of you who know 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.

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 on this page.

12.2 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 major action or event that has taken place concerning a ticket. The entries of the indicated type(s) will be visible in the ticket history when the user has selected the respective visibility level. This is of importance when the display mode *Display all entries* is used.

> ℹ️ The visibility of e-mails, 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 e-mails and comments is named *default_class* and the default class for attachments is named *default_attachment_class*).

Available action types
(list cannot be modified)

Visibility level
configuration for the
action types
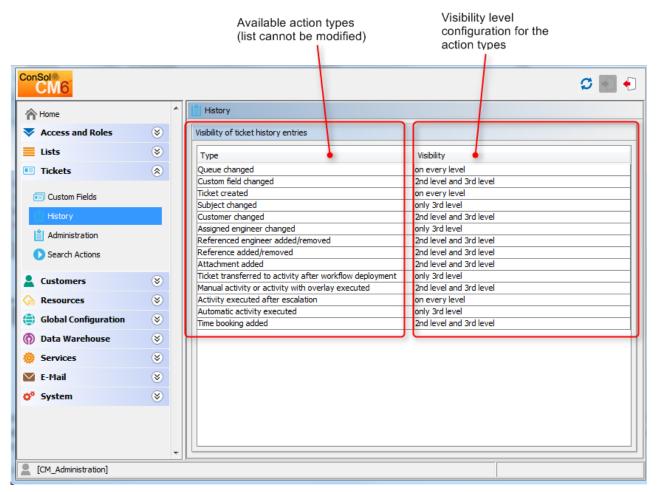
Fig. 1: ConSol*CM Admin Tool - Configuration: Ticket History

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

- **Type**

  The type of action that has been performed.
- **Visibility**

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

The following figures show the action type *time booking added* configured for *2nd level and 3rd level*.

Fig. 2: ConSol*CM Web Client - Time Booking Entry Not Visible on 1st Level

Fig. 3: ConSol*CM Web Client - Time Booking Entry Visible on 2nd Level

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

Fig. 4: ConSol*CM Admin Tool - Selecting 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*:

Fig. 5: ConSol*CM Web Client - Time Booking Entry Visible on 1st Level

12.3 General Information about the Visibility of Ticket History Entries

There a **two visibility settings** configuring the visibility of ticket history entries:

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

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

- **Communication**
  Refers to comments, e-mails and attachments. 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 e-mails and comments is named *default_class* and the default class for attachments is named *default_attachment_clas*s).
- **Other entries** For example, entries about a ticket changing the queue or the 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.

12.3.1 The Display Mode

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

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

Display mode *Display communication*

Shows only comments, e-mails and attachments (not the full attachment, but an entry, that an attachment has been added and a link to open the attachment). Whether an communication entry is shown and in which length they are shown, depends an the selected visibility level (including the additional information, whether entries shall be displayed *full* or *short* within the chosen visibility level).

Display mode *Display all entries*

Shows all other entries in addition the the communication entries (e.g. entries about a ticket changing the
queue or the 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 this entries.
These *other entries* are visible only when 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 entires, there is no differentiation between *full* and *short* visibility for these
other entries.

Fig. 6: ConSol*CM Web Client - Setting the Display Mode

12.3.2 The Visibility Level

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 e-mails and comments is named
default_class and the default class for attachments is named *default_attachment_clas*s).

In addition to the general visibility of an 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 entires, there is no
differentiation between *full* and *short* visibility for these other entries.

Fig. 7: ConSol*CM Web Client - Setting the Visibility Level

12.4 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 custom field groups or single custom 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* does not have to be set for the respective custom fields or custom field groups.
6.10.2 and up	dwh-no-history	dwh-no-history-field	For the custom field groups or single custom fields with this annotation, no history data will betransferred to the DWH.

Thus, when 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. when some internal programing data is written into custom fields, the 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.

Information:

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

For keeping old system configurations valid, for all custom fields and custom 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.

13 Configuration of the Web Client Dashboard

Starting with version 6.9.4, the ConSol*CM Web Client provides a *dashboard* for engineers which is displayed on the Web Client *Overview* page. The dashboard is composed of one or more so called widgets. As default configuration, the dashboard contains only one widget which displays a graphic with the 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.

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

Fig. 1: ConSol*CM Web Client - Web Client Dashboard with Default Graphic

Fig. 2: 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 two possible types:

- **Chart**
  for graphical representation of data

- **Table**
  for table representation of data

You as an administrator can design the layout of the dashboards as required, i.e. you can place chart widgets and table widgets on the dashboard page. They are placed on the page based on a grid layout.
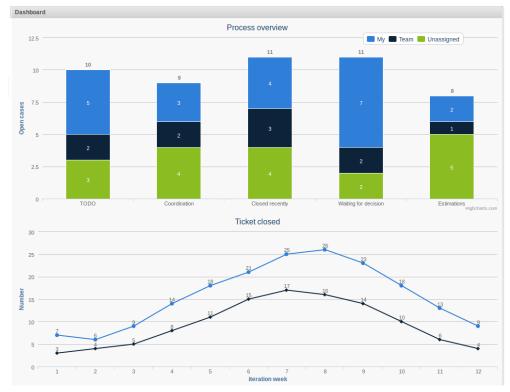
Fig. 3: ConSol*CM Web Client - Example for a Dashboard with Two Graph Widgets in One Column

Fig. 4: ConSol*CM Web Client - Example for a Dashboard with One Chart and One Table Widget in One Column

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.

14 ConSol CM Administrator Manual 6.10.2 - Page Customization

14.1 Page Customization

- General Introduction to Page Customization
- Page Customization in the Web Client
- Page Customization Using Attributes
    - Possible Pages (Scopes) for Page Customization
        - companyEditPage
        - contactCreatePage
        - contactEditPage
        - resourceDashboard
        - resourceType (Resource Type Page)
        - resource (Resource Detail Page)
        - officeTemplatePage (Only When CM.Doc Has Been Activated)
        - searchDetailPage
        - templateViewPage
        - ticketCreatePage
        - ticketEditPage
        - userProfilePage
        - welcomePage

14.1.1 General Introduction to Page Customization

In addition to the design of the Web Client GUI in the process of defining Custom Fields (see section Custom Field Administration) and Data Object Group Fields (see section Data Object Group Field Management and GUI Design), an administrator can configure more GUI layout features and functionalities using **page customization**.

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 functionalities.

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 E-Mail Editor is open, you can also configure e-mail 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.

14.1.2 Page Customization in the Web Client

When you start the page customization mode, the *Page Customization Definition Section* (PCDS) 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 editor section for this element in the PCDS is opened, and the element is marked in the tree. That way you can easily identify the component you would like to modify.

Fig. 1: ConSol*CM Web Client - Page Customization Definition Section

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

```
http://cm6-demo.int.consol.de
/cm-client/
    └ navigationLinks
        └ ticketEditPage
    └ globalSearchField
        └ ticketEditPage
    └ accordionTicketList
        └ ticketEditPage
            └ ticketList
    └ ticketPanel
        └ ticketEditPage
    └ customerSectionPanel
        └ ticketEditPage
    └ acimSection
        └ ticketEditPage
    └ attachmentSection
        └ ticketEditPage
```

Fig. 2: ConSol*CM Web Client - Page Customization Tree

Icon	Description
	Configuration of all components of this type
	Configuration of this specific component 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 PCDS, e.g. for the element *navigationLinks* (see following figure).

Fig. 3: ConSol*CM Web Client - Selected Tree Element in PCDS

The entire page is built according to a strictly hierarchical structure and every element is defined by a type, a scope, and a class. These are displayed in the blue header section of the editor in the PCDS 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 and classes. When you work on the *scope* level you define the attributes for all (sub-)elements of this scope. When you work on the *class* level, you define the attributes for this class only.

Please see the following example for ticket list configuration.

- **Type level:**

Fig. 4: ConSol*CM Web Client - Defining Parameters on Type Level

- **Scope level:**

Fig. 5: ConSol*CM Web Client - Defining Parameters on Scope Level

- **Subscope level:**

Fig. 6: ConSol*CM Web Client - Defining Parameters 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, the ticketEditPage / contactSection subscope can be configured for two different types: *customerSectionPanel* and *sectionList*.

Fig. 7: 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 check box.

Fig. 8: ConSol*CM Web Client - Configuration Script for Defining Values

E.g. there should be only one e-mail recipient (here: the main customer) for medium and low priority tickets, but 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'];
```

It has to be stored in the *Scripts* section of the Admin Tool (see section 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.
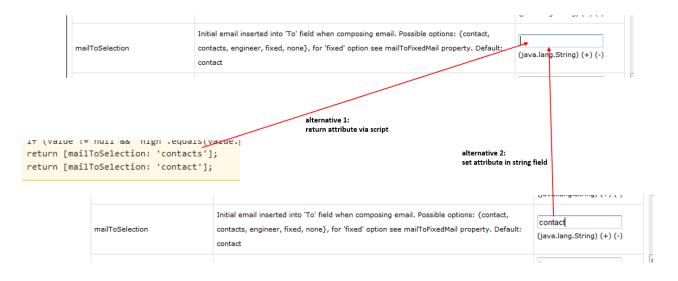
Fig. 9: Two Alternatives for Setting Page Customization Attributes

14.1.3 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.

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

Fig. 10: companyEditPage for Page Customization

contactCreatePage

Fig. 11: ConSol*CM Web Client - Create Contact Page

contactEditPage

Fig. 12: ConSol*CM Web Client - Edit Contact Page

resourceDashboard

Fig. 13: ConSol*CM Web Client - Resource Pool Dashboard Page Customization

resourceType (Resource Type Page)

Fig. 14: ConSol*CM Web Client - Resource Type Page for Page Customization

resource (Resource Detail Page)

Fig. 15: ConSol*CM Web Client - Resource Detail Page for Page Customization

officeTemplatePage (Only When CM.Doc Has Been Activated)

Fig. 16: ConSol*CM Web Client - Word Template Page (Only When CM.Doc is Activated)

searchDetailPage

Fig. 17: ConSol*CM Web Client - Detailed Search Page

templateViewPage

Fig. 18: ConSol*CM Web Client - Template View Page

ticketCreatePage

Fig. 19: ConSol*CM Web Client - Create Ticket Page

ticketEditPage

Fig. 20: ConSol*CM Web Client - Edit Ticket Page

userProfilePage

Fig. 21: ConSol*CM Web Client - User Profile Page

welcomePage

Fig. 22: ConSol*CM Web Client - 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

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

acimSection (Type or Subscope)

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

- a comment
- an e-mail which has been sent from the ticket
- an e-mail 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.

Fig. 23: ConSol*CM Web Client - ACIM Group and Item

> ⊖ **Warning:**
>
> 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

Fig. 24: ConSol*CM Web Client - Valid Date Formats for ACIM Date Configuration

Attributes:

- **acimGroupActionEntryDateFormat**
  Date format for group of ACIM without text or e-mail 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)

Fig. 25: ConSol*CM Web Client - Display for Format: dd.MM.yyyy HH.mm|dd.MM.yyyy HH.mm|dd.MM.yyyy HH.mm

Fig. 26: 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, e-mail, 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)

Fig. 27: ConSol*CM Web Client - Display for Format: dd.MM.yyyy HH.mm|dd.MM.yyyy HH.mm|dd.MM.yyyy HH.mm

Fig. 28: 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 e-mail 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 e-mail entry). (boolean)

Fig. 29: 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 clone for another item is selected simultaneously. Possible
  values are *append*, *replace*. Default is *append*. (java.lang.String)
- **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)
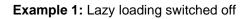
Example 1: Lazy loading switched off

Fig. 30: ConSol*CM Web Client - Lazy Loading Switched Off

Example 2: headHistoryElementsCount and tailHistoryElementsCount set to *1*

Fig. 31: ConSol*CM Web Client - headHistoryElementsCount and tailHistoryElementsCount Set to 1

- **mailToSelection**
  Initial e-mail address inserted into TO-field when composing e-mail. Default: *contact* (java.lang.String)
  Possible options:
    - **contact**
      The e-mail address of the main contact is copied into the TO-field when the Ticket E-Mail Editor is opened.
    - **contacts**
      The e-mail addresses of the main contact and all additional contacts are copied into the TO-field when the Ticket E-Mail Editor is opened.
    - **engineer**
      The e-mail address of the engineer is copied into the TO-field when the Ticket E-Mail Editor is opened.
    - **fixed**
      for *fixed* option see *mailToFixedMail* attribute.
    - **none**
      The TO-field is left empty when the Ticket E-Mail Editor is opened.
- **mailToFixedMail**
  Fixed e-mail address used when attribute *mailToSelection* is set to *fixed*. (java.lang.String)
  Example: E-mail to a fixed e-mail address, *mailToFixedMail* set to *foo@bar.de*, *mailToSelection* set to *fixed*.

Fig. 32: ConSol*CM Web Client - E-Mail to a Fixed E-Mail Address

- **recordLastUsedAcimTab**
  Records last used ACIM tab, i.e. when you open the editor again, the tab (comment/e-mail /attachment/time booking) will be opened that was open when you closed the editor last time. (boolean)
- **reloadPageIfIE8onAcimSubmit**
  Reloads page after a new ACIM submit, only for *IE8*. This is a workaround for the problem that adding comments/e-mails may take a long time when using IE8. (boolean)
- **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 E-Mail Editor to the Comment Editor, the text you have typed will remain in the editor panel and will not be removed)
- **timeBookingFeature**
  Activates or deactivates the time booking support in acimSection (i.e. display the link to time booking and the tab in the editor). (boolean, default = *true*)

Fig. 33: ConSol*CM Web Client - Activate Time Booking Support (timeBookingFeature = true)

Fig. 34: ConSol*CM Web Client - De-Activate Time Booking Support (timeBookingFeature = false)

> ℹ **Information:**
>
> Please note that the value *false* in the *timeBookingFeature* hides the hyperlink from the time booking editor, see figure above. The engineer cannot show it again by using the *Display* option in the ticket's menu!

Please keep in mind that the visibility of the time booking section on the *userProfilePage* is configured via the *timeBookingSection* attribute *visible*!

accordionTicketList (Type)

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

Attributes:

- **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 immediately after two necessary libraries are 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*.

- **quickAssignLinkShowsTicketPageFlag**
  Whether the *quick assign* link (represented by the arrow rendered for each unassigned ticket) opens the assigned ticket immediately (boolean, default is *false*).

attachmentSection (Type or Subscope)
Attributes:

- **defaultVisibilityFlag**
  The visibility of the attachment section for engineers who did not change 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 higher versions (i.e. 6.9.4 and up) with the new ticket section management, use the *state* attribute in sectionList / ticketEditPage / attachmentSection.

Fig. 35: ConSol*CM Web Client - Visibility of Attachment Section (defaultVisibilityFlag = true)

Fig. 36: ConSol*CM Web Client - Visibility of Attachment Section (defaultVisibilityFlag = false)

autocomplete

(available e.g. on *userProfilePage*)

Attributes:

- **suffixCharactersToRemove**
  Occurrence of any 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.

Fig. 37: ConSol*CM Web Client - Input in a Search Field Where Charactes 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 (eg: 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 at 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.)

Fig. 38: 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 MS Exchange Calendar Integration for a detailed explanation of the feature with a list of all available attributes.

cmRichTextEditor

Attributes:

- **editorFeatures**

  Additional editor features. As default value, all values are set, i.e. the respective menu entries are available. (java.lang.String)

  Possible values:

  - **SUB_SUP** X_2 X^2

    Allows subscript and superscript.

  - **INDENTS**

    Provides the possibility to add indents to the text.

  - **LISTS**

    Allows to insert/format lists.

  - **TABLES**

    Allows to insert tables.

  - **INSERT**

    Allows to insert text elements; for finer control:

    - **INSERT_EMOTION**

      Allows to insert emoticons.

    - **INSERT_CHAR** Ω

      Allows to insert special characters.

    - **INSERT_IMAGE**

      Allows to insert images.

- **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 offered 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=6pt,regular size=10pt, large=12pt".

- **imagePasteEnabled**

  Flag, whether direct pasting of images from clipboard into editor is enabled. Note that enabling this requires a Java Applet to run (boolean, default = false). Web browsers (i.e. Internet Explorer, Firefox) might show different behavior concerning display of the images.

customerGroupSelector

Available directly under root node in PCDS.

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.

Fig. 39: ConSol*CM Web Client - hiddenCustomerGroups

customerSectionPanel

Here you can define if 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*)
  The visibility of the link for editing referenced units (boolean, default is *true*).
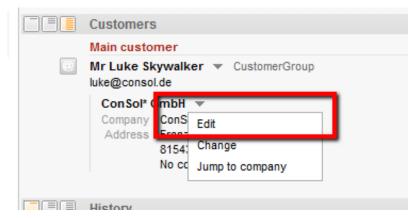
Fig. 40: ConSol*CM Web Client - Visibility of Edit Link (referencedUnitEditLinkVisible = true)

Fig. 41: 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*.
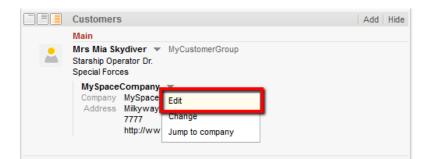
Fig. 42: ConSol*CM Web Client - Visibility of Company Ddit Link (companyEditLinkVisible = true)

Fig. 43: 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.

Fig. 44: ConSol*CM Web Client - Page Customization for Sort Order of Additional Coustomers (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 Data Object descriptions used for the display of contacts and companies are taken from the template *<Contact>.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*.

Fig. 45: ConSol*CM Web Client - Page Customization for Sort Order of Additional Customers (2)

Fig. 46: ConSol*CM Web Client - Page Customization for Sort Order of Additional Customers (3)

detailSearch

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 Custom 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 Custom Fields or Data Object Group 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

- **maxGridTicketsNumber**
  Maximum number of tickets shown in grid view, i.e. in the entire grid, not in one column. The default value is *120*.

Fig. 47: ConSol*CM Web Client - Using Page Customization to Adapt Maximum Number of Tickets Shown in Grid

Fig. 48: 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.

Fig. 49: ConSol*CM Web Client - Defining a Script for Maximum Number of Tickets in Grid View Using Page Customization

Fig. 50: ConSol*CM Admin Tool - Script for Maximum Number of Tickets in Grid

Fig. 51: ConSol*CM Web Client - Number of Tickets in Ticket Grid for Different Engineer Roles

engineerAutocomplete

(available on *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, no limit.
- **suffixCharactersToRemove**
  Occurrence of any of these characters will be removed from the tail of each search pattern word. (string, default: empty)
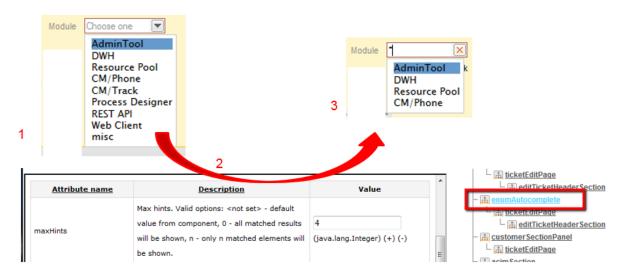
Fig. 52: ConSol*CM Web Client - Using enumAutocomplete Parameter to Reduce Number of Hints

generalFeedback

This type is used to manipulate log entries in a clustered environment.

Attributes:

- **appendHostInfo**
  Configures if 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 *lappendHostInfo.* There are two properties which can be used:
    - cmas.http.host.port (default)
    - cmas.clusternode.id

globalSearchField

Here you can define the layout and the behavior of the *Quick Search* input field.
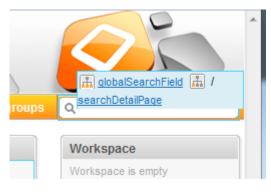
Fig. 53: ConSol*CM Web Client - Quick Search Input Field

Attributes:

- **maxHints**

  Defines the maximum number of suggestions which is displayed. When set to *0*, all suggestions are displayed, 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

Attributes:

- **addingManyTemplateEnabled**

  Makes it possible to compose an e-mail by using more than one template. (boolean)

- **engineerPersonalMailsIncluded**

  Enable appending personal e-mail feature. (boolean)

- **includeTextBlocksInEmailTemplate**

  Whether text blocks from e-mail template will be included by default. (boolean)

- **mailBodyLockedAfterTemplateSelection**

  Indicates whether e-mail body will be locked after template selection. (boolean)

- **mailEncryptionAvailable**

  Makes e-mail encryption option available. (boolean)

- **mailSelectionComponentWidth**

  The width of the e-mail selection component (in pixels). (java.lang.Integer)

- **mailTemplateSortStrategy**

  E-mail 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 e-mail
  recipients. Integer. Default 1.
- **quotingFeature**
  Activate the quoting function in e-mail content. (boolean)
- **selectionComponentWidth**
  The width of the mail/comment selection component (in pixels). Integer.
- **showUniqueEmails**
  Results in autocomplete e-mail fields will be compared by e-mail, only first e-mail will be used in
  results. If set to *true* then results will be compared by whole e-mail description. (boolean)
- **templateSortStrategy**
  Template list sorting strategies. Default value is: USAGE, NAME. Possible comma separated options
  are: USAGE, NAME. String. Default USAGE,NAME.

navigationLinks

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* link can be shown (boolean, i.e. possible values are *true* or *false*, default
  value is *true*).

  > ⓘ **Information:**
  >
  > In addition to this attribute being set to *true*, engineers must have appropriate permissions
  > to see the *createContact* link.

- **createTicketLinkVisible**
  Whether the *createTicket* link can be shown (boolean, i.e. possible values are *true* or *false*, default
  value is *true*).

  > ⓘ **Information:**
  >
  > In addition to this attribute being set to *true*, engineers must have appropriate permissions
  > to see the *createTicket* link.

- **externalLinks**
  External links which will be appended to navigation bar. This attribute may configure more than one external link (the order matters).
  Format (compatible with wiki): [http://link description]
  This attribute might 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 on the Internet and/or intranet.
  Example: [http://www.consol.com ConSol*][http://www.somewhere.com Somewhere]
- **manageTemplateLinkVisible**
  Whether the *Manage Templates* link (for the start of the *ConSol*CM Template Manager*) can be shown (boolean, i.e. possible values are *true* or *false*, default value is *true*). See also section The ConSol*CM Template Manager.

  > ⓘ **Information:**
  >
  > In addition to this attribute being set to *true*, engineers must have appropriate permissions (*Global Permissions* - *Write template*) to see the *Manage Template*s link.

- **officeTemplateLinkVisible**
  Whether the *officeTemplate* link (for the start of the *Doc Template Manager*) can be shown (boolean, i.e. possible values are *true* or *false*, default value is *true*).

  > ⓘ **Information:**
  >
  > In addition to this attribute being set to *true*, engineers must have appropriate permissions (*Global Permissions* - *Write template*) to see the *officeTemplate* link. CM.Doc has to be enabled in the system (see section CM.Doc).

- **overviewLinkVisible**
  Whether the *Overview* link can be shown.

resourceRelations

Available e.g. on Ticket Page

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_preload,
  hidden], default: 'expanded'

resourceRelationsPanel

Available e.g. on Ticket page.

Attributes:

- **preserveOrder**
  It allow to preserve order of elements. By default elements will be placing in optimal position based
  on available vertical space. Default is *false*.

resourceTypes

Available e.g. on Resource Dashboard

Attributes:

- **preserveOrder**
  It allow to preserve order of elements. By default elements will be placing in 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)

Avaiable in the following scopes.

- ticketEditPage
- contactEditPage
- companyEditPage

Attributes:

- **counterVisible**
  Whether the counter should be shown in the section header. The default value is true.

counterVisible = *true*   counterVisible = *false*

Fig. 54: ConSol*CM Web Client - Attribute counterVisible
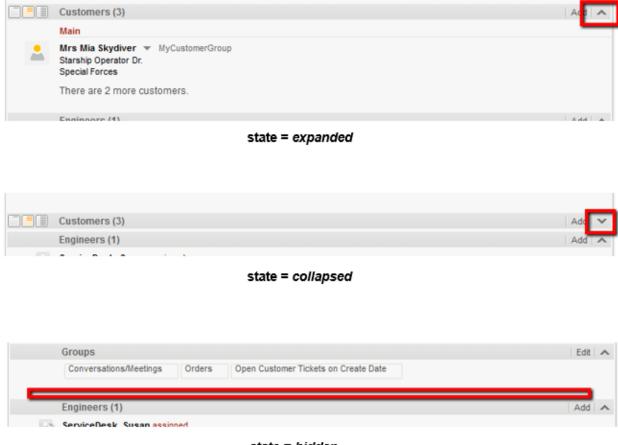
- **state**
  The visibility mode of the section, possible values are [expanded, collapsed, collapsed_and_preload, hidden], default: 'expanded'. The default value is 'expanded'. This mode defines the initial visibility mode after start of a session. The engineer can change the visibility of a section afterwards, but this will not be saved. When the next session is started (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 by database or JBoss Admin Console access ⚠

⊖ **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 by database access or using the JBoss Admin Console ⚠ So please think twice before applying this value ⚠

---

state = expanded

state = collapsed

state = hidden

Fig. 55: ConSol*CM Web Client - Attribute state

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.

ticketList (Subscope)

see *accordionTicketList* (type)

ticketsAutocomplete

(available on the relations addition form)

Attributes:

- **maxHints**
  Defines the maximum number of suggestions which is displayed. When set to *0*, all suggestions are displayed, no limit.

ticketsBookingAutocomplete

(available on the time booking addition form of the *userProfilePage*)

Attributes:

- **maxHints**
  Defines the maximum number of suggestions which is displayed. When set to *0*, all suggestions are displayed, no limit.

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:**

Fig. 56: ConSol*CM Web Client - Button "go to top and bottom page" Not Visible

- **topBottomPageButtonVisible = true:**

Fig. 57: ConSol*CM Web Client - Button "go to top and bottom page" Visible

TicketRelation

Available in the following scopes:

- TicketRelationSection (e.g. on Resource Detail Page)

Attributes:

- **compactViewLimit**
  If the limit is exceed the table is presented along with filtering feature. The default limit is set to 10.
- **defaultRelationSortStrategy**
  Additional resource relation sorting strategies. Possible options are(default is CREATION_DATE):
  CREATION_DATE, FIRST_SORTABLE_COLUMN(sort by first column which is sortable)

timeBookingSection

(available e.g. on *userProfilePage*)

Attributes:

- **visible**
  The visibility of the time booking section on the *userProfilePage*. (boolean, 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*!

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, 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.

Fig. 58: ConSol*CM Web Client - Suggestions for the Ticket Contact

UnitResourceRelation

Available in unitRelationsSection (e.g. on Resource Detail Page).

Attributes:

- **compactViewLimit**
  If the limit is exceed the relations are presented along with filtering feature. The default limit is 10.
- **defaultRelationSortStrategy**
  Additional resource relation sorting strategies. Possible options are(default is CREATION_DATE): CREATION_DATE, FIRST_SORTABLE_COLUMN(sort by first column which is sortable)
- **numberOfRelations**
  The default number of the presentedt relations. 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*)

Fig. 59: ConSol*CM Web Client - EXTENDED aidLevel in the Unit Search

unitRelationSection (Type UnitSection)

Available on the

- contactEditPage
- companyEditPage

Attributes:

- **compactViewLimit**
  If the limit is exceed the relations are presented along with filtering feature. The default limit is 10.

Fig. 60: ConSol*CM Web Client - Effect of Attribute compactViewLimit

- **numberOfRelations**
  The default number of the presented relations. 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, see next code example.

```
JSon Object for Customer Data Format in Box Preview

{    "layout": [                  ["firstname", "lastname", "lastname"],                  ["email",
"email",    null]                ]}
```

For different unit definitions in the FlexCDM use the following syntax:

```
Example Value for Customer Data Preview Box Layout

{''unit_definition_1": <json description>, "unit_definition_2": "component_name_used_to_display_
preview"}.
```

In Json we set the configuration for a particular unit definition. Json document can contain unit preview or just the name of a component (spring bean) used to render the preview.

unitSearchHeader

(available on *ticketCreatePage* in the company section)

Attributes:

- **companyCreateLinkVisible**
  The visibility of the link for referenced company creation. (boolean)

viewDiscriminatorsSection (Type)

(available e.g. on *userProfilePage*)

Attributes:

- **visibilityFlag**
  The visibility of the *View criteria* section (section for attribute settings for dynamic views on *userProfilePage*). (boolean, default value = *true*)

Fig. 61: ConSol*CM Web Client - Visibility of View Criteria Section (visibilityFlag=true)

Fig. 62: ConSol*CM Web Client - Visibility of View Criteria Section (visibilityFlag=false)

welcomePage

The most important configuration on the *welcomePage* is the Page Customization for the Web Client Dashboard.

14.1.4 Order and Priorities of Page Customization

In case there are more than one values which are set for an attribute, the following hierarchy is applied:

1. **Highest priority:** script
2. **Medium priority:** scope definition
3. **Lowest priority:** type definition
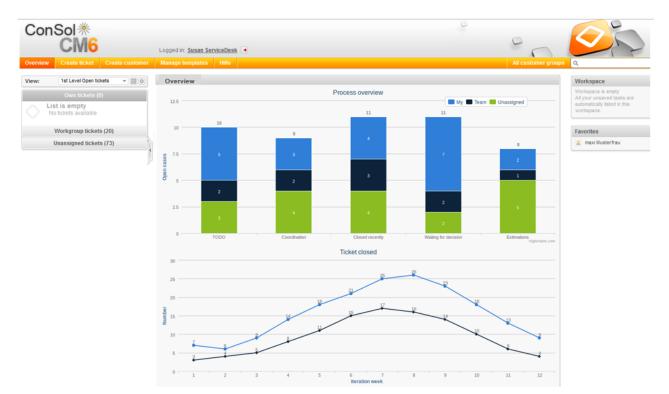
Example for the value of *maxHints* in the *Global Search* field on the *ticketEditPage* :

- **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 definitoin (GlobalSearchField): maxHints = 5
    - => maxHints will be 7

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

For keeping old system configurations valid, all fields and groups annotated with *no-history-field* and *no-history* get assigned correspondingly: *dwh-no-history-field* and *dwh-no-history* annotations during update from previous CM versions.

14.2 Page Customization for the Web Client Dashboard

14.2.1 Introduction

Fig. 1: ConSol*CM Web Client - Web Client Dashboard with two Widgets

The **Web Client Dashboard** is configured using *page customization*.
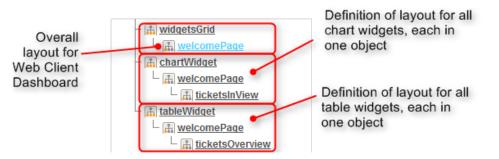
Log in as *admin*, open the *Overview* page and select *Enable page customization* in the main menu. Besides other attributes which are not relevant for the Dashboard and are explained on in the section Page Customization, the following (Dashboard-relevant) elements can be configured (i.e. attributes can be set). Each of the three elements is represented by a subtree in the page customization tree.

1. **widgetsGrid / 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. 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 chartWidgets 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 chartWidget, e.g. *chartWidget / welcomePage / ticketsInView*.
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 tableWidgets 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 tableWidget, e.g. *tableWidget / welcomePage / ticketsOverview*.

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.

Fig. 2: ConSol*CM Web Client - Page Customization Subtrees for Web Client Dashboard Layout

14.2.2 Definition of the Overall Dashboard Layout

The overall layout of the entire Web Client Dashboard is defined using the page customization attribute *widgetsGrid / welcomePage*.

Attributes:

- **layout**

  This defines the layout of the entire Dashboard on the welcomePage based on the following principles:

  - 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], a new table widget named *myTickets* will appear in the page customization tree (see 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, i.e 'ticketsInView: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 shown below.
  - *null* is a reserved key word 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 widgets name, afterwards, it can be omitted, e.g. [ticketsInView:Chart, ticketsInView, ticketsInView].
  - 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 figures show the organization of an example grid and its representation in the page customization.

Fig. 3: Organization of an Example Grid for a welcomePage

The value of the respective *layout* attribute would be:

```
[ticketsInView:Chart, ticketsInView, myTickets:Table], [ticketsInView, ticketsInView,
ticketsOverview:Chart]
```

14.2.3 Configuration of Widgets

Configuration Script for Widgets

Each chart widget and each table widget has a configuration script. This script is a Groovy script which is stored in the Admin Tool script section ad is referenced by its name. The scripts has to be of type *Page customization*. Select the widget in the PCDS and enter the script name:

Fig. 4: ConSol*CM Web Client - Script Definition for a Chart Widget

Fig. 5: 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 thus will overrides any widget attribute value set in the customization, so please make sure that the desired property is not set within the script, if you want to use the page customization for property 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 the current engineer has at least read permissions, but will return absolutely 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 the attributes which should be set, as a HashMap (see return statement in the code example above which uses the attributes *series*, *visible*, *chart*, *title*, *tooltip*, and *localization*). A detailed explanation of all attributes and the respective hyperlinks are 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 the attributes which should be set, as a HashMap. Please see section Attributes for Table 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.

```groovy
Standard ConSol*CM Table Widget Script ticketsInViewDataWidget.groovy

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'}"];
```

The following chart is defined by the script above. For a detailed explanation, please refer to the section
Example for a Chart Widget.
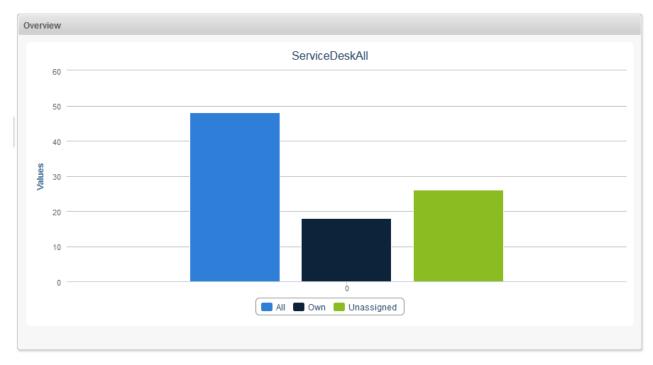
Fig. 6: ConSol*CM Web Client - Example Chart Widget

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 a attribute which is set within the Admin Tool script of a widget always overwrites the respective attribute which has been set as 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 section about layout

- **refreshOnViewChange**

  Indicates whether the Dashboard should be refreshed when the engineer changes the view, i.e. uses the drop-down view list to select another view for the ticket list, default value is *true*

Examples for Visibility Configuration

Example 1: The chart should not be displayed.

```
Visibility Switched off (Set in Admin Tool Script)

return [visible: 'false']
```

Example 2: The chart should only be displayed if the selected view is *service_customer and the* engineer has *the consultant* role.

```
Visibility Depending on Engineer Role (Set within Admin Tool Script)

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}
}
```

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.

```
$("#container").highcharts({
    ▶ chart : { … }
      colors : [ … ]
    ▶ credits : { … }
    ▶ data : { … }
    ▶ drilldown : { … }
    ▶ exporting : { … }
    ▶ labels : { … }
    ▶ legend : { … }
    ▶ loading : { … }
    ▶ navigation : { … }
    ▶ noData : { … }
    ▶ pane : { … }
    ▶ plotOptions : { … }
    ▶ series : [{ … }]
    ▶ subtitle : { … }
    ▶ title : { … }
    ▶ tooltip : { … }
    ▶ xAxis : { … }
    ▶ yAxis : { … }
});
```

Fig. 7: Highcharts Configuration Options

General attributes:

- **localization**
  localized values, i.e -> "de: {subject:'Thema', yes:'Ja'}, en: {subject:'Subject', yes:'Yes'}"
- **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:

```
Example for Chart Object

chart =  "type:'column', pltShadow:false, backgroundColor:'#4dc245', height:
300";"items: [{html:'sometext', style: { left: '100px'; }}]"
```

- **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).

```
Example for Lables Object

labels = "items: [{html:'sometext', style: { left: '100px'; }}]"
```

- **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 hovers over a series or point (http://api.highcharts. com/highcharts#tooltip)

- **visible**
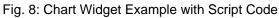
  Indicates whether the widget is shown
- **xAxis**

  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 for 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. Here, the lines of code are set in relation to the GUI elements which they configure.

Fig. 8: Chart Widget Example with Script Code

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:

- **localization**

  localized values, i.e -> "de: {subject:'Thema', yes:'Ja'}, en: {subject:'Subject', yes:'Yes'}"
- **visible**

  defines if the widget is displayed, *true* or *false*.

Datatables-specific attributes:

- **columns**

  Options which you can apply to the columns objects (http://datatables.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'}"

- **options**

  Options (http://datatables.net/reference/option/)

- **visible**

  Indicates whether the widget is shown

Example for a Table Widget

The following demonstrates the basic principle of the implementation of a table widget based on the
Datatables library.

Example Admin Tool Script for a Table Widget

```groovy
// 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
]
```

In the Web Client, the table is displayed as follows (all other widgets have been set to invisible).

First name	Last name	Job title	Expertise level	Hire date
Homer	Simpson	Nuclear disaster	3	25.03.1989
Zaphod	Beeblebrox	President of the Galaxy	0	12.09.1979
Sheldon	Cooper	Mad scientist	321	01.04.2006
Robin	Scherbatsky	Anchorwoman	25	10.09.2004
Elmer	Fudd	Duck hunter	1	15.12.1962
Eric	Cartman	Pupil	10	23.02.1995
Mickey	Mouse	Private investigator	111	04.11.1932
Wilma	Flintstone	Housewife	64	07.01.1964
Charlie	Harper	Composer	12	16.07.2001
Daenerys	Targaryen	Mother of dragons	238	08.05.2010

Showing 1 to 10 of 12 entries   Previous   1   2   Next

Fig. 9: ConSol*CM Web Client - Web Client Dashboard with One Example Table Widget

Example for a Composed Dashboard

The Dashboards looks like this:
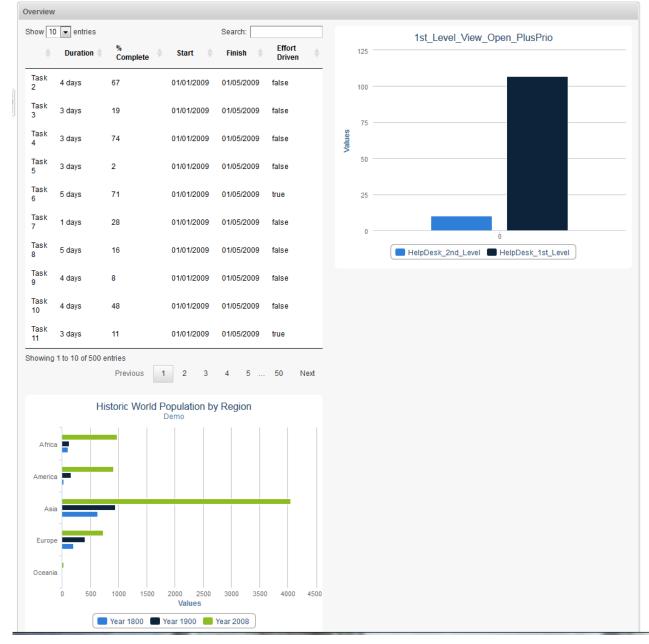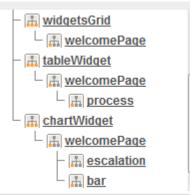
Fig. 10: ConSol*CM Web Client - Example of Composed Dashboard

The page customization attribute *widgetsGrid / welcomePage / layout* is set as follows:

```
Layout Attribute for Composed Dashboard

[process:Table, escalation:Chart], [process:Table, null], [bar:Chart, null]
```

The page customization attributes subtree looks like this:

Fig. 11: ConSol*CM Web Client - Example Page Customization Attributes Subtree

The **table widget** (named *process*) has an associated Admin Tool script:

Fig. 12: ConSol*CM Web Client - Configuration of Example Table Widget

The Admin Tool script (named *table*) for the table widget (named *process*):

```
Example Admin Tool Script Associated with Table Widget

def data = []
(1..500).each { i ->
  data.add("""
  {'title': 'Task ${i+1}',
  'duration': '${Math.round(Math.random() * 5) + 1} days',
  'percentComplete': '${Math.round(Math.random() * 100)}',
  'start': '01/01/2009',
  'finish': '01/05/2009',
  'effortDriven': '${i % 5 == 0}'}
  """)
}
return [
"columns": """[
    {data: 'title'},
    {title: 'Duration', data: 'duration'},
    {title: '% Complete', data: 'percentComplete'},
    {title: 'Start', data: 'start'},
    {title: 'Finish', data: 'finish'},
    {title: 'Effort Driven', data: 'effortDriven'}]""",
"options": """{
    'order': []
    }""",
"data": "[${data.join(",")}]" as String
];
```

The first chart widget (named *escalation*) also has an associated Admin Tool script (named *chart*):

Fig. 13: ConSol*CM Web Client - Configuration of Example Chart Widget

The Admin Tool script for the first chart widget (upper right, named *escalation*):
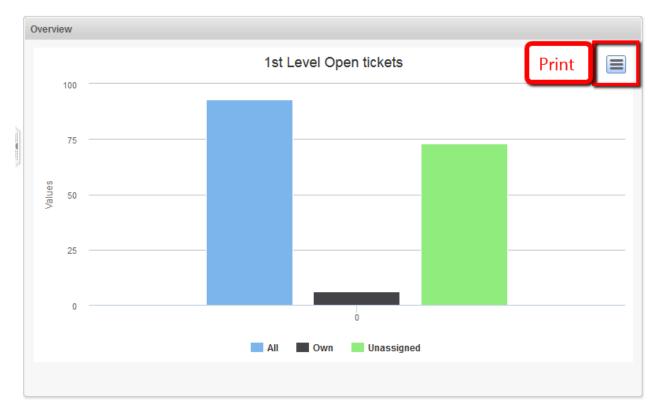
```
Example Admin Tool script Associated with Chart Widget

import com.consol.cmas.common.model.ticket.*;
import java.util.*;
import java.util.Map.Entry;
if (viewId == -1) {
  return [visible: 'false']
}
def view = viewService.getById(viewId);
TicketCriteria crt = new TicketCriteria();
crt.setStatus(view.name.toLowerCase() =~ 'close' ?
    TicketCriteria.Status.CLOSED : TicketCriteria.Status.OPEN);
tickets = ticketService.getByCriteria(crt);
def queues = [:]
tickets.each { ticket ->
  def qname = ticket.queue.name
  if (queues.containsKey(qname)) {
    queues[qname] = queues[qname] + 1
  } else {
    queues[qname] = 1
  }
}
def data = []
queues.each { entry ->
  data.add("{name: '$entry.key', data:[$entry.value]}")
}
return [series: "[${data.join(',')}]" as String, visible: 'true',
    chart: "{type: 'column'}", title: "{text: '${view.name}'}" as String];
```

The second chart widget (lower line, named *bar*) does not have an associated script. It is configured entirely by attributes using the page customization. The following values have been set:

Attribute	Value
chart	type:'bar'
credits	enabled: false
series	[{ name: 'Year 1800', data: [107, 31, 635, 203, 2] }, {name: 'Year 1900', data: [133, 156, 947, 408, 6]}, {name: 'Year 2008', data: [973, 914, 4054, 732, 34] }]

Attribute	Value
subtitle	text:'Demo'
title	text: 'Historic World Population by Region'
visible	true
xAxis	categories: ['Africa', 'America', 'Asia', 'Europe', 'Oceania'], title: {text: null}
yAxis	min: 0, title: {text: 'Population (millions)', align: 'high'}, labels: { overflow: 'justify'}

14.2.4 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 operation system.

Fig. 14: 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*.

14.2.5 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.

> ⚠ **Caution:**
>
> 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!
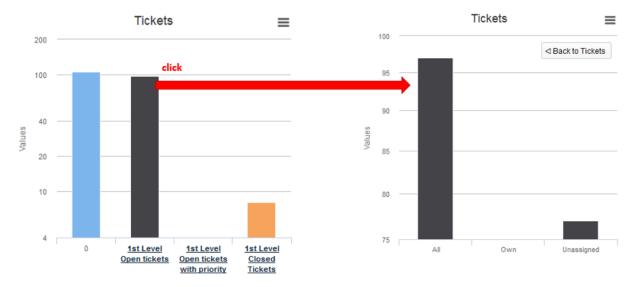
An example script for showing the default chart in a 3D view is listed below. It sets the 3D options in the return value of the script. The additional options in comparison to the standard script are highlighted.

```
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'}"];
```

14.2.6 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 detail view of this subset is shown, illustrated by the right screenshot.

Fig. 15: 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 left 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}"
];
```

15 Labels

- Introduction
- Configuring Labels Using the Admin Tool

15.1 Introduction

Starting with CM version 6.10, 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.

Currently, the configuration is possible for terms which concern the CM.Resource Pool. In future CM versions, more adaptations will be possible.

15.2 Configuring Labels Using the Admin Tool

In the Admin Tool, labels are configured on the navigation item *Labels* in the navigation group *Global Configuration*.

Fig. 1: ConSol*CM Admin Tool - Label Configuration

The following labels are currently available:

Label	GUI location
cmweb.rp.resource.base	Header for Resource Detail Page
cmweb.rp.resource.new	Header on Create Resource form
cmweb.rp.resource.plural	Expression for more than one resources, e.g. on Resource Pool Dashboard in *Search or create Res ources* section or tab on Detailed Search page
cmweb.rp.resource.to	Entry in context menu which leads to resource, default *Jump to resource*
cmweb.rp.resource_pool.base	Entry in main menu
cmweb.rp.resource_relation.base	Expression for resource relation

Label	GUI location
cmweb.rp.resource_type.base	Expression for Resource Type, e.g. on Detail Search page for resources

16 Customer Data Model Section

Customer Data Model (FlexCDM)

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:

- The CM Customer Data Model: FlexCDM
    - Introduction to FlexCDM
    - A Short Introduction to FlexCDM-Specific Web Client Functionalities
    - Setting Up the Customer Data Model
    - Data Object Group Field Management and GUI Design
    - Templates for Customer Data
- Managing Customer Groups
- Customer Roles
- Customer Relations
- Action Framework - Data Object Actions
- Address Autocomplete

17 ConSol CM Administrator Manual 6.10.2 - The CM Customer Data Model - FlexCDM

17.1 The CM Customer Data Model: FlexCDM

Starting with version 6.9, ConSol*CM offers a very flexible and powerful customer administration based on the *FlexCDM*, the Flexible Customer Data Model.

In the following sections, all aspects of the new data model are explained.

- Introduction to FlexCDM
- A Short Introduction to FlexCDM-Specific Web Client Functionalities
- Setting Up the Customer Data Model
- Data Object Group Field Management and GUI Design
- Templates for Customer Data

17.2 Introduction to FlexCDM

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.

17.2.1 FlexCDM at a Glance

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 organitazional 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 model** (where a company can contain several contacts)
- only a *contact* or only a *company* level:
  then we talk about a **one-level customer model**
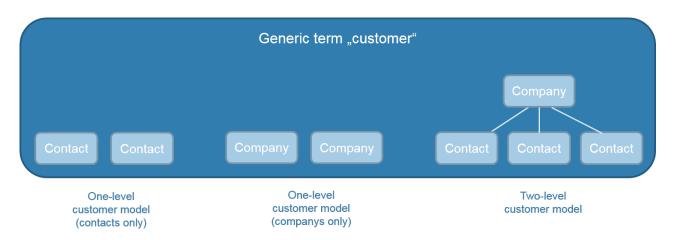
Fig. 1: Types of Customer Data Models in ConSol*CM

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, e-mail, and room number for a contact).
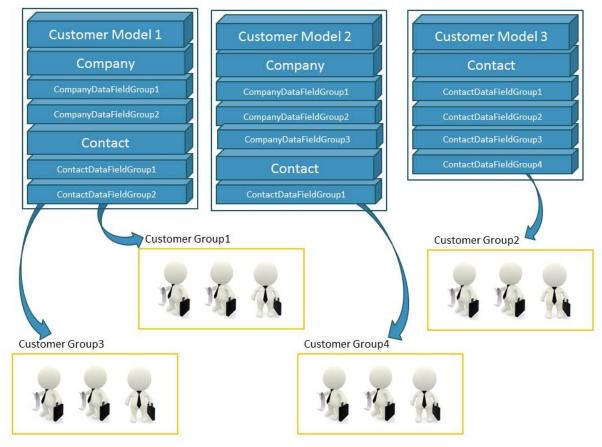
Fig. 2: ConSol*CM FlexCDM - General Principle

> ⓘ **Information:**
>
> **For a two-level customer 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 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.
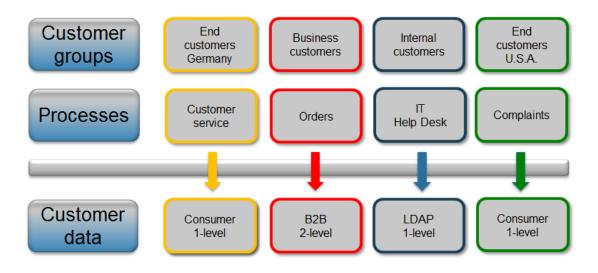
Fig. 3: ConSol*CM FlexCDM - Customer Data Model

Please see section Setting Up the Customer Data Model for a detailed description of the customer management.

17.2.2 Introduction to FlexCDM Objects

In this section, we will give you an overview of all objects which are relevant for the FlexCDM.
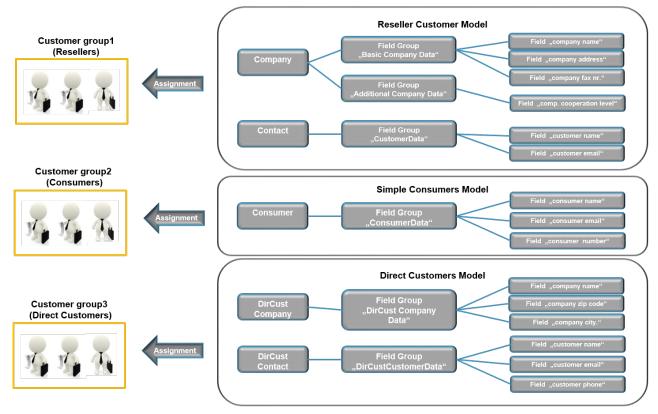
Fig. 4: ConSol*CM FlexCDM - Example Configuration

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**

  Data object of type *company*, company level.

- **Contact**

  Data 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.

- **Data object**

  An object within the customer data model. The object type can be *customer* (often times a person) or *company*. The technical (Groovy) equivalent is an object of class *Unit*.

- **Data object definition**
  All definitions pertaining to the unit. For a company these are e.g. all data object 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!)

- **Data object group**
  A group comprising one or more data field(s) (data object group fields), analog to a custom field group for ticket data. A data object group can be shown or hidden or it can be displayed as a tab in the (new) customer data group section.

- **Data object group field**
  A single data field (types like custom field types) that can contain customer data, analog to custom 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 data object groups and data object group 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.

17.2.3 Management of FlexCDM Objects Using the Admin Tool

In the Admin Tool, the new FlexCDM configuration options are to be found in the navigation group
Customers.
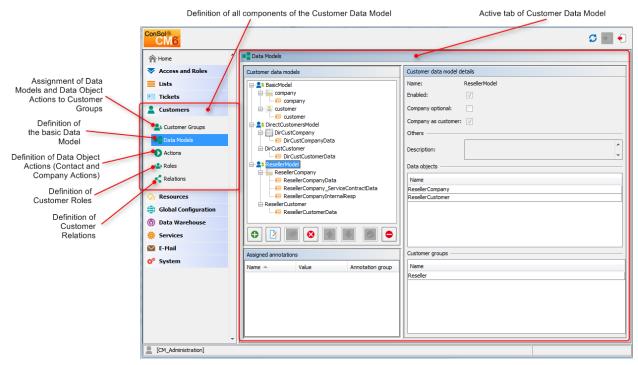
Fig. 5: ConSol*CM Admin Tool - Customer Navigation Items Relevant in FlexCDM

Three tabs are relevant for the definition and management of the customer data model:

- **Customer groups**
  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
    - Data Object Actions
    - Search Actions
    - CM.Phone configuration, if CM.Phone is active
- **Data Models**
  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 GUI Design for details.
- **Actions**
  Definition of Data Object Actions, i.e. contact actions and company actions, please see section
  Action Framework - Data Object Actions for details.
- **Roles**
  Definition of Customer Roles, see section Customer Roles for details.

- **Relations**
  Definition of Data Object Relations, which represent references between customer (i.e. contact and company) objects, please see section Customer (Data Object) Relations for details.
  (Relations from Resources to customers have to be defined in the Resource Pool data model, they cannot be configured here.)

17.3 A Short Introduction to FlexCDM-Specific Web Client Functionalities

- Introduction
- Working with the ConSol*CM Web Client with FlexCDM
  - Example 1: Selecting the Customer Group
  - Example 2: Creating a New Company and Contact
  - Example 3: Using Company and Contact Page
    - Company Page
    - Contact Page
  - Example 4: Setting a Company as Main Customer of a Ticket
  - Example 5: Using Company and Contact Actions
  - Example 6: Setting Relations between Contacts and Companies
  - Example 7: Deactivate a Customer (i.e. a Company or a Contact)
  - Example 8: Using the Ticket Filters on Company or Contact Pages
    - Ticket Filter on Company Page
    - Ticket Filter on Contact Page
  - Example 9: Customer Group Displayed in Quick Search
  - Example 10: Using Search Actions for Company or Contact Data

17.3.1 Introduction

You as an administrator might wonder why a Web Client GUI introduction is provided in an administrator's manual. However, when you want to work with the new customer data model, the *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 for the work with the new customer data model.

All configuration details which are required to understand the system's behavior will be explained in the corresponding sections of the manual.

17.3.2 Working with the ConSol*CM Web Client with FlexCDM

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.

Fig. 1: 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 detail search, the criterion *customer group* is only offered when *All customer groups* has been selected in the drop-down menu. Otherwise the search uses implicitly only data from the selected customer group.
- In the detail 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 for which the selected customer group has been assigned.
- In the ticket list, only views are available which contain tickets from queues for 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 Page Customization, hiddenCustomerGroups for details.

Example 2: Creating a New Company and Contact

In case engineers have access to several customer groups (and have selected *All customer groups* 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. In case 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.

Fig. 2: ConSol*CM Web Client - Creating a New Company within a Customer Group

Example 3: Using Company and Contact Page

Provided there is a two-level customer data model (company and contact) there is **a separate company and a contact page**. 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 which improves the usability.

Company Page

Fig. 3: ConSol*CM Web Client - Top Section of Company Page

Fig. 4: ConSol*CM Web Client - Bottom Section of Company Page

The *Company* page contains the following sections:

- **Company data**

  company data like address, phone number, service data (i.e. the data object group fields you have defined in the Customer Data Model). All company data might be placed in the ticket section or there might be one or more tabs in the *Groups* section.

- **Tickets**

  All tickets for the company are listed in this section. Starting with 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**

  This section shows a list of all contacts belonging to this company. Click on a contact name to open the contact page.

- **Additional details**

  There are two tabs:

  - **Comments**

    Here, all comments concerning this company are listed.

  - **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**

  Here, all relations to and from this company are listed.

- **History**

  In this section, all actions which have been performed with this company object are listed, e.g. the change of a name or any other value of one of the data object group fields.

Please refer to the *ConSol*CM User Manual* for a detailed introduction of how to work with companies.

Contact Page

Fig. 5: ConSol*CM Web Client - Contact Page

These are the sections on the *Contact* page:

- **Contact data**
    - contact data like address, phone number, service data (i.e. the data object group fields you have defined in the Customer Data Model). All contact data might be placed in the ticket section or there might be one or more tabs in the *Groups* section.
- **Tickets**

    All tickets for the contact are listed in this section.
- **Additional details**

    There are two tabs:
    - **Comments**

        Here, all comments concerning this contact are listed.
    - **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**
  Here, all relations to and from this contact are listed, e.g. if the contact has a relation to a certain resource like a laptop to represent an employee's working environment.
- **History**
  In this section, all actions which have been performed with this contact object are listed, e.g. the change of a name or any other value of one of the data object group 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.

> ⚠ **Attention:**
>
> 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 *Additional details* section of tickets:
>
> - Details read
> - Details write
> - Details delete

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.

Fig. 6: ConSol*CM Web Client - Using the Company as Main Customer for a Ticket

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 following figure), or can update the contact data from another database, or create a ticket for the contact (see figure after next).

Automatic actions can be performed when a system action takes place (create/update/delete of a customer). The company and contact actions are based on the *Action Framework*.

Fig. 7: ConSol*CM Web Client - Manual Company Actions

Fig. 8: ConSol*CM Web Client - Manual Contact Action

> ⚠ **Attention:**
>
> 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

Example 6: Setting Relations between Contacts and Companies

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 which improves *Customer Relationship Management* (CRM) functionalities.

In the Web Client, relations between companies and/or contacts are established and displayed similar to ticket relations. In the example, *MyNewSpaceCompany* sells products to the end customer *Mr. Sample*.

Fig. 9: ConSol*CM Web Client - Establishing a Company-Contact Relation

Fig. 10: ConSol*CM Web Client - Display of Company-Contact Relation

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. In case the customer has to be deleted, all of its 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 to deactivate (and reactivate)
companies and contacts, i.e. only then the *Deactivate/Activate* menu items will be displayed in the
Web Client:

- Deactivate/activate

Fig. 11: 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 ticket to another customer.

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 activated the checkbox "include
  deactivated customers").

> ⚠ **Attention:**
>
> **Deactivation in a two-level model**
> When a company (or more generally spoken: an object on 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 *deactivated* option is not selectable.
>
> **Reactivation in a two-level model**
> In case a company is reactivated, the assigned contacts will **not** be reactivated **automatically**. They have to be reactivated **manually**.

Example 8: 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 Filter on Company Page

Fig. 12: ConSol*CM Web Client - Ticket Filter on Company Page

Available options:

- **Closed tickets**

  Closed tickets where the company is the main customer or additional customer.

- **Company tickets**

  Tickets where the company is the main customer.

- **Open tickets**

  Open tickets where the company is the main customer or additional customer.

- **Open tickets of contacts**

  Open tickets of contacts of this company.

Ticket Filter on Contact Page

Fig. 13: ConSol*CM Web Client - Ticket Filter on Contact Page

Available options:

- **Closed tickets**

  Closed tickets where the contact is the main customer or additional customer.

- **Company tickets**

  Tickets where the company of the customer is the main customer or an additional customer.

- **Open tickets**

  Open tickets where the contact is the main customer or additional customer.

- **Own tickets**

  Tickets where the contact is the main customer.

Example 9: 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 data object group> (<localized name of customer group>)
```


Fig. 14: ConSol*CM Web Client - Search Results for Quick Search

Example 10: Using Search Actions for Company or Contact Data

See section Action Framework - Search Actions.

17.4 Setting Up the Customer Data Model

17.4.1 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, the following steps have to be performed:

1. Create a customer data model.
   (This implies you have already decided if 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 data objects within this model**
   A data 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 Data Object Group Fields**
   These are the data fields for the data objects, i.e. either the Data Object Group Fields for company data (e.g. ZIP, address, phone) or the Data Object Group Fields for contact data (e.g. name, forename, e-mail address).

17.4.2 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.
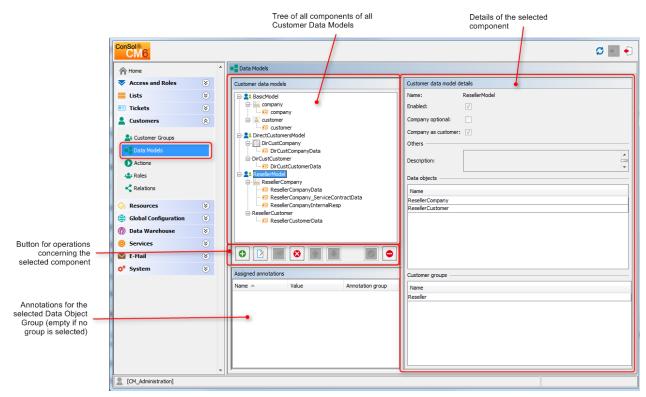
Fig. 1: ConSol*CM Admin Tool - Customer Data Model Definiton

To explain how to work with the customer data model, we will walk you through an example in the next sections.

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 reseller 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 data object (1st level)
- the Data Object Group Fields for the company

- the contact data object (2nd level)
- the Data Object Group 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 Data Object

When you create a new customer data model, you have to add a data object and the respective Data Object Group 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 ⊕ button to open the pop-up window.
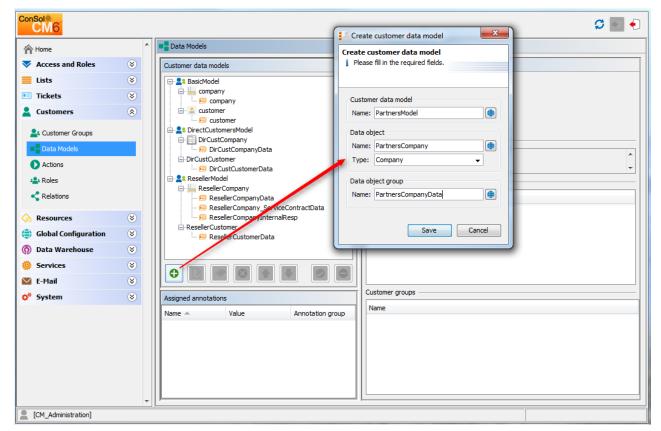
Fig. 2: ConSol*CM Admin Tool - Creating a New Customer Data Model

You have to fill-in the following fields:

- **Customer data model**
    - **Name**
      The name of the new customer data model. As usual in ConSol*CM6, there is one (unique) technical object name. By using the 🌐 button you can open a pop-up window where you can enter the values for the name in various languages.

- **Data object**
  - **Name**

    The unique technical name of the company/contact object, can also be localized using the 🌐
    button.
  - **Type**

    Select *Contact* or *Customer*. There can be only one company object and one contact object
    within one customer data model.
- **Data Object Group**
  - **Name**

    The unique technical name of the first Data Object Group for company data within the defined
    data object. More Data Object Groups can be added later on.

Step 2: Create Another Data Object

In the next step, you have to add the contact object. Select the object *ResellerCompany* (to set the correct
level for the following *Add* operation) and use the ⊕ button to open the pop-up window.
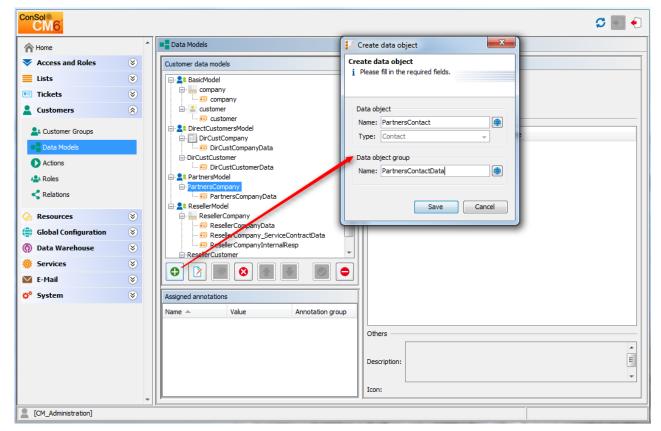
Fig. 3: ConSol*CM Admin Tool - Adding a New Data Object

You have to fill-in the following fields:

- **Data object**
    - **Name**
      The unique technical name of the contact object, can also be localized using the ⊕ button.
    - **Type**
      Here, *Contact* is pre-selected and cannot be modified, because a company object is already present in the customer data model.
- **Data Object Group**
    - **Name**
      The unique technical name of the first Data Object Group for contact data within the defined data object. More Data Object Groups can be added later on.

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 (*ResellerModel* in our example) or mark the customer data model in the list and click on 🖉 to open the pop-up window where you can define the parameters for the model.
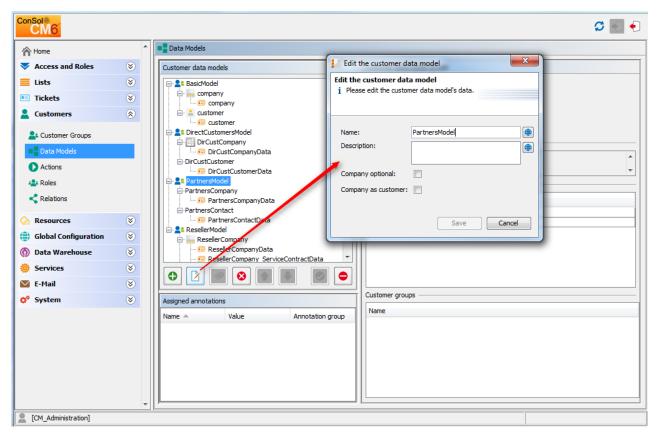
Fig. 4: ConSol*CM Admin Tool - Parameters for a Customer Data Model

You can fill-in the following fields:

- **Name**
  Check or modify the existing name of the model.
- **Description**
  Optional description.
- **Company optional**
  If this check box is marked 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 the system to work with single contacts, even within a two-level customer data model.
- **Company as customer**
  Mark this check box if it should be allowed to create tickets not only for contacts but also for companies within the model.

Parameters for the Data Object

Double-click on the name of a data object, e.g. the *PartnersCompany* or select the object and press the *Edit* button to open the pop-up menu where you can configure the parameters for this object.

Fig. 5: ConSol*CM Admin Tool - Parameters for a Data Object

You can fill-in the following fields:

- **Name**
  The unique technical name of the data object with technical name and localized name(s).

- **Description**

  The description of the data object. Will be used in future ConSol*CM versions.

- **Type**

  A read-only field which displays the type (*contact* or *company*) of the data object.

- **Icon**

  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 from the file system using the file explorer button ![icon] .

- **Templates**

  The templates which are used to render the data of the data 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 Data Object Group

Double-click on the name of the Data Object Group to change the technical and/or localized name(s) of the group.

To define the Data Object Group Fields, use the GUI elements on the right-hand side. In the following figure, an example for the definition of Data Object Group Fields for the *ResellerCompany* is shown. Here, only one Data Object Group (*ResellerCompanyData*) is used. You can use as many Data Object Groups in one data object as you consider suitable for your system.
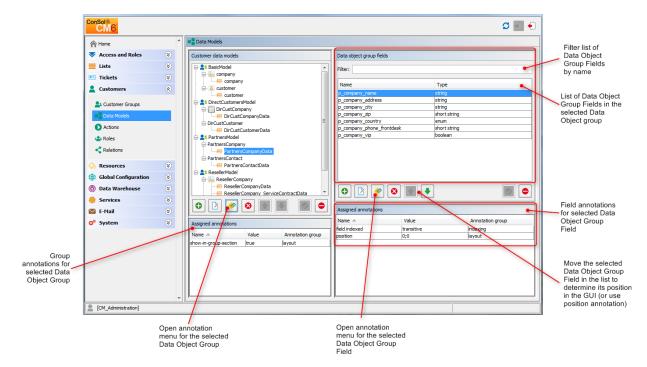
Fig. 6: ConSol*CM Admin Tool - Parameters for the Data Object Group

The definition of Data Object Group Fields within customer data models is based on the same principles as the definition of custom fields for ticket data. For a detailed introduction to the definition and management of custom fields, please refer to sections Custom Field Administration and Data Object Group Field Management and GUI Design.

The available Data Object Group Field annotations are listed in section Appendix A (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*).

> ⚠ **Attention:**
>
> Please make sure that the annotation *field indexed* is set for all fields which should be searchable. This concerns the quick search, the detail search, and all auto-complete operations! See also section Search Configuration and Indexer Management.

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 ahead to assign the model to one or more customer group(s).

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 *Resellers* which will use the new *ResellerModel*.

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.

Fig. 7: ConSol*CM Admin Tool - Definition of a New Customer Group

You can fill-in the following fields:

- **Name**
  The unique technical name (and localized name) of the new customer group.
- **Customer data model**
  Select the desired data model from the drop-down menu.
- **Actions**
  Here, the customer actions can be defined. This will be explained in detail in section Action Framework - Data Object Actions.

For an engineer who has access permissions on four customer groups, the Web Client will look as shown in the following figure.

Fig. 8: ConSol*CM Web Client - Creating a New Company and Customer

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 reseller data sets or to modify them, you have to grant access permissions for the user groups to one or more roles.

See section Permissions on Customer Groups.

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.

17.5 Data Object Group Field Management and GUI Design for Customer Data

17.5.1 Introduction

One feature of ConSol*CM version 6.9 is the great flexibility as far as customer data model (*FlexCDM*) and GUI design are concerned. You as an administrator can define any data field (Data Object Group Field) which is required and place it on the user interface wherever it is suitable. The basic principle is now the same as the one you know for ticket Custom Fields: full flexibility.

The management of the ticket data model and GUI design is explained in section Custom Field Administration. The management of objects within the customer 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 good knowledge of those topics.

17.5.2 Defining Data Object Group Fields for Customer Data Using the Admin Tool

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 data objects within the customer data model are called *Data Object Group Fields*. The work with Data Object Group Fields is based on the same principle as the work with ticket data fields (Custom Fields): data fields are managed in groups and the groups as well as the single fields can be annotated.
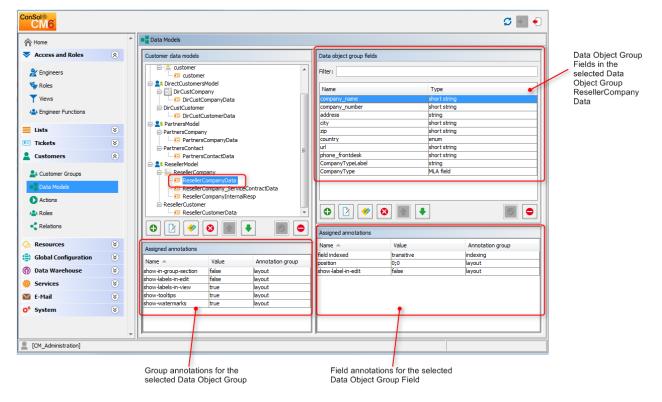
Fig. 1: ConSol*CM Admin Tool - Definition of Data Object Group Fields

Data Object Groups

Like the Custom Fields in previous versions, Data Object Group Fields are placed in groups, the *Data Object Groups*. Each data object within a customer data model can have as many Data Object Groups as required. For example, for a reseller company, there can be a Data Object 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 Data Object Group with general data is defined.

Fig. 2: ConSol*CM Admin Tool - Customer Data Model with Several Data Object 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 Data Object Group ...

- can be faded in and out in the GUI during the process, but only the whole group, not single fields (= Data Object Group Fields) of it.
- can be displayed as tab or in the customer data section. The title (and mouse-over) of the tab is the (localized) name of the Data Object Group.
- is configured by the group annotations.
- is placed on the GUI based on its position in the Data Object Group list (defines e.g. the order of tabs).

Fig. 3: ConSol*CM Web Client - Company Data Organized in Tabs (Based on Data Object Groups)

Data Object Group Field Definition

The definition of Data Object Group Fields (i.e. data fields like *name*, *address*, or *phone number*) is based on the principle which has been used for Custom Fields in previous ConSol*CM versions.

A Data Object Group Field ...

- is defined by a data type.
- is configured using Custom Field annotations (e.g. *position* or *field-indexed*), see Annotations.

> ⓘ   In contrast to Custom 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 defintion of Data Object Group Fields.

Labels for Data Object Group Fields

In CM versions prior to 6.9.4, the labels for Data Object Group Fields (e.g. name, address) are displayed in
the Web Client as watermarks per default. If a distinct label is required, a separate Data Object Group Field
of type *string* with the annotation *text-type = label* has to be used, as shown in the following figures.

Fig. 4: ConSol*CM Web Client - Default Display of Data Object Group Field Labels in Versions Prior to 6.9.4

Fig. 5: ConSol*CM Web Client - Labels as Distinct Data Object Group Fields in Versions Prior to 6.9.4

Now (as of CM version 6.9.4) you as an administrator can decide if watermarks or labels (or both) should be used, labels are now implemented similar to the labels for Custom Fields (i.e. the data fields for ticket data). Furthermore, tool tips can be added. The display modes of labels for Data Object Group Fields is controlled by annotations.Annotations can be set on Data Object Group level and on Data Object Group Field level. The field annotations overwrite the group annotations. In tis way, you can define a group display behavior but can modify one or more single fields. Please see the example which is explained below.

Annotations for Data Object 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 Data Object Group Fields:

- layout:show-label-in-view
- layout:show-label-in-edit
- layout:show-watermark
- layout:show-tooltip

Fig. 6: ConSol*CM Admin Tool - Configuration for Reseller Data Object Group, One Field Configured Field-Specific

Fig. 7: ConSol*CM Web Client - (Edit Mode) View for Reseller Data Object Group, One Field Configured Field-Specific

Fig. 8: ConSol*CM Web Client - (View Mode) View for Reseller Data Object Group, One Field Configured Field-Specific

The labels are inserted automatically, so in the end, there might be six columns as maximum.In the following example, two columns are used.

Fig. 9: Example for Data Object Group 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 model will be preserved during the update! All annotations will be set accordingly and can be modified later.

For all existing Data Object Groups the annotation setting layout:*show-labels-in-edit* = *false* (which will hide standard labels in edit mode) is made during the update.

Default Mode for New Data Object Groups and Data Object Group Fields

All new Data Object Groups will be rendered with the new default configuration. This means:

- standard labels
- watermarks disabled
- tool tip enabled

Fig. 10: ConSol*CM Web Client - Default Display Mode for Data Object Group Fields

17.5.3 Scripting Using Objects from the FlexCDM

> ⚠ In this book we use the terms *data object* and *data object definition*. However, the names of the corresponding Java classes are *Unit* and *UnitDefinition*. All other Java classes which deal with customer data objects also are still named *Unit...* Please keep that in mind when you work on the administrator level as well as on the programmer's level. Please refer to the *ConSol*CM Java API Doc* and to the *ConSol*CM Process Designer Manual* for details.

New Scripting Features Since ConSol*CM Version 6.9

Up to ConSol*CM version 6.8, a *Unit* could have only one Custom Field Group. The expression *unit.get ("name")* was always valid because one Custom Field with a specified name could exist only once, for example *"group1.name"*.

Starting with ConSol*CM version 6.9, a data object (*Unit*) may have one or more Data Object Group Fields with the same field name, e.g *"name"* can be represented as *"group1.name"*, *"group2.name"* . In such cases, the expression *Unit.get("name")* is not valid and throws an exception.

> ⚠ **Attention:**
>
> For backwards compatibility the code *unit.get("name")* will work as long as the Custom Field *"name"* is unique. When another Custom Field with the same name is added, the code *unit.get ("name")* will no longer work!

Now use the following notation to retrieve unit field (Data Object Group Field) data:

- **For one field:**
  ```
  unit.get("group1 :name ")
  ```
- **For numerous fields:**
  ```
  unit.get(" group1:field1.group2:field2 ")
  ```

```
Example to Get Contact Company Name

unit.get("contactFields:companyReference.companyFields:name")
```

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 concerns scripting for ticket and other objects as well as objects from the customer data model, i.e. Units (data objects: objects on contact or company level). Here, the improvements concerning units (data objects) will be explained. 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
  - E-mail
  - Clone
  - Default values
  - Data object action
  - Data object condition
  - Workflow

17.5.4 Changes in Scripting from Consol*CM Version 6.8 to Version 6.9

In ConSol*CM version 6.9, some methods were declared deprecated and have to be replaced by new methods. This is only relevant if you have scripts from version 6.8 or older. In this case, the scripts have to be redesigned using the new methods.

AbstractField

Removed custom value accessors for each Custom Field type.

Removed/changed method	Replacement
StringField.getStringValue()	StringField.getValue()
NumberField.getNumberValue()	NumberField.getValue()

ActivityFormFieldsSet

Removed accessors with plain *FieldDefinition*, use *ActivityFormElement* instead.

Removed/changed method	Replacement
ActivityFormFieldsSet.addFieldDefinition(new FieldDefinition)	ActivityFormFieldsSet.addElement(new ActivityFormElement(new FieldDefinition()))
ActivityFormFieldsSet.getFields() returns List<FieldDefinition>	ActivityFormFieldsSet.getElements() returns List<ActivityFormElement>
ActivityFormFieldsSet.setFields (List<FieldDefinition>)	ActivityFormFieldsSet.setElements (List<ActivityFormElement>)
ActivityFormFieldsSet.removeFieldDefinition (FieldDefinition)	ActivityFormFieldsSet.removeElement (ActivityFormElement)
ActivityFormFieldsSet.removeAllFieldDefinitions()	ActivityFormFieldsSet.removeAllElements()
ActivityFormFieldsSet.getFieldDefinition(index) returns FieldDefinition	ActivityFormFieldsSet.getElement(index) returns ActivityFormElement
ActivityFormFieldsSet.addFieldDefinition(new FieldDefinition, index)	ActivityFormFieldsSet.setElements(ordered list of elements)

ContentFile

Added size parameter to input stream methods.

Removed/changed method	Replacement
new ContentFile(filename, inputstream)	new ContentFile(filename, inputstream, streamsize)
ContentFile.setInputStream(inputstream)	ContentFile.setInputStream(inputstream, streamsize)

ContentResource

Same changes as in *ContentFile*.

Removed/changed method	Replacement
new ContentResource(filename, inputstream)	new ContentResource(filename, inputstream, streamsize)

Removed/changed method	Replacement
ContentResource.setInputStream(inputstream)	ContentResource.setInputStream(inputstream, streamsize)

FieldLogEntry

Removed modification accessors.

Removed/changed method	Replacement
FieldLogEntry.setModification(Modification)	FieldLogEntry.setValue(value) + FieldLogEntry. setPreviousValue(value)
FieldLogEntry.getModification()	FieldLogEntry.getValue() + FieldLogEntry. getPreviousValue()

Ticket

Removed renamed Custom Fields accessors and other changes.

Removed/changed method	Replacement
Ticket.getField(), Ticket.setFieldValue(), Ticket. removeField()	Previously mixing *groupName* and *fieldName* parameters worked, now only the order *groupName, fieldName* is accepted.
Ticket.setField(AbstractField)	Ticket.addField(AbstractField)
Ticket.addOrUpdateField(AbstractField)	Ticket.setFieldValue(pGroupName, pFieldName, Object pValue)
Ticket.getEnumValue	EnumValue enumValue = getFieldValue(String pGroupName, String pFieldName) String enumName = enumValue.getName();

Removed/changed method	Replacement
Ticket.setEnumValue(fieldName, groupName, enumName)	EnumValue enumValue = enumService. getEnumValue(enumGroupName, enumValueName); Ticket.setFieldValue(pGroupName, pFieldName, enumValue); For workflow usage: Ticket.setFieldValue(pGroupName, pFieldName, getEnumValueByName(enumGroupName, enumValueName));

Removed/changed method	Replacement

TimerTrigger

Removed *setDuedate* method.

Removed/changed method	Replacement
TimerTrigger.setDuedate	TimerTrigger.setDueTime

Unit

Removed renamed Custom Fields accessors.

Removed/changed method	Replacement
Unit.getFieldsSet()	Unit.getFields()
Unit.setFieldsMap(Map)	Unit.addFields(Set)
Unit.setField(AbstractField)	Unit.addField(AbstractField)

New (Convenience) Methods

Examples for new 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.

Example 1: Search for the Tickets of a Contact or of a Company

```
TicketCriteria ticketCriteria = new TicketCriteria();
Unit patternContact = new Unit("contact", customerGroup);
mdcmCriteriaBuilder.setReferencedContactCriteria(ticketCriteria, patternContact);
```

Example 2: Search for the Tickets of the Contact Who is Member of a Certain 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);
```

Example 3: Search for Contacts of a Certain Company

```
UnitCriteria unitCriteria = new UnitCriteria();Unit c
ompanyPattern = new Unit("company", customerGroup);
mdcmCriteriaBuilder.setReferencedCompanyCriteria(unitCriteria, companyPattern);
```

ⓘ **Information:**

For detailed information about the methods, including input parameters (method signatures) and output data type, please refer to the *ConSol*CM Java API Doc*.

17.5.5 New Objects in ConSol*CM 6.9 and Up

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:

Starting point	Script	Objects	Example
Company page	data object action script	unit represents the company	def contacts = unit.get ("contacts()")
Contact page	data object action script	unit represents the contact	List tickets = unit.get ("tickets()")
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()

17.6 Templates for Customer Data

- Introduction to Using Templates for the Display of Customer Data
- Coding Templates
  - Examples for Templates
- Template Types
  - Standard (Default)
  - REST
  - Dragged
  - E-Mail
  - Quick Search
  - Data Object Search Result
  - Ticket Search Result
  - Ticket Page
  - Ticket List
  - Ticket Relation
  - Workspace and Favorites
  - History
  - Suggestion
  - CM.Phone Customer Details
  - CM.Phone Customer List

17.6.1 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 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 (very similar to the principle used in previous ConSol*CM versions):

- An annotation is assigned to a Data Object, i.e. to a contact or company definition, in the navigation item *Data Model* in the navigation group *Customers* in the Admin Tool. It defines for which section of the Web Client the annotation is valid.
- The referenced template must be the name of a template which is stored in the *Script and Template* section of the Admin Tool. You as an administrator are free to assign names to the templates. All you have to make sure is that the referenced name in the *Edit Data Object* window and the template name in the *Script and Template* section are identical.

Fig. 1: ConSol*CM Admin Tool - Template Annotations for Data Object (Here: Company)

In the following paragraphs, the syntax and coding for templates and all possible template types are explained.

17.6.2 Coding Templates

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. **Name of the Data Object**
    i.e. the name of a company or contact object.
2. **Name of the Data Object group**
    within the Data Object (a Data Object group is similar to a custom field group for ticket data).
3. **Name of the Data Object group fields**
    within the Data Object group.

See the following figure for an example:
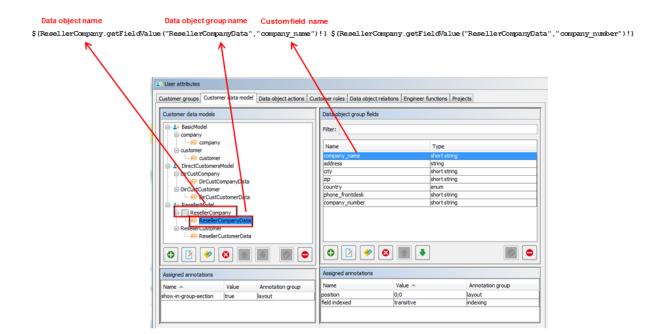
Fig. 2: ConSol*CM Admin Tool - Writing Customer Templates

> ⚠️ **Attention:**
>
> The customer templates must be single-row! They must not contain line-breaks!

Examples for Templates

Here are some examples for templates:

```
Example for engineer description template name (has to be written in one line!)

<#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>
```

```
Example for company-contact-template (has to be written in one line!)

${company.getFieldValue("company", "name1")!}${company.getFieldValue("company", "name2")!}
${company.getFieldValue("company", "mainaddr_city")!},${customer.getFieldValue("customer", "firs
tname")!}
${customer.getFieldValue("customer", "name")!}
```

```
Example for search-customer-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", "n
ame")!}
```

```
Setting number format: removing

<#setting number_format="#"/>${customerModelCompany.getFieldValue("groupName", "numberValueField
")!}
```

17.6.3 Template Types

Standard (Default)

This template, respectively format, is used at all following locations if no special templates have been defined, i.e. all other annotations could be omitted if a standard template is defined.

> ⚠ **Attention:**
>
> 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!

REST

The display of customer data 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 will only be effective 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 which is formatted using the REST template. In the example, the company name ("ConSol*") and the company number ("4711") are part of the template.

```
Example REST template

${ResellerCompany.getFieldValue("ResellerCompanyData","company_name")!} ${ResellerCompany.
getFieldValue("ResellerCompanyData","company_number")!}
```

Fig. 3: REST API - Request for All Units

Fig. 4: REST API - Request for One Unit by ID

Dragged

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.

Fig. 5: ConSol*CM Web Client - Customer Dragged Template

E-Mail

In the Ticket E-Mail Editor an automatic search provides search results in-line in form of a drop-down list. The format of those search results can be configured using this template.

Fig. 6: ConSol*CM Web Client - Customer E-Mail Template

```
E-mail template (has to be written in one line!)

<#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>
```

Quick Search

This template defines the format of the search result for contacts or companies in the quick search. The template has to be short and single-line.
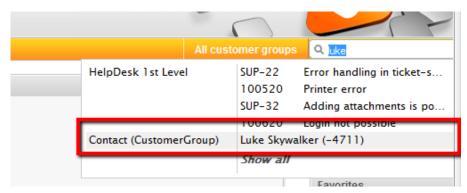
Fig. 7: ConSol*CM Web Client - Customer Quick Search Template

Data Object Search Result

This template defines the search results for automatic searches in auto-complete fields.

Fig. 8: ConSol*CM Web Client - Customer Data Object Search Result Template

This is the applied template:

```
ResellerCompany search result template (has to be written in one line!)

${ResellerCompany.getFieldValue("ResellerCompanyData","company_name")!}
${ResellerCompany.getFieldValue("ResellerCompanyData","company_number")!}
```

Ticket Search Result

In the results page of the detail 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.

Fig. 9: ConSol*CM Web Client - Customer Ticket Search Result Template

Ticket Page

This template defines the presentation of customer data in the *Customers* section of a ticket. The template defines the appearance of the customer data only for the minimum display level 🔲 .

In the medium and large level, the first line (positions 0;x) of the Data Object Group Fields definitions are displayed.

Fig. 10: ConSol*CM Web Client - Customer Ticket Page Template

Ticket List

This template defines the presentation of the customer data in the ticket list.

> ⚠ **Attention:**
>
> If you would like to work with this template type, please make sure that the page customization parameter *accordionTicketList.mainCustomerDescriptionVisible* has been set to *true*. Otherwise customer data cannot be displayed in the ticket list.

Fig. 11: ConSol*CM Web Client - Customer Ticket List Template

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*.

Fig. 12: ConSol*CM Web Client - Customer Ticket Relation Template

Workspace and Favorites

This template defines the presentation of customer data in the *Favorites* section.

Fig. 13: ConSol*CM Web Client - Customer Favorites Template

History

This template defines the presentation of customer data in the ticket protocol, i.e. in the *History* section of a ticket.

Fig. 14: ConSol*CM Web Client - Customer History Template

Suggestion

This template defines the presentation of customer data for suggestions which are displayed when a ticket is created.

Fig. 15: ConSol*CM Web Client - Customer Suggestion Template

CM.Phone Customer Details

See section CTI with ConSol*CM: CM.Phone.

CM.Phone Customer List

See section CTI with ConSol*CM: CM.Phone.

18 Managing Customer Groups

- Basic Principle for Customer Data Models and Customer Groups
- Managing Customer Groups Using the Admin Tool
    - Customer Groups List
    - Customer Group Details
    - Creating a New Customer Group
    - Editing a Customer Group
    - Deleting a Customer Group
    - Disabling and (Re-)Enabling a Customer Group
- Assigning Access Rights for Customer Groups

18.1 Basic Principle for Customer Data Models and Customer Groups

In a ConSol*CM system several 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 three customer groups, each with its specific customer data model.

18.2 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*.
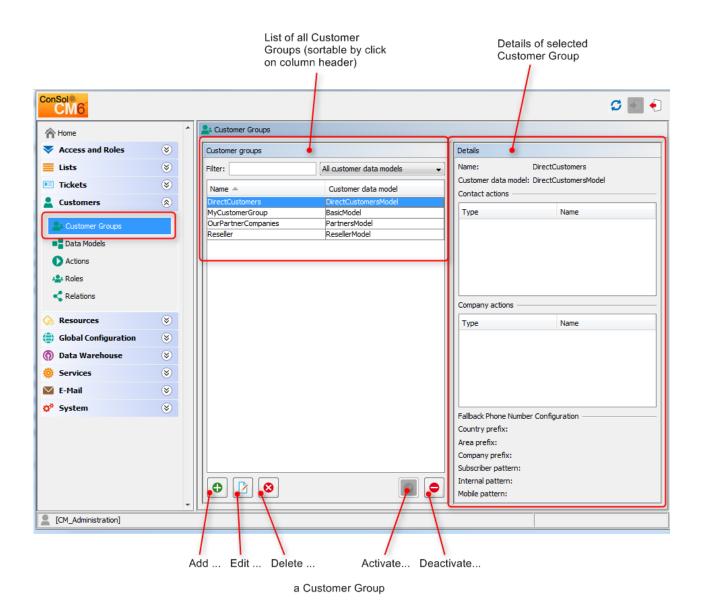
Fig. 1: ConSol*CM Admin Tool - Managing Customer Groups

18.2.1 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.

18.2.2 Customer Group Details

On the right side, the details of the customer group, which is selected in the list, will be displayed. An explanation of all parameters is given in the following section.

18.2.3 Creating a New Customer Group

You create a new customer group by clicking on below the group list. A pop-up window is opened where you have to enter the customer group parameters.

Fig. 2: ConSol*CM Admin Tool - Parameters for a Customer Group

- **Name**

  The technical and unique name of the customer group. Click on 🌐 to enter the localized name of the customer group for all languages that are available in the system. The localized queue name will be displayed in the Web Client in the ticket header. If no localized values are provided, the name will be displayed in the default language.

- **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 Data Object Actions can be assigned to a customer group. All Data Object Actions which have already been defined will be offered in the lists. You can assign those actions for contacts and/or for companies of the customer group. Automatic actions will be triggered when a Data Object is created, edited or deleted. The entire configuration is explained in detail in section Action Framework - Data Object Actions.

- **Manual Actions**
  Manual actions can also be assigned to contacts and/or to companies. All Data Object Actions which have already been defined will be offered in the lists. The actions will be offered 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 Data Object Actions are explained in section Action Framework - Data Object Actions.

- **Search Actions**
  Search Actions are offered as activities for result list of detail 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 Data Object Search Actions (*bulk Data Object Actions*) 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.

- **CM.Phone**
  Tab for all CM.Phone parameters. Only available if CM.Phone is active, see section CTI with ConSol*CM: CM.Phone.

18.2.4 Editing a Customer Group

If you want to edit a customer group, select it in the list and click on ▣ or just double-click the name of the customer group. Modify the customer group parameters and click *Save* to store your modifications.

18.2.5 Deleting a Customer Group

Select the customer group you want to delete in the list and click on ⊗ . If you confirm the following dialog with *Yes*, the customer group will be deleted and is no longer 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.

18.2.6 Disabling and (Re-)Enabling a Customer Group

To disable a customer group, select the customer group in the list and click on ⊖ . The entry in the list is now shown in italics. Just click on ⊘ at the bottom of the page, if you want to enable the customer group again. When 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.

18.3 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.
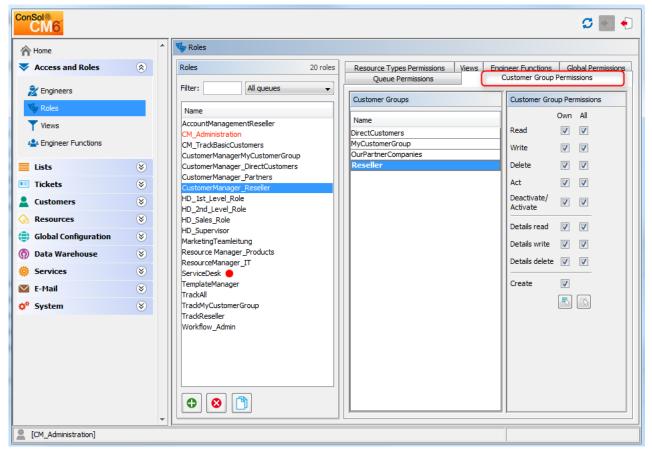
Fig. 3: ConSol*CM Admin Tool - Assigning Permissions on Customer Groups to a Role

The access rights which can be granted have been modified compared to previous ConSol*CM versions. New rights have been added which concern a new section of the customer page. The customer page in the Web Client has a new section, the *Details* section.
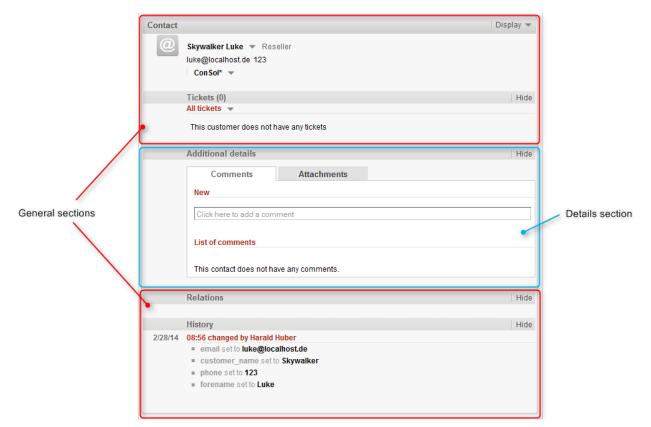
Fig. 4: ConSol*CM Web Client - Contact Page: Details Section

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 additional engineer.
    - **All**
      All customers.
- **General sections**
    - **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 for details about customer actions).
    - **Deactivate/activate**
      Deactivate and (re-)activate the customer. It is not possible to create tickets for a deactivated customer.

- **Details section**
    - **Details read**

      Read customer data in the *Details* section.
    - **Details write**

      Write/modify customer data in the *Details* section.
    - **Details delete**

      Delete customer data in the *Details* 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.

> ⚠ **Attention:**
>
> 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!

19 Customer Roles

19.1 Introduction

Customer Roles help you distinguish different customers who are linked to a ticket. For example, it will help a great deal if the engineer finds a trouble ticket and instantly knows who of the assigned 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. A customer can only have none or one role, not more than one.

Besides helping he engineer in his everyday work, customer roles can also be helpful for designing processes and writing scripts, e.g. write an e-mail to all managers of all open Service Desk tickets of a company to deliver a certain information.

19.2 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.

Fig. 1: ConSol*CM Admin Tool - Customers: 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. Information in the Web Client is provided (e.g. you would not want to send a log file to a *manager* but to the *IT contact*).
2. The customer role can be used in workflow programming to control the process flow (e.g. send an e-mail to all *IT contacts* but not to contacts with other roles).

Fig. 2: ConSol*CM Web Client - Setting a Customer Role for an Additional Customer

19.2.1 Create or Edit a Customer Role

A customer role is defined by its name. By clicking on [+] a pop-up window appears where you can enter the name. Using the globe icon [🌐] next to the name field you can localize the name subsequently (see below). The check box *Enabled* is already selected to set the customer role active in the system (see also Disable or Enable a Customer Role). You will get the same window when you click on [✎] in order to edit a customer role.

Fig. 3: ConSol*CM Admin Tool - Create or Edit a Customer Role

19.2.2 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 you can only disable this customer role (see below).

In order to delete a customer role, select it in the list and click on [✖] . After choosing *Yes* in the confirmation dialog the customer role will be removed from the list and the system.

19.2.3 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 on [⊖] . The entry in the list is shown in italics afterwards. The customer role cannot be assigned anymore. Just click on [✓] 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 check box *Enabled*. When you create a customer role this check box is automatically selected.

19.2.4 Localize a Customer Role

Click on the globe icon [🌐] in the create or edit window to enter the localized name of a customer role. In the pop-up window *Localize* all languages that are available in the system are listed. Enter the customer role name in the *Value* field for each additional language on the right and click *Save*. The localized customer role name, according to the locale of the web browser, will be displayed in the Web Client. If no localized value is found, the default value is displayed. This is the value of the default language. If this has not been defined either, the technical name of the customer group is displayed.

20 Customer (Data Object) Relations

- Introduction to Customer (Data Object) Relations
- Management of Customer Relations Using the Admin Tool
- Creating Customer Relations Using the Web Client
- Scripting Using Relations
- Data Object (Customer) Relations to Resources

20.1 Introduction to Customer (Data Object) Relations

Customer Relations represent relations between customers (Data Objects), 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.
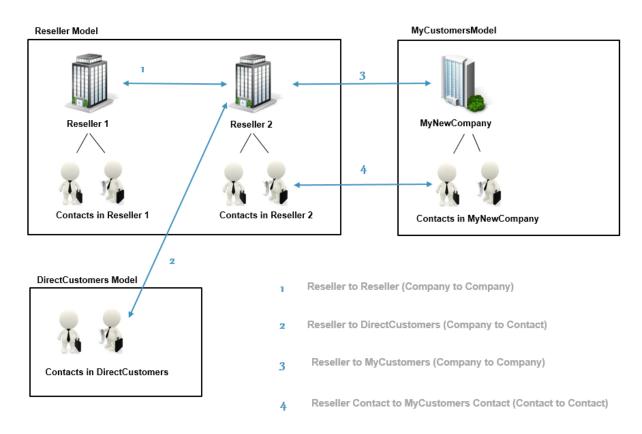
Fig. 1: ConSol*CM FlexCDM - Examples of Customer Relations

20.2 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, or old ones can be deleted.
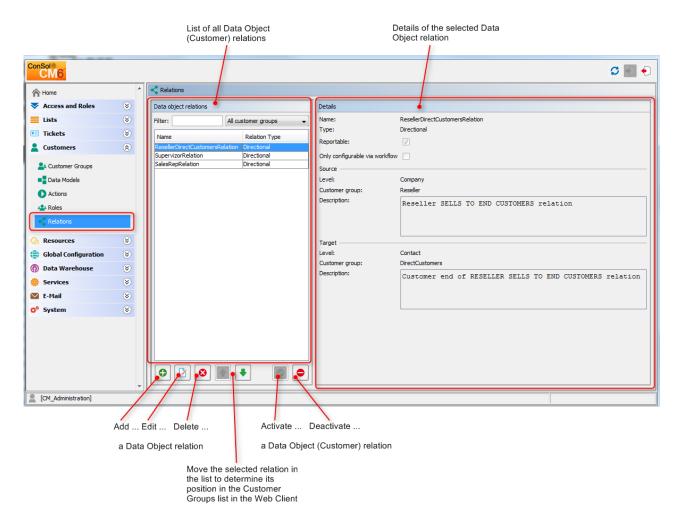
Fig. 2: ConSol*CM Admin Tool - Managing Customer Relations

The following elements are available:

- **List of relations**
- **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.
- **Add button**
    Add a new relation. The pop-up window *Create Data Object relation* with the details fields (see next section) is opened.

- **Edit button** 📝
  Modify the parameters of a relation. The pop-up window *Edit Data Object relation* with the details fields (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.
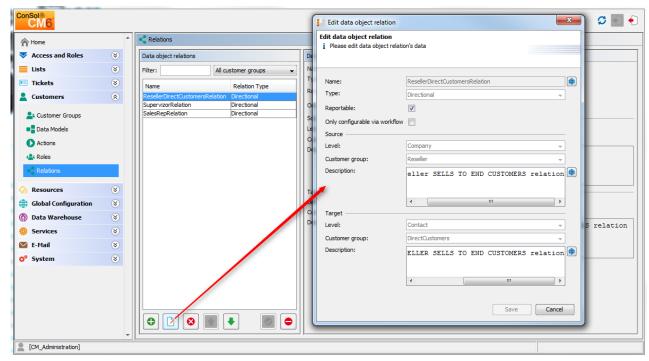
Fig. 3: ConSol*CM Admin Tool - Details of a Customer Relation

To create a new relation, use the ⊕ button, to edit a relation use the 📝 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), the localized name will be displayed in the Web Client as for most fields in ConSol*CM.

- **Type**

  Select one of two types:

  - **Directional**

    A directional relation has a defined source and a defined target. A Data Object 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). Or a relation between two contacts of a company: *is boss of*. The other direction *works for* can also be used. However, a consistent structure for the entire system should be designed to avoid misunderstandings.

  - **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 check box is marked the relation is not available in the Web Client but can only be created via workflow scripts. Therefore 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 Data Object.

    - **Description**

      Will be displayed in the Web Client as 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 Data Object.

    - **Description**

      Will be displayed in the Web Client as description of the relation on the target side.

20.3 Creating Customer Relations Using the Web Client

In spite of this book 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 Data Object to another in the *Relations* section of the source Data Object. 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 Data Object-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 products to* from a company in the *Reseller* customer group to a contact in the *Direct 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 press *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 the work with Customer Relations.

Fig. 4: ConSol*CM Web Client - Setting a Relation

20.4 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 *Data Object* and *Data Object definition*. However, the names of
> the corresponding Java classes are *Unit* and *UnitDefinition*. All other Java classes which deal with
> customer Data Objects also are still named *Unit...* Please keep that in mind when you work on the
> administrator level as well as on the programmer's level. Please refer to the 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.

20.5 Data Object (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 Data Objects. 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.

21 Action Framework - Data Object Actions

- Introduction to Data Object Actions
- Managing Data Object Actions Using the Admin Tool
    - Step 1: Write the Data Object Action Script
    - Step 2: Create Data Object Action(s) Which Use the Script
    - Step 3: Assign Data Object Actions to Customer Groups
- Using Data Object Actions as an Engineer (User)
- Examples for Data Object Action Scripts
    - Example 1: Simple Manual Action
    - Example 2: New Ticket for a Customer
- Scripts for the Action Framework: Programming Data Object Actions
    - Data Object Action Scripts
        - Automatic Data Object Action Scripts
        - Manual Data Object Action Scripts
            - Create a Unit
            - Create a Ticket
            - Open a Unit Page
            - Open a Ticket Page
            - Open a Web Page
    - Data Object Condition Scripts
    - Important Groovy Objects
        - Object UnitActionScriptResult (CM Version 6.9)
        - Object actionScriptResult (CM Version 6.10 and Up)

21.1 Introduction to Data Object Actions

Data Object Actions are components of the ConSol*CM Action Framework. Data Object Actions are sometimes also called *Unit Actions*. They can be performed for a Data Object (unit), i.e. a contact or a company. 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 Data Object Actions for use cases like the following:

- Load additional data into a company's data set.
- Build an automatic report about the company-specific KPIs.
- Transfer ConSol*CM data to another system (e.g. an ERP system).
- Create/update a Google Maps link from the address data.

You can use the following types of Data Object Actions:

- **Automatic** actions which are performed by the system after one of the following Data Object operations:
  - CREATE
  - UPDATE
  - DELETE
- **Manual** actions which are performed by the engineer using *Activities* links in the Data Object page (*company* or *contact* page) of the Web Client (similar to *Workflow activities* for tickets). Manual actions are executed for the Data Object which is 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.

⚠ 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 Data Object Actions, i.e. only then the *Activities* will be displayed in the Web Client:

- Act

Fig. 1: ConSol*CM Web Client - Example for Manual Data Object Activities

Data Object Actions are defined as Groovy scripts which are stored in the *Script and Template* section of the Admin Tool.

The execution of Data Object Actions can be controlled using condition scripts, i.e. you can implement a condition script which is executed before the Data Object Action itself. Only when this script returns *true*, the following action script is executed.

So there are two types of scripts you have to deal with when you use the ConSol*CM Action Framework:

- **Data Object Action scripts**
  Defines the action which should be performed.
- **Data Object 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 use a Data Object Action you have to proceed in three steps:

1. Create Data Object Action script (either action script only or action script and condition script).
2. Create the Data Object Action(s) which use(s) the script(s).
3. Assign the Data Object 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.

21.2 Managing Data Object Actions Using the Admin Tool

> ⚠ In this book we will use the terms *Data Object* and *Data Object definition*. However, the names of the corresponding Java classes are *Unit* and *UnitDefinition*. All other Java classes which deal with customer Data Objects are also still named *Unit...* Please keep that in mind when you work on the administrator level as well as on the programmer's level. Please refer to the the *ConSol*CM Java API Doc* for details.

21.2.1 Step 1: Write the Data Object Action Script

Create a new Admin Tool script of type *Data Object Action*. If required, create another script of type *Data Object 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 Data Object Actions, please read section Scripts for the Action Framework in this chapter.

Fig. 2: ConSol*CM Admin Tool - Scripts for Data Object Actions

Example data object action script for CM version 6.9.4

```
//create and return action result that will tell the web to create 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)
```

Example data object action script for CM version 6.10

```
// 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)
```

21.2.2 Step 2: Create Data Object 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 *Plus* button . The pop-up window is the same for adding and editing a Data Object Action.

Fig. 3: ConSol*CM Admin Tool - Editing a Data Object Actions

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 🌐 .The localization is required for manual actions, because the localized name is displayed under *Activities* in the Web Client.
- **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.
  - **Delete**
    This script will be executed automatically when the contact/company is deleted.
  - **Manual**
    This script will be offered on the contact/company page as manual activity.
- **Condition Script**
  In case a condition script should be executed before the action script, the name of the condition script has to be entered here. Only when the condition script has returned *true*, the action script will be executed. If there is no condition, just leave this field empty.
- **Execution Script**
  The name of the action script which should be executed. This has to be the exact name under which the script is stored in the *Script and Template* 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 customer groups. Please see following step.

21.2.3 Step 3: Assign Data Object Actions to Customer Groups

For the Data Object Action to become effective, you have to assign it to a customer group. Then 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 Data Object 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 on 🖉 to open the pop-up window to assign the Data Object Actions. All Data Object Actions which have been stored under *Actions* (see step 2) will be offered 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 offered as *Update* action.
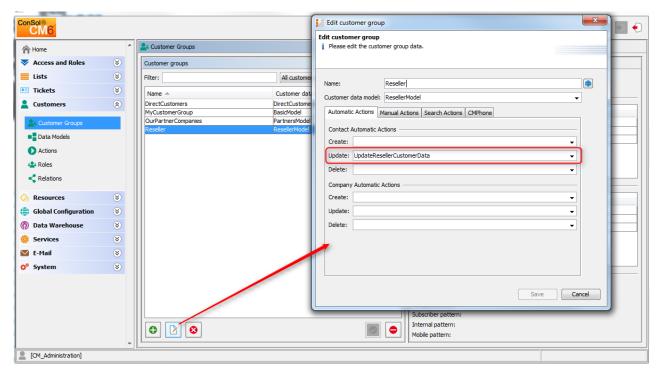
Fig. 4: ConSol*CM Admin Tool - Assigning Data Object Actions to a Customer Group

You can define the following action types:

- **Automatic Actions**
    - **Contact Automatic Actions**

        The script will be executed for the contact. Manual actions are offered on the contact page only.

    - **Company Automatic Actions**

        The script will be executed for the company. Manual actions are offered on the company page only.

        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.
        - **Delete**

            This script will be executed automatically when the contact/company is deleted.
- **Manual Actions**

    This script will be offered on the contact/company page as manual activity.
- **Search Actions**

    Those will be explained in section Action Framework - Search Actions.

21.3 Using Data Object Actions as an Engineer (User)

As an engineer (user), only the Data Object 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.

21.4 Examples for Data Object Action Scripts

21.4.1 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.

Fig. 5: ConSol*CM Admin Tool - Data Object Action Script to Be Used as Company Script

Fig. 6: ConSol*CM Admin Tool - Defining the Company Action

Fig. 7: ConSol*CM Admin Tool - Assigning a Company Action

The engineer can use the action manually in the Web Client.

Fig. 8: ConSol*CM Web Client - Using a Manual Company Action

A script similar to the following could be used:

```
Example Data Object Action script, CM version 6.10

// 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
c
// ... do something, e.g. reference an external system to find the current service status of
the company) ...
// here, the example value "ok" is used, this has to be an enum value in the service_status list
def ser_stat = enumService.getValueByName("service_status","ok")
// set the new service status for the company
unit.setFieldValue("ResellerCompanyData","service_status",ser_stat)
return actionScriptResultFactory.getPostAction(PostActionType.SUCCESS, "action.result.success").
withRefreshContent();
```

21.4.2 Example 2: New Ticket for a Customer

The following script opens the *Create ticket* page for the contact or company from which the action has been performed. The target queue is *ServiceDesk*. That way a new Service Desk ticket can be created in no time for the open contact or cpmpany. For an introduction to Admin Tool scripts for the Action Framework, please read the following section.

```
Example contact script (CM version 6.9.4)

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)
```

```
Example customer action script (CM version 6.10)

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)
```

ⓘ   Please note that, in case a ticket should be created with a company as main customer, the check
    box *Company as customer* must be checked for the customer data model of the respective
    customer group.

21.5 Scripts for the Action Framework: Programming Data Object Actions

Data Object Actions are defined by Admin Tool scripts, i.e. by Groovy scripts which are stored in the *Script and Template* 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:

- Data Object Action scripts
- Data Object Condition scripts

21.5.1 Data Object Action Scripts

The actions in this script are executed either triggered automatically by the system operations *CREATE*, *UPDATE*, or *DELETE* or by a manual action (using *Activities* in the Web Client) of the engineer.

Automatic Data Object Action Scripts

```
Example script: Set a value in customer data and update the unit

unit.set("personalData.name", "Skywalker")
unitService.update(unit)
```

> ⚠️ **Attention:**
>
> When you use *unitService.update(unit)* as in the example above, you can use a Data Object condition script to avoid infinite loops. See note in section *Data Object Condition Scripts*.

Manual Data Object Action Scripts

For manual Data Object Action scripts you can make use of some specific methods and objects.

CM version 6.9:

- **Methods (fields of the Interface PostActionType,** com.consol.cmas.common.model.scripting.unit **)**
  - **CREATE_UNIT**

    Open create unit page.
  - **CREATE_TICKET**

    Open create ticket page.
  - **GOTO_UNIT**

    Open unit detail page.
  - **GOTO_TICKET**

    Open ticket page.
  - **GOTO_PAGE**

    Open a web page (URL).
- **Objects**
  - UnitActionScriptResult

CM versions 6.10 and up:

- **Methods (fields of the Interface PostActionType,** com.consol.cmas.core.server.service.action.
  PostActionType**)**
  - **CREATE_RESOURCE**

    Open create resource page
  - **CREATE_UNIT**

    Open create unit page.
  - **CREATE_TICKET**

    Open create ticket page.
  - **GOTO_RESOURCE**

    Open resource detail page
  - **GOTO_UNIT**

    Open unit detail page.
  - **GOTO_TICKET**

    Open ticket page.
  - **GOTO_PAGE**

    Open a web page (URL).
- **Objects**
  - actionScriptResult

For a detailed explanation of the PostActionTypes in CM versions 6.10 and up, 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 decide for which customer group a new unit has to be
created and optionally the map of Data Object group fields (*PostActionParameter.FIELDS_MAP*) to fill the
unit's Data Object group fields with the passed values.

Example company action script which fills some unit data, CM version 6.9.4

```
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)
```

Example company action script which fills some unit data, CM version 6.10

```
// 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)
```


Fig. 9: ConSol*CM Web Client - Company Action Script

Fig. 10: ConSol*CM Web Client - Create Unit Page Opened and Pre-Filled by Company Action Script

Of course, the names of the Data Object Group 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.

Fig. 11: ConSol*CM Admin Tool - Script for Creating Unit Page as Company Action

Create a Ticket

(*PostActionType.CREATE_TICKET*) redirects the user to a create ticket 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 custom fields map *PostActionParameter.FIELDS_MAP*.

```
Script creates and returns action result that will tell the web to create a new ticket with
unit as the main contact, CM version 6.9.4

// 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)
```

```
Script creates and returns action result that will tell the web to create a new ticket with
unit as the main contact, CM version 6.10

// 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)
```

> ⚠ **Attention:**
>
> Please remember that the customer group for which the script should be applied has to be assigned to the queue where the ticket is to be created (*HelpDesk_1st_Level* in the example).

Fig. 12: ConSol*CM Web Client - Example Contact Action Script

Fig. 13: ConSol*CM Web Client - Create Ticket Page Opened and Pre-Filled by Contact Action Script

Open a Unit Page
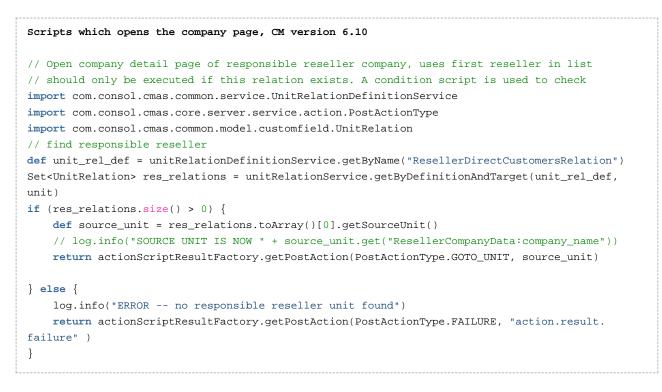
(*PostActionType.GOTO_UNIT*) redirects to a unit page. It uses the obligatory parameter
PostActionParameter.UNIT_ID with the ID of the unit.

```
Scripts which opens the company page, CM version 6.9.4

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)
```

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 have been defined, the first relation in the list is used.

Fig. 14: ConSol*CM Web Client - Data Object Action on Contact Detail Page (Activity Opens Company
Detail Page of Responsible Reseller, CM Version 6.10)

```
Scripts which opens the company page, 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" )
}
```

In case the activity OpenResponsibleResellerPage should only be offered in the Web Client if a relation to the responsible reseller has been set, yo can work with a data object condition script. It has to be ceated as Admin Tool script of type *Data object condition* and has to be assigned to the data object action script in the *Actions* section.
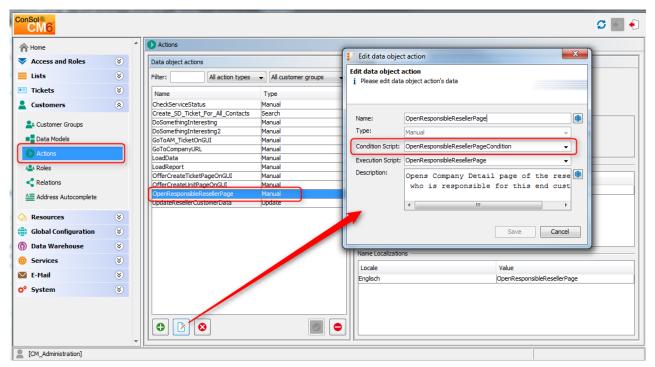
Fig. 15: Admin Tool: Assignment of Data Object Condition

```
Data Object Condition script which checks if reseller relation is present

// 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
}
```

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.

```
Open a ticket page in view mode, CM version 6.9

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
```

```
Open a ticket page in view mode, CM version 6.10

 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
```

> ⚠ **Attention:**
>
> When you use on e of the scripts above, please keep in mind that the ticket search requires that
> the Data Object group fields of the company be indexed (annotation *field-indexed = true*).

Fig. 16: ConSol*CM Web Client - Company Action Available on Company Page (1)

Fig. 17: ConSol*CM Web Client - AM Page Opened after Company Action Go to Ticket

Open a Web Page

(*PostActionType.GOTO_PAGE*) redirects to URL. It uses the obligatory *PostActionParameter.URL* with the URL.

The following code shows a simple example with a fixed URL for each company.

```
Scripts which opens a cerain web site (URL), CM version 6.9.4

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)
```

```
Scripts which opens a cerain web site (URL), CM version 6.10

// 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)
}
```

The string *error.script.no.url* is a label which has been defined in the navigation group Labels, see section
Labels.

Fig. 18: ConSol*CM Web Client - Company Action Available on Company Page (2)

> ⚠ **Attention:**
>
> To open a fixed URL, you can use a Data Object group 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
> Data Object Action scripts is only recommended when a URL is built dynamically within a script.

21.5.2 Data Object Condition Scripts

A Data Object condition script defines whether the action should be shown in the Web Client or not. It is
executed before executing the Data Object Action script. If it returns *false*, the Data Object Action script will
not be executed.

```
Example data object condition script

if(unit.getFieldValue("customer.personalData") == null) {
    return true
} else {
    return false
}
```

To put a data object condition script into operation, you have to permorm the following steps:

1. Write an Admin Tool script of type *data object condition*. The script has to define the conditions for the return values *true* and *false*. See example above. At this point you already have to 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 data object condition script to one or more data object action script(s). This is done in the *Actions* section, navigation group *Customers*.

That's it. You do not have to assign the condition script to a customer group. This happens implicitly when the respective data object action script is assigned to contacts or companies of certain customer groups.

21.5.3 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 having executed 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 (singleto) 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.

> ⚠ **Attention:**
>
> The Data Object Actions *CREATE*, *UPDATE*, and *DELETE* are executed in the core methods *create*, *update*, and *delete* of the object *unitService*.
>
> So if the update action script updates the Data Object using the *unitService.update(Unit)* method then a *java.lang.StackOverflowError* error can be thrown, because the update action will be executed infinitely. In that case a Data Object condition script can be used to avoid such infinite loops.

22 Address Autocomplete

22.1 Introduction

In some ConSol*CM systems it is required that engineers enter or edit a great number of customer data manually. In that case it can be helpful to have system support to

- provide suggestions for the input into some data fields (like e.g. zip or address) to ensure that the entered address really exists
- avoid entering duplicates

To provide those functionalities, ConSol*CM offers the feature *Autocomplete Address*. 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 data object group fields:

- one or more fields which contain the zip code
- one or more fields that contain the city
- one or more fields that 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 Detail 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/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 *Autocomplete Address* 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 CM Web Client to learn how 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 a Autocomplete Strategy Definition, please see section Define the Autocomplete Strategy Using the ConSol*CM Admin Tool for details.

Fig. 1: ConSol*CM Web Client - Suggestions of Autocomplete Address Feature, Example 1

Example 2: The *ZIP* field has already been filled. For the *City* field, only the correct possible values are suggested.

Fig. 2: ConSol*CM Web Client - Suggestions of Autocomplete Address 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.

Fig. 3: ConSol*CM Web Client - Suggestions of Autocomplete Address 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 CM standard feature *Autocomplete Search* works and will display suggestions for the customers which are already part of the CM database.

Fig. 4: ConSol*CM Web Client - Standard Autocomplete / Suggestion Feature in Combination with Autocomplete Address

To configure your ConSol*CM system for this feature, the following steps have to be performed:

1. Switch on the *Autocomplete Addresses* feature using the Admin Tool, see section Switch on the *Autocomplete Address* Feature Using the Admin Tool
2. Import zip/city/address data into the CM database. See section Import Zip/City/Address Data into the CM Database
3. Define the autocomplete strategy using the ConSol*CM Admin Tool, see section Define the Autocomplete Strategy Using the ConSol*CM Admin Tool
4. Refresh the index, see section Refresh the Index

22.2 Switch on the Autocomplete Address 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 CM installation and has to be added manually.

Fig. 5: ConSol*CM Admin Tool - System Property for the Autocomplete Addresses Feature

Restart the Admin Tool. 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 Autocomplete Strategy using the ConSol*CM 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.

Fig. 6: ConSol*CM Admin Tool - Address Autocomplete Tab in User Attributes Section

22.3 Import Zip/City/Address Data into the CM Database

Import the data into the table *cmas_autocomplete_address* in your 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    |                |
+--------+--------------+------+-----+---------+----------------+
```

Fig. 7: ConSol*CM Database - Table cmas_autocomplete_address

22.4 Define the Autocomplete Strategy Using the Admin Tool

To define the autocomplete strategy, you have to perform the following steps:

1. Create one or more Autocomplete Strategy Definitions
2. Configure the behavior for each Autocomplete Strategy Definition

22.4.1 Create One or More Autocomplete Strategy Definitions

An Autocomplete Strategy Definition represents a mapping of a data object group field to one of the key fields in the *cmas_autocomplete_address* table, e.g. you want to define the following definition "When the engineer starts the input in the field *ResellerCompanyData.zip*, the system should search in *ZIP*, and when the engineer starts typing in the data object group field *ResellerCompanyData.city*, the system should search in the city field".

You have to define each of the Autocomplete Strategy Definitions in the Admin Tool in the navigation item Address Autocomplete in the navigation group *Customers*. Add a new Autocomplete Strategy Definition by pressing the plus ⊕ button. Enter the name of the new strategy definition and press *Save*.

Fig. 8: ConSol*CM Admin Tool - Adding a New Autocomplete Strategy Definition

22.4.2 Configure the Behavior for Each Autocomplete Strategy Definition

To configure an Autocomplete Strategy Definition, mark the definition in the list and add one or more Autocomplete Address Fields. These are the mapping rules from the data object group fields to the key fields in the *cmas_autocomplete_address* table.

Fig. 9: ConSol*CM Admin Tool - Definition of Autocomplete Address Fields

The following fields have to be defined for each Autocomplete Address 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 to other fields.
  if switched off: The field will not be an autocomplete field itself but will be filled by dependence 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 react to engineer input itself.
- **On Input:**
  Here you define the number of characters the engineer has to type into the data object group 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 data object group field in the Web Client, leave the configuration field here empty.
- **Cut off:**
  Here you define the number of suggestions in the (drop-down) list.
- **Field Definition:**
  Here you select one data object group field which should be sensitive for the autocomplete input. Only single selection is possible.

In the following example, two Autocomplete Address Fields have been defined for the Autocomplete Address definition *AutocompleteDefinitionZIP1*, see the following figure.

Fig. 10: ConSol*CM Admin Tool - Complete Autocomplete Address Definition

22.4.3 Refresh the Index

When you have entered and configured all required Autocomplete Strategy Definitions, you have to refresh the index. For details about the index, please refer to section Search Configuration and Indexer Management.
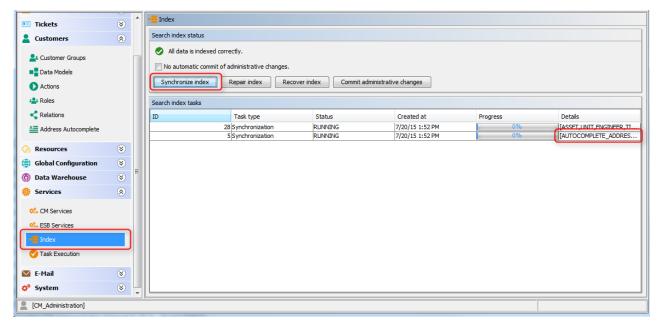
Fig. 11: ConSol*CM Admin Tool - Index Refresh

22.5 Edit an Autocomplete Strategy Definition

In order to edit an existing Autocomplete Strategy Definition, use the edit edit button. You can edit the Autocomplete Strategy Definition and/or each of the Autocomplete Address Fields.

22.6 Delete an Autocomplete Strategy or Autocomplete Address Fields

In order to delete an Autocomplete Strategy Definition or Autocomplete Address Fields, mark the strategy definition or the field in the respective list and press the delete button.

23 CM.Resource Pool Section

CM.Resource Pool

In this section, you learn how to configure all elements of the CM.Resource Pool. In the following chapters, all topics are covered which you might need to configure and operate a ConSol*CM system with the CM. Resource Pool module.

All sections are listed on the CM.Resource Pool Start Page.

24 ConSol CM Administrator Manual 6.10.2 - CM Resource Pool

24.1 CM.Resource Pool

Starting with version 6.10, ConSol*CM offers the Add-on *CM.Resource Pool*. Using this module, you can extend the CM database to store data objects for any type of asset or object in you 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, and 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 CM customization process. To learn about how to work with the module, read the following sections:

For a general introduction to the CM.Resource Pool, read the section CM.Resource Pool - Introduction.

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 give you as an administrator an impression of how the Resource Pool objects are managed by engineers working with the Web Client we have written the section A Short Introduction to CM.Resource Pool Functionalities in the Web Client.

One of the first steps when you start work with the Resource Pool is the set-up of the Resource Data Model. This is covered in section CM.Resource Pool - Setting Up the Basic Resource Model.

Similar to Unit templates, you can define templates which control the appearance of resource data in te Web Client (GUI). 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.

According to 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 to all Resource Pool objects. The Dashboard configuration is explained in section CM.Resource Pool - The Resource Pool Dashboard.

24.2 Introduction to CM.Resource Pool

- Introduction to CM.Resource Pool
- CM.Resource Pool at a Glance
    - CM.Resource Pool Structure
    - Important Terms
- Defining Resource Templates Using the Admin Tool
- Licensing
- Programming with Resource Pool Objects

24.2.1 Introduction to CM.Resource Pool

Starting with CM version 6.10, the Add-on **CM.Resource Pool** is available. The *Resource Pool* provides a database extension which enables 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 Group* (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 the ConSol*CM 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.**

24.2.2 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 Groups. Each Resource Group represents a subtree within the Resource Pool. You can manage any number of Resource Groups, 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 Groups/subtrees.

Relations between resources and other CM objects can mirror various constellations, e.g. a resource of resource group *printer* and of resource type *HP printer* can have a relation to a ticket, because in this ticket, a problem concerning this specific printer has to be solved. Or the printer is related to all contacts which use this printer. Relations between resources are also possible. For a detailed explanation of the work with Resource Relations, see section CM.Resource Pool - Resource Relations.

The data fields are defined for each Resource Type in a way similar to the field definition for ticket data (Custom Fields) and customer data (Data Object Group Fields). Data fields are 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 Unit 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.
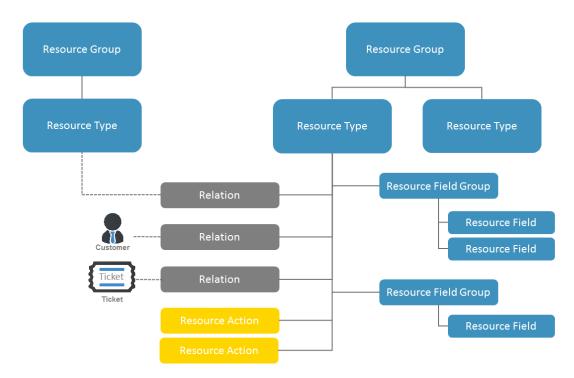
Fig. 1: CM.Resource Pool - Objects in a Resource Model

Important Terms

- A **Resource Group**
  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 Group *Printers*. A Resource Group 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 other 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 Detail Page, a Resource Field group can be displayed as tab, just like the Custom 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 Custom 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. Is is built using the Admin Tool, navigation group *Resources*, tab *Data Models*.

- A **Resource Relation**

  is a relation of a resource to another 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 Type Page like Workflow Activities for Tickets.

- The **Resource Detail Page**

  is the page in the Web Client where all data for one real-world resource is displayed, e.g. the Resource Type 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 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 Groups and Resource Types are displayed according to the access permissions of the current engineer.

24.2.3 Defining Resource Templates Using the Admin Tool

To be able to create real-world resources (e.g. HP printer #4711), a resource template has to be defined first, 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 templates have to be defined using the Admin Tool. This applies to:

- Resource Groups (e.g. Resource Group *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 Groups. 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.
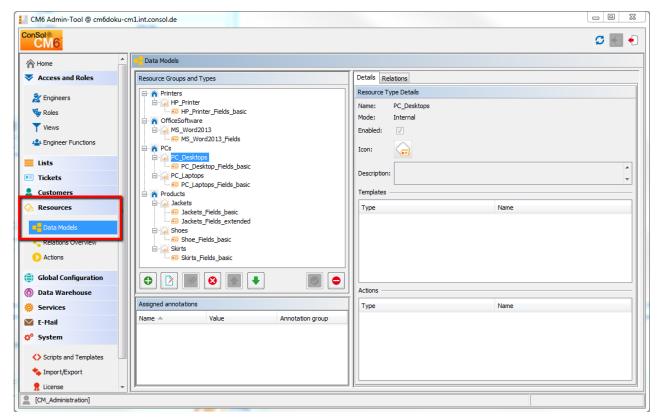
Fig. 2: ConSol*CM Admin Tool - Simple Resource Pool Configuration

Once you have defined the Resources (i.e. the templates for the real resources) using the Admin Tool, the engineers can start work with real resource objects using the Web Client. This will be explained briefly in section A Short Introduction to CM.Resource Pool Functionalities in the Web Client. A detailed explanation is provided in the *Consol*CM User Manual*.

24.2.4 Licensing

CM.Resource Pool has to be licensed separately. Please ask your CM sales representative for more information.

24.2.5 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.

24.3 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***

  In case you use MLAs and/or enums (sorted lists) in Resource Group Fields

- **Navigation group *Resources*** (see following figure)

  - **Data Models**

    Definition of the Resource Data 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
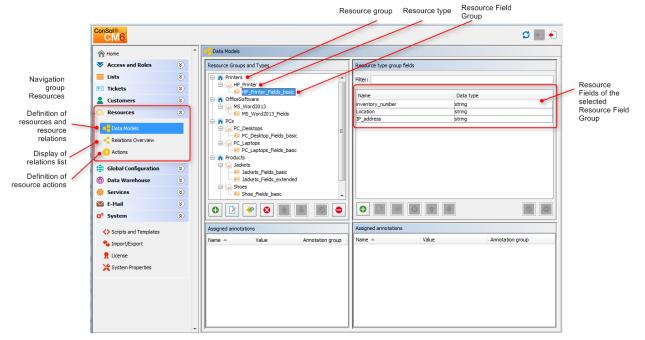
Fig. 1: ConSol*CM Admin Tool - Main Elements for Configuration of CM.Resource Pool
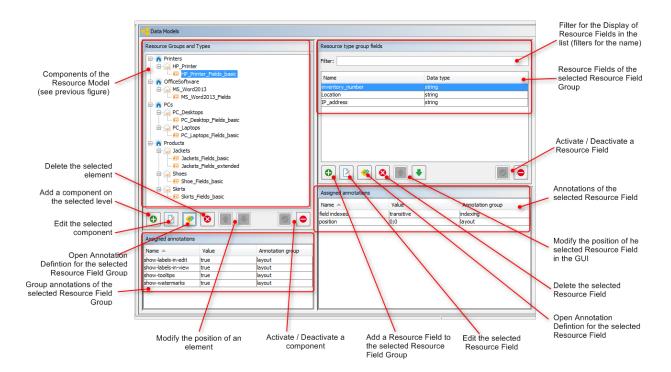
Fig. 2: ConSol*CM Admin Tool - CM.Resource Pool Configuration

24.4 A Short Introduction to CM.Resource Pool Functionalities in the Web Client

- Overview of Resource Pool Operations in the Web Client
- Creating Resources Using the Web Client
  - How You as an Administrator Can Manipulate the Layout of the Web Client Dashboard Section "Search or create Resources"
  - How you as an Administrator Can Manipulate the Layout of the Resource Detail Page
- Working with the Resource Type Page
  - How You as an Administrator Can Manipulate the Layout of the Resource Type Page
- Setting a Resource as Favorite
  - How You as an Administrator Can Manipulate the Display Format of the Resource Concerning Drag-and-Drop to Favorites
- Defining and Using Resource Relations
  - How You as an Administrator Can Manipulate the Display of Resource Relations
- Defining and Using Resource Actions
  - How You as an Administrator can Manipulate the Display of Resource Actions
- Use the Quick Search to Find Resources
  - How You as an Administrator can Manipulate the Display of Resources in the Quick Search
- Use the Detailed Search to Find Resources
  - How You as an Administrator Can Manipulate the Display of Resources in the Detail Search

24.4.1 Overview of Resource Pool Operations in the Web Client

When you as an administrator have defined the Resource Pool templates (Resource Groups and the required Resource Types with their Resource Field Groups and Resource Fields) using the Admin Tool, an engineer can start work with real-world Resource Pool objects, provided he has the required access permissions. Permissions will be treated in section CM.Resource Pool - Assigning Permissions for Resources, in this section here, 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
- Use the Quick Search to Find Resources
- Use the Detailed Search to Find Resources

24.4.2 Creating Resources Using the Web Client

In the Admin Tool, you only define the templates for the objects (in case you are a developer: this is comparable to defining classes in OO programming). The real objects (the instances of the defined templates) 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 templates which have been defined in the Admin Tool (see section CM.Resource Pool - Admin Tool Elements). The engineer who is logged in has access permissions for all Resource Types.

The label for the Resource Pool in the main menu (like 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 section Labels for an explanation how to change the label.

Fig. 1: ConSol*CM Web Client - Resource Pool Dashboard

When the templates 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 show the steps which you should perform.

Fig. 2: ConSol*CM Web Client - Resource Pool Dashboard: Create a New Resource of Type HP_Printer, Step 1

Fig. 3: ConSol*CM Web Client - Create a New Resource of Type HP_Printer, Step 2

Fig. 4: ConSol*CM Web Client - Resource Detail 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 #4711 which is located on the top floor. The Resource Detail 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 is happening when you work with the Admin Tool configuration for the CM Resources.

How You as an Administrator Can Manipulate the Layout of the Web Client Dashboard Section "Search or create Resources"

- The roles of an engineer determine if 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 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 (or better: at least one of his roles) has the access permission CREATE on this Resource Type.

How you as an Administrator Can Manipulate the Layout of the Resource Detail Page

- In the top section of the Resource Detail page, the localized values of the Resource Fields are displayed (analog to ticket data of Custom 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 Custom 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 for each Resource Type.
- The template for the display of the resource name is defined for the Resource Type, here on the Resource Detail 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
- In case Resource Actions are defined for resources of this type (not shown in the figure above), they will be offered on the Resource Detail 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 *Additional Details* might not be displayed if his roles do not have the permission *Details read*, see section CM.Resource Pool - Assigning Permissions for Resources for details.
- Only if relations to other resources, to contacts or companies or to tickets have been defined in the Admin Tool, there might be relations displayed in the respective sections of the Resource Detail page, if real relations have been created for the resource by an engineer. See section CM.Resource Pool - Resource Relations for details.

24.4.3 Working with the Resource Type Page

The Resource Type Page provides an overview over 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.

Fig. 5: ConSol*CM Web Client - Opening a Resource Type Page

Fig. 6: ConSol*CM Web Client - Resource Type Page

On the Resource Type Page, all resources of this type are listed. They can be filtered according to 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 pull-down menu for the search, only fields are offered which have been indexed, see section Search Configuration and Indexer Management for details.
- The column headers represent the localized values of the names of the Resource Fields which you have (or another admin has) defined, see section CM.Resource Pool - Setting Up the Basic Resource Model for details.

24.4.4 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 Detail page.

Fig. 7: ConSol*CM Web Client - Resource Detail 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 in the *Default* template, for details see section CM.Resource Pool - Templates for Resource Data.
- The format for the display of the resource in the drag-and-drop box is based in the *Default* template, for details see section CM.Resource Pool - Templates for Resource Data.

24.4.5 Defining and Using Resource Relations

The same principle as for Resource Groups and Resource Types also applies to Resource Relations:

- First, the Resource Relation templates have to be defined 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 will be explained in detail in section CM.Resource Pool - Resource Relations. Here, a short example is given.

To link a resource (e.g. a PC desktop) to tickets in the queue Service Desk, first, this relation type (i.e. resource-ticket) has to be defined in the Admin Tool.

Then an engineer can create a relation of a PC to a ticket, e.g. when an incident ticket has come in which covers a problem with this machine.

Fig. 8: ConSol*CM Admin Tool - Defining a New Resource Relation
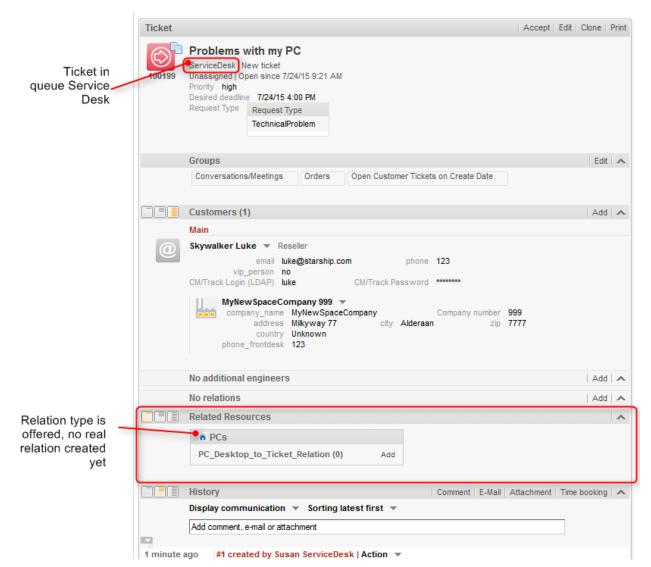
Fig. 9: ConSol*CM Web Client - Ticket Where a Relation to a PC Can Be Created

Fig. 10: ConSol*CM Web Client - Creating a Relation between a Ticket and a PC

Fig. 11: ConSol*CM Web Client - Ticket with Relation to a PC

When the relation has been created successfully, the counter of the relation is set to 1. The engineer can then display details of the relation by clicking on the relation name.

Fig. 12: 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 Detail 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 BOOLEAN field *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 multiplicity of the relation is set to *many-to-one, one-to-many* or *many-to-many*.

24.4.6 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 always is defined on the tab *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 Service Desk, directly from the Resource Detail page. Thus, when a customer calls to complain about a problem with the PC, the Service Desk agent can start by opening the PC Detail page for information about this asset. If necessary, he or she can directly create a new Service Desk ticket for the customer.

Fig. 13: ConSol*CM Admin Tool - Definition of a Resource Action

Fig. 14: ConSol*CM Web Client - Resource Activity (Based on Resource Action)

Fig. 15: ConSol*CM Web Client - Service Desk 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 you as an administrator have 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 for the resource activity in the Web Client.

24.4.7 Use the Quick Search to Find Resources

In the result list of the Quick Search, resources are displayed in a distinct section:

Fig. 16: 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 in 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* will define the maximum number of hits displayed in the result list.

24.4.8 Use 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*).

Fig. 17: ConSol*CM Web Client - Detailed Search for Resources

Fig. 18: 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 the value will be *transitive*.
- The value of the system property *cmweb-server-adapter*,*searchPageSize* defines the number of hits which are initially displayed

24.5 CM.Resource Pool - Setting Up the Basic Resource Model

24.5.1 Introduction

This chapter will guide you through the complete set-up of a resource model. You will learn how to

- create a Resource Group
- create two Resource Types within this Resource Group
- 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 want to manage their IT assets as well as their product 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.
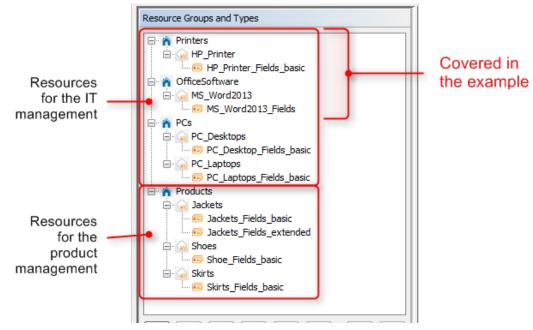
Fig. 1: ConSol*CM Admin Tool - Example resource model

24.5.2 Creating a Resource Group with the First Resource Type

A Resource Group represents the template for objects on the highest hierarchical level in the Resource Pool. In our example, all printers are managed in a Resource Group *Printers* and all office software packages are managed in a Resource Group called *OfficeSoftware*. The following figure shows some Resource Groups and Resource Types in the Web Client. This should help you keep the overview over the configuration in the Admin Tool and its consequences in the GUI. 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.
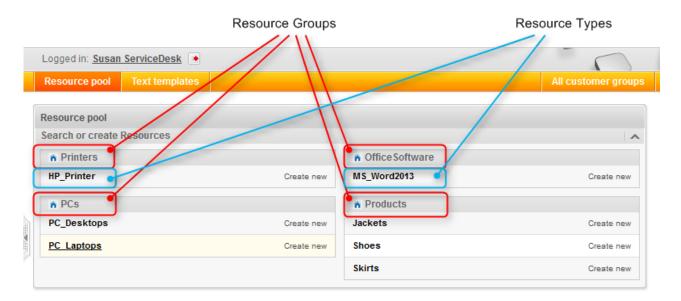
Fig. 1: ConSol*CM Web Client - Resources in the Resource Pool Dashboard

The first step in a new (empty) resource model is to create a new Resource Group which already contains one Resource Type.The Resource Type already contains one Resource Field Group.
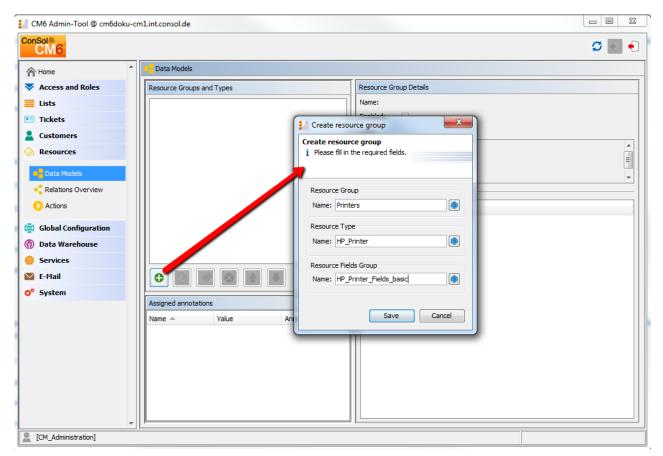
Fig. 2: ConSol*CM Admin Tool - Creating a New Resource Group

To create a Resource Group, enter the values in the following fields:

- **Resource Group**, Name
  The technical name of the Resource Group, it can be localized using the Globe button.
- **Resource Type**, Name
  The technical name of the first Resource Type within the Resource Group. The name can be localized using the Globe button.
  More Resource Types for the Resource Group can be added later on.
- **Resource Field Group**, Name
  The name of the first Resource Field Group (similar to a Custom Field Group for ticket data) within the Resource Type. The name can be localized using the Globe button.
  More Resource Field Groups for the same Resource Type can be added later-on.

Click Save to create the Resource Group, Type and first Field Group.

When you have defined the Resource Group, you can add some more data for the created objects.

Editing the Resource Group

Fig. 3: ConSol*CM Admin Tool - Editing a Resource Group

Mark the Resource Group and press the Edit button to edit some Resource Group parameters:

- **Description**
  You can enter the description directly in the pop-up window, this will become the default description. Or you can use the Globe button to enter localized values for each language . The Resource Group description is displayed as information in the Admin Tool only, not in the Web Client.

- **Icon**
  Select one of the CM standard icons by clicking on the icon or use the file browser to upload an icon of your choice. The Resource Group icon will be displayed in the Web Client next to the name of the Resource Group. In this way, you can, for example, select a Printer icon for printers, a PC icon for PCs etc. Allowed formats are: jpg, jpeg, png, gif.

Fig. 4: ConSol*CM Web Client - Customized Icon for a Resource Group

Click *Save* to commit your changes.

Editing the Resource Type

Fig. 5: ConSol*CM Admin Tool - Editing a Resource Type

Mark the Resource Type and press the Edit button to edit some 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 Globe 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**
  - **Internal**: resource data is fully managed by CM6
  - **On the fly**: resource data are retrieved from an external backend system whenever requested (available only in a later release). The mode "On the fly" relies on a customer specific implementation of the class *ResourceExternalSource*.
  - **Cached**: resource data come from an external backend system and are cached in CM6, a refresh a an item can be triggered manually (available only in a later release). The mode "Cached" relies on a customer specific implementation of the class *ResourceExternalSource*.
  - **Imported**: resource data come from an external backend system and are stored in CM5, an update required a new bulk data import (available only in a later release). The mode "Imported" requires a previous import by ETL

- **Icon**
  Select one of the CM standard icons by clicking on 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, jpeg, png, gif.

- **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 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 Resource Actions for a detailed explanation.

- **Manual 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 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 Search Actions section of the Admin Tool (navigation group *Tickets*). See section Search Actions for a detailed explanation.
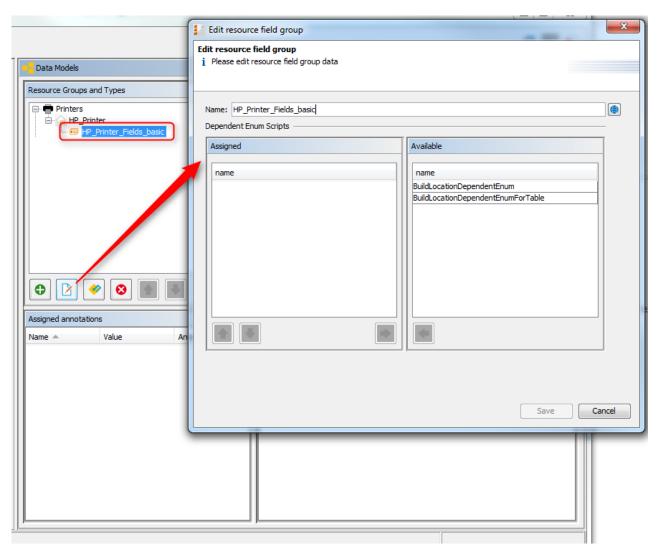
The Resource Relations are also defined for a Resource Type. This is described in detail in the section CM. Resource Pool - Resource Relations.

Editing the Resource Field Group

A Resource Field Group is a collection of Resource Fields, the data fields for resource data (analog to a Custom Field Group which is a collection of Custom 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 press *Edit* to edit those values (see next section).

Editing Resource Field Group-Specific Parameters

Fig. 6: ConSol*CM Admin Tool - 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 this is really required, because it might have side effects for all scripts (workflow and Admin Tool) which use the Resource Group name ⚠

- **Dependent Enum Scripts**
  Here, you can assign a Dependent Enum to the Resource Field Group. The Dependent Enum has to be defined as Admin Tool Script first, see section Admin Tool Scripts, Dependent Enums.

Setting Resource Field Group Annotations

A Resource Field Group can have annotations. Select the Resource Field Group and press the Annotations button to set or unset annotations.The annotations are the same as for Custom Field groups. All annotations are listed and explained in Appendix A (Annotations).
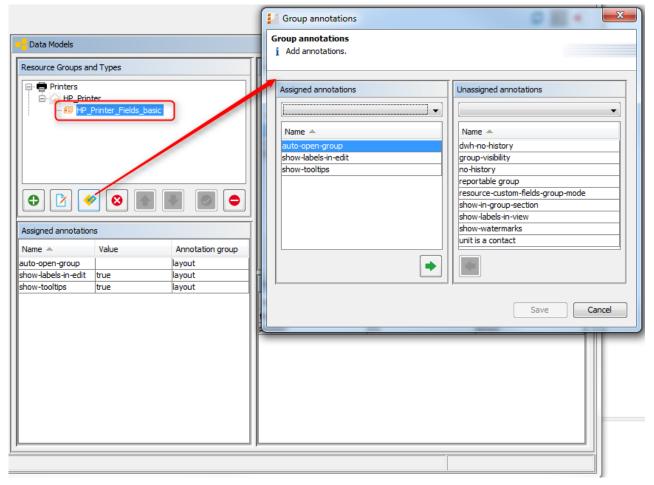
Fig. 7: ConSol*CM Admin Tool - Resource Field Group Annotations

Creating and Editing Resource Fields

Resource Fields are containers for the data of resources (analog to Custom 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 side of the Admin Tool.

Fig. 8: ConSol*CM Admin Tool - Managing Resource Fields

Creating A New Resource Field

To create a new Resource Field, press the Plus button and enter the required data in the pop-up window.

Fig. 9: ConSol*CM Admin Tool - Creating a New Resource Field

Enter the following data:

- **Name**
  The technical name of the Resource Field. This is used in scripts and templates. Set the localized
  value by using the Globe button.
- **Data type**
  The data type of the Resource Field. You can select a data type from a pull-down list, like for Custom
  Fields. A detailed explanation of all available data types is provided in section Custom Field
  Administration (Setting Up the Ticket Data Model), Data type.

Setting Resource Field Annotations

The characteristics of a Resource Field, e.g. the visibility, the position in the Web Client GUI, if the field
should be indexed or if it should be reportable, are defined using annotations. You might know this principle
from your work with Custom Field annotations.

Fig. 10: ConSol*CM Admin Tool - Defining Resource Field Annotations

The annotations which are available are the same as for Custom Fields and are listed and explained in detail in Appendix A (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 Detail Page which is explained in section A Short Introduction to CM.Resource Pool Functionalities in the Web Client, Resource Detail Page.

As for ticket layout you can place the Resource Fields on 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

Fig. 11: Example Matrix for Position Annotation

> ℹ️ In contrast to Custom 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 Resource Fields.

24.5.3 Creating Further Resource Types within a Resource Group

To add a new Resource Type within a Resource Group, mark another Resource Type of the desired Resource Group in the list and press the Add button. Then fill in all required data as described in the section Editing the Resource Type.

24.5.4 Create 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 press the *Add* button.

Then enter all the required data as described in the section Editing the Resource Field Group.

24.5.5 Creating Further Resource Groups

To create a new Resource Type (OfficeSoftware in our example), select another Resource Type in the list and click the Add button.

Enter all the data as for the first Resource Type, see section Editing the Resource Group.

Fig. 12: ConSol*CM Admin Tool - Adding a New Resource Group

To fill the new Resource Group with Resource Types and Resource Fields, please proceed as described in the previous sections.

24.5.6 Example for a Complete Resource Model

In our example, the following resource model has been created. It will be used in the further course of this Manual.
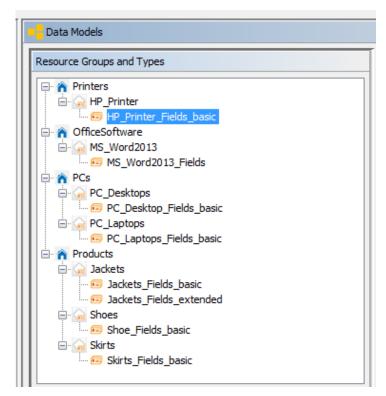
Fig. 13: ConSol*CM Admin Tool - Example for a Resource Model

Fig. 14: ConSol*CM Web Client - Example for a Resource Model (in Resource Pool Dashboard)

24.5.7 Defining Resource Relations

This is explained in detail in the section Resource Relations.

24.5.8 Defining Resource Actions

This is explained in detail in the section Resource Actions.

24.5.9 Assigning Access Permissions for Resources to Engineers Using Roles

This is explained in detail in the section Role Administration, Resource Types Permissions.

24.6 Templates for Resource Data

- Introduction to Using Templates for the Display of Resource Data
- Creating and Editing Resource Data Format Templates
- Coding Resource Data Format Templates
    - General Principle
    - Examples for Templates
- Template Types
    - Default
    - Quick Search
    - Search
    - Rest
    - History
- Web Client Locations for the Use of Templates
    - Header of the Resource Detail Page
    - Dragged Resource in Drag-and-Drop Operations
    - Favorites
    - Quick Search
    - Search
    - History Sections of Resource Detail Page, Customer Detail Page and Ticket
    - REST

24.6.1 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_Printers* are a Resource Type, i. e. the templates will be valid for all real-world HP printers which are stored in the CM system.

The configuration is based on the following principle (very similar to the principle used for the display of contact 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 is that the referenced template name in the Resource Type definition and the template name in the *Scripts and Templates* section are identical.

24.6.2 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.Choose the names as you like or according to the naming conventions of your company.

Fig. 1: ConSol*CM Admin Tool - 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. For details, please refer to the following section.

24.6.3 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!
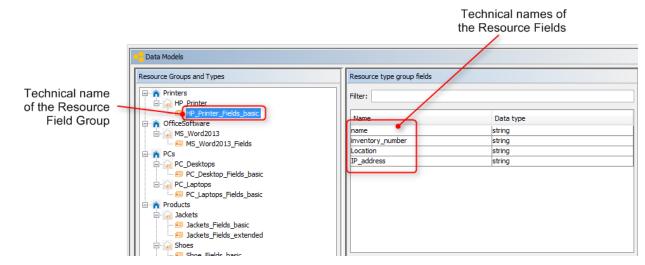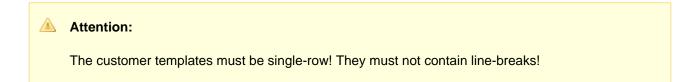
Fig. 2: ConSol*CM Admin Tool - Objects Used in Resource Data Template Definition

> ⚠ **Attention:**
>
> The customer 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("vehicleProp
erties", "vehicleLicense") + ')'}
```

```
// Example 2
// // Resource Field group is named publicTransportProperties, Resource Fields is named passName
${resource.get("publicTransportProperties", "passName")}
```

24.6.4 Template Types

The following Template types can be defined:

- Default
- Quick Search
- Search
- REST
- History

Default

This template always has to be defined. If it is not defined, 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 if no other templates have been defined. The templates which are described in the subsequent sections will overwrite the default template for the specific GUI location.

Example:

```
HP Printer: ${resource.getFieldValue("HP_Printer_Fields_basic","inventory_number")!}
```

The template is used for

- the header of the Resource Type Page
- the dragged resource in drag-and-drop operations
- the display of the resource in the 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 customer data are displayed in the 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 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.

24.6.5 Web Client Locations for the Use of Templates

The following locations are in use:

- the header of the Resource Detail Page
- the dragged resource in drag-and-drop operations
- the display of the resource in the Favorites
- the results of the Quick Search
- the results of the search in suggestion lists
- the REST API
- History sections

Header of the Resource Detail Page

Fig. 3: ConSol*CM Web Client - Header of the Resource Type Page

Used template: *Default* (see example above)

Dragged Resource in Drag-and-Drop Operations

Fig. 4: ConSol*CM Web Client - Dragged Resource in Drag-and-Drop Operations

Used template: *Default* (see example above)

Favorites

Fig. 5: ConSol*CM Web Client - Resource Name in Favorites

Used template: *Default* (see example above)

Quick Search

Fig. 6: ConSol*CM Web Client - Results for Quick Search (Resource)

Used template: *Quick Search*, if this is defined. If not *Default*.

Example template used here:

```
HP Printer: ${resource.getFieldValue("HP_Printer_Fields_basic","name")!} - ${resource.
getFieldValue("HP_Printer_Fields_basic","inventory_number")!}
```

Search

Fig. 7: ConSol*CM Web Client - Resource in Suggestion List, Formatted Based on Search template

Used template: *Search*, if this is defined. If not: *Default*.

Example template used here:

```
HP Printer: ${resource.getFieldValue("HP_Printer_Fields_basic","name")!} - - ${resource.
getFieldValue("HP_Printer_Fields_basic","IP_address")!}
```

History Sections of Resource Detail Page, Customer Detail Page and Ticket

The history template for a resource will be displayed, for example, in the ticket history for the extendede level, when Show all entries has been selected.

Fig. 8: 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 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 Printer") and the inventory number ("4711") are part of the template.

Fig. 9: REST API - List of All Resources

Fig. 10: REST API - Details of One Selected Resource (ID 2)

Example template used here:

```
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*.

24.7 CM.Resource Pool - Resource Relations

24.7.1 Introduction to Resource Relations

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 one or more relation to the following types of 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)
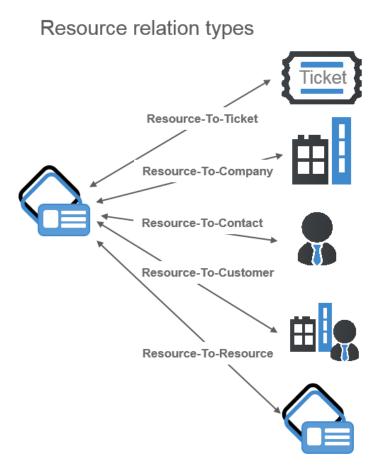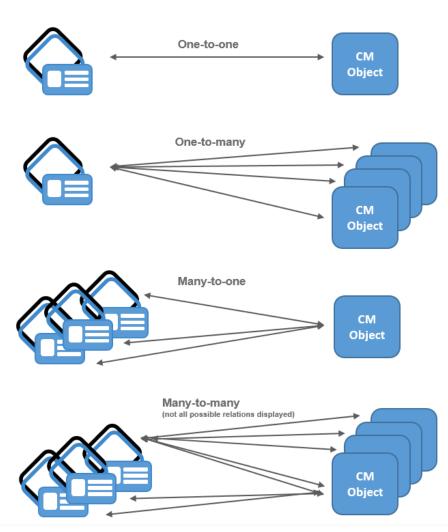
Fig. 1: ConSol*CM Resource Relation Types

For a Resource Relation, the multiplicity has to be defined, i.e. for each relation is is defined to how many objects of the selected object type a relation can be created. The following configurations are possible for the Resource Relation multiplicity:

- **One-to-one**: Each resource can have only one relation with a single target object
- **One-to-many**: Each resource can have many relations with different target objects (e.g. one laptop can have resource-Resource Relations to one, two, or more printers)
- **Many-to-one**: Many resources can have one relation each with only one single target objects
- **Many-to-many**: Many resources can have many relations with different target objects

Resource relation multiplicity

Fig. 1: ConSol*CM Resource Relation Multiplicity

24.7.2 Creating Resource Relation Templates Using the Admin Tool

Resource Relations are always defined for a Resource Type, e.g. for the Resource Type *HP_Printers*.

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*).

> ℹ️ Please note that you cannot create or edit Resource Relations using the navigation item *Relations Overview*. This panel is used only for display of the relations and for defining the order of the relations for their display in the Web Client!

Fig. 2: ConSol*CM Admin Tool - Creating and Editing Resource Relations

To define a new Resource Relation, press the *Add* button, and fill-in the required fields in the pop-up window.
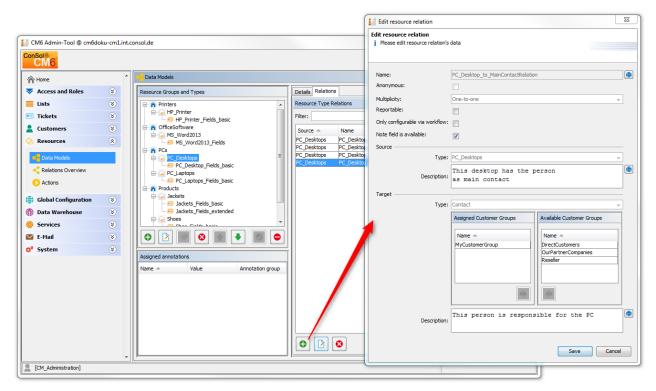
Fig. 3: ConSol*CM Admin Tool - Creating a New Resource Relation

- **Name**
  The technical name of the Resource Relation. You can localize it using the *Globe* button. This name
  will be displayed in the Web Client when a Resource Relation of this type has been established.

- **Anonymous**
  If this checkbox is ticked, *related* will be displayed as 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 for the relation (see the explanation of the relation multiplicity above).

- **Reportable**
  If this checkbox is ticked, the relations of this type will be transferred into the DWH.

- **Only configurable via workflow**
  If this checkbox is ticked, the relation can only be set / removed using workflow scripts. The relation
  will not be offered in the respective menus in the Web Client.

- **Note field is available**
  Isf this checkbox is ticked, a note field will be displayed for each resource when a new Resource
  Relation is established (see the following two figures)

Fig. 4: ConSol*CM Web Client - Establishing a Resource-Resource Relation with Note Field

Fig. 5: 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 header for any relation
      of this type on the source side. The description can be localized using the *Globe* button.
- **Target**
    - **Type**

      The object type of the target of this relation. For an explanation, please see the section about
      Relation Types.

- **Assignment table**

  Here, you can assign objects of the selected type to the relation, i.e. the Resource Relation will only be available for those objects. Depending on the selected target object type, there will be different object 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 header for any relation of this type on the target side. The description can be localized using the *Globe* button.

24.7.3 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.
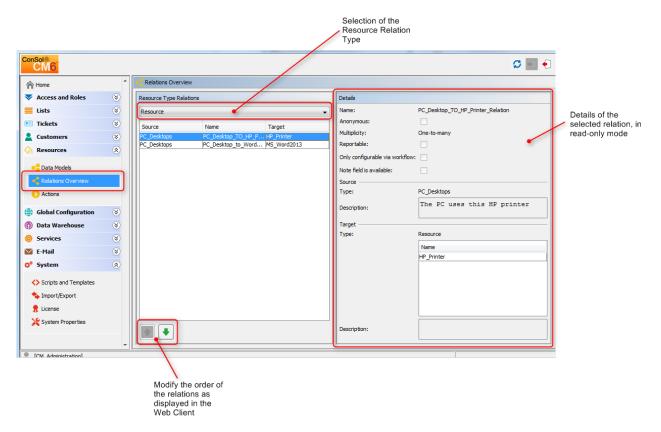
Fig. 6: ConSol*CM Admin Tool - Relations Overview

Use the filter (pull-down menu) in the top row to select the target object type. 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 he filter above the list. You always have to select one Resource Relation type:

- Resource
- Customer
- Contact
- Company
- Ticket

Only relations of this type will be dispayed in the list.

Two functionalities are available in the *Relations Overview:*

- Display relation details
  Select a relation. The relation details will be displayed on the right side in read-only mode.
- 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 relations list in case more than one relation is offered for selection in the Web Client.

24.7.4 Creating Resource Relations Using the Web Client

Resource Relation 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 the work with Resource Relations, see section A Short Introduction to CM. Resource Pool Functionalities in the Web Client. A detailed explanation if all resource-related Web Client functionalities is provided in the *ConSol*CM User Manual*.

24.7.5 Configuration of Resource Relations in the Web Client

Some parameters concerning the display mode of Resource Relations can be configured using Page Customization. There are:

- the default limit for the number of Resource Relations which is required to display the table-filter control elements
- the sorting strategy for Resource Relations

For details about the parameters concerning unit pages, please see section Page Customization, unitResourceRelation.

For details about the parameters concerning the Resource Detail Page, please see section Page Customization, TicketRelation.

For details about the parameters concerning the ticket page, see section Page Customization, resourceRelations.

24.8 CM.Resource Pool - Resource Actions

24.8.1 Introduction to Resource Actions

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 which has to be taken care of.
- Find all companies who 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 data object operations:
    - CREATE
    - UPDATE
    - DELETE
    - SEARCH
- **Manual** actions which are performed by the engineer using *Activities* links on a Resource Detail 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 the Activities will be displayed in the Web Client:
>
> - Act

Fig. 1: 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 action* scripts can be controlled using condition scripts, i.e. you can implement a condition script which is executed before the Resource Action itself. Only when this script returns *true*, the following action script is executed.

So there are two types of scripts you have to deal with when you use ConSol*CM Resource Actions:

- *Resource action* **scripts**
  Defines the action which should be performed.
- *Resource condition* **scripts**
  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 *Resource action* script (either action script only or action script and 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.

24.8.2 Creating Resource Actions Using the Admin Tool

Step 1: Write the Data Object 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.

Fig. 2: ConSol*CM Admin Tool - Resource Action Script in Scripts Section

```
Example 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) {
    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)
```

Step 2: Create Resource Action(s) Which Use the Script

To create, edit or delete Resource Actions, open the tab *Actions*, navigation group *Resources*, in the Admin Tool.

To create / add a new action press the *Plus* button ⊕ and fill-in the required data in the pop-up window.

Fig. 3: ConSol*CM Admin Tool - Creating a Resource Action

- **Name**

  The technical name of the Resource Action. You can localize the value by using the Globe button.

- **Type**

  The Resource Action type. Select one of the following types. If a type has been defined, it cannot be modified afterwards (when you edit an existing Resource Action).

  - **Create**

    This script will be executed automatically when the resource is created

  - **Update**

    This script will be executed automatically when theresource is updated, i.e. when the data has been modified (either manually or automatically) and is saved again**.**

  - **Delete**

    This script will be executed automatically when the resource is deleted.

  - **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 Search Actions.

- **Condition Script**

  In case a condition script should be executed before the action script, the name of the condition script has to be entered here. Use the pull-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. Only when the condition script has returned *true*, the action script will be executed. If there is no condition, just leave this field empty.

- **Execution Script**

  The name of the action script which should be executed. Use the pull-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 Action and/or resource condition scripts to resource types, open the tab *Data Models*, navigation item *Resources*, in the Admin Tool. Select the resource type you would like to edit and click on ⬚ to open the pop-up window to assign the Resource Actions. Only Resource Actions which have been stored as Admin Tool scripts of types *Resource action* will be offered here. Scripts of type *Resource condition* are not offered, because they are not directly assigned to a resource type but assigned to a *Resource action* script.

In the following example (next figure), a manual Resource Action is assigned to the resource type *HP_Printer*.

Fig. 4: ConSol*CM Admin Tool - Assigning Resource Actions to a Resource Type

You can define the following action types:

- **Automatic Actions**
  Those actions will be executed automatically when the respective type of event (create, update, or delete) has been triggered. Select a script for each type which is required. Only the actions with the correct type (which have ben defined as Resource Actions before, see Step 2) are offered in each pull-down menu. For example, for an automatic action of type CREATE, only Resource Actions of type CREATE (which have been defined in the Actions section) will be offered.

- **Manual Actions**
  Those actions are displayed as activities on the Resource Detail 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 reource condition script has returned *true*.

- **Search Actions**
  See section Search Actions.

24.8.3 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 Search Actions.

The *DELETE*, *UPDATE*, and *CREATE* scripts run in the background.

24.8.4 Programming Scripts for Resource Actions and Resource Conditions

Please read the section about Working with the Action Framework for a general introduction about important principles, classes and methods for action and condition scripts.

24.8.5 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_Desktop. 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_desktop. To implement the Resource Action, perform the following steps.

Wirite the *Resource action* script:

```
Resource action script for PC-Desktops to create a new Service Desk ticket for responsible PC
contact

// 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)
```

```groovy
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)
```

Create a Resource Action based on the script:

Fig. 5: ConSol*CM Admin Tool - Creating a New Resource Action

Assign the action to the correct resource type:

Fig. 6: ConSol*CM Admin Tool - Assign the Resource Action to the Correct Resource Type

Check the functionality using the Web Client:

Fig. 7: ConSol*CM Web Client - Resource Detail Page with the Resource Action
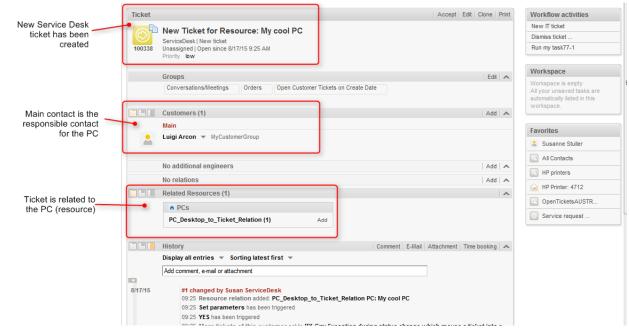
Fig. 8: ConSol*CM Web Client - New Service Desk Ticket Created by the Resource Action

Fig. 9: ConSol*CM Web Client - Resource Detail 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 when a ticket is opened. The following example demonstrates this functionality. For a resource, a HP printer, a maintenance ticket should be opened. During the (sub-) process, an ACF should be offered to ask for the next.

To implement the Resource Action, perform the following steps.

Write the *Resource action* script:

```
Resource action script which opens a ticket and uses an ACF

// 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);
```

Create a Resource Action based on the script.

Fig. 10: ConSol*CM Admin Tool - Create the Resource Action for the HP Printer Maintenance ticket

Assign the action to the correct resource type:

Fig. 11: ConSol*CM Admin Tool - Assigning the Resource Action to the Correct Resource (HP Printer)

Check the functionality using the Web Client:

Fig. 12: ConSol*CM Web Client - Resource Action for HP Printer
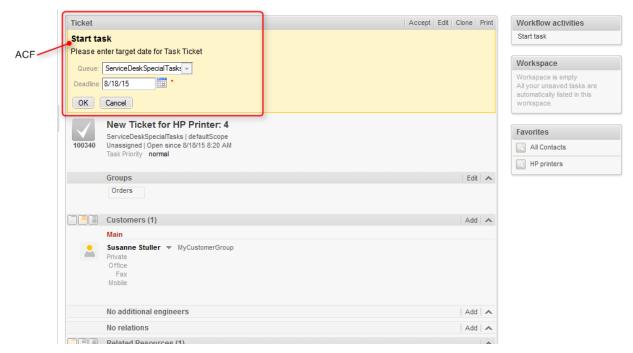
Fig. 13: ConSol*CM Web Client - New Maintenance Ticket for Resource (HP Printer), ACF

24.9 Assigning Permissions for Resources

- Introduction to Permissions Concerning Resources
- Assigning Resource Permissions Using the Admin Tool

24.9.1 Introduction to Permissions Concerning Resources

According to the ConSol*CM basic principle, an engineer can only access resources if he has the required permissions, i.e. if he has at least one role which has the required permissions. Resource permissions are granted on 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*.

24.9.2 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.

Fig. 1: ConSol*CM Admin Tool - Assigning Resource Permissions

For a detailed description of the permissions, see section Role Administration, Tab Resource Types Permissions.

24.10 The Resource Pool Dashboard

24.10.1 Introduction

The Resource Pool Dashboard provides an overview of all objects in the Resource Pool. According to CM principles, only the objects are displayed which are covered by the access permissions of the engineer who is logged in and opens the dashboard.

The Resource Pool Dashboard will always contain the overview of all resources (section *Search or create resources*, see following figure). It might also contain some charts and/or tables which display figures /reports about resources.

Open the Resource Pool Dashboard by clicking on *Resource pool* (or the label which has been configured in your CM system) in the main menu:

Fig. 1: ConSol*CM Web Client - Resource Pool Dashboard without Reports
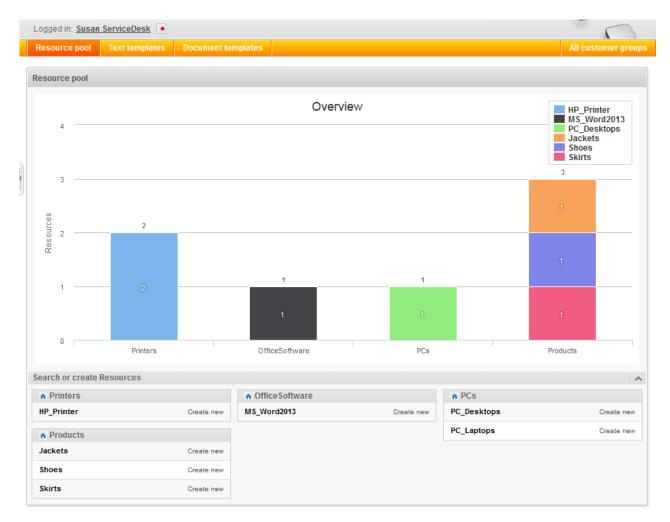
Fig. 2: ConSol*CM Web Client - Resource Pool Dashboard with Report

The Resource Pool, including the Resource Pool Dashboard, is a new feature which has been introduced in 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. Any Resource Pool Dashboard report (chart and/or table widgets) has to be configured manually.

24.10.2 Configuring Reports for the Resource Pool Dashboard

When you configure reports with graphs and tables (or both) for the Resource Pool Dashboard, you work according to the same principles which are applied 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 *Overview* page and select *Enable page customization* in the main menu. Besides other attributes which are not relevant for the Dashboard and are explained on in the section Page Customization, the following (Resource 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**
   Here, the Resource Pool Dashboard can be switched on or off. If a correct 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 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 / 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 chartWidgets 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 chartWidget, e.g. *chartWidget / resourceDashboard / ResourceDashboardOverview1*
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 / *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 tableWidgets 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 tableWidget, e.g. *tableWidget / welcomePage / ResourceDashboardOverview2_table* .

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.

Fig. 3: ConSol*CM Web Client - Page Customization Subtree for Resource Pool Dashboard Configuration

24.10.3 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 principles:
    - 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 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, i.e ' *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 shown below.
    - *null* is a reserved key word 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 widgets name, afterwards, it can be omitted, e.g. [*ResourceDashboardOverview1* :Chart, *ResourceDashboardOverview1* , *ResourceDashboardOverview1*] for a three-columns chart.
    - 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 figures 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 show for the Web Client dashboard, please take a look at the respective section in section Page Customization for the Web Client Dashboard.

Fig. 4: 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.

24.10.4 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 script section and is referenced by its name. The scripts has to be of type *Page customization*. Select the widget in the PCDS and enter the script name:

Fig. 5: ConSol*CM Web Client - Script Definition for a Chart Widget

Fig. 6: 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.

> ⚠ **Attention:**
>
> 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 property is not set within the script, if you want to use the page customization for property 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 the current engineer has at least read permissions, but will return absolutely 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 the attributes which should be set, as a HashMap (see return statement in the code example below which uses, for example, the attributes *visible*, *chart*, *title*, *legend*, *align*, *backgroundColor*). A detailed explanation of all attributes and the respective hyperlinks are 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 the attributes which should be set, as a HashMap. Please see section Attributes for Table Widgets.

> ⚠ **Attention:**
>
> Please note: Very complex scripts can decrease system performance!!!

The following example shows the script *ResourceDashboardOverview1 .groovy.*

Example script for ConSol*CM chart widget ticketsInViewDataWidget.groovy:

```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);
```

```groovy
  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
];
```

The following chart is defined by the script above. For a detailed explanation, please refer to the section
Example for a Chart Widget.
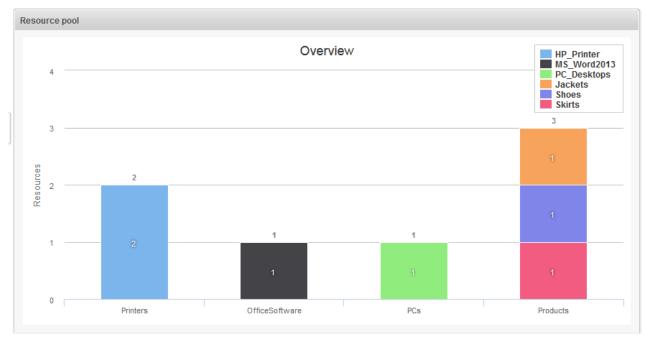
Fig. 7: Example Chart Widget

Configuration Attributes for Widgets

These are the same as for the Web Client Dashboard. Please refer to section Page Customization for the Web Client Dashboard, Configuration Attributes.

25 Global Configuration Section

Global Configuration

In this section, you learn how to configure some global settings in the CM system:

- Languages
- Queue Administration
- Projects
- Working with Calendars
  - Business Calendars
  - MS Exchange Calendar Integration
- Classes of Text

26 Languages

ConSol*CM can be configured to display the GUI (Web Client) in one or more of a great 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.

Fig. 1: ConSol*CM Admin Tool - Global Configuration: Languages

Configured Locales:
In this list, the locales 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 Custom Field values). The displayed values for those activities or fields then depend on the locale of the web browser which is used to display the ConSol*CM Web Client.

- Click on ![icon] to add more locales.
- Click on ![icon] to remove the selected locale from the list.
- Click on ![icon] to set the selected locale as default locale. The default locale will be used if the browser locale is not present in CM, e.g. when the engineer has set the browser locale to FR and in the CM administration only English (default), German, and Polish are available, the English values will be displayed.

⚠ **Attention:**

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.

26.1 The Use of Locales

For an engineer who works with the ConSol*CM Web Client, the GUI 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.

When an administrator configures Custom Fields or Data Object Group Fields he/she can always indicate a translation for each configured language/locale, see following figure.

Fig. 2: ConSol*CM Admin Tool - Custom Field Administration: Localized Values for a Custom Field

In the Process Designer, the locales which have been configured in the Admin Tool are offered. However, you can also delete locales in the Process Designer. Please refer to the *ConSol*CM Process Designer Manual* for details.

27 Queue Administration

- Introduction to Queue Administration
- Queue Administration Using the Admin Tool
  - Filter the Queue List
  - Create a Queue
  - Edit a Queue
  - Delete a Queue
  - Copy a Queue
  - Enable or Disable a Queue

27.1 Introduction to Queue Administration

Queues are a central element of ConSol*CM. Tickets are grouped in queues, e.g. for certain tasks or work groups. To each queue a single workflow is assigned which controls the processing steps of all tickets in this queue. For example, there might be one queue *Helpdesk*, one queue *Marketing*, and one queue *Sales*.

In a queue you define:

- 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 e-mails 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. Only for customers of those customer groups tickets can be created in the queue (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 (Custom Fields) which should be available in tickets of the queue. They are defined by assigning Custom 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).

> ℹ️ **Information:**
>
> As a central element the queue uses various objects and elements that have been defined at another place, i.e. on another page of the Admin Tool, so usually the elements which are later required for the queue definition are defined first. However, except for the workflow, all parameters can be modified even after a queue definition has been saved. 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.

27.2 Queue Administration Using the Admin Tool

In the Admin Tool, queues are managed on the navigation item *Queues* in the navigation group *Global Configuration*.
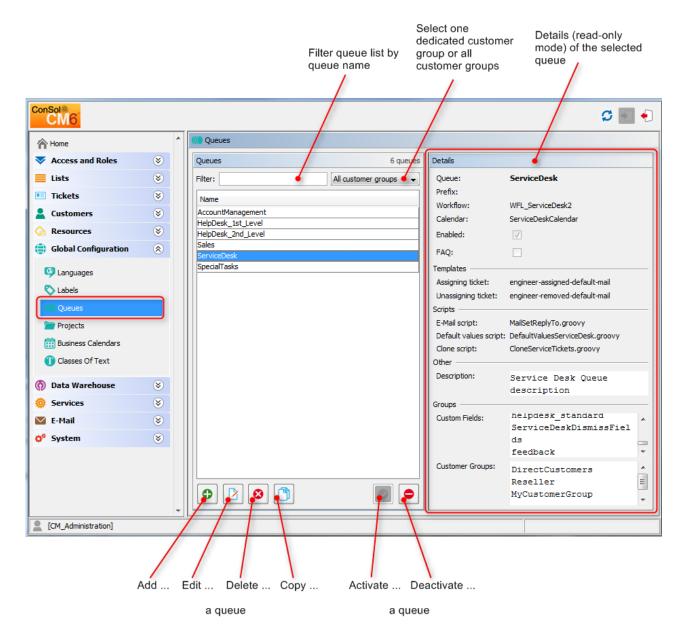
Fig. 1: ConSol*CM Admin Tool - Queue Administration

27.2.1 Filter the Queue List

Queues you want to edit or copy can be found faster, 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 specific for customer groups.

27.2.2 Create a Queue

You create a new queue by clicking on ⊕ below the queue list. The following pop-up window appears:

Fig. 2: ConSol*CM Admin Tool - Queue Administration: Create a Queue

Here, you can define the queue details:

- **Queue:**
  Enter the technical queue name in this field. Click on [icon] 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 name will be displayed in the default language.
- **Workflow:**
  Choose the workflow for the queue from this list.

> ℹ **Best Practice:**
>
> 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!

> ⊖ **Warning:**
>
> 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. if the ticket ID shall indicate to which queue
  or organizational structure it belongs.

> ⚠ **Attention:**
>
> The prefix will remain with the ticket name if the ticket is moved to another queue.

- **Calendar:**
  Choose the business calendar for the queue from the list. Calendars define working hours, holidays
  and the valid timezone (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 the
      work with time triggers.
- **Enable:**
  If this check box is ticked, the queue is immediately available in the system after saving. If the check
  box is not ticked, the queue is disabled. In enabled queues, you can create tickets, in disabled
  queues, this is not possible.

- **FAQ:**
  Ticking this check box marks the queue as a knowledge base for CM.Track users. They can search
  for tickets of this queue in CM.Track, the ConSol*CM Web Portal. Please see also section CM.Track:
  FAQs in CM.Track on this topic.

- **Ticket assignment templates:**
  Here you can choose e-mail templates that shall be used for an automatic e-mail which is sent to the
  (new) engineer when a ticket is assigned to an engineer (*Assign*) or to the (old) engineer when a
  ticket is retrieved from an engineer (*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. When you do not want the CM system to send an automatic e-mail
  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 Appendix C (System Properties) for details.

- **Scripts**:
  Scripts are used to automate recurring tasks and activities. They are managed and stored in the
  Scripts & Templates Administration (see section Scripts). You can assign:

  - **E-mail script**
    Choose a script from the list if outgoing e-mails for this queue should be modified by the script,
    e.g. to contain default values like the sender or address fields. The e-mail script indicated here
    is the last script that processes an outgoing e-mail so all former settings will be overwritten
    (except for REPLY-TO, see warning below!) in case a variable has been set before. All scripts
    of type *E-mail* that are stored in the script section are available, please make sure to pick the
    correct one.

    > ⊖ When you work with the configuration of the REPLY-TO e-mail address, please note
    > the following technical behavior of ConSol*CM and adapt your system accordingly!
    > **The technical background:**
    > There are three 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-E-Mail Editor in the Web Client.
    >   If it is really the effective REPLY-TO address in an e-mail depends on the
    >   configuration in the queue-specific outgoing e-mail script. See next topic.
    >   2. The **REPLY-TO address which is set in a queue-specific outgoing e-mail**
    >   **script**. Since the outgoing e-mail script is the last instance which processes
    >   an outgoing e-mail, 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, it will not really be used (but it will be displayed in the Ticket E-Mail
    >   Editor which might cause some confusion! What that means for your system
    >   configuration is explained in the next section).
    >   3. The **e-mail address of the current engineer** (the engineer who is logged in
    >   to the Web Client). This personal e-mail address is used as REPLY-TO
    >   address for e-mails from the Web Client if neither the *mail.reply.to* property is
    >   set nor a queue-specific outgoing e-mail script is configured.
    > In the Web Client, in the ticket history, the REPLY-TO address which was really used
    >   is always displayed for outgoing e-mails. So even in case there should be a

difference between the address which was displayed in the Ticket E-Mail Editor (the *mail.reply.to* property) and the REPLY-TO address which was really used (the REPLY-TO in the queue-specific outgoing e-mail 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 e-mail script

or

- use the *mail.reply.to* system property.

However, since the e-mail communication should take place via ConSol*CM and not using personal e-mail addresses, one of the two system settings mentioned above should be used to prevent CM from using personal e-mail addresses as REPLY-TO. The latter would automatically lead to customer e-mails being sent to an engineer's personal e-mail account instead of CM.

What that means for your system configuration:

1. The simplest way to set a REPLY-TO address by using the *mail.reply.to* system property. It will be displayed in the Ticket E-Mail Editor and will be the effective REPLY-TO address.

2. If you work with queue-specific outgoing e-mail scripts, you have to configure a script for each queue! The *mail.reply.to* system property should then be left empty! You can

   a. write queue-specific scripts which are only used in one queue (e.g. to set a team-specific REPLY-TO address), e.g. *ChangeOutgoingMail. groovy_Queue1,ChangeOutgoingMail.groovy_Queue2, ... ChangeOutgoingMail.groovy_Queue_n*.

   b. write a standard outgoing e-mail script which is configured for all queues, e.g. *ChangeOutgoingMail.groovy_Queue_standard*

   c. combine a and b, i.e. to set a standard script for most queues and configure specific scripts for specific queues

What that means when you work with workflow scripts which send e-mails:

(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 e-mail.

- Use the *Mail* object when the queue-specific script should be used: e.g. *mail. useDefaultScript()* . This will overwrite the *mail.reply.to* property! Hence, use either property or script.

If neither the system property nor the queue-specific outgoing e-mail 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 e-mail 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*.

- **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.

- **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.

- **Tab Custom Fields:**

  In order to show data fields (Custom Fields) in tickets of the queue, you have to assign the respective custom field groups here.

- **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 on ⬅ and remove it by clicking on ➡ .

Click on *Save* afterwards to create the queue. The details of the new queue are displayed on the right side of the page.

27.2.3 Edit a Queue

If you want to edit a queue, select it in the list and click on 🖻 or just double-click the name of the queue. Modify the queue details and click *Save* to store your modifications.

> ⚠ **Attention:**
>
> You cannot change the workflow of a queue, once the queue has been saved for the first time!

27.2.4 Delete a Queue

Select the queue you want to delete in the list and click on ⊗ . If you confirm the following dialog with *Yes*, the queue will be deleted and is no longer available in the system.

> ⚠ **Attention:**
>
> 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.

27.2.5 Copy a Queue

The icon 🗐 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 on the name or click on 🖉 to open the edit window where you can modify the name and details of the queue. Click *Save* to store your modifications.

> ⚠ **Attention:**
>
> You cannot change the workflow of a queue, once the queue has been saved for the first time!

27.2.6 Enable or Disable a Queue

You can disable a queue to prevent that new tickets can be opened in this queue. That way you can re-activate the queue later and do not have to delete it. To disable a queue, select the queue in the queue list and click on ⊖ . The entry in the list is now shown in italics. Just click on ⊙ at the bottom of the page, if you want to enable the queue again.

You can still read tickets in a disabled queue (provided you have the read access rights for this queue), but you cannot process tickets, i.e. they cannot be moved to the next step in the process using workflow activities.

28 Projects

- Introduction
- Managing Projects Using the Admin Tool
    - Create or Edit a Project
    - Delete a Project
    - Disable or Enable a Project
    - Localize a Project

28.1 Introduction

With ConSol*CM you can book amounts of time on projects, please see section Time Booking for a detailed explanation.

28.2 Managing Projects Using the Admin Tool

The projects are managed on the navigation item *Projects* in the navigation group *Global Configuration*. For a project to become 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 amounts of time on tickets that are in one of the queues where the project has been assigned. An engineer can see his/her time bookings on the engineer profile page.
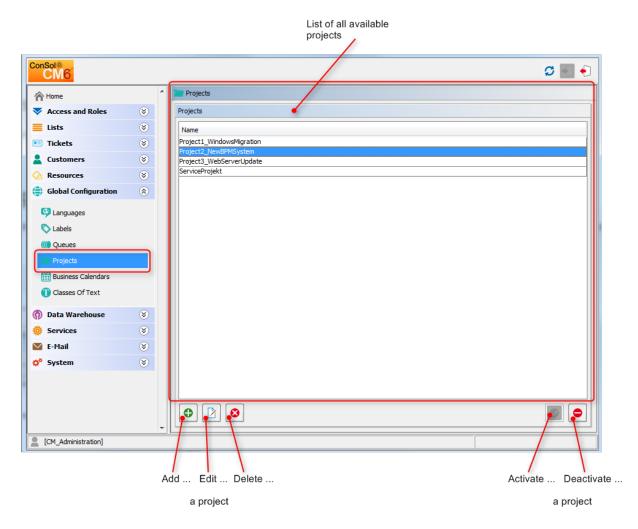
Fig. 1: ConSol*CM Admin Tool - Global Configuration: Projects

28.2.1 Create or Edit a Project

A project is defined by its name. By clicking on ⊕ a pop-up window appears where you can enter the name. Using the globe icon 🌐 next to the name field you can localize the name subsequently (see below). The check box *Enabled* is already selected to set the project active in the system (see also Disable or Enable a Project). You will get the same window when you click on 📝 in order to edit a project.

Fig. 2: ConSol*CM Admin Tool - Global Configuration: Create or Edit a Project

28.2.2 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 you can only disable this project (see below).

In order to delete a project, select it in the list and click on 🞫 . After choosing *Yes* in the confirmation dialog the project will be removed from the list and the system.

28.2.3 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 on 🛇 . The entry in the list is shown in italics afterwards. The project is not available for time bookings anymore. Just click on ✅ 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 de-selecting the check box *Enabled*. When you create a project this check box is automatically selected.

28.2.4 Localize a Project

Click on the globe icon 🌐 in the create or edit window to enter the localized name of a project. In the pop-up window *Localize* all languages that are available in the system are listed. Enter the project name in the *Value* field for each additional language on the right and click *Save*. The localized project name, according to the locale of the web browser, will be displayed in the Web Client in the time booking section of a ticket. If no localized value is found, the default value is displayed. This is the value of the default language. If this has not been defined either, the technical name of the project is displayed.

29 ConSol CM Administrator Manual 6.10.2 - Working With Calendars

29.1 Working with Calendars

In ConSol*CM, there are two configurations which concern calendars:

1. You can define Business Calendars which can be used to manage and rule business hours for one or for all teams who work with CM. This is covered in section Business Calendars.
2. You can configure CM to offer access to Microsoft Exchange Server calendars. This is explained in section MS Exchange Calendar Integration.

29.2 Business Calendars

29.2.1 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 free time.

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.

> ⓘ **Example:**
>
> Tickets which have not been assigned to an engineer more than one hour after opening shall be 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.

SLA = Reaction time 4 hours within the regular business hours
Monday - Friday 9:00 - 17:00

Without Business Calendar:

With Business Calendar:
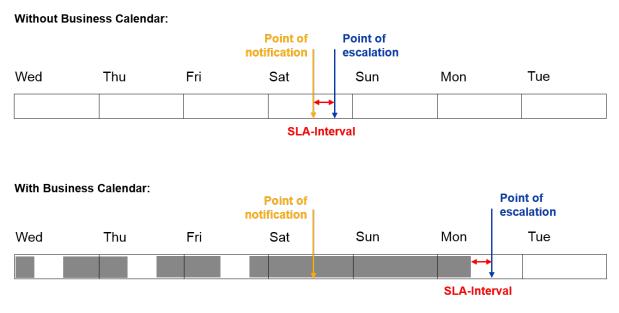
Fig. 1: ConSol*CM Principle - Business Calendar

Besides 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 the basis for time calculations (Process Designer), this is explained in the *ConSol*CM Process Designer Manual*.

29.2.2 Configuration of Business Calendars Using the Admin Tool

In the Admin Tool, Business calendars are defined on the navigation item *Business Calendars* in the navigation group *Global Configuration*.
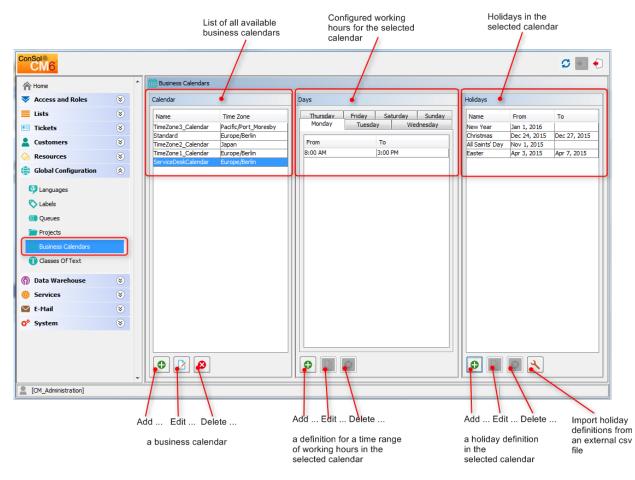
Fig. 2: ConSol*CM Admin Tool - Global Configuration: Business Calendars

Creating a New Calendar

Click on the ⊕ in the left part of the page to create a new calendar. The following window appears:

Fig. 3: ConSol*CM Admin Tool - Business Calendars: New Calendar

- **Name:**
  Enter a unique name for the calendar.
- **Timezone:**
  Choose the time zone to be used for the calendar.

> ⚠️ **Attention:**
>
> 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 timezone. A time trigger which uses the business trigger would fire according to the Berlin time and not the Detroit time.

Click on *Save* afterwards to create the calendar.

Clicking on 📝 you can modify a selected calendar in the same way. Click on ❌ , if you want to delete the selected calendar.

Defining the Working Hours for a Calendar

Select a calendar on the left and click on ➕ in the middle part of the page to create the days and hours for this calendar. The following window appears:

Fig. 4: ConSol*CM Admin Tool - Business Calendars: Working Hours of a Calendar

- **Time range**
Enter the time range for which the automatic workflow escalations shall be active.
- **Days**
Mark the check boxes of the days for which the time range shall be valid. It is possible to choose individual or all days (check box *All*).

> ⓘ **Information:**
>
> In case, the system detects an inconsistency of the time you define here with an already existing time, you will get a corresponding message.

Click on *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 on 🖉 and change the time range in the window that appears. Or click on ⊗ if you want 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 on ⊕ in the right part of the page to create a new holiday entry. The following window appears:

Fig. 5: ConSol*CM Admin Tool - Business Calendars: Holidays of a Calendar

- **Name:**
Enter the name of the holiday here.
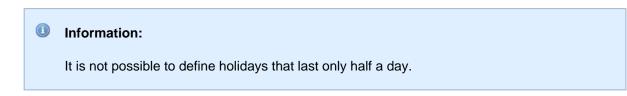
- **From:**
  Enter the date of the holiday in this field.
- **To:**
  If it is a multi-day holiday (e.g. Christmas), you can enter the last date of the holiday here.

> ⓘ **Information:**
>
> It is not possible to define holidays that last only half a day.

Click on *Save* afterwards to create the holiday.

If you want to edit a selected holiday entry just click on 🖹 . Clicking on ⊗ deletes one or more selected entry/entries.

Importing Holidays for a Calendar from an .csv File

Holiday data can be imported from a .csv file which is based on 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

```
.csv file for holiday import

Christmas,24/12/2014,26/12/2014
New Year,01/01/2015,01/01/2015
Easter,03/04/2015,06/04/2015
```

In the Admin Tool navigation group *Global Configuration,* navigation item *Business Calendars*, select a calendar and click on 🔧 *Import holidays* and enter the path for the .csv import file.

Fig. 6: Consol*CM Admin Tool - Business Calendars: Importing Holidays

The new holidays will be imported in the holiday list of the selected business calendar.

Fig. 7: Consol*CM Admin Tool - Business Calendars: Newly Imported Holidays

29.3 MS Exchange Calendar Integration

29.3.1 Introduction

Starting with CM version 6.10, it is possible to include an MS Exchange calendar view into the Web Client.

Supported MS Exchange versions are:

- Exchange Server 2010 SP2
- Exchange Server 2013

The calendar view can be offered ...

- on the ticket page
- on the customer detail page, i.e.
    - on the contact detail page
    - on the company detail 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 the full access has been configured)

Fig. 1: ConSol*CM Web Client - Ticket with Calendar Section (Monthly View)

Fig. 2: ConSol*CM Web Client - Adding an Appointment in the Calendar

29.3.2 Configuring the MS Exchange Calendar Integration

Basic Configuration

The configuration of an MS Exchange server which should provide the 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 will be explained.

Perform the following steps:

1. **Make the calendar section visible (example for the Ticket Edit page):**
    Log in as 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 you 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 *Calendar* 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 will follow in step 2.

Fig. 3: ConSol*CM Web Client - Using Page Customization to Make the Calendar Section of a Ticket Visible

Fig. 4: ConSol*CM Web Client - Page Customization Definition Section (PCDS) for the Calendar Section of Tickets

2. **Configure the calendar script:**

   The connection to the calendar of an MS Exchange server is implemented using an Admin Tool script. The script has to be stored in the Admin Tool script section, i.e. navigation group *System*, tab *Scripts and Templates* and must be of script type *Calendar integration*. The name of this script has to be entered as value for the attribute *calendarInitializationScript* in the page customization (see step 1).

   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'
]
```

   For *somebody*, you have to insert the desired name of the MS Exchange user. The calendar on the ticket page or on the customer detail page will be based on the view of this user in the Exchange calendar. For example, *somebody* could be the Exchange login of the CM engineer who is currently logged in. In case it is the same login as in CM, you can retrieve it using the method *workflowApi.getCurrentEngineer().getName()*.

3. **Enable *edit* mode for the calendar:**

   To allow full access (i.e. to create and/or edit appointments) for engineers, set the attribute *editable* to *true*.

Advanced Configuration: Page Customization Parameters for the MS Exchange Calendar Integration

Short Background Information about the MS Exchange Calendar Integration

The integration (or better the *display*) of MS Exchange calendars in ConSol*CM is based on the jQuery solution *fullcalendar*. If you would like to go into details, please refer to the fullcalendar web site.

Pages for the Configuration of the calendarSection

The *Calendar* section can be configured on three pages:

- **Ticket page**
  Use *calendar/ticketEditPage/calendarSection*.
- **Customer detail pages**
  - **Contact detail page**
    Use *calendar type/contactEditPage/calendarSection*.
  - **Company detail 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**
  Determines if the "all-day" slot is displayed at the top of the calendar. (boolean)
- **appointmentBackgroundColor**
  Sets the background color for all appointments on the calendar. You can use any of the CSS color formats such as *#f00*, *#ff0000*, *rgb(255,0,0)*, or *red*. (java.lang.String)
- **appointmentBorderColor**
  Sets the border color for all appointments on the calendar. You can use any of the CSS color formats such as *#f00*, *#ff0000*, *rgb(255,0,0)*, or *red*. (java.lang.String)
- **appointmentColor**
  Sets the background and border colors for all appointments on the calendar. You can use any of the CSS color formats such as *#f00*, *#ff0000*, *rgb(255,0,0)*, or *red*. (java.lang.String)
- **appointmentConstraint**
  Limits appointment dragging and resizing to certain windows of time. (java.lang.String)
  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. Same format as *businessHours* attribute. Days of week are optional. Examples: *10:00-18:00; 1,2,3,4* or *10:00-18:00*
- **appointmentDurationEditable**
  Allows appointments' durations to be editable through resizing. (boolean)
- **appointmentOverlap**
  Determines if appointments on the calendar, when dragged and resized, are allowed to overlap each other. (boolean)

- **appointmentStartEditable**
  Allows appointments' start times to be editable through dragging. (boolean)
- **appointmentTextColor**
  Sets the text color for all appointments on the calendar. You can use any of the CSS color formats such as *#f00*, *#ff0000*, *rgb(255,0,0)*, or *red*. (java.lang.String)
- **aspectRatio**
  Determines the width-to-height aspect ratio of the calendar. If not set default value *1.35* will be set. (java.lang.String)
- **businessHours**
  Emphasizes certain time slots on the calendar. (java.lang.String)
  Format: <start-time>-<end_time>;<days_of_week>
  Example: 10:00-18:00; 1,2,3,4 (from 10am to 6pm, Monday-Thursday)
- **calendarEventHandlerScript**
  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**
  Name of the script which produces calendar configuration. If value is empty or script returns *null* the calendar won't be shown. (java.lang.String)
- **contentHeight**
  Will make the calendar's content area a pixel height. By default, this option is not set and the calendar's height is calculated by *aspectRatio*. (java.lang.String)
- **defaultAllDayAppointmentDuration**
  A fallback duration for all-day appointments without a specified *end time* value. If not set, the default value will be set to *1* (one day). (java.lang.String)
- **defaultDate**
  The initial date displayed when the calendar first loads. Accepts an ISO8601 date string like *2014-02-01*. (java.lang.String)
- **defaultTimedAppointmentDuration**
  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 appointment during creation. (java.lang.String)
- **defaultView**
  Default view.
  Possible values: month, basicWeek, basicDay, agendaWeek, agendaDay.
  View examples at Available Views. (java.lang.String)
- **editable**
  Whether the appointments can be dragged and resized. This value overwrites source configuration. (boolean)
- **firstDay**
  The day that each week begins with. (Sunday=0, Monday=1, Tuesday=2, etc.). If empty, value will be based on browser's locale. (java.lang.String)

- **forceAppointmentDuration**
  A flag to force calculation of an appointment's end if it is unspecified. (boolean)
- **handleWindowResize**
  Whether to resize the calendar automatically when the browser window resizes. (boolean)
- **headerCenter**
  Defines the buttons and title at the top of the calendar (*middle*). See *headerLeft* description for details. (java.lang.String)
- **headerLeft**
  Defines the buttons and title at the top of the calendar (*left side*). Comma or space separated list values (values separated by a comma will be displayed adjacently). (java.lang.String)
  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**
  Defines the buttons and title at the top of the calendar (*right side*). See *headerLeft* attribute description for details. (java.lang.String)
- **height**
  Will make the entire calendar (including header) a pixel height. By default, this option is not set and the calendar's height is calculated by *aspectRatio*. (java.lang.String)
- **hiddenDays**
  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. (java.lang. String)
- l**azyFetching**
  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. (boolean)
- **maxTime**
  Determines the end time (exclusively) that will be displayed, even if the scrollbars have been scrolled all the way down. Default value is *24:00:00*. (java.lang.String)

- **minTime**

  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*. (java.lang.String)

- **nextDayThreshold**

  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 occur in *month view*, *basicDay*, *basicWeek* and the *all-day slots* in the agenda views. (java.lang.String)

- **rightToLeftMode**

  Displays the calendar in right-to-left mode. (boolean)

- **scrollTime**

  Determines how far down the scroll pane is initially scrolled down. Default is: *06:00:00* (6am). (java.lang.String)

- **slotAppointmentOverlap**

  Determines if timed appointments in agenda view should visually overlap. (boolean)

- **slotDuration**

  The frequency for displaying time slots. Default is *00:30:00* (30 minutes). (java.lang.String)

- **snapDuration**

  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*. (java.lang.String)

- **state**

  The visibility mode of the section, possible values are [expanded, collapsed, collapsed_and_preload, hidden]. Default: *expanded*. (java.lang.String)

- **weekNumbers**

  Determines if week numbers should be displayed on the calendar. (boolean)

- **weekends**

  Whether to include Saturday/Sunday columns in any of the calendar views. (boolean)

30 Classes of Text

30.1 Introduction

A class of text is a class that you assign to a ticket entry. This entry can be:

- a comment
- an e-mail that was sent from the ticket
- an e-mail that was received in the ticket
- an attachment

Assigning a class of text can have 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 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-over 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.

Fig. 1: ConSol*CM Web Client - Using a Class of Text for an Internal Important Note

30.2 Managing Classes of Text Using the Admin Tool

30.2.1 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 following figure).
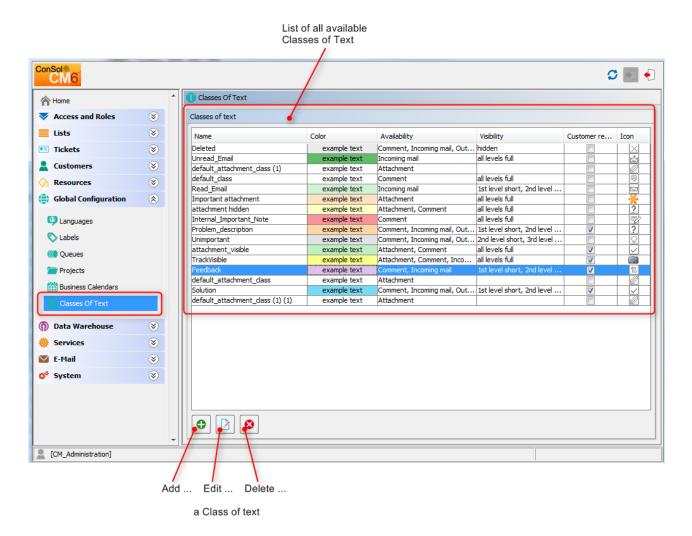
Fig. 2: ConSol*CM Admin Tool - Global Configuration: Classes of Text

You define a new class of text by clicking on ⊕ below the list. The following pop-up window appears:

Fig. 3: ConSol*CM Admin Tool - New Class of Text

Here, you have to define the class details:

- **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 by clicking on it. You can check the selected color in the *Preview* area. Click on *OK* to save your choice. Click on *Reset* if you want to return to the last saved color.
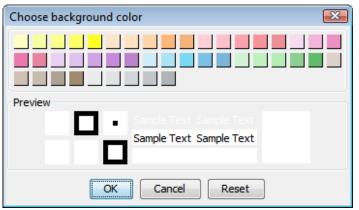
Fig. 4: ConSol*CM Admin Tool - Choose a Color for the Text Class

- **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**

  (see also General Information about the Visibility of Ticket History Entries)

  There are three ticket history levels [ ] [ ] [ ] History 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 if a text entry marked with this class of text is displayed on the level or not.

  The addition *short* and *full* indicates, whether entries are shown *within* the chosen visibility level in full length (*full*) or if they are shortened after a certain number of characters (*short*):
  - **short** - shortened
  - **full** - full length

Fig. 5: ConSol*CM Web Client - Example: Visibility Level "Extended" (2nd Level) Shows Entries in Full Length ("full")

Fig. 6: ConSol*CM Web Client - Example: Visibility Level "Basic" (1st Level) Shows Entries Shortened ("short")

Select in the drop-down menu on which history levels the marked ticket information shall be visible (see picture below).

Fig. 7: ConSol*CM Admin Tool - 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 Ticket History, General Information about the Visibility of Ticket History Entries .

- **Icon**

  When you click into the box next to *Icon* you will get a selection of standard CM icons. Select one of these icons for the new class of text or load your own individual icon by clicking on the *Browse...* button.

Fig. 8: ConSol*CM Admin Tool - Choose an Icon for the Text Class

- **Customer readable**

  Select this check box if ticket information marked with this class of text shall be visible for customers in CM.Track, the 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. In the Web Client (in the context menu for classes of text, see next figure) the name will be displayed in the respective language of the locale of the web browser. If you do not make an entry here the object name, i.e. the content of the *Name* field, will be taken instead.

Click on OK to save the details of the new class of text and to close the window.
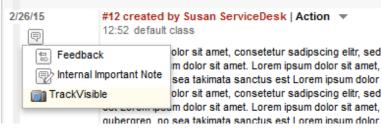
Fig. 9: ConSol*CM Web Client - Context Menu for Classes of Text in Ticket History Section

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.

30.2.2 Edit a Class of Text

If you want to edit a class of text, select it in the list and click on ⬚ . The same window as described above for creating a class will appear. You can modify all details and save your changes by clicking *OK*.

30.2.3 Delete 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 select it in the list and click on ⊗ . If you confirm the following dialog with *Yes*, the class will be removed from the list and the system.

30.2.4 Setting the Default Class of Text

To define the default class of text, use the system property *cmweb-server-adapter*, *defaultContentEntryClassName* (see Appendix C). 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.

30.2.5 Working With Classes of Text in Scripts
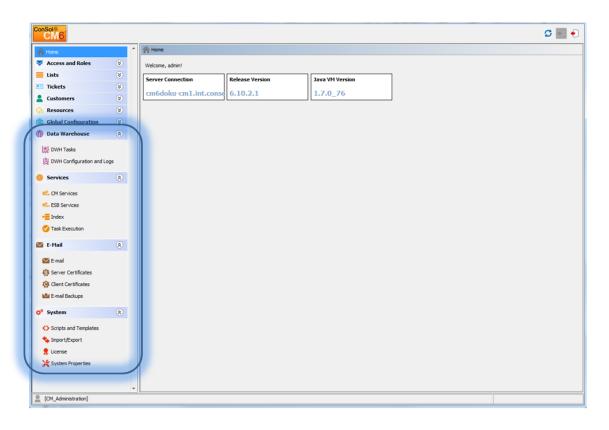
You can work with classes of text in scripts which are used in workflow activities or which are located in the *Script* section of the Admin Tool. For details about programming, please read the *ConSol*CM Process Designer Manual*.

In scripts, you can assign a class of text to a TextEntry, even when the class of text is not assigned to the queue! This can help automate processes.

30.2.6 Transfer of Classes of Text to the DWH

All classes of text (default classes and special ones) used in history entries (comments, e-mails) are transferred from 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.

31 Advanced System Configuration

Advanced System Configuration

In this section, you learn how to configure some advanced settings in the ConSol*CM system:

The following topics will be 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 which is the basis for CM Reporting.
- Services: You will learn which services work within the CM application.
- Search in ConSol*CM: You will learn all about search in CM, e.g. how to manage the Indexer or how to implement Search Action for the Action Framework.
- The Task Execution Framework (TEF): You will learn what the TEF is and how you can configure it in your CM system.
- E-Mail Configuration: You will learn all about the e-mail functionality in ConSol*CM and how to configure the respective modules.
- Script and Admin Tool Template Administration: You will learn which script and templates types are managed with the Admin Tool and how to code those scripts and templates yourself.

- **Deployment (Import/Export):** You will learn all about CM scenarios which are used for CM system export and import.
- **License Management:** You will learn how to install a new license in your CM system.
- **System Properties:** You will learn what system properties are and what you can do with them to configure basic settings of CM.
- **Working with Text Templates:** You will learn all about E-mail and Doc templates which can be used as integral part of your CM system.
- **Time Booking:** You will learn what you have to do to prepare your ConSol*CM system for time booking.
- **Authentication Methods for Engineers in the CM Web Client:** You will learn various configurations for the engineer authentication in the Web Client.
- **CTI with ConSol CM: CM.Phone:** You will learn all about the CTI module of ConSol*CM which has to be licensed separately.
- **The Customer Portal: CM.Track:** You will learn all about CM.Track configuration, the CM customer portal which is also a separate module based on an additional license.
- **System Architecture:** You will get a first glimpse at the CM system architecture. More information on this topic is provided in the *ConSol*CM Set-Up Manual* and the *ConSol*CM Operations Manual*.

32 Ticket Administration

32.1 Introduction to Ticket Administration

In the ticket administration you can:

- **Delete tickets**
  e.g. if a ticket was created by mistake.
- **Reopen tickets**
  e.g. if a ticket has been closed too early.

> ⚠️ **Attention:**
>
> Please keep in mind that a ticket, which is reopened, starts in the process at the start node of the respective workflow. So when a ticket has passed 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 way is to modify the workflow to contain a shortcut.

32.2 Ticket Administration Using the Admin Tool

In the Admin Tool, you can manage tickets using the navigation item *Administration* in the navigation group *Tickets*.

Fig. 1: ConSol*CM Admin Tool - Ticket Administration after Ticket Search

32.2.1 Delete or Reopen Tickets

For those operations you can either use the buttons below the list, or you can use the context menu.

1. Buttons:
   Select the desired tickets in the list and click on [×] to delete tickets resp. click on [↻] to reopen tickets. If you confirm the following dialog with *Yes*, the corresponding action will be executed.
2. Context menu:
   Select the desired tickets in the list and use the right mouse button to open the context menu. Select the desired operation.

32.2.2 Switching off the Delete Functionality Using a System Property

The delete functionality can be switched off 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.

Fig. 2: ConSol*CM Admin Tool - Ticket Administration Page with cmas-app-admin-tool, delete.ticket. enabled = true

Fig. 3: ConSol*CM Admin Tool - Ticket Administration Page with cmas-app-admin-tool, delete.ticket. enabled = false

32.2.3 Search Tickets

To search for tickets you want to delete or reopen click on 🔍 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.

Fig. 4: ConSol*CM Admin Tool - Ticket Administration: Ticket Search

The following parameters can be used for searching:

- **Ticket id:**

  You can enter an ID range for the tickets here.

- **Creation date:**

  Via calendars you can define a time period within which the tickets have been opened.

- **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:**

  Here you can specify the maximum number of tickets displayed in the list.

- **Ticket State:**

  Using the radio buttons you can determine if you want to search for *open*, *closed*, or *all* tickets.

- **Queues:**

  The list on the right shows the available queues. Select the queues to search in here and click on
  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.

Click on *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 last 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 just start it again.

33 Data Warehouse (DWH) Management

- Introduction
  - Data Warehouse
  - ConSol*CM Data Warehouse and ConSol*CM Reporting Framework
- DWH Management Using the Admin Tool
  - DWH Configuration and Logs
    - Basic DWH Configuration
    - Initialization of the DWH
    - First DWH Synchronization
    - DWH Synchronization During System Operation
  - DWH Tasks
  - DWH Troubleshooting and Repair
- DWH-Related System Properties
- Transfer Modes: JMS or DIRECT
  - Configuring the Transfer Mode
  - JMS Mode
  - DIRECT Mode

> ⚠️ **Attention:**
>
> To set up a DWH, a ConSol*CM Reporting Framework (CMRF) that is up and running is required. If your system does not include a CMRF yet, please talk to your ConSol*CM manager or contact ConSol* Software.

33.1 Introduction

33.1.1 Data Warehouse

A data warehouse is a collection of data from one or more systems and/or databases that provides the basis for reporting and for data analysis. Often, the data has been combined or rearranged (integrated) in a way that a perfect basis for reporting and for data analysis is provided.

33.1.2 ConSol*CM Data Warehouse and ConSol*CM Reporting Framework

A ConSol*CM default installation comprises all modules that are required to build a data warehouse. The core component is the **ConSol*CM Reporting Framework (CMRF)**.

This is a Java EE application which synchronizes the data between the ConSol*CM database and the DWH database. Please see the following picture for a system architecture example with DWH and CMRF. We recommend that you use two servers, one for ConSol*CM and one for CMRF. Please refer to the current *System Requirements* for information about the supported application servers and RDBMS.
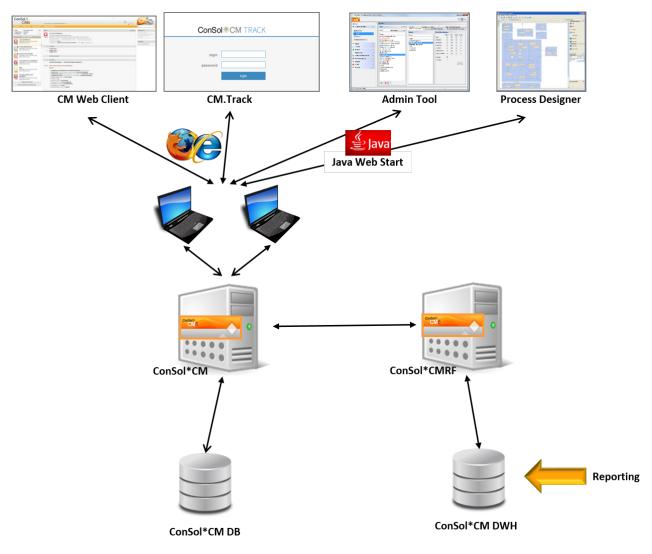
Fig. 1: ConSol*CM - System architecture with DWH and CMRF (2 Servers)

There are two different modes to synchronize the DWH with the CM database:

- **LIVE mode**
  In this mode, every change that has been submitted to the CM database is immediately synchronized with the DWH.
- **ADMIN mode**
  In this mode, the administrator has to trigger the synchronization manually.

> ⚠️ **Attention:**
>
> Only data from custom fields and data object group fields with the annotation *reportable = true* will be synchronized with the DWH!

33.2 DWH Management Using the Admin Tool

To manage the DWH, use the navigation items in the navigation group *Data Warehouse*:

- DWH Tasks
- DWH Configuration and Logs

33.2.1 DWH Configuration and Logs
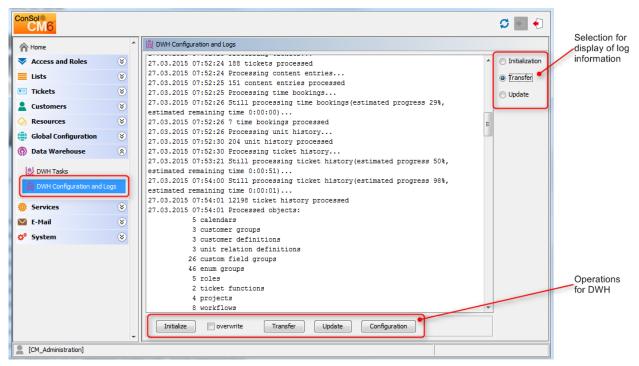
Fig. 2: ConSol*CM Admin Tool - DWH Configuration and Logs

In the center area, the log file information of DWH operations is displayed. Use the radio buttons on the right-hand side to select which log file should be displayed. The DWH operations are available as buttons in the row below the center area.

- **Initialize**
  Create tables during DWH set-up, see Initialization of the DWH.
- **overwrite**
  Used for re-initialization, see Initialization of the DWH.
- **Transfer**
  Start initial data transfer after set-up, see First DWH Synchronization.
- **Update**
  Transfer new/additional data to the DWH, see DWH Synchronization During System Operation.
- **Configuration**
  Open DWH configuration panel, see Basic DWH Configuration.

The radio buttons and the buttons for DWH operations do not influence one another, i.e. when you select an operation, the log file display is not changed. See the following paragraphs for detailed explanations about all operations.

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.

In order to prepare the system for the DWH synchronization, you have to configure the database and the DWH mode. In the Admin Tool, open the navigation group *Data Warehouse,* navigation item *DWH Configuration and Logs*. Click on *Configuration*, open the tab *Configuration*, and insert all values of the CMRF server.

Fig. 3: ConSol*CM Admin Tool - DWH Configuration and Logs: DWH Configuration

At *DWH Mode Selection*, choose one of the available options:

- **LIVE**
  In this mode, every change that has been submitted to the CM database is immediately synchronized with the DWH.
- **ADMIN**
  In this mode, the administrator has to trigger the synchronization manually.
- **OFF**
  No transfer of data to the DWH.

You can also see the DWH mode that is in operation by taking a look at the corresponding DWH system property *cmas-dwh-server*, *dwh.mode* (see Appendix C).

Fig. 4: ConSol*CM Admin Tool - System Property for DWH Mode

For the connection, the following parameters are required:

- For **JBoss**:
  - **Initial context factory**
    The Java class that is used for the connection. No changes are required here since ConSol*CM selects the correct value during system set-up.
  - **URL factory packages**
    The Java package that comprises the required connection classes. No changes are required here since ConSol*CM selects the correct value during system set-up.
  - **CMRF URL**
    The URL of the CMRF, i.e. the URL to which the CM system should connect in order to provide the information about new synchronization tasks. The general notation

    ```
    <CMRF_HOST_IP>:<JNDI_PORT>
    ```

    (i.e. *192.168.0.1:1099*) can be used. Please note that the default JNDI port is *1099*. In case you are using different JBoss port mappings, then the JNDI port will also differ. I.e. when using *ports-01* then the JNDI port is *1199*, for *ports-02* it is *1299*, and so on.
- For **Weblogic**:
  - **Initial context factory**
    The Java class that is used for the connection. Use:

    ```
    weblogic.jndi.WLInitialContextFactory
    ```

  - **URL factory packages**
    The Java package that comprises the required connection classes. Use:

    ```
    weblogic.jndi.factories:weblogic.corba.j2ee.naming.url:weblogic.corba.client.
    naming
    ```

- **CMRF URL**

    The URL of the CMRF, i.e. the URL to which the CM system should connect in order to
    provide the information about new synchronization tasks. The *t3* protocol has to be used, i.e.

    ```
    t3://<CMRF_HOST_IP>:<JNDI_PORT>
    ```

    (i.e. *t3://localhost:7010*).

In the tab *Notification* you can configure the format of the messages (e-mails) which are sent by the system
concerning DWH operations. These might be errors, success messages, or an information about an
unsuccessful operation.

The values are saved in the DWH notification properties (see Appendix C for details).

Fig. 5: ConSol*CM Admin Tool - DWH Configuration: Tab Notification

The following fields are available:

- **Protocol**

  Required - The protocol that is used to send the message, usually this is SMTP.

- **Host**

  Required - The e-mail server. You can enter a name (DNS-resolvable) or an IP address.

- **Port**

  Required - The port on the e-mail server where the mail daemon is listening.

- **User**

  Optional - User name if a user authentication is required at the e-mail server.

- **Password**

  Optional - Password of the e-mail user if a user authentication is required at the e-mail server.

- Tabs **Error/Successful/Unsuccessful**

  Here the e-mail parameters for e-mails that are sent by the system concerning the DWH 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 e-mail address for messages (maybe this is another FROM address than the one used for e-mails to customers and to engineers).

  - **To**

    The e-mail address of the recipient of the DWH messages. Initially this will be the ConSol*CM administrator e-mail address which has been entered during system set-up.

  - **Subject**

    The (e-mail) subject of the error/success/unsuccessful message.

  - **Description**

    The body (text) of the message.

Initialization of the DWH

When the basic configuration has been performed, the DWH initialization can be started. Press *Initialize* and follow the entries in the protocol panel. Be sure to have marked *Initialization* (radio button) in the top right corner to display initialization events.

Fig. 6: ConSol*CM Admin Tool - DWH Initialization

During this step, the database structure in the DWH is created with all tables and relations. No data will be transferred yet. Depending on the amount of tables, this might take some time.

If the DWH has been in operation and has to be set-up a second time, a reinitialization has to be performed. Check the *overwrite* option in order to delete the old database structure and create a new one. Then press *Initialize*.

First DWH Synchronization

To fill the data warehouse with the CM data for the first time, press *Transfer*. The initial transfer is started. Depending on the amount of tables, this might take some time (even several hours). You can follow the log entries by opening *Transfer* in the protocol panel.

DWH Synchronization During System Operation

If the DWH is running in *ADMIN* mode, the DWH administrator has to start the transfer manually by clicking on *Update*. Then all data that is supposed to be transferred, i.e. all data from fields with the *reportable = true* annotation that has been added or changed since the last transfer, is transferred. When the *Update* button has been clicked, all required operations will be created as tasks and all open tasks will be listed in the *DWH tasks* panel.

If a custom field or data object group 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 and/or customers is transferred.

You can follow the log entries for the DWH operation by opening the *Update* part of the protocol panel.

> ⊖ **Warning:**
>
> 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!

33.2.2 DWH Tasks

In this list you will find entries (one entry for one 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.
- custom field or data object group field annotations have been set to *reportable = true* and the check box *Automatic commit of administrative changes* has not been checked.

You can mark a task in the list and execute it manually.

If the check box *Automatic commit of administrative changes* has been checked, the tasks will be run automatically by the system.

33.2.3 DWH Troubleshooting and Repair

If any errors have occurred during initialization, transfer, or update, the log entries are displayed in the respective protocol panel.

You can also check the original log file under the following path:

- **JBoss:**

```
<JBOSS_HOME>\server\<CMRF_SERVER_NAME>\log\cmrf.log
```

- **Weblogic:**

```
<ORACLE_HOME>\Middleware\user_projects\domains\consolcm6_domain\cmrf-logs\cmrf.log
```

Please note that these are the standard paths. In ConSol*CM, *Log4J* is used. If you have configured a different path for your log files in the *log4j.xml* file, you will know where to find them.

Usually the log file and/or protocol panel entries give good hints at the initial reason for the transfer failure. If you cannot fix the problem and you have a maintenance contract with ConSol*, please contact our support team.

33.3 DWH-Related System Properties

A list of all system properties which are relevant for a specific DWH configuration can be found in section Appendix D - Important System Properties - Ordered by Area of Application, DWH.

33.4 Transfer Modes: JMS or DIRECT

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.

There are two possible modes:

- JMS mode
- DIRECT mode

33.4.1 Configuring the Transfer Mode

The mode is set using the system property *cmas-dwh-server*, *communication.channel*. Possible values are:

- "JMS": only possible value in versions prior to 6.8.5.0, default value before 6.9.4.1
- "DIRECT": database communication channel, exists since version 6.8.5.0 (has to be added manually in version 6.8.5.0 up to 6.9.4.0), default value since 6.9.4.1

33.4.2 JMS Mode

In JMS mode, the CM server sends messages to the CMRF by using the JMS-Queues *transfer*, *live*, and *control*. These JMS queues are located either in

- the table JBM_POSTOFFICE in the CM DB (in overlay mode)

or

- in a separate JBM_POSTOFFICE table in the DWH DB (standalone mode).

The CMRF server reads the messages from the JMS queues and writes the data into the DWH database schema.

33.4.3 DIRECT Mode

In DIRECT mode, the CM server sends messages to the CMRF by using direct access to database-tables (INT_CONTROL_QUEUE, INT_LIVE_QUEUE, INT_TRANSFER_QUEUE in the DWH database). JMS is not involved. CM is able to do that because in DIRECT mode, the CMRF datasource was made accessible for CM as well, i.e.

- in overlay mode: CM and CMRF use the same application server anyway
- in standalone mode: the CMRF database configuration file is also located on (copied to) the CM application server

The CMRF server reads the messages from the tables and writes the data into the DWH database schema.

34 ConSol CM Administrator Manual 6.10.2 - Services

34.1 Services

The Java EE application ConSol*CM comprises several services.

In the Admin Tool, you can manage two types of those services in the navigation group *Services*.

- CM Services, see section CM Services.
- ESB Services, see section ESB Services.

34.2 CM Services

CM Services are managed on the navigation item *CM Services* in the navigation group *Services*. On this tab you can start ▶ or stop ■ the individual services of the CM system, e.g. data indexing or DWH transfer.

Fig. 1: ConSol*CM Admin Tool - Services: CM Services

> ⚠ **Attention:**
>
> 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 Admin Tool and stores them in CM DB. The entries are used for the log protocol in the Admin Tool. See section Data Warehouse (DWH) Management.
- **DWH transfer service**
  Controls DWH transfer.

- **Job Executor**

  Controls the escalations for processes resp. workflows.

- **Kerberos v5 authentication provider**

  Required if Kerberos authentication is in operation.

- **ESB service**

  Retrieves incoming e-mails when the server is running in ESB/Mule Mail mode. This service is deactivated when NIMH is used as incoming mail module. Please see also section ESB Services.

- **NIMH**

  Retrieves incoming e-mails when the server is running in NIMH mode. This service is deactivated when ESB/Mule Mail is used as incoming mail module.

- **Remote client pooling**

  Controls that Web Clients get changes from Admin Tool.

- **Rest API service**

  Activates or deactivates REST (*Representation State Transfer* interface).

- **Server Session Service**

  Checks sessions and stops session when end of client or Admin Tool session end has been reached. 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 threads pool used for tasks execution.

- **Index changes notifier**

  Creates JMS (*Java Message Service*) messages with notifications that there has been one or more change(s) that concern(s) the index.

- **Index changes receiver**

  Reads 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).

34.3 ESB Services

- Introduction to ESB Services
- Starting and Stopping ESB Services Using the Admin Tool

ⓘ **IMPORTANT INFORMATION**

Since ConSol*CM version 6.9.4, there are two modes to receive incoming e-mails:

- **ESB/Mule** - this has also been available in all previous CM versions
- **NIMH** (New Incoming Mail Handler) - new in version 6.9.4

For all configurations/settings which are valid for both modes, no further notes are added. For all settings which vary depending on the mode, this will be explained in separate (i.e. ESB/Mule or NIMH specific) sections.

⚠ The ESB services are only active if ESB/Mule Mail is active! If your systems runs in NIMH mode, the navigation item *ESB Services* is disabled!

34.3.1 Introduction to ESB Services

The ESB services are in operation for incoming e-mails. Please see the following figure for the functions of the ESB services and also section E-Mail Scripts for a detailed description of the general principle of ConSol*CM mailing.

ESB stands for *Enterprise Service Bus* and ConSol*CM has integrated an ESB (Mule ESB^{TM}) as one of the application modules.
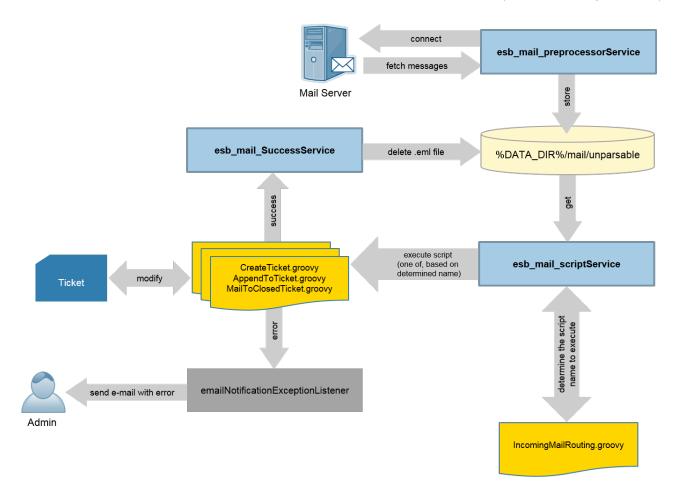
Fig. 1: ConSol*CM ESB Services

ESB services:

- **esb_mail_preprocessorService**
  Responsible for fetching e-mail messages from the configured incoming mailboxes. Retrieved messages are stored in the directory *%DATA_DIR%/mail/unparsable* as *.eml* files. Stopping this service will cause the ConSol*CM server to disconnect from configured e-mail servers. This means that e-mails will not be fetched and initially processed. After starting this service again, ConSol*CM will connect to configured e-mail servers and process all queued messages.

- **esb_mail_scriptService**
  This service calls the *IncomingMailRouting.groovy* script to determine the script name to execute. It can be one of *CreateTicket.groovy*, *AppendToTicket.groovy*, or *MailToClosedTicket.groovy*. Then the determined script is executed. On success, the *esb_mail_SuccessService* is called. On error, an e-mail with detailed cause is sent to the administrator. When this service is stopped e-mail messages will be retrieved from the mailboxes and stored in the directory *%DATA_DIR%/mail/unparsable*. Then the processing will stop. After the service is started again, the messages will be picked up from the *unparsable* directory and processed.

- **esb_mail_SuccessService**
  Responsible for deleting e-mail files that were processed correctly from the backup folder. Stopping this service will cause e-mail copies to remain in the backup folder (*%DATA_DIR%/mail/unparsable*) after processing.

> ⊖ **Warning:**
>
> When this service (*esb_mail_SuccessService*) is started again, it will delete all messages which were not removed when it was stopped.

34.3.2 Starting and Stopping ESB Services Using the Admin Tool

ESB Services are managed in the Admin Tool on the tab *ESB Services* in the navigation group *Services*. In this tab you can start and stop the sub-services of the *Enterprise Service Bus* (ESB). You should only change the service status upon request of CM consulting or CM support!

Fig. 2: ConSol*CM Admin Tool - Services: ESB Services

35 ConSol CM Administrator Manual 6.10.2 - Search in ConSol CM

35.1 Search in ConSol CM

ConSol*CM offers a powerful search engine.

Learn how to configure the CM system for your search requirements in section Search Configuration and Indexer Management.

Starting with 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 section Search Actions.

Result sets which have been retrieved in the Web Client can be exported as csv list. This feature has been added in CM version 6.10.1. It is explained in section CSV Export of Search Results.

35.2 Search Configuration and Indexer Management

ConSol*CM provides a powerful search 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 will 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.

35.2.1 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, please see section Templates for Customer Data for details. Please keep in mind that you can adapt the size of the result list using the system property *cmweb-server-adapter*, *globalSearchResultSizeLimit*, see Appendix C (System Properties) for details.
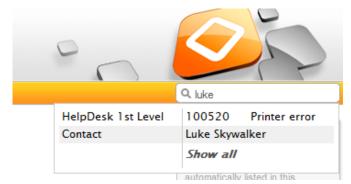
Fig. 1: ConSol*CM Web Client - Quick Search

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.

Fig. 2: ConSol*CM Web Client - Detailed Search

Please keep in mind that the size 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*. See Appendix C (System Properties) for an explanation.

> ⓘ **Information:**
>
> Please refer to the *ConSol*CM User Manual*, section *Searching for Tickets and Customers* to learn how to use the search functionality.

Configure 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 Data Object Group *ResellerCompanyData* which belongs to the Customer Data Model *ResellerModel* contains the following Data Object Group Fields:

company_name: order-in-result = 1

company_number: order-in-result = 2

Thus, in a search result list, e.g. a detail search which shows a result set with companies of a customer group which uses the ResellerModel, will contain the two colums company-name and companynumber.

Fig. 3: ConSol*CM Admin Tool - Setting the Annotation order-in-result for a Data Object Group Field

Fig. 4: ConSol*CM Web Client - Search Result Set with the Two Annotated Columns (order-in-result)

> Please note that when at least one column is annotated with the *order-in-result* annotation, columns which are not annotated with the *order-in-result* annotation will not be displayed in the search results.
>
> When no field is annotated with the *order-in-result* annotation, all fields are displayed in the search result list.

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).

Fig. 5: ConSol*CM Web Client - Autocomplete Search

Fig. 6: ConSol*CM Web Client - Suggestions for an Autocomplete Search

Starting with CM version 6.10.2, a search in all contacts of the CM database is also performed in the Ticket E-Mail Editor when the engineer statrs typing in the TO, CC or BCC field:

Fig. 7: ConSol*CM Web Client - Ticket E-Mail Editor: Automatic Search for Contacts as E-Mail Receivers

The number of characters which is required to initiate the automatic search can be configured using page customization. Use the page customization attribute *minMailInputLength* (Integer, Default 1) in the scope *mailTemplate*.

35.2.2 Fields Which Can Be Searched

For Custom Fields and Data Object Group Fields which should be searched, the annotation *field indexed* has to be set. See section Field Indexed Annotation. This makes the field available for the Quick Search and for the Detailed Search.

There are two types of fields concerning the search functionalities:

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
    - E-Mail
    - Firstname
    - Lastname
    - ID
- Ticket data
    - Attachment content
    - 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 datafields which have been added and configured by the CM administrators, the annotation *field indexed* has to be set to make the fields available in the search. A field which is indexed is available in

- the Quick Search
- the Detail Search
- all searches in Workflow and Admin Tool scripts, which use, for example, criteria classes/objects (e. g. *TicketCriteria*, *UnitCriteria*, *ResourceCriteria*), this concerns
    - scripts which have been manually added by the administrator

- scripts which are present by default (e.g. the script *createTicketgroovy*. In this script, the correct contact for a new ticket can only be set if the data object group field which is used to retrieve the correct contact or company is inddexed)

You as an administrator can define the system's behavior concerning search results. Depending on the values of the *fiel indexed* annotation, the search results for a contact might include all the tickets for the contact as well. The following table show 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 single data field, e.g. *name* and *e-mail* 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 you deal with two perspectives 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 annotation field-indexed.
> 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 annotation field-indexed. Depending on this value you might see only the objects you have searched for (e.g. only contacts) or also obkects which are related to the objects you have searched for (e.g. companies of the conatcts). Please see the following table for a detailed exaplanation.

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 the search - no other objects which are related to a ticket in any way will be retrieved - we recommend to accept the default value *transitive*			the ticket data will be NOT available for the search
CONTACT	- the contact data will be available for the search - the tickets of the contact will	- the contact data will be available for the search - the search for company	- the contact data will be available for the search - no other objects which are	the contact data will be NOT available for the search

Object type/ value of field indexed annotation	transitive	unit	local	not indexed
	also be found when you search for the contact • the search for company by its contact field is NOT possible	by its contact field is NOT possible	related to a contact in any way will be retrieved	
COMPANY	• the company data will be available for the search • 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 the search • the contacts of the company will also be found when you search for the company	• the company data will be available for the search • no other objects which are related to a company in any way will be retrieved	• the company data will be NOT available for the search
RESOURCE	• no difference between the three values			

Object type/ value of field indexed annotation	transitive	unit	local	not indexed
	- the resource data will be available for the search - no other objects which are related to a resource in any way will be retrieved - we recommend to accept the default value *transitive*			the resource data will be NOT available for the search

35.2.3 Administrator Tasks Concerning the Indexer

For the administrator it is important 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

Those tasks will be described in the following sections.

First, some background knowledge about the Indexer, the system which manages the search in ConSol*CM, is provided. This will help you as an administrator to look behind the scenes and understand the configuration in a better way.

35.2.4 Introduction to the ConSol*CM Indexer

The Indexer is a module of ConSol*CM which creates indexes. For each data field that should serve as search criterion (see section Fields Which Can Be Searched above), an index is created.

The indexes are stored on the hard disc in a sub-directory of the data directory that you have indicated during system set-up (the data directory is stored as a system property: *cmas-core-shared*, *data.directory*). The following picture shows an example for index files of a ConSol*CM installation (here used for a demo environment). The *demo_Datadir* is the data directory you have provided during set-up, all other directories are created automatically by ConSol*CM.
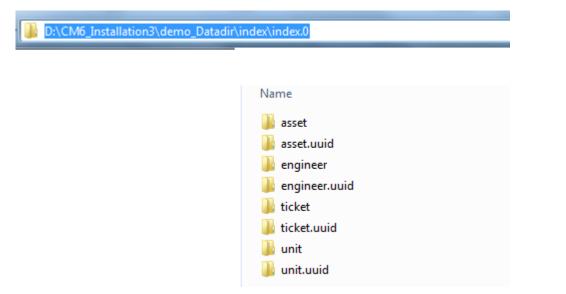
Fig. 8: ConSol*CM Indexer - Directory demo_Datadir

Please make sure that ...

- the data directory is always available for the ConSol*CM server system if it has been created on another server and is linked to (or mounted on) the application server.
- that the ConSol*CM datadir is part of the daily backup (and can be restored if required).

If the index directory should be corrupt or the index should not be available, the index can be rebuilt or repaired. Please see the following sections for details.

35.2.5 Indexer and Index Management Using the Admin Tool

Field Indexed Annotation

By default, the entire ticket text and all attachments are indexed. For all Custom Fields Data Object Group Fields and Resource Fields, the field(s) which should be indexed has/have to have the annotation *field indexed*. Please refer to the sections Custom Field Administration, Data Object Group Field Management and GUI Design for Customer Data and Setting Up the Basic Resource Model for details about setting annotations to Custom Fields (ticket data), Data Object Group 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 triggeerd by those values is provided in the table above.

Nested fields have to have all 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 Custom Fields which should be searched have to have the value *transitive* for the annotation *field indexed*.

Indexer Management: Navigation Item Index

Usually, you do not have to do anything concerning the Indexer. ConSol*CM will handle everything regarding the indexing automatically. There are only two cases where you have to perform manual administrative operations:

- You would like to change the configuration concerning the commitment of changes concerning the *field-indexed* annotation.
- 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.
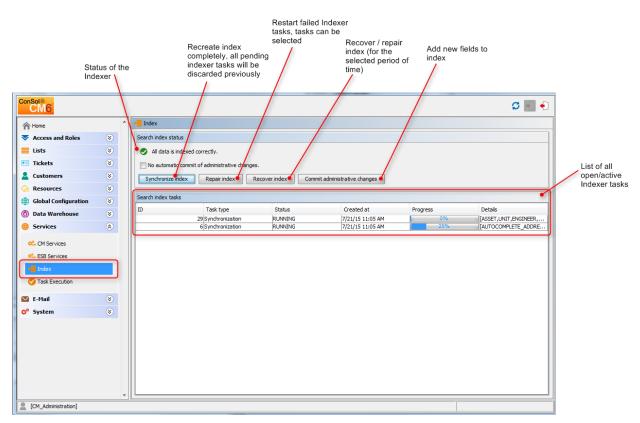
Fig. 9: ConSol*CM Admin Tool - Navigation Group Services: Navigation Item Index

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 is completed successfully it remains green. If there is any problem 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.

- **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 Custom Field or Data Object Group Field to *field-indexed* that was not indexed before. This has to be used if the check box *No automatic commit of administrative changes* has been selected. If the check box is inactive, the changes will be committed automatically when you have set the new annotation(s).

> ⚠ **Attention:**
>
> There is a difference in CM versions concerning the meaning of *administrative changes* - this influences the manual index management!
> In **versions prior to 6.9.3.0**, setting the annotation *field indexed* from *false* (or not set) to *true* is ⚠ **not** considered an administrative change. That means, when you have added the annotation *field indexed* = *true* for a Custom Field which was present before, you have to modify the index manually by using either *Synchronize index* or *Recover index*.
> **Starting with version 6.9.3.0**, setting the annotation *field indexed* from *false* (or not set) to *true* **is** considered an administrative change. That means, when you have added the annotation *field indexed* = *true* for a Custom Field which was present before, you have to
>   - modify the index manually by clicking on *Commit administrative changes* in case *No automatic commit of administrative changes* **is** set.
> **or**
>   - do nothing in case *No automatic commit of administrative changes* is **not** set.

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 whole index.
    - Triggered manually using the Admin Tool, *Synchronize index* command.
    - Before the start all other index tasks are removed.
  - **Administrative changes**
    - Created automatically when one of the following was updated:
      scope, queue, enum value, ticket function, ticket engineer, supported locale, role.
    - Processed automatically if *No automatic commit of administrative changes* option is unchecked.
    - Using *Commit administrative changes* command will start all administrative changes tasks.
  - **Retry**
    - Created automatically when error was encountered during index update process.
    - Holds information about entities which caused problems.

- Using *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 percentage of the task that has already been executed.
- **Details**

  A list of objects that are (re-)indexed in the task.

35.2.6 Indexer and Index-Relevant System Properties

The following system properties are also relevant for the Indexer, see following figure. Please refer to Appendix C for a detailed explanation of the Indexer system properties and Appendix D - Important System Properties - ordered by area of application, Indexer.

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.seconds	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.history	false
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.assets.per.thread.in.memory	200
cmas-core-index-common	indexed.engineers.per.thread.in.memory	300
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	b556aa28-df54-11e4-96aa-bbf9dddd9c29
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

Fig. 10: ConSol*CM Admin Tool - System Properties for Indexer

35.2.7 CM Indexer Services

For indexing, two ConSol*CM services are important:

- **Index changes notifier**

  Creates JMS (*Java Message Service*) messages with notifications that there has been one or more change(s) that concern(s) the index.

- **Index changes receiver**
  Reads JMS queue and starts update in Indexer.

Please see also section CM Services.

> ⚠ Stopping *index changes receiver* is safe. After restart it will pick up all of the missing changes from the persistent store.
>
> 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 to *RED* i.e. signal that the index needs a full synchronization.

35.2.8 Systems with More Than One Indexing Server

In case ConSol*CM is run in a cluster of application servers, each cluster node holds a local index in its *${cmas.data}/index/* directory. The index gets updated on the (single) master server, which is then polled by all slave servers to update their copies. As a result all indexing servers are eventually consistent, i.e. updates are not visible right away, usually with a small delay.

35.3 Search Actions

35.3.1 Introduction to Search Actions

It is possible to define actions on search results. These actions are presented like workflow activities for result lists of

- tickets
- data objects (= units, = customers)
- resources

Search (Result) Actions operate on the whole result set of a search. You can configure Search (Result) Actions to offer different bulk operations to the CM engineers, e.g. assigning all tickets in the result list to the current engineer (see following figure), sending a specific e-mail to all customers in the result list, or creating a new maintenance ticket for all PCs (resources) in the result list.

Fig. 1: 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 to all elements of the result set (if any)
> - the condition-script has returned *true* (if such a script exists)

Each Search Action is implemented based on Admin Tool scripts of the following script types.

- **Used for tickets:**
  - Bulk ticket action
  - Bulk ticket condition
- **Used for resources:**
  - Bulk resource action
  - Bulk resource condition
- **Used for customers:**
  - Bulk data object action
  - Bulk data object condition

> ℹ 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
> .

35.3.2 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

- Define and write the Admin Tool script
    1. mandatory: a script of type *bulk ticket action* (see following figure)
    2. optional: a script of type b*ulk ticket condition*

Fig. 2: ConSol*CM Admin Tool - Script for a Bulk Ticket Action

Example script:

```groovy
// 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();
```

> ⓘ   For a detailed explanation about how to write scripts for the ConSol*CM Action Framework, please
> read section Scripts for the Action Framework.

- Define the Search Action on the navigation item *Search Actions*, navigation group *Tickets*.

Fig. 3: ConSol*CM Admin Tool - Defining a New Ticket Search Action

Steps to Perform to Implement a Search Action for Resources

- Define and write the Admin Tool script
    1. mandatory: a script of bulk resource action
    2. optional: a script of type bulk resource condition

Example script which sets the next maintenance date of all retrieved HP printers to a date in two weeks:

```groovy
// 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();
```

> ⓘ For a detailed explanation about how to write scripts for the ConSol*CM Action Framework, please read section Scripts for the Action Framework.

- Define the Search Action on the navigation item *Actions*, navigation group *Resources*.

Fig. 4: ConSol*CM Admin Tool - Defining a Resource Search (Result) Action

- Assign the action to one or more Resource Type(s), navigation item *Data Models*, navigation group *Resources*, tab *Search Actions* for each Resource Type.

Fig. 5: ConSol*CM Admin Tool - Assigning a Search Action to a Resource Type

- Check if it works in the Web Client.

Fig. 6: ConSol*CM Web Client - Search (Result) Action on Resources, 1

Fig. 7: ConSol*CM Web Client - Search (Result) Action on Resources, 2

Steps to Perform to Implement a Search Action for Data Objects (= Units = Customers)

- Define and write the Admin Tool script
    1. mandatory: a script of resource bulk data object action
    2. optional: a script of type bulk data object condition, in this example we will work with a condition script

Example action script which creates a ticket for each contact in the result list:

```groovy
// 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();
```

> ⓘ   Please make sure that
>
>   • you only use Data Object Group 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, as a first step before implementing the Search (Result) Action script, you might want
>   to 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 let the Search (Result) Action only be displayed if the list has more than five
entries.

```
if (units.size() >= 5){
    return true
} else {
   return false
}
```

> ⓘ   For a detailed explanation about how to write scripts for the ConSol*CM Action Framework, please
>   read section Scripts for the Action Framework.

   • Define the Search Action on the navigation item *Actions*, navigation group *Customers.*
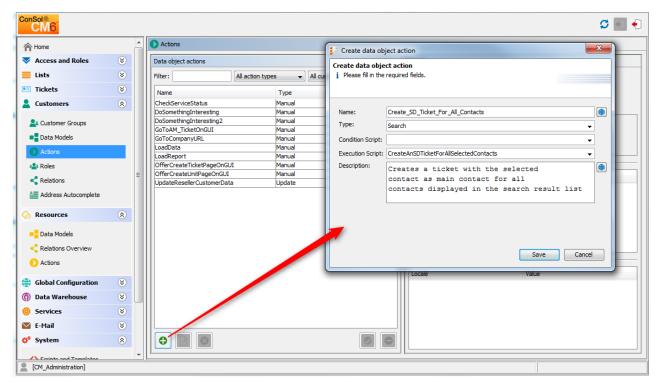
Fig. 8: ConSol*CM Admin Tool - Definining a Unit Search (Result) Action

- Add the condition script.

Fig. 9: ConSol*CM Admin Tool - Defining a Search Script Condition

- Assign the action to one or more Customer Groups, navigation item *Customer Groups*, navigation group *Customers*, 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 both offered in one table.

Fig. 10: ConSol*CM Admin Tool - Assigning a Search Action to a Contact Object within a Customer Group

- Check if it works in the Web Client.

Fig. 11: ConSol*CM Web Client - Search (Result) Action on Units (Here: Contacts), 1

Fig. 12: ConSol*CM Web Client - Search (Result) Action on Units (Here: Contacts), 2

Fig. 13: ConSol*CM Web Client - Search (Result) Action on Units (Here: Contacts), 3

35.3.3 Tips and Tricks for Search Action Admin Tool Scripts

Objects Implicitly Availalbe 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, neither do you have to instanciate the object from a class.

Objects which are implicitly (without further instanciation) in AT scripts:

All Search Action scripts

- pageSize (Integer)
- pageNumber (Integer)

Script types *Bulk ticket action* and *Bulk ticket condition*

- criteria (TicketCriteria)
- tickets (List<Ticket> for ticket Search Action)

Script types *Bulk resource action* and *Bulk resource condition*

- criteria (ResourceCriteria)
- resources (List<Resource> for resource Search Action)

Script types *Bulk data object action* and *Bulk data object condition*

- 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.

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();
```

> ⓘ   For a detailed explanation about how to write scripts for the ConSol*CM Action Framework, please read section Scripts for the Action Framework.

Fig. 14: ConSol*CM Web Client - SUCCESS Message after Search (Result) Action

Example for a negative feedback:

```
return actionScriptResultFactory.getPostAction(PostActionType.FAILURE, "cmweb.search.assigned").
withRefreshContent();
```

> ⓘ   For a detailed explanation about how to write scripts for the ConSol*CM Action Framework, please
> read section Scripts for the Action Framework.

Fig. 15: ConSol*CM Web Client - FAILURE Message after Search (Result) Action

35.4 CSV Export of Search Results

- Introduction
- Activating the CSV Export Functionality

35.4.1 Introduction

Starting with CM version 6.10.1, search result lists can be exported in csv format. This functionality is not activated by default but has to be activated using page customization.

In case the csv export is active in a CM system, the engineer can use the *Export table* option which is offered on search result pages for lists of

- tickets
- contacts
- companies
- resources

Fig. 1: ConSol*CM Web Client - CSV Export Option for a Ticket Search Result List

Available csv export options in the Web Client:

- **With column header**
  This adds the column labels of the result table in the first lines as a content description.
- **Timestamp as File Name**
  If this is selected, the preselected 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 which is configured to open csv files on the client machine. In the file there will be no pagination, the result lines will be listed uninterrupted. Icons in the result display will be ignored. Custom field string values will be quoted by standard double quotes.

35.4.2 Activating the CSV Export Functionality

The csv export functionality has to be activated selectively for distinct search result pages and is activated based on page customization.

Fig. 2: 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. Unless you have at
least once produced a result table, the *table* scope will not be available in the page customization
tree.

The following attributes are available:

- **enabled**

  Enable/Disable 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.

36 The Task Execution Framework (TEF)

36.1 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 CM script. The TEF tasks can be executed in an asynchronous manner. A new API has been added to ConSol*CM (in version 6.9.4.0) in order to provide the TEF extensions. TEF scripts can be started (i.e. the TEF API is available in):

- manually in the Admin Tool
- from workflow activity scripts
- from e-mail scripts
- from unit action scripts

A task is stored as Admin Tool Script of type *Task*. This script type is explained in the respective section below.
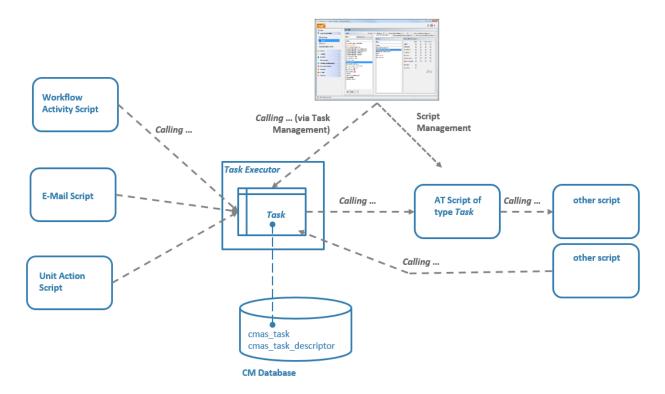
Fig. 1: ConSol*CM Task Execution Framework

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 of 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 Task Execution Using the Admin Tool. Here, a task can be started immediately, it can not 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
    - e-mail 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 or the start time of the task execution. Using the task descriptor, a newly defined task can be executed immediately or can be scheduled for a later execution time. This can be applied in scripts. Please see section Programming with Tasks for programming details.

36.2 Admin Tool Scripts of Type Task

36.2.1 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 provided when the script is created initially.

```
Methods in Admin Tool Script of Type Task

def onInitialize(taskDescriptor) {}
def onExecute(taskDescriptor) {}
def onError(taskDescriptor) {}
def onCancel(taskDescriptor) {}
```

36.2.2 Example Admin Tool Script of Type Task

```
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!")
}
```

36.2.3 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 *start.groovy.task.enabled* in module *cmas-app-*
> *admin-tool* has to be set to the value *true*. This property is not present in a default installation and
> has to be added manually.
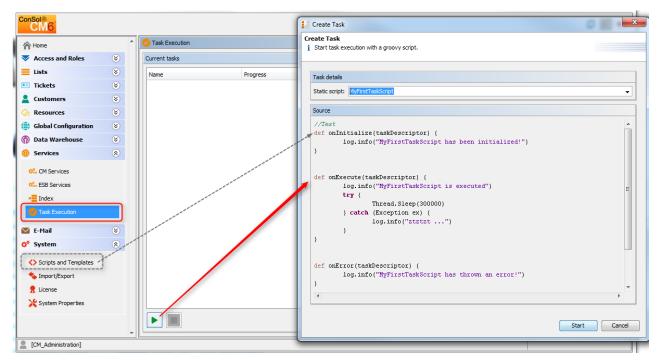
Fig. 2: ConSol*CM Admin Tool - Admin Tool Task

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* and the script will be executed 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 the explanation below.

When a task is running, a progress bar is shown. You can stop (cancel) the task using the *Stop* button.

Fig. 3: ConSol*CM Admin Tool - Running Task in the Admin Tool
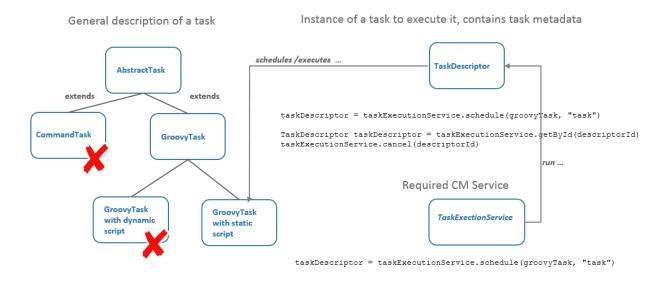
36.3 Programming with Tasks

36.3.1 Introduction

As of CM version 6.9.4, one type of tasks is available, the **Groovy Task with a static script**. This refers to the Admin Tool script which defines the task, see sections above.

The **Task Execution Service** (Groovy class TaskExecutionService, a singleton) runs in the background and scans the CM database for tasks (DB table cmas_task_descriptor) with the status INITIALIZED. Like all CM services it is implicitly available as object named *taskExecutionService* (see following examples). When the start date of the task has been reached, the task is started.

All parameters for the new task have to be set using the **task descriptor** (Groovy class TaskDescriptor), e. g. the start date of the task. The task descriptor will also provide information about the running task, like the task's progress.

Fig. 4: Some TEF Groovy Classes

36.3.2 Coding Examples

Creating a Task

```
Example for creating a task descriptor

GroovyTask groovyTask = new GroovyTask();
groovyTask.setStaticScript(scriptSourceService.getByName("someATScript.groovy"));
taskDescriptor = taskExecutionService.schedule(groovyTask, "task");
```

Canceling (Killing) a Task

Part 1: Create the task descriptor and save the Id somewhere.

```
Example for cancelling a task

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
```

Part 2, maybe also used at a later point of time:

```
Cancelling a task

taskExecutionService.cancel(myTaskDescriptorId)
```

Repeating a Task

If you set another execution date for a task after its job has been done, it will be rescheduled. So this code has to be used within the Admin Tool Task script.

```
Repeating a task

def onInitialize(taskDescriptor) {}

def onExecute(taskDescriptor) {
    //some code to execute
    ..........................
    //here, we set the new future execution date for the task, we also need to return a special
steering object
    taskDescriptor.setExecutionDate(new Date(new Date().getTime() + 15000));
    return new ExecutionSpecification().setRetryRequested(true);
}
```

```
def onError(taskDescriptor) {}
def onCancel(taskDescriptor) {}
```

Defining the (First) Execution Date

For a script which should not be started immediately, you can define a start time in the *onInitialize()* method.

```
(Pre-) Scheduling a task

def onInitialize(taskDescriptor) {
    taskDescriptor.setExecutionDate(yourDate)
}
```

Repeating a Task after an Error Occurred

```
Repeating a Task after an error occured

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) {}
```

Working with the ContextReference

Using the *ContextReference*, it is possible to see in which context a script has been called. For example, when a TEF-script is called from a workflow script in a certain activity and again in the process from another activity, it might be required to react differently. In this case, the *ContextReference* will help you to deduce from which workflow script the TEF-script has been called. 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* from where it has been called.

Use the methods

- *void GroovyTask.setContextReference(String ContRef)*
- *String TaskDescriptor.getContextReference()*

See also example #2 below.

36.3.3 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. The engineer does not have to do anything in between.

Fig. 5: ConSol*CM Web Client - Workflow Activity for Task Execution
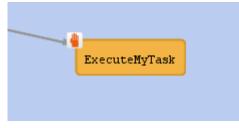
Fig. 6: ConSol*CM Process Designer - Workflow Activity for Task Execution

```
Workflow activity script for task execution

def myNewTask = new GroovyTask()
myNewTask.setStaticScript(scriptSourceService.getByName("MyFirstTaskScript"))
def myTaskDescriptor = taskExecutionService.schedule(myNewTask, "myTaskGroup")
myTaskDescriptor.setExecutionDate(new Date())
```

```
Log output from 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.
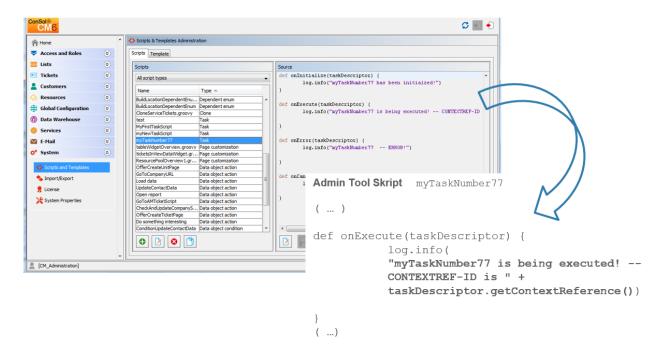
Fig. 7: ConSol*CM Admin Tool - Admin Tool Script for a TEF Task

In different workflow activities, different ContextReferences as identifiers are set. When the TEF script is called, it will always display (in the log output) the ContextReference specifically for the calling workflow activity.

```
import com.consol.cmas.common.model.task.*;

GroovyTask run77 = new GroovyTask();
run77.setStaticScript(scriptSourceService.getByName(
"myTaskNumber77"));
run77.setContextReference("77-1");

taskExecutionService.schedule(run77,
"myTaskGroup77");
```



```
( … )
run77.setContextReference("77-2");
```

```
:S NOW null
10.0.6.200:0-] Task Executor task-executor:10.0.6.200:0 is executing task: TaskDescrip
1ss com.consol.cmas.common.model.task.GroovyTask
10.0.6.200:0-] myTaskNumber77 is being executed    -- CONTEXTREF-ID is 77-1
10.0.6.200:0-] Task execution successful  removing my task : TaskDescriptor{group='myTask
1s.common.model.task.GroovyTask}
Removing timer after regular execution: workflow instance id: 199, timer name: default:


NTINENT IS NOW null
executor:10.0.6.200:0-] Task Executor task-executor:10.0.6.200:0 is executing task: TaskDescrip
  type=class com.consol.cmas.common.model.task.GroovyTask}
executor:10.0.6.200:0-] myTaskNumber77 is being executed!    - CONTEXTREF-ID is 77-2
executor:10.0.6.200:0-] Task execution successful  removing task : TaskDescriptor{group='myTask
onsol.cmas.common.model.task.GroovyTask}
```

Fig. 8: Workflow Activities and Log Output When One of the Activities Has Called the TEF Script

36.4 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 AdminTool
`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

37 ConSol CM Administrator Manual 6.10.2 - E-Mail Configuration

37.1 E-Mail Configuration

ConSol*CM includes a sophisticated e-mail interface. CM fetches e-mails from one or more mail servers and can send e-mails. Incoming as well as outgoing e-mails are included in the respective tickets, hence you always have access to the complete information about one case.

Basic CM e-mail functionalities are explained in the following sections:

- E-Mail
- E-Mail Backups

In case your company want to work with e-mail encryption, you can employ server and/or client certificates, see section:

- E-Mail Encryption

37.2 E-Mail

- Introduction to E-Mails in ConSol*CM
  - Sending E-Mails from ConSol*CM
    - Manual E-Mails
    - Automatic E-Mails
  - Receiving E-Mails with ConSol*CM
- E-Mail Configuration Using the Admin Tool
  - General E-Mail Configuration (Navigation Item E-Mail)
    - Administration
    - Incoming E-Mail
    - Outgoing E-Mail
  - E-Mail Modes: ESB/Mule Mail or NIMH
    - E-Mail Configuration Using ESB/Mule Mail
    - E-Mail Configuration Using NIMH
    - Switching from ESB/Mule Mail to NIMH
    - Running NIMH in a Clustered Environment
- E-Mail Duplication in the ConSol*CM Web Client

In this section, the navigation item *E-mail* in navigation group *E-Mail* in the Admin Tool will be explained. Furthermore, the e-mail-related system properties and the configuration for e-mail duplication will be shown.

37.2.1 Introduction to E-Mails in ConSol*CM

Before we explain the administration of e-mail accounts using the ConSol*CM Admin Tool, we will give you a short introduction on the subject *e-mail with ConSol*CM*, because this represents a core functionality of the application. ConSol*CM can send and receive e-mails.

Sending E-Mails from ConSol*CM

Manual E-Mails

E-mails can be sent manually by an engineer or automatically by the system. *Manual* e-mails are sent using the *Ticket E-Mail Editor*. In most systems, as default, the ticket's main customer is the receiver of the e-mail, but the engineer can select or type any other e-mail 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 e-mail templates and/or quote ticket text. Please see the *ConSol*CM User Manual* for a detailed introduction about working with the Ticket E-Mail Editor.

Fig. 1: ConSol*CM Web Client - Ticket E-Mail Editor

Automatic E-Mails

Automatic e-mails might 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 take care of the ticket.
    b. when the customer should receive an automatic confirmation that a ticket has been opened for him/her.
    c. when the customer should receive an automatic confirmation that his/her ticket 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 e-mail can be sent to every valid e-mail 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. e-mail error
    c. DWH synchronization (error or success)
        Usually, those e-mails are sent to the ConSol*CM administrator. However, for most special error cases a special receiver e-mail address can be configured using system properties. Please see section Appendix C (System Properties) for details.
3. Initiated by the CM system to remind engineers

a. When an engineer receives a ticket or a ticket is retrieved from the engineer, an e-mail can be sent to this engineer. This can be configured for every queue, see section Queue Administration.

Receiving E-Mails with ConSol*CM

The ConSol*CM system can fetch e-mails from one or more mailboxes (= e-mail accounts) on one or more e-mail server(s). The mailboxes are configured in the Admin Tool (E-mail configuration). Please keep in mind that ConSol*CM works with mailboxes here. Each of the mailboxes can be reached by at least one e-mail address. In certain cases, one mailbox might be used for more than one e-mail address. This can be of importance for writing e-mail scripts.

ConSol*CM acts towards the e-mail server like a regular e-mail client by fetching the e-mails using a standard mail protocol: IMAP(s), POP3(s). Depending on the mail server configuration and on the ConSol*CM system property *cmas-esb-mail*, *mail.delete.read*, the e-mails are deleted from the mailbox on the e-mail server after ConSol*CM has picked them up. The default setting is *mails are not deleted after pick-up*.

> ⚠️ In case you do not want ConSol*CM to delete e-mails from the e-mail server, please make sure to control the mailbox(es) manually to avoid a data overflow and server or performance problems.

All incoming e-mails are first stored in an incoming mail pool in ConSol*CM and are then processed in a chain of mail scripts. Please see section E-Mail Scripts for a detailed explanation of those scripts. When an e-mail cannot be processed, the administrator will receive a notification e-mail. The unprocessed e-mail is listed under E-mail Backups.

There are different possibilities concerning the default system behavior concerning an incoming e-mail:

- The subject of the e-mail 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 e-mail does contain a ticket number with a valid syntax (RegEx) and the ticket is still open:
  The e-mail is attached to the existing ticket.
- The subject of the e-mail 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 e-mail scripts (see section E-Mail Scripts), 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!

37.2.2 E-Mail Configuration Using the Admin Tool

> **ⓘ IMPORTANT INFORMATION**
>
> Since ConSol*CM version 6.9.4, there are two modes to receive incoming e-mails:
>
> - **ESB/Mule** - this has also been available in all previous CM versions
> - **NIMH** (New Incoming Mail Handler) - new in version 6.9.4
>
> For all configurations/settings which are valid for both modes, no further notes are added. For all settings which vary depending on the mode, this will be explained in separate (i.e. ESB/Mule or NIMH specific) sections.

General E-Mail Configuration (Navigation Item E-Mail)

In this navigation item you can set parameters for the e-mail connection.

Fig. 2: ConSol*CM Admin Tool - E-Mail

Administration

Admin e-mail:

Enter the e-mail address which shall receive general messages or warnings from the system. Multiple addresses separated by commas are possible, the total number of characters should not exceed 72. If there are many recipients we recommend using a mailing list on the mail server system.

Incoming E-Mail

The configuration of incoming e-mail is divided into two areas:

- **Configured accounts:**
  Here you can use a pop-up window to add or edit accounts from which e-mails 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 as system property *cmas-esb-mail*, *mail.incoming.uri*. Please see the e-mail properties section in Appendix C for detailed information. Required values are:
  - **Protocol**
    The protocol used to retrieve e-mails from the server. Supported protocols are IMAP4, IMAP4s, POP3, and POP3s. Please keep in mind that ConSol*CM acts towards the e-mail server like a regular e-mail 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 name or IP address of the e-mail server.
  - **Port**
    The port on the e-mail server where the mail daemon/service is listening.
  - **User name**
    The user name of the e-mail account.
  - **Password**
    The password of the e-mail account.

    > ⚠ **Attention:**
    >
    > Please keep in mind that one e-mail account can have more than one e-mail 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 e-mails, it might be required to use the e-mail address. The e-mail address is also required when you configure the REPLY-TO-address, the FROM-address, and queue-specific e-mail addresses! So be sure to use the correct parameter: mailbox or e-mail address!

- **Configuration:**
  - **Incoming e-mail subject pattern:**
    Describes the elements that the subject of an incoming e-mail has to contain in order to assign

this e-mail 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 (<Ticketnumber>)*.

- **Outgoing e-mail subject template:**
  Describes the pattern which is used to create the ticket ID in the subject of an outgoing e-mail. The template should be matched by the incoming e-mail subject pattern. Via the *Edit* button on the right you can modify the incoming e-mail subject pattern and outgoing e-mail subject template and verify if they match.
  Example: *Ticket (${ticketName})* would match the example RegEx above.

> ⚠ **Attention:**
>
> You can check if the pattern for the incoming e-mail subject pattern and for the outgoing e-mail subject template match by using the *Edit* button and the editor that is opened. Please make sure that the e-mail 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 e-mails. Valid for all e-mail pollers.
- **Error e-mail address:**
  E-mail 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 e-mail you might also want to check the e-mail-related system properties, see Appendix C (System Properties). Particularly, the polling interval (the time interval for fetching e-mails from the mail server, system property *cmas-esb-mail*, *mail.polling.interval*) might be of interest.

Outgoing E-Mail

The connection data for outgoing e-mails are set here:

- **Outgoing e-mail connector**
  Use the following format:

```
smtp://<IP address of mail server>:<port>
```

Example with standard port:

```
smtp://10.0.1.151:25
```

Example for SMTP servers with authentication:

```
smtp://test%40consol.com:ConSol12345@123.123.123.123
```

If a username and password have to be provided, the '@' has to be coded as %40.

For the example, the following parameters have been used:

protocol: SMTP

Username: test

Password: ConSol12345

Host/Mailserver: 123.123.123.123

Port: 25

E-Mail Modes: ESB/Mule Mail or NIMH

Starting with version 6.9.4, there are two modules for the retrieval of incoming e-mails:

- **ESB/Mule Mail**
  Which has been available since the start of ConSol*CM version 6.
- **NIMH**
  The New Incoming Mail Handler, which is a new module since version 6.9.4. This will be the only available incoming e-mail module in future CM versions.

In the current CM version, you as an administrator can decide which module to use: you can run CM in ESB /Mule mode **or** in NIMH mode. ESB/Mule and NIMH use different system settings which are stored as system properties. They are explained in the following two sections. The "switch" which changes the incoming e-mail mode is the system property *cmas-core-server*, *nimh.enabled* which can be set to *true* or *false.*

The sending of e-mails, i.e. the SMTP server configuration is not influenced by the incoming e-mail mode.

E-Mail Configuration Using ESB/Mule Mail

Mule is the internal ESB (Enterprise Service Bus) which is - besides other functions - used to retrieve incoming e-mails.

When ESB/Mule Mail is enabled, the following mechanisms apply:

- ESB/Mule Mail runs as services, see sections CM Services and ESB Services.
- E-mails are retrieved from the configured mailboxes.
- It is not possible to retrieve e-mails from the file system.
- E-mails which could not be processed are stored in a separate directory (*unparsable*) in the main ConSol*CM data directory and can be re-sent to the CM system, see section E-Mail Backups, ESB /Mule Mail.
- ESB/Mule Mail system properties have to be set. Please keep in mind that you can set most of the properties using the graphical user interface of the Admin Tool, navigation item *E-Mail* (see sections above).

The following figure provides an example of ESB/Mule Mail-related system properties. Please also refer to the detailed explanation of ESB/Mule Mail system properties in the System Properties Section in Appendix D (Important System Properties, Ordered by Area of Application)

| mail.cluster.node.id |
| mail.db.archive |
| mail.delete.read |
| mail.incoming.uri |
| mail.max.restarts |
| mail.mime.strict |
| mail.mule.service |
| mail.polling.interval |
| mail.process.error |
| mail.process.retry.attempts |
| mail.process.timeout |
| mail.redelivery.retry.count |

Fig. 3: ESB/Mule Mail System Properties

E-Mail 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.
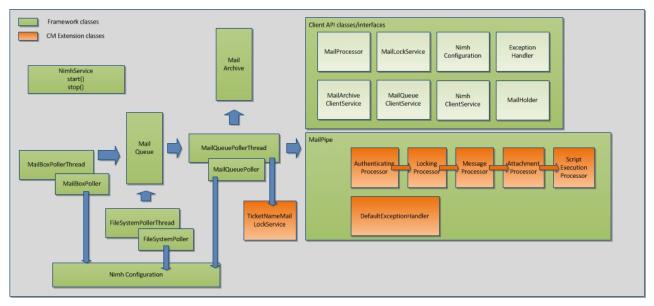
Fig. 4: NIMH Components

When NIMH is enabled, the following mechanisms apply:

- NIMH runs as a (single) CM service, see section CM Services. When NIMH is active, the ESB/Mule Mail services are disabled.
- E-mails from mailboxes on a mail server are retrieved by CM using the MailBoxPoller.
- E-mails can also be fetched from a data directory using the FileSystemPoller.
- All CM e-mails are stored in the CM database (nothing is stored in the file system).
- The MailQueuePoller retrieves the e-mails from the database and forwards them to the core CM system where the mail runs through the e-Mail script pipeline.
- NIMH uses e-mail scripts very similar to the ones used by ESB/Mule Mail, see section Admin Tool Scripts, Scripts of type e-mail (NIMH).
- E-mails which could not be processed are stored in a separate database table and can be re-sent to the CM system, see section E-Mail Backups, 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 the property is not set as 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 the system properties section in Appendix D (Important System Properties, Ordered by Area of Application).

NIMH System Properties

nimh.enabled	true

mailbox.1.connection.host	
mailbox.1.connection.password	**Customer-specific mailbox #1**
mailbox.1.connection.port	
mailbox.1.connection.protocol	
mailbox.1.connection.username	

mailbox.2.connection.host	
mailbox.2.connection.password	**Customer-specific mailbox #2**
mailbox.2.connection.port	
mailbox.2.connection.protocol	
mailbox.2.connection.username	

mailbox.default.connection.host	**Default values, are used when no**
mailbox.default.connection.password	**mailbox-specific equivalent is set**
mailbox.default.connection.port	
mailbox.default.connection.protocol	
mailbox.default.connection.username	
mailbox.default.session.mail.debug	
mailbox.default.session.mail.imap.connectiontimeout	
mailbox.default.session.mail.imap.timeout	
mailbox.default.session.mail.mime.address.strict	
mailbox.default.session.mail.pop3.connectiontimeout	
mailbox.default.session.mail.pop3.timeout	
mailbox.default.task.delete.read.messages	
mailbox.default.task.enabled	
mailbox.default.task.interval.seconds	
mailbox.default.task.max.message.size	
mailbox.default.task.max.messages.per.run	
mailbox.default.task.timeout.seconds	

mailbox.polling.threads.number	
mailbox.polling.threads.shutdown.timeout.seconds	**General NIMH settings**
mailbox.polling.threads.watchdog.interval.seconds	

queue.polling.threads.number	
queue.polling.threads.shutdown.timeout.seconds	
queue.polling.threads.watchdog.interval.seconds	
queue.task.error.pause.seconds	
queue.task.interval.seconds	
queue.task.max.retries	
queue.task.timeout.seconds	**MailQueuePoller settings**
queue.task.transaction.timeout.seconds	

filesystem.polling.threads.number	
filesystem.polling.threads.shutdown.timeout.seconds	
filesystem.polling.threads.watchdog.interval.seconds	
filesystem.task.enabled	
filesystem.task.interval.seconds	
filesystem.task.polling.folder	
filesystem.task.timeout.seconds	
filesystem.task.transaction.timeout.seconds	**FileSystemPoller settings**

Fig. 5: NIMH System Properties, 1

mail.attachments.validation.info.sender	
mail.attachments.validation.info.subject	
mail.db.archive	**Extended NIMH settings**
mail.error.from.address	
mail.error.to.address	
mail.on.error	
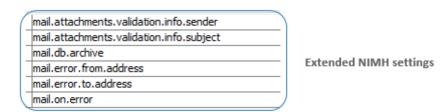
Fig. 6: NIMH System Properties, 2

> ⓘ **Information about System Property mail.on.error**
>
> In ConSol*CM versions up to 6.9.4.1, the default value for the property *mail.on.error* (module *cmas-nimh-extension*) is *false*.
>
> In ConSol*CM versions 6.9.4.2 and up, the default value for the property *mail.on.error* (module *cmas-nimh-extension*) is *true*.
>
> The value will be automatically set to *true* during an update from versions 6.9.4.1 and below to a version 6.9.4.2 and up. Please be aware that you may have to disable these mail notifications after an update in case you intentionally have set the property value to *false*.

Switching from ESB/Mule Mail to NIMH

If you want to switch from ESB/Mule ("old") mode to NIMH ("new") incoming mail mode, you have to perform the following steps, please also note the info box *Step-By-Step Guide for a Switch From ESB/Mule Mail to NIMH*.

- Copy the Admin Tool mail scripts and rename the new scripts with the same name with Nimh as Prefix, see also section Admin Tool Scripts, Scripts of type e-mail (NIMH).
    - AppendToTicket.groovy -> NimhAppendToTicket.groovy
    - CreateTicket.groovy -> NimhCreateTicket.groovy
    - IncomingMailRouting.groovy -> NimhIncomingMailRouting.groovy
    - MailToClosedTicket.groovy -> NimhMailToClosedTicket.groovy
- Adapt the new Admin Tool scripts, see also section Admin Tool Scripts, NIMH. There are new Groovy classes which have to be used to replace the old ones
    - import required NIMH classes
    - change methods according to the table displayed in the section Admin Tool Scripts, NIMH
    - change the section in mailRouting script for target handlers
    - control / set NIMH-specific system properties for mailboxes, see section above and system properties section in Appendix D (Important System Properties, Ordered by Area of Application)
    - switch system property to activate NIMH: *cmas-core-server.nimh.enabled = true*

> ⓘ **Step-by-step guide for a switch from ESB/Mule Mail to NIMH**
>
> If you have to switch your CM system from ESB/Mule Mail to NIMH, perform the following steps in
> the order listed here:
>
> 1. Prepare the NIMH environment:
>   a. Create the Admin Tool scripts as described in the section above
>   b. Adjust the system properties. The NIMH system properties will be automatically
>   added to the system configuration during an update from a previous version to CM
>   version 6.9.4. Only the required values have to be set. Mailbox properties are added
>   automatically for each mailbox which is added using the Admin Tool.
> 2. Shut down ESB/Mule Mail (stop all ESB services using the ESB Services and stop the ESB
>   service using the CM Services. Make sure CM finishes processing e-mails, review e-mail
>   backups and re-processing or delete the remaining e-mail backups. "Old" ESB/Mule e-mail
>   backups will not be displayed when you have switched to NIMH (but will be displayed when
>   you switch back to ESB/Mule Mail).
> 3. Start NIMH:
>   a. Set the system property *cmas-core-server.nimh.enabled = true*
>   b. Start the NIMH service

For a mapping of ESB Mail system properties to NIMH system properties, please refer to section Appendix
D (Important System Properties, Ordered by Area of Application).

Running NIMH in a Clustered Environment

In a cluster, NIMH can run only on one node. Here, the according system property has to be set: *cmas-core-server*, *nimh.enabled.CLUSTER_NODE_ID* = true
for example: cmas-core-server.nimh.enabled.1 = trueThis property replaces the general property *cmas-core-server.nimh.enabled = true*. On all other nodes NIMH and Mule (and ESB services) have to be disabled.
Furthermore, set the system property *cmas-nimh*, *mailbox.polling.threads.mail.log.enabled = true.* Without
this setting there is the chance that incoming e-mails may be processed several times by different cluster
nodes

37.2.3 E-Mail Duplication in the ConSol*CM Web Client

Please see explanations on the *Page Customization* page at showCloneOption and
appendOrReplaceOnClone.

37.3 E-Mail Backups

37.3.1 Introduction

Incoming e-mails which could not be processed are stored in a special store in the CM system (the location varies depending on the mode of the incoming mail configuration, see ESB and NIMH sections below).

You as an administrator can then try to re-send the e-mails to the system manually. The e-mails stored here can also be deleted, e.g. spam e-mails.

> **ⓘ IMPORTANT INFORMATION**
>
> Since ConSol*CM version 6.9.4, there are two modes to receive incoming e-mails:
>
> - **ESB/Mule** - this has also been available in all previous CM versions
> - **NIMH** (New Incoming Mail Handler) - new in version 6.9.4
>
> For all configurations/settings which are valid for both modes, no further notes are added. For all settings which vary depending on the mode, this will be explained in separate (i.e. ESB/Mule or NIMH specific) sections.

37.3.2 E-Mail Backups in the Admin Tool

All e-mails which could not be processed are listed in the navigation item *E-mail Backups* in the navigation group *E-Mail*. If your system is running in ESB/Mule Mail mode, the ESB/Mule backup e-mails are displayed, if the system is running in NIMH mode, the NIMH backup e-mails are displayed. Backups of both types are never ⚠ displayed at the same time!
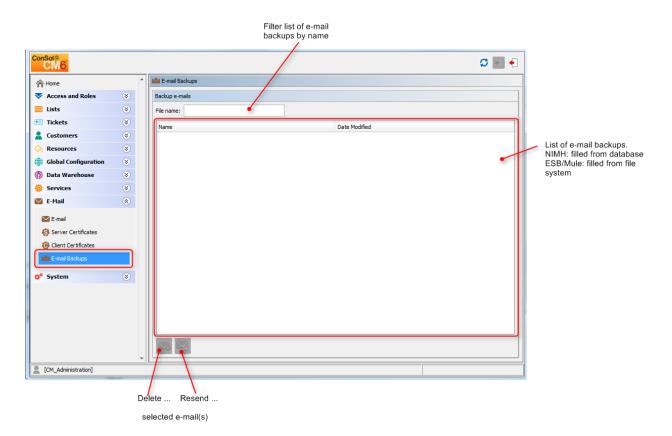
Fig. 1: ConSol* Admin Tool - E-Mail: E-mail Backups

The list panel for unparsable e-mails contains the following elements:

- **File name**
  This field provides a filter. When you enter the name or part of the name of e-mail files, only the matching file names will be displayed in the list.
- **Name**
  The name of the e-mail file (usually with an *.eml* extension).
- **Date modified**
  The last date when the file was modified. Usually the date when the e-mail has been stored on the CM server.

In an error-free ConSol*CM system, the list panel for unparsable e-mails should be empty. In case there are e-mails listed, an error has occurred concerning the processing of the incoming e-mail(s) in the system. Please see section E-Mail Scripts for a detailed explanation of the processing pipeline.

To delete an e-mail from the list, select the list entry and press the *Delete* button. Please keep in mind that the information will be lost! It will not be saved or transferred to CM in any way!

You can also try to re-send the e-mail to the processing pipeline (e.g. when a script was not working correctly and has been fixed now) by selecting it in the list and by pressing the *Resend* button.

37.3.3 Management of E-Mail Backups with ESB/Mule

Incoming e-mails which could not be processed are stored in the file system, in the following directory (as *.eml* files):

```
<CMAS_DATADIR>/mail/unparsable
```

E-Mails which were stored in the *unparsable* directory and were re-sent successfully are transferred from the *unparsable* directory to the following directory (as *.eml* file):

```
<CMAS_DATADIR>/mail/reimported
```

37.3.4 Management of E-Mail Backups with NIMH

Incoming e-mails which could not be processed are stored in the CM database, in the following table (as *.eml* files):

```
cmas_nimh_archived_mail
```

37.4 E-Mail Encryption

37.4.1 Introduction

Due to increasing security policies, it might be required to encrypt the e-mail traffic (including the e-mails which are sent and received by the ConSol*CM installation) using standard S/MIME encryption.

In order to enable ConSol*CM to work with encrypted e-mails, you first have to enable the e-mail encryption in the system:

1. **Mandatory:**
   Set the system property *cmas-esb-mail*, *mail.encryption* to *true*. It is set to *false* as default value. This is the basic configuration for the entire system to enable e-mail encryption.
2. **Optional:**
   Set the page customization property *mailEncryptionAvailable* to *true.* This activates the possibility in the Web Client to choose if the e-mail should be encrypted.

General Explanation about E-Mail Encryption in ConSol*CM

There are two types of certificates:

- **Personal Information Exchange Certificates**
  For incoming e-mails (here, server certificates are relevant).
    - The *Personal Information Exchange* certificate can be manually imported to the system from the *PKCS12* (.p12) files. This file contains the public and the private key for the corresponding e-mail address. If the certificate file is protected with a password, the administrator must enter it during the import process.
- **Security Certificates**
  For outgoing e-mails (here, client certificates are relevant).
  Security certificates can be imported into the system in two ways:
    - **Manually**
      By selecting the *X.509* (.cer or .crt) file.

- **Automatically**
  From the LDAP repository which holds it in the same format as for the file import. This can be done on demand during the e-mail sending.

> ⚠ **Attention:**
>
> The certificates treated here, are used for e-mail encryption only and **not** for the access of ConSol*CM (as e-mail client) to the e-mail server! This has to be managed by certificates which are stored in the security store of the application server.

Requirements

- The client certificate must contain the e-mail address of the customer in the attribute *SubjectDN* (*E=* or *EMAILADDRESS=*) or the *X509v3 Subject Alternative Name* element from the *Extensions* section of the certificate must contain the e-mail address.
- *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

If LDAP is configured, ConSol*CM will lookup the certificate for the requested contact in the LDAP repository. This is done in the following way:

1. Someone tries to send an encrypted e-mail.
2. The cryptography service is looking for a client certificate for the recipient.
3. If a client certificate is found, the e-mail is encrypted 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 e-mail is encrypted and sent.
6. If it is not found, the e-mail is sent unencrypted.

The following configuration properties have to be set for the certificate lookup via LDAP:

- ldap.certificate.basedn
- ldap.certificate.searchattr
- ldap.certificate.content.attribute

Please see section LDAP certificate parameters in Appendix C (System Properties) for details.

37.4.2 Certificates Management in the Admin Tool

In the Admin Tool, the tabs *Server Certificates* and *Client Certificates* in the navigation group *E-Mail* are used to configure the CM environment for e-mail encryption.

Server Certificates

Server certificates are used to decipher incoming e-mail messages and also to encrypt outgoing e-mail messages. They each contain the public and the private key for the given e-mail address. If you define an incoming e-mail account (see section above), you have to upload a server certificate for that e-mail address (or for all e-mail addresses covered by this mailbox) to be able to receive encrypted messages. 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 e-mail 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 on 🔼 and use the file browser to find the required certificate. The certificate is checked 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 a user. It allows encrypting messages going to that user.

When you open the tab *Client certificates*, a list of all existing client certificates is displayed. To add a new client certificate, click on 🔼 and use the file browser to find the required certificate. The certificate is checked 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 filename extensions for *X.509* certificates are:
    - .cer
    - .crt
    - .der
    - .pem

Fig. 1: ConSol*CM Admin Tool - Pop-up Window for Adding a Client Certificate

Here are some example use cases:

1. An engineer uses the ConSol*CM Web Client and writes an encrypted e-mail using the Ticket E-Mail Editor. When he/she presses the *Send* button, the ConSol*CM system looks up the receiver address in the list of mail addresses under *Client certificates* and uses the public key of the recipient to encrypt the outgoing e-mail. If ConSol*CM cannot find a matching certificate (the e-mail address is not mentioned in the list), the e-mail is loaded from LDAP. If this does not work either, the e-mail is sent unencrypted. If one of the recipients is the same as one of the incoming e-mail accounts, then also the server certificate will be used to encrypt that message.
2. ConSol*CM receives an e-mail and checks the TO-address. If this is found in the list under *Client 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 E-Mails

Choosing if E-Mails Should be Sent Encrypted from the Web Client

If the page customization property mailEncryptionAvailable has been set to *true*, a check box *Send encrypted* is available in the Ticket E-Mail Editor in the Web Client. Thus, the user can choose if the e-mail should be sent encrypted.

Fig. 2: ConSol*CM Web Client - Send Encrypted E-Mail

Sending an Encrypted E-Mail from the Workflow

An encrypted e-mail can be sent by using the method *enableEncryption()*. Please see the *ConSol*CM Process Designer Manual* for a detailed explanation.

Sending Encrypted E-Mails by Default

If the system property *cmas-esb-mail*, *mail.encryption* is set to *true*, all outgoing e-mails from the workflow and Web Client are encrypted by default.

If users would like to send selected e-mails unencrypted, they can uncheck the check box *Send encrypted* in the Web Client. For e-mails sent by the workflow the method *disableEncryption()* has to be used.

38 ConSol CM Administrator Manual 6.10.2 - Script and Admin Tool Template Administration

38.1 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 concerning this topic. However, various scripts are also stored in the Admin Tool, in the *Scripts* section. This will be explained in section Admin Tool Scripts.

Templates are also stored in several locations:

- Accessible via Web Client:
    - In the Template Manager (e-mail templates)
    - In the Doc Template Manager (MS Word or OpenOffice templates in CM.Doc)
- Accessible via Admin Tool:
    - In the *Scripts and Templates* section
        - Some special e-mail templates
        - Templates for customer data
        - Templates for resource data
        - Password reset template
        - CM.Phone templates
        - <more templates depending on customizing>

For explanations of the work with e-mail templates using the 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.

38.2 ConSol CM Administrator Manual 6.10.2 - Admin Tool Scripts

38.2.1 Admin Tool Scripts

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* is opened initially.

Fig. 1: ConSol*CM Admin Tool - Scripts & Templates Administration

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. 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:
  - **Bulk ticket action**
    This script type is part of the Action Framework, namely the Search Actions and is described in section Search Action Scripts.
  - **Bulk ticket condition**
    This script type is part of the Action Framework, namely the Search Actions and is described in section Search Action Scripts.
  - **Bulk resource action**
    This script type is part of the Action Framework, namely the Search Actions and is described in section Search Action Scripts.
  - **Bulk resource condition**
    This script type is part of the Action Framework, namely the Search Actions and is described in section Search Action Scripts.
  - **Bulk data object action**
    This script type is part of the Action Framework, namely the Search Actions and is described in section Search Action Scripts.
  - **Bulk data object condition**
    This script type is part of the Action Framework, namely the Search Actions and is described in section Search Action Scripts.

- **Calendar integration**

  Scripts of this type provide the connection information for the integration of MS Exchange calendars. See section MS 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.

- **Data object action**

  Script which is executed when a Data Object Action has taken place, see section Action Framework - Data Object Actions for details.

- **Data object condition**

  Script which is executed to evaluate if a Data Object Action should be offered in the Web Client, see section Action Framework - Data Object 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*, a structure that provides hierarchical lists. Please see section Scripts of Type Dependent Enum for details.

- **E-mail**

  Scripts of this type are used to manage incoming and outgoing e-mails. Please see section Scripts of Type E-mail 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 The Resource Pool Dashboard for details.

- **PostActivityExecutionScript**

  The (optional) script which is executed after every manual workflow activity. See section PostActivityExecutionScript for details.

- **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 if a Resource Action should be offered in 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).

- **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 that is selected in the list on the left is displayed. Here you can write the script source code when you have selected the *edit* mode.

The Source Code Editor

The Source Code Editor provides an editing panel with syntax highlighting. You have to check for correct code yourself.


```
Source
import com.consol.cmas.common.model.customfield.UnitReferenceField
import com.consol.cmas.common.model.customfield.meta.FieldKey
import com.consol.cmas.common.model.content.AttachmentEntry
import com.consol.cmas.common.model.content.ContentEntryCategory
import com.consol.cmas.common.model.content.MailEntry
import com.consol.cmas.esb.mail.MailContextService
import javax.activation.DataHandler
import org.mule.transport.email.MailProperties
import javax.mail.internet.MimeUtility

if(mailLog.isDebugEnabled()) {
    mailLog.debug("Creating ticket from message $msg")
}

String contactUnitType = "customer"            // type of contact unit
String contactEmailFieldName = "email"         // name of contact unit email string field
String contactNameFieldName = "name"           // name of contact unit name string field
String customerGroupName = "CustomerGroup"     // name of contact unit customer group
String contactCompanyRefName = "companyRef"    // name of contact unit company reference field

String companyUnitType = "company"             // type of unit which represents company
String companyNameFieldName = "name1"          // name of company unit name string field
String companyNameFieldValue = "ConSol* GmbH"  // name of company referenced by contact

String ticketQueueName = "HelpDesk_1st_Level"; // name of queue for created ticket
String ticketPriorityFieldGroupName = "helpdesk_standard"  // name of queue field group
String ticketPriorityFieldName = "priority"        // name of queue enum field
String ticketPriorityFieldValue = "normal"         // value of ticket priority enum field


findContact = {
    String email = mailContextService.extractMailFromField(msg)
```

Fig. 2: ConSol*CM Admin Tool - Source Code Editor

In the lower section of the Source Code Editor, there are the following buttons:

- **Edit**
  Press 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 changing something accidentally.

- **Quit and save**
  Save the script and quit *edit* mode, i.e. switch to *read-only* mode again.

- **Quit without saving**
  Switch to *read-only* mode again, without saving the changes you might have made to the source code.

- **Open script from file**
  Import a script from a file. This will import the file and display it in the Source Code Editor. No checks are performed.

- **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 you are working on.

Script Types

In the following section, the possible script types are explained. Some examples are provided to give you an impression of the potential of scripts in the system.

The following script types will be 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 the respective sections.

- Clone
- Default values
- Dependent enums
- E-mail
- Workflow
- PostActivityExecutionScript

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.
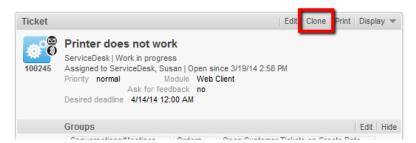
Fig. 3: ConSol*CM Web Client - Clone Option in Ticket Menu

The following two pictures show the cloning of a ticket without a clone script.

Fig. 4: ConSol*CM Web Client - Original Ticket

Fig. 5: ConSol*CM Web Client - Cloning the Ticket (without Clone Script)

> ⓘ **Information:**
>
> 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 it was set)
> - the values of all Custom Fields (Ticket Data Section and Group 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
>   - e-mails
>   - attachments
> - the ticket history
> - additional engineers
> - ticket relations

In case the queue where the ticket is located uses a *Clone* script (see section Queue Administration), this script will be executed when the engineer has clicked on *Clone*. The script can be used similar to a *Default values* script (see respective section below): you can preset values in the ticket which will be created. In the cloning process the values are pre-filled in the Custom Fields in the Web Client, i.e. before the ticket is generated. The engineer can change the values if required.

> ⓘ **Information:**
>
> 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 wrong service dates in the (cloned) *Service Desk* ticket.

```
Clone Script to Reset Custom Field for Desired Deadline

ticket.set("serviceDesk_fields.desiredDeadline", null)
```

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 offered to the engineer (see following image).

Fig. 6: ConSol*CM Web Client - Cloned Ticket (with Clone Script Assigned to the Queue)

Scripts of Type Default Values

Sometimes it is required that a data field of a ticket has a certain value when the ticket is to be created, i.e. when the engineer presses *New ticket* and the respective form is opened in the Web Client, one or more value(s) should be preset. This saves the engineer from setting 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 specifically defined for each queue. Please see the following example.

Fig. 7: 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 Custom Field is to be found (in which Custom Field Group and under which name) within the *Custom Field Administration*. See section Custom Field Administration for details.

> **ⓘ Information:**
>
> 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!

Fig. 8: ConSol*CM Admin Tool - Custom Field Administration

Since *priority* is an *enum* field (i.e. an ordered list), we have to check the possible values that can appear in the list and we have to look for the required default value within the *Enum Administration*.

Fig. 9: ConSol*CM Admin Tool - Priority Values in Enum Administration

The group, the field, and the correct value can then be used in the respective Java method in the new script.

Fig. 10: ConSol*CM Admin Tool - Include Group, Field, and Value in Script

In the *Queue Administration* we have to assign the script to the queue where the default value should be used.

Fig. 11: ConSol*CM Admin Tool - Assign Default Values Script to Queue in Queue Administration

When the engineer starts the *create ticket* operation now in the Web Client, the default value *normal* will be set in the *Priority* field.

Fig. 12: ConSol*CM Web Client - New Ticket with Default Value

> ⓘ **Important information:**
>
> Please keep in mind that for every queue there can be only one *Default values* script. So 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.
>
> In case the same default value has to be set in different queues and is set together with other default values, this has to be coded in one script for each queue also.

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 of the first script will be persistent.

If a *Default values* script should overwrite already existing values, the *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()
```

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 on the sub-level, is opened. There do not have to be sub-lists for every list entry, so in graph notation, the list might be a combination of nodes and leaves. A dependent enum script is assigned to the Custom Field Group where it is required.

Please see the following example:
In help desk tickets a category can be selected. When the user has selected *hardware* a hardware sub-list is displayed, when the user has selected *software*, the software sub-list is displayed. All Custom 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 and more complex.

The *Dependent enum* script is placed in the Admin Tool. Please ask our CM consultants for support when creating and/or editing the script since this is a rather complex task.

Fig. 13: ConSol*CM Admin Tool - Dependent Enum Script

The *Dependent enum* script is assigned to the Custom Field Group where it is required.

Fig. 14: ConSol*CM Admin Tool - Assign Dependent Enum Script to Custom Field Group

In the Web Client the engineer only sees the sub-list of the value selected in the first level list.

Fig. 15: ConSol*CM Web Client - Sub-list of Category Hardware

Fig. 16: ConSol*CM Web Client - Sub-list of Category Software

Scripts of Type E-Mail

Scripts of this type are used for several functionalities. 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.

ℹ️ **IMPORTANT INFORMATION**

Since ConSol*CM version 6.9.4, there are two modes to receive incoming e-mails:

- **ESB/Mule** - this has also been available in all previous CM versions
- **NIMH** (New Incoming Mail Handler) - new in version 6.9.4

For all configurations/settings which are valid for both modes, no further notes are added. For all settings which vary depending on the mode, this will be explained in separate (i.e. ESB/Mule or NIMH specific) sections.

E-Mail Scripts for the Processing of Incoming E-Mails in ESB/Mule Mail Mode

When an e-mail is received by ConSol*CM, it is processed by several scripts, see following figure.
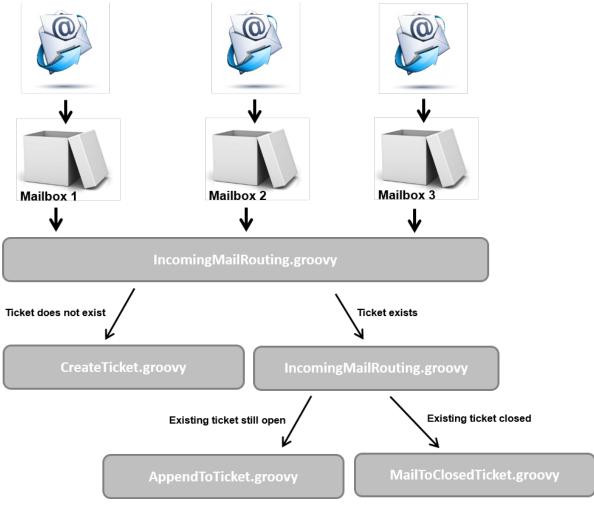
Fig. 17: ConSol*CM Admin Tool - E-Mail Scripts (ESB/Mule Mail)

- **IncomingMailRouting.groovy**
  Standard script. This is the first script that is executed when an e-mail comes in. Here, it is decided if a new ticket has to be created or if the e-mail concerns an existing open ticket (then *AppendToTicket. groovy* is executed) or if it concerns a closed ticket (then *MailToClosedTicket.groovy* is executed). This script does not require any changes to adapt it for a customer-specific environment.

- **CreateTicket.groovy**
  Standard script which is responsible for the creation of a ticket when an e-mail has been received in one of the ConSol*CM-configured mailboxes (see section E-Mail for details). If the ticket subject does not match the regular expression for appending the e-mail to an existing ticket, this script is performed. All e-mails which are received by ConSol*CM (and have not been assigned to an existing ticket) are processed here, no matter from which mailbox they have been collected. In the script, the default queue for incoming e-mails has to be defined, more values of Custom Fields can be defined (like e.g. the default priority for e-mail tickets). Or decisions can be made in which queue the new ticket should be created, depending on the TO-address or other parameters. So usually, this script has to be adapted heavily. Please ask a CM consultant for support concerning this task.

- **AppendToTicket.groovy**
  Standard script which is responsible for appending an e-mail to a ticket that already exists. The

assignment of the e-mail to the ticket is performed using the comparison between the ticket subject and the required regular expression. Please see section E-Mail for a detailed explanation of this context. Usually no changes are required for this script.

- **MailToClosedTicket.groovy**
  Standard script which is responsible for handling the e-mail when it concerns a closed ticket. The default system behavior is to create a new ticket for the customer (sender of the e-mail) and to create a reference to the old/closed ticket. So usually, no changes are required in this script.

E-Mail Scripts for the Processing of Incoming E-Mails in NIMH Mode

When an e-mail is received by ConSol*CM, it is processed by several scripts, see following figure.
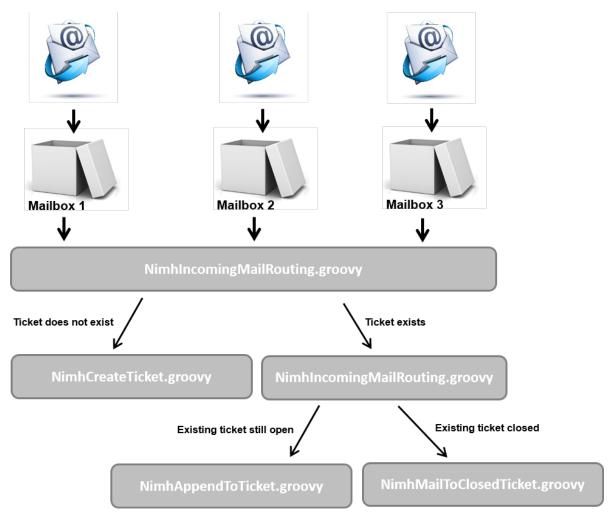
Fig. 18: ConSol*CM Admin Tool - E-Mail Scripts (NIMH)

- **NimhIncomingMailRouting.groovy**
  Standard script. This is the first script that is executed when an e-mail comes in. Here, it is decided if a new ticket has to be created or if the e-mail concerns an existing open ticket (then *NimhAppendToTicket.groovy* is executed) or if it concerns 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 e-mail has been received in one of the ConSol*CM-configured mailboxes (see section E-Mail for details). If the ticket subject does

not match the regular expression for appending the e-mail to an existing ticket, this script is performed. All e-mails which are received by ConSol*CM (and have not been assigned to an existing ticket) are processed here, no matter from which mailbox they have been collected. In the script, the default queue for incoming e-mails has to be defined, more values of Custom Fields can be defined (like e.g. the default priority for e-mail tickets). Or decisions can be made in which queue the new ticket should be created, depending on the TO-address or other parameters. So usually, this script has to be adapted heavily. Please ask a CM consultant for support concerning this task.

- **NimhAppendToTicket.groovy**
  Standard script which is responsible for appending an e-mail to a ticket that already exists. The assignment of the e-mail to the ticket is performed using the comparison between the ticket subject and the required regular expression. Please see section E-Mail for a detailed explanation of this context. Usually no changes are required for this script.

- **NimhMailToClosedTicket.groovy**
  Standard script which is responsible for handling the e-mail when it concerns a closed ticket. The default system behavior is to create a new ticket for the customer (sender of the e-mail) and to create a reference to the old/closed ticket. So usually, no changes are required in this script.

Differences between Scripts in ESB/Mule Mail and NIMH

When NIMH is enabled, other Groovy classes have to be used in the e-mail scripts than in ESB/Mule Mail mode. Please see the mapping in the table below.

Mule	NIMH
mailContextService	mailSupportService
msg	mailHolder
mailLog	mailLog
spring cm services	spring cm services
all places where *mailHolder* ist used	pipeContext (used as parameter in *mailSupportService* invocations)

The steps you have to perform when you want to switch your ConSol*CM system from ESB/Mule Mail mode to NIMH mode are described in section E-Mail - Switching from ESB/Mule Mail to NIMH.

E-Mail Scripts for Outgoing E-Mails

For every queue, an *E-mail* script can be configured. Please see section Queue Administration for an explanation how to configure this. An e-mail 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 *E-mail* script you can change or set queue-specific settings for the outgoing e-mail. 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 *E-mail* script is the following script which is part of the CM default application:

- **ChangeOutgoingMail.groovy**

  Standard script that is not in operation but serves as a template for outgoing *E-mail* scripts. You might want to use them to configure queue-specific *E-mail* scripts.

Within the outgoing *E-mail* script, the Java object *mailEntry* is implicitly available as object *mail*. You have to set all required attributes for the outgoing e-mail using the *mail.setAttribute()* or *mail.setAttributes()* methods.

Example:

```
def queueReplyAddress = "serviceteam@mycompany.com"
// you might also use system properties for the queue-specific e-mail addresses and fetch an
// address using the configurationService!
mail.setAttribute('Reply-to', queueReplyAddress)
```

Common e-mail attributes are:

- BCC
- From
- Reply-to
- To
- CC
- Subject

In case you would like to read a very detailed description of the e-mail format, please refer to RFC 5322.

⊖ When you work with the configuration of the REPLY-TO e-mail address, please note the following technical behavior of ConSol*CM and adapt your system accordingly!

The technical background:

There are three 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-E-Mail Editor in the Web Client. If it is really the effective REPLY-TO address in an e-mail depends on the configuration in the queue-specific outgoing e-mail script. See next topic.
2. The **REPLY-TO address which is set in a queue-specific outgoing e-mail script**. Since the outgoing e-mail script is the last instance which processes an outgoing e-mail, 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, it will not really be used (but it will be displayed in the Ticket E-Mail Editor which might cause some confusion! What that means for your system configuration is explained in the next section).
3. The **e-mail address of the current engineer** (the engineer who is logged in to the Web Client). This personal e-mail address is used as REPLY-TO address for e-mails from the Web Client if neither the *mail.reply.to* property is set nor a queue-specific outgoing e-mail script is configured.

In the Web Client, in the ticket history, the REPLY-TO address which was really used is always displayed for outgoing e-mails. So even in case there should be a difference between the address which was displayed in the Ticket E-Mail Editor (the *mail.reply.to* property) and the REPLY-TO address which was really used (the REPLY-TO in the queue-specific outgoing e-mail 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 e-mail script

or

- use the *mail.reply.to* system property.

However, since the e-mail communication should take place via ConSol*CM and not using personal e-mail addresses, one of the two system settings mentioned above should be used to prevent CM from using personal e-mail addresses as REPLY-TO. The latter would automatically lead to customer e-mails being sent to an engineer's personal e-mail account instead of CM.

What that means for your system configuration:

1. The simplest way to set a REPLY-TO address by using the *mail.reply.to* system property. It will be displayed in the Ticket E-Mail Editor and will be the effective REPLY-TO address.
2. If you work with queue-specific outgoing e-mail scripts, you have to configure a script for each queue! The *mail.reply.to* system property should then be left empty! You can
   a. write queue-specific scripts which are only used in one queue (e.g. to set a team-specific REPLY-TO address), e.g. *ChangeOutgoingMail.groovy_Queue1, ChangeOutgoingMail.groovy_Queue2, ... ChangeOutgoingMail.groovy_Queue_n.*
   b. write a standard outgoing e-mail script which is configured for all queues, e.g. *ChangeOutgoingMail.groovy_Queue_standard*
   c. combine a and b, i.e. to set a standard script for most queues and configure specific scripts for specific queues

What that means when you work with workflow scripts which send e-mails:

(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 e-mail.
- Use the *Mail* object when the queue-specific script should be used: e.g. *mail. useDefaultScript()* . This will overwrite the *mail.reply.to* property! Hence, use either property or script.

If neither the system property nor the queue-specific outgoing e-mail 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 e-mail client.

Scripts of Type Workflow

Scripts of that 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 workflow(s). It is easier, less error-prone, and less time-consuming to store the scripts at one central location (Admin Tool) and just reference them in the workflow(s) than to edit the same code at 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 in operation at once whereas when editing a workflow it has 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
("initializeEscalationTriggers.groovy"))
```

In the Admin Tool, the respective script is stored:

Fig. 19: ConSol*CM Admin Tool - 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*.

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 GUI after the workflow activity has been executed. From a user's (engineer's) point of view, the Web Client GUI *jumps* to the next ticket. The latter can be a child ticket or another 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*, see Appendix C (System Properties).

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 will happen)
- What should happen?

Starting with CM version 6.10.2, 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 detail 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	

Fig. 20: ConSol*CM Admin Tool - Property for Definition of postActivityExecutionScriptName

Fig. 21: ConSol*CM Admin Tool - postActivityExecutionScript

The PostActivityExecutionScript also allows to *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)
}
```

Another **example** which also shows the Web Client behavior:

Fig. 22: ConSol*CM Admin Tool - PostActivityExecutionHandlerScript

The script:

```
switch(activity.name){
  case 'defaultScope/Service_Desk/New_ticket/Open_Page_of_Main_Contact':
    def main_cont = ticket.getMainContact()
      return main_cont
  // ( ... )
}
```
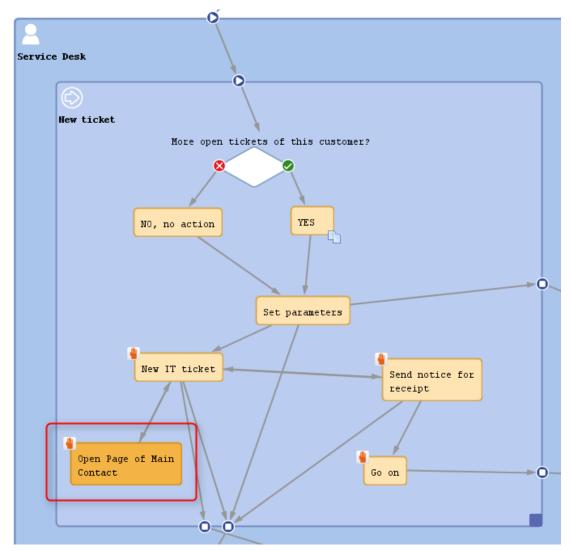
Fig. 23: ConSol*CM Process Designer - Activity Which Will Be Controlled Using the PostActivityExecutionScript

Fig. 24: ConSol*CM Web Client - Ticket with Workflow Activity Which Will Jump to Contact Page

Fig. 25: ConSol*CM Web Client - Contact Page Which Has Been Opened Directly from Ticket

38.2.2 Scripts for the Action Framework

Introduction

The ConSol*CM Action Framework offers the possibility 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 CM version 6.10) of three components which are treated in the respective manual sections in detail.

- **Unit Actions (Data Object Actions)**

    Actions based on contact or company objects, see section Action Framework - Data Object 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 Search Actions.

In this section here, you will learn more about Action Framework-specific programming.

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.

> ⓘ **Information:**
>
> 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 has returned *true*

The following Action Framework-related script types are available:

- For **Data Object Actions (= Unit Actions):**
  - Data object condition
  - Data object action
- For **Resource Actions**:
  - Resource condition
  - Resource action
- For **Search Actions (= Bulk Actions):**
  - For actions based on ticket result lists:
    - Bulk ticket condition
    - Bulk ticket action
  - For actions based on data object (= unit) result lists, i.e. contact or company result lists:
    - Bulk data object condition
    - Bulk data object action
  - For actions based on resource result lists:
    - Bulk resource condition
    - Bulk resource action

Script Return Values

The return value of a condition script has to be either *true* or *false*.

The return value of a manual action script has to be one of the following PostActionTypes:

- An object where the object detail page is opened (e.g. a unit page).
- A create page for an object.
- An URL.
- A return status which is also displayed as message in the Web Client.

Please see the details in the following section.

Action Types

An action can be of one of the following types:

- **Automatic**
    - Create
    - Update
    - Delete
    - Search
- **Manual**

Programming with the Action Framework

One core component of programming scripts for manual actions in the ConSol*CM Action Framework is the class **ActionScriptResultFactory**. An instance of this class is available as **actionScriptResultFactory** in each action script. The type of the returned PostActionScriptResult, i.e. the **PostActionType**, defines the step directly following the execution of a manual ⚠ action script. By working with the **PostActionType** you can implement the desired system behavior.

Open the Create Ticket Page

- return actionScriptResultFactory.getPostAction(PostActionType.CREATE_TICKET, ticket)
  Redirects to ticket create page with fields filled with ticket data.
- return actionScriptResultFactory.getPostAction(PostActionType.CREATE_TICKET, ticket, unit)
  Redirects to ticket create page with fields filled with ticket data and existing unit as main contact.

Example:

```
import com.consol.cmas.core.server.service.action.PostActionType
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")
return actionScriptResultFactory.getPostAction(PostActionType.CREATE_TICKET, ticket)
//to additionally set main contact use
//return actionScriptResultFactory.getPostAction(PostActionType.CREATE_TICKET, ticket, unit)
```

Open the Ticket Page in Display-Only Mode

- return actionScriptResultFactory.getPostAction(PostActionType.GOTO_TICKET, ticket)
  Redirects to ticket page of given ticket.

Example:

```
Example for a Unit Action script (unit implicitly available)

import com.consol.cmas.core.server.service.action.PostActionType
import com.consol.cmas.common.model.ticket.Ticket

Ticket ticket = ticketService.getByContact(unit).iterator().next()
return actionScriptResultFactory.getPostAction(PostActionType.GOTO_TICKET, ticket)
```

Open the Ticket Page in Display-Only Mode and Show ACF Before

- return actionScriptResultFactory.getPostAction(PostActionType.GOTO_TICKET, ticket,
  activityControlFormExecutionContext)
  Redirects to ticket page of given ticket and shows given activity control form obtained via
  `ActivityControlFormService.getExecutionContext(Ticket, String)`.

Example:

```
( ... )
def executionContext = activityFormDefinitionService.getExecutionContext(newtic, "defaultScope
/TaskInProgress/AcceptTask")
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);
```

> ⓘ **Information:**
>
> Please note that the implementation class for the bean *activityFormDefinitionService* is
> *ActivityControlFormService*. Please refer to the class documentation for details. Usage in scripts is
> done by the bean name, see script above.

Open the Create Unit (Contact or Company) Page

- return actionScriptResultFactory.getPostAction(PostActionType.CREATE_UNIT, unit)
  Redirects to unit create page with fields filled with unit data.

Example:

```
Example for a Unit Action script (customer implicitly available)

import com.consol.cmas.common.model.customfield.Unit
import com.consol.cmas.core.server.service.action.PostActionType

Unit contact = new Unit("customer", unit.getCustomerGroup());
contact.set("firstname", "Luke");
contact.set("name", "Skywalker");
return actionScriptResultFactory.getPostAction(PostActionType.CREATE_UNIT, contact);
```

Open a Unit (= Data Object, i.e. Contact or Company) Page in Display-Only Mode

- return actionScriptResultFactory.getPostAction(PostActionType.GOTO_UNIT, unit)
  Redirects to unit page of given unit.

Example:

```
Example for a Unit Action script (unit implicitly available)

import com.consol.cmas.common.model.customfield.Unit
import com.consol.cmas.core.server.service.action.PostActionType

Unit contact = unitService.getByCustomerGroup(unit.getCustomerGroup()).get(0)
return actionScriptResultFactory.getPostAction(PostActionType.GOTO_UNIT, contact)
```

Open a Create Resource Page

- return actionScriptResultFactory.getPostAction(PostActionType.CREATE_RESOURCE, resource)
  Redirects to resource create page with fields filled with resource data.

Example:

```
import com.consol.cmas.common.model.resource.Resource
import com.consol.cmas.common.model.resource.meta.ResourceType
import com.consol.cmas.core.server.service.action.PostActionType

ResourceType type = resourceTypeService.getByName("resource type 1")
Resource resource = new Resource(type)
resource.setFieldValue("group1", "stringField1", "value1")
resource.setFieldValue("group2", "numberField1", 1L)
return actionScriptResultFactory.getPostAction(PostActionType.CREATE_RESOURCE, resource)
```

Open a Resource Page in Display-Only Mode

- return actionScriptResultFactory.getPostAction(PostActionType.GOTO_RESOURCE, resource)
  Redirects to resource page of given resource.

Example:

```
import com.consol.cmas.common.model.resource.Resource
import com.consol.cmas.core.server.service.action.PostActionType

Resource resource = resourceService.getAll().iterator().next()
return actionScriptResultFactory.getPostAction(PostActionType.GOTO_RESOURCE, resource);
```

Open an URL

- return actionScriptResultFactory.getPostAction(PostActionType.GOTO_PAGE, "http://consol.de")
  Redirects to page of given URL.

Example:

```
import com.consol.cmas.core.server.service.action.PostActionType

return actionScriptResultFactory.getPostAction(PostActionType.GOTO_PAGE, "http://consol.de");
```

Display Result Message in Web Client: SUCCESS

- return actionScriptResultFactory.getPostAction(PostActionType.SUCCESS, localizedLabelKey)
  Remains on the same page with green success feedback panel and a localized message:
    - action.result.success=Action succeed
      Also considers actionScriptResult.withRefreshContent() to reload page data.

For details, please refer to section Return Values and Return Messages.

Display Result Message in Web Client: FAILURE

- return actionScriptResultFactory.getPostAction(PostActionType.FAILURE, localizedLabelKey)
  Remains on the same page with red failure feedback panel and a localized message:
    - action.result.failure=Action failed
      Also considers actionScriptResult.withRefreshContent() to reload page data.

For details, please refer to section Return Values and Return Messages.

Return Values and Return Messages

To indicate the return value of a ConSol*CM action for the engineer in the Web Client, you have to use one of two PostActionTypes:

- **PostActionType.SUCCESS**
  The message will be displayed in green in the Web Client.
- **PostActionType.FAILURE**
  The message will be displayed in red in the Web Client.

You have to set a return value for every action script. It is not possible to end an action script without the return value (a runtime error will be thrown in that case).

Example for a positive feedback:

```
return actionScriptResultFactory.getPostAction(PostActionType.SUCCESS, "cmweb.search.assigned").
withRefreshContent();
```


Fig. 1: ConSol*CM Web Client - SUCCESS Message after Search (Result) Action

Example for a negative feedback:

```
return actionScriptResultFactory.getPostAction(PostActionType.FAILURE, "cmweb.search.assigned").
withRefreshContent();
```


Fig. 2: ConSol*CM Web Client - FAILURE Message after Search (Result) Action

The text which is displayed (e.g. *Action succeeded* or *Action failed*) can be retrieved using one of two possible methods:

- Use a CM standard value which is defined in standard property files, see section CM Standard Values for Return Messages.
- Use a label which has been defined specifically for your CM system in the Labels Administration, see section Use System-Specific Labels.

CM Standard Values for Return Messages

```
action.result.success=Action succeeded
```

```
action.result.failure=Action failed
```

Use System-Specific Labels

You can define your own labels for the display of messages in the Web Client. See section Labels for a detailed explanation.

38.3 Admin Tool Templates

- Introduction to Templates in the Admin Tool
- The Admin Tool Template Editor
- Working with Admin Tool Templates
    - System Templates
    - Templates for Definition of Customer Format in the Web Client
    - Ticket Assignment Templates
    - Password Reset Templates
        - Password Reset Template for Engineers in the Web Client
        - Password Reset Template for Customers Using CM.Track
    - Custom Defined Templates

38.3.1 Introduction to Templates in the Admin Tool

In ConSol*CM, several types of templates are used:

- **E-mail templates** are stored in one of the following places:
    - the *Template Manager*, see section The ConSol*CM Template Manager
    - the *Templates* section of the Admin Tool. Details on that topic will be explained in the present chapter.
- **Document templates** are stored in
    - the Doc 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 Templates for Resource Data
- **General templates** are stored in
    - the *Scripts & Templates Administration* of the Admin Tool. They will be explained in the present chapter.

In this chapter, the general templates in the *Scripts & Templates Administration* of the Admin Tool will be explained.

Admin Tool templates are written according to the *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.

38.3.2 The Admin Tool Template Editor

To work with templates, open the item *Scripts and Templates* in the navigation group *System* and switch to the *Templates* 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 sort the templates in the templates list. They do not have a technical implication.

To open a template in the editor panel, mark it in the list and open it by clicking on the edit button [icon]. 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. Each template always has to be defined for the default language.
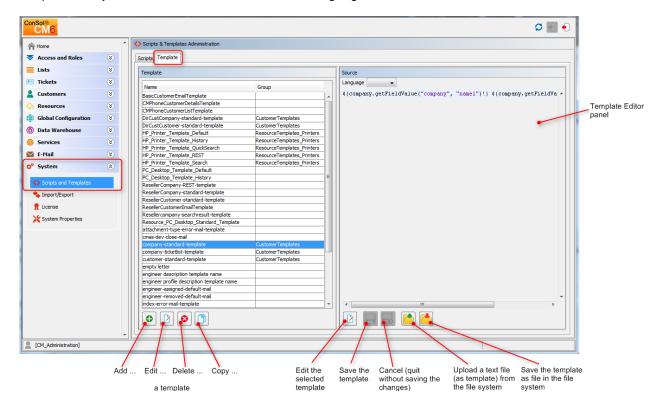
Fig. 1: ConSol*CM Admin Tool - Template Editor

38.3.3 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 Data Object Group Fields, Resource Fields, and Custom Fields are referenced by group name and field name, e.g. the company name within the Data Object Group *ResellerCompany* will be referenced as shown in the following example.

```
${ResellerCompany.getFieldValue("ResellerCompanyData","company_name")!}
```

For a detailed explanation of the work with Custom Fields, please see section Custom Field Administration. Data Object Group 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.

> ⚠ **Attention:**
>
> 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 to an administrator. Please see the following list for an overview of the system templates:

- **attachment-type-error-mail-template**
  An e-mail with this template is sent to the e-mail administrator (e-mail address given in system property *mail.process.error*) when the attachment type of an incoming or outgoing e-mail is not supported and thus the e-mail cannot be processed.
- **cmas-dev-close-mail**
  Not used. Removed from the standard installation in CM version 6.10.1.
- **engineer description template name**
  Template used to render the engineer label, e.g. ticket owner.
- **engineer profile description template name**
  Template used to render the label on a header of the page, next to logout button.
- **index-error-mail-template**
  Not used. Removed from the standard installation in CM version 6.10.1.
- **password-reset-template**
  Template for the body of the e-mail which is sent when a user requests a password reset (on login page).
- **representation_info_email_html**
  All e-mails 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 e-mail.
- **representation_info_email_plain_text**
  Same as above, as plain text.

Templates for Definition of Customer Format in the Web Client

The appearance of **customer data** (e.g. name, phone number, and room number or name and forename only) in different sections of the Web Client can be formatted using templates. The definition has to be made

for each data 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 a data 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 CM system). For a detailed explanation, please refer to section Templates for Resource Data.

In the following example, the customer data within the *ResellerCustomer* data object should be represented in the standard template with first name and name.

Fig. 2: ConSol*CM Admin Tool - Example of a Customer Format Definition Template

```
Example of a Customer Format Definition Template

<#if ResellerCustomer.getFieldValue("ResellerCustomerData","customer_name")?has_content &&
ResellerCustomer.getFieldValue("ResellerCustomerData","forename")?
has_content>${ResellerCustomer.getFieldValue("ResellerCustomerData","customer_name")!},${Reselle
rCustomer.getFieldValue("ResellerCustomerData","forename")!}<#else> ${ResellerCustomer.
getFieldValue("ResellerCustomerData","customer_name")!}</#if>
```


Fig. 3: 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 two use cases *assign* and *remove*.

- The *assign* template (*Assign*) is used as text template for an automatic e-mail, 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 e-mail, which is sent by the system to the (old) engineer when a ticket has been removed 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 to use a name which describes the use of the template, as shown in the following example.

Fig. 4: ConSol*CM Admin Tool - Example of an E-Mail Template for Ticket Assignment to an Engineer

Text of the example template *engineer-assigned-default-mail*:

```
Subject: Ticket #${ticket.name} assigned to you
<#setting number_format="0.######"/>
The ticket #${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}
```

Fig. 5: ConSol*CM Admin Tool - Configuration of a Ticket Assignment Template for a Queue

Password Reset Templates

When a user has forgotten his password, he can request a new password using CM standard functionalities. 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 CM administrator has to configure CM in a way that an e-mail can be sent to the user who has to set a new password. This e-mail is based on a template which is stored in the Admin Tool, in the *Templates* section.

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 e-mail with a link to an URL where the engineer can set the new password is sent to the engineer.

> ⚠️ **Attention:**
>
> Please note that this can only work if a valid e-mail account is available for this engineer and if the respective value has been entered as e-mail address for the engineer in the engineer data!

The e-mail which is sent to the engineer is based on the template *password-reset-template.* This template name is mandatory - otherwise the password reset will not 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")}.
```
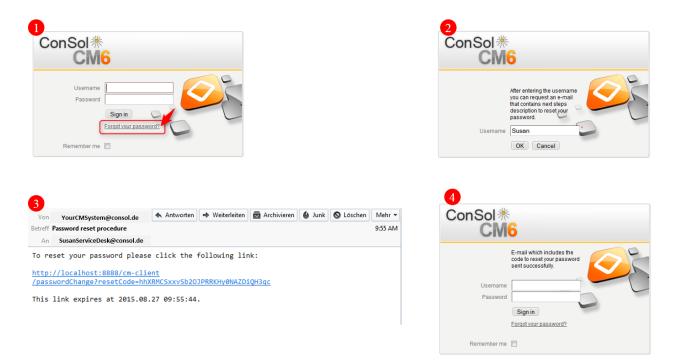

Fig. 6: ConSol*CM Web Client - Password Reset by an Engineer

> ℹ️ **Information:**
>
> 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 for Engineers in the Web Client for an explanation of all possible authentication modes.

Password Reset Template for Customers Using CM.Track

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 e-mail with a link to an URL where the customer can set the new password is sent to the customer.

> ⚠ **Attention:**
>
> Please note that this can only work if a valid e-mail account is available for this customer and if the respective value has been entered as e-mail address for the customer in the respective Data Object Group Field.

The e-mail which is sent to the customer is based on the template *track-password-reset-template.* This template name is mandatory - otherwise the password reset will not 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://cm6doku-cm1.int.consol.de:8280/cm-track/#track=set_new_password/resetCode-${resetCode}
(Reset Code: ${resetCode})
Valid 24 hours only, please visit before expiry!
```

As FROM-address for the e-mail to the customer, the administrator e-mail address is used.

Fig. 7: CM.Track - Password Reset by a Customer

ⓘ   Please note that 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 CM.Track:
      Authentication Modes for the Portal 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 you use it in automatic e-mails which are sent by a workflow
activity, you can always use the workflow API *renderTemplate()* method to reference a template. However,
most e-mail templates should be managed using the Template Manager (see section The ConSol*CM
Template Manager). There are only very few use cases which might require that e-mail templates or parts of
e-mail templates have to be stored in the *Scripts & Templates Administration* of the Admin Tool.

39 Deployment (Import/Export)

- Introduction
- Scenarios
- Deployment (Import/Export) Using the Admin Tool
    - Export
    - Import
    - Workflow Deployment (for Deployment Error Recovery Only)

39.1 Introduction

ConSol*CM offers the possibility to export the system configuration with or without runtime data into a file and to import this file into another 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 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 functionalities in both systems. In case a transfer (export/import) is performed later on, there will be duplicate objects! Please read the detailed explanations below.

39.2 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 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 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-*
- all page customizations

A scenario **can** contain, depending on the selection of the administrator (see figure below):

- runtime data
- configuration data

A scenario will **never** contain:

- general (not customer-specific) system properties (e.g. mail server, LDAP directory etc.)

39.3 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 CM installations. A typical example is transfer of the configuration from a test system to a production system.

Fig. 1: ConSol*CM Admin Tool - Import/Export

> ⊖ **Warning:**
>
> The import of external data can modify 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.

39.3.1 Export

- **Export-Archive:**
  Enter path and name of the file you want to create. Alternatively, you can click on 📁 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):

Fig. 2: ConSol*CM Admin Tool - 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 and customer data is exported completely **and** the complete configuration is exported. When you select the check box *All*, all other check boxes are selected automatically.
    - **All without tickets**
      The complete installation beside the tickets is exported, i.e. the customer data **and** the complete configuration. When you select the check box *All without tickets*, all other check boxes except for *All* are selected automatically.
    - **Only customer data**
      Only customer data (i.e. the customer data model and the actual customer data) is exported. Nothing else. (The check box *Customer model* is checked automatically.)
- **Configuration data**
  This refers only to the configuration in the Admin Tool, no runtime data is exported.

- **All**

    The complete configuration is exported. When you select the check box *All*, all other check boxes under *Configuration data* are selected automatically.

- **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).

- **Templates**

    Only the templates that are stored in the Template Manager (see section The ConSol*CM Template Manager for details) are exported.

- **Word templates**

    Only the MS Word templates or OpenOffice templates are exported, this is only relevant when CM.Doc is in operation (see section CM.Doc for details).

- **Customer model**

    Only the data object group fields that are used to define the customer model are exported. No runtime customer data is included.

- **Queue related and other data**

    Only queue configuration and general configuration settings are exported (workflows, queues, custom fields, enum values, MLAs, roles, views, properties, ...), in short: everything which is not included above.

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.

39.3.2 Import

The *general principle* of ConSol*CM scenario import is:

- If the check box *Delete existing data* has **not** been **selected**, the scenarios are **merged**, based on the following principles:
    - 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** the one of the original scenario.
      Example: For the field *priority*, there is the annotation *position = 0;2* in the imported scenario. The original scenario contains the value *position = 2;2* for the field *priority*, i.e. 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: In the imported scenario, there is the annotation *visibility = none* for the field *PersonID*. In the original scenario, the field *PersonID* is present, but does not have the annotation, i.e. 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/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 does no longer contain the annotation *visibility = none*, but the original scenario does contain the annotation, it will remain, i.e. in the resulting scenario the field *PersonID* is still invisible.
    - **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 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.
- If the check box *Delete existing data* has been **checked**, the entire system is deleted, i.e. **all** existing data are **deleted**. All data means:
    - Configuration data
    - Runtime data
  That means when *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.

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 on 📁 to open the file browser.
- **Mode:**
  Here you can choose what the import shall 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 when an error appeared. The problem can be probably handled afterwards in a short time. A new import might take longer to perform.
    Example: A referenced object is not found, e.g. 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 doing an error analysis.

Click on *Import* afterwards to start the data import.

39.3.3 Workflow Deployment (for Deployment Error Recovery Only)

Usually, all operations concerning workflow design and deployment are performed using the Process Designer. However, in case an error has occurred 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 it has to find the last consistent position in the process.
- **Restart process**
  The ticket goes back to the *START* node of the process/workflow.

Please read also the detailed explanation of the workflow deployment process in the *ConSol*CM Process Designer Manual*.

40 License Management

- General Information about Licenses in ConSol*CM
- Managing the ConSol*CM License Using the Admin Tool

40.1 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 Web Client, REST section. Ten licenses have been purchased.

```
[CONCURRENT_USERS]
contractParty = Demo-Licence ConSol
products = WEB_CLIENT,REST
version = 6.9
expirationDate = 31.12.2015
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*, Appendix C (System Properties)).

40.2 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 software contracts with ConSol*.

Please ask your consultant for details. The license is a plain text file. The license can be modified during CM operation, no system downtime is required.

> ⚠️ **Attention:**
>
> There is no *Back* button to undo changes with one click when you enter or delete text in the *Licence* field. If you accidentally changed 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.
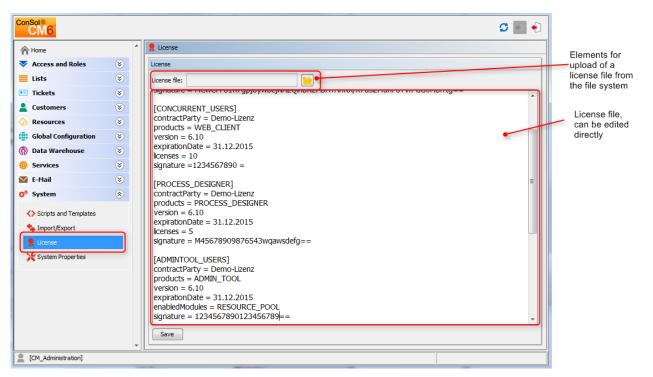
Fig. 1: ConSol*CM Admin Tool - System: License

Choose one of the two ways to import your ConSol*CM license file:

- Insert the entire text of the license file by copy and paste. In case an old license is present, just replace the entire text. Click on *Save*.
- Load the license using the file browser next to the field *Licence file*. Click on *Save*.

You should receive a message that the license has been imported into the system successfully. It is in operation at once, without further action.

41 System Properties

- Introduction
- System Property Overview
- Setting System Properties
- Programming with System Properties

41.1 Introduction

On the tab *System Properties* in the navigation group *System*, you can add, modify or delete system settings, so called *properties*, for the CM application. Properties are used to store, for example, values for the number of seconds for session timeout, for admin e-mail addresses or for the configuration of the search page size.

> **DO NOT WORK ON THIS TAB UNLESS YOU KNOW EXACTLY WHAT YOU ARE DOING !!!**
>
> On this page, you have access to basic system setting called *properties*.
>
> Do not modify, edit or delete any values on this page unless you know exactly what the impact will be ⚠

41.2 System Property Overview

A system property has the following parameters:

- **Module**
  (mandatory) This indicates in which CM module the property will be used.
- **Property** (name)
  (mandatory) The name of the property. By this name the property is referenced throughout the system.
- **Type**
  (mandatory) Data type of the 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.
  - **E-mail**
    A field which contains an e-mail address, i.e. it has to be formatted according to the standard e-mail format (<address>@<domain>)
  - **Boolean**
    A Boolean (true/false) field
  - **Integer**
    A whole number field (no points). Used, for example, for time intervals or number of restarts.
- **Value field**
  The value of the property. Muss be set according to the given data type.
- **Description**
  (optional) A text description for the property.
  ⚠ In case you add company-specific properties which contain integers for time intervals, we strongly recommend to use the description to explain the unit of measurement since it makes a big difference if an escalation goes off within 5 minutes, 5 hours or five days!
- **Restart required**
  Boolean field (checkbox) to indicate if 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 property can be NULL.
    - The property can be deleted, e.g. using the Admin Tool.
  - optional = *false*:
    - The value of the property cannot be NULL.
    - The property cannot be deleted, i.e. there must be a value in the value field.

Fig. 1: ConSol*CM Admin Tool - System Properties

The system properties are managed in a table where three of the parameters of a property are displayed as columns:

- **Module**
  Module name
- **Property**
  The name of he property.
- **Value**
  The value of he property.

You can sort the table according to a column by clicking on the column header. Another click reverses the order.

41.3 Setting System Properties

The values for properties are set in different ways. Properties can be ...

- filled automatically with default values by the CM system, e.g. *cmweb-server-adapter, ticketListRefreshIntervalInSeconds* is set to 180 seconds. You can modify the values when necessary.
- prepared by the system, i.e. the properties are in the list but they are not filled with the required values. For example, LDAP-related properties will only be filled (manually ba 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 *E-Mail* configuration, the respective values will be entered into the property *cmas-esb-mail,mail.incoming.uri* (for an Mule/ESB system). ⚠ In those cases, you should always use the graphical configuration and not edit the 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 property *cmas-app-admin-tool, start.groovy.task. enabled* and setting its value to *true*.
- added to a system manually as customer-specific properties. If the module name starts with *custom*-, those properties are exported in a scenario (see section Deployment (Import/Export)). For example, you can define and add your own properties to manage escalation times. In this way, you can store the values here in the System Properties Management and an administrator can change the values without having any programming knowledge. In the script code where the properties are referenced, only the property name is used. This is explained in the following section.

41.4 Programming with System Properties

To use a property in a Groovy script, i.e. to retrieve the 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.

Fig. 2: ConSol*CM Admin Tool - Customer-Specific System Property

42 ConSol CM Administrator Manual 6.10.2 - Working with Text Templates

42.1 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 e-mails or ticket-comments where text, headers, and footers can be specified. Another example are documents which have to be edited using MS Word or OpenOffice.

In both cases, the templates do not offer only texts 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 *Template Manager* for editing and managing e-mail and ticket-comment templates (see section The ConSol*CM Template Manager)

and

- *CM.Doc* with the *Doc Template Manager* for editing and managing MS Word templates (see section CM.Doc).

42.2 The ConSol*CM Template Manager

42.2.1 Introduction to the Work with E-Mail and Ticket Text Templates

Using the *Template Manager*, two types of templates can be defined:

- **E-mail templates** for e-mails written from the ConSol*CM system
    - manual e-mails (written by an engineer using the Ticket E-Mail Editor)
    - automatic e-mails 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

E-Mail Templates

Why E-Mail Templates?

When a system works with e-mails, several criteria have to be considered. If all those requirements are met, e-mail templates are a very helpful tool in every-day working life.

- The e-mails have to have a strictly defined layout, usually according to a 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 software company should be required.

ConSol*CM provides the function set to take all those criteria into consideration.

E-Mails in ConSol*CM

E-mails are used for core functionalities in ConSol*CM. Those functionalities have been described in detail in section E-Mail, so here, only a short review is given.

ConSol*CM can receive and send e-mails. Sending e-mails can serve various purposes:

- **An engineer writes an e-mail directly from the ticket, using the Ticket E-Mail Editor.**
  This can be an e-mail to the customer, to a co-worker, or to any other person with a valid e-mail 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 e-mail 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 e-mail templates are not static, but dynamic.
  E-mails which are sent manually either do not use a template or are based on a template from the *Template Manager*. Templates from the *Script and Template Administration* of the Admin Tool are not available here.
- **The system sends an e-mail automatically.**
  This can be an internal e-mail like a reminder for an engineer when the ticket has entered the escalation status or an internal e-mail to a supervisor when a ticket needs approval to be continued. Or it can be an external e-mail to the customer like a confirmation of receipt or a notice that a ticket has been solved. The e-mail is generated automatically based on the respective e-mail template.This can be an e-mail template from the *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 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 e-mail templates. Only the *Used within* parameter is set in a different way when the template is created using the Template Manager.

> ⚠️ **Attention:**
>
> Technically, there is no difference between e-mail and ticket text templates! So for each template you as a template manager working with the Template Manager can decide if the template should be used as ticket text template, as e-mail template, or both.

E-Mail and Ticket Text Templates in ConSol*CM

E-Mail and Ticket Text Template Components

In e-mail and ticket text templates in ConSol*CM you can use free text and all data that is available for a customer, an engineer, and/or the ticket. In section The Library of Markers all available components will be explained.
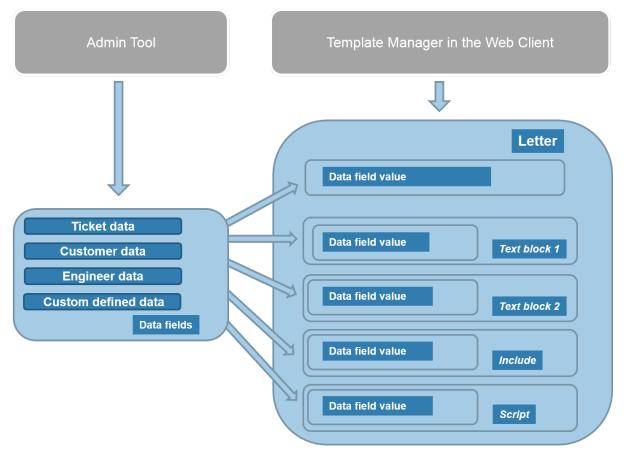
Fig. 1: ConSol*CM - Availabe Components / Data for E-Mail Templates

Please refer to the *ConSol*CM User Manual* section *Creating a New Ticket*, *Editing a Ticket*, and *Creating and Sending an E-Mail* for a detailed description how to use the ticket editing functionalities and the Ticket E-Mail Editor.

Storage and Management of E-Mail and Ticket Text Templates

E-Mail Templates

E-Mail templates are stored and managed at two different locations in ConSol*CM:

1. In the *Template Manager*
2. In the *Script and Template Administration* of the Admin Tool (this will not be treated 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 *Template Manager* (here, ticket text templates are managed)
2. In *CM.Doc* (here, MS-Word documents can be stored for use with CM.Doc, see section CM.Doc)

42.2.2 Introduction to the Template Manager

The ConSol*CM Template Manager is a Web Client-based tool for the creation and management of e-mail and ticket text templates. See section Work with the Template Manager.

Group	Template	Language	Type	None	None	Usage	Used within workflow	Used within email	Used within ticket create	Used within ticket edit
Rückfrage	Catalogue 1 0.0.1	en	Text Block			0	no	yes	yes	yes
Rückfrage	Catalogue2 0.0.1	en	Text Block			0	no	yes	yes	yes
Rückfrage	Katalog PC 0.0.1	de	Text Block			0	no	yes	yes	yes
Vertrag	New offer 0.0.1	de	Text Block			0	no	yes	yes	yes
Werbung	New products 0.0.1	en	Text Block			0	no	yes	yes	yes
general	SalutationScript 0.0.1	en	Script			0	yes	yes	yes	yes
allgemein	Signatur Company 0.0.1	de	Include			0	no	yes	yes	yes
allgemein	Signatur standard (E-mail standard) 0.0.1	de	Letter				no	yes	no	no

Fig. 2: ConSol*CM Web Client - Template Manager

Every user who has been assigned a role with the permission *Write template* can access the item *Manage templates* in the main menu (which opens the Template Manager).

Fig. 3: ConSol*CM Admin Tool - Permissions for Role TemplateManager

> ✅ **Consulting Best Practice:**
>
> We recommend to create a role (*e.g. TemplateManager*) that has only the permission *Write template*, no queue permissions or other permissions are granted. Every user who should have access to the Template Manager can be given this role. That way, there is no merge between regular user permissions and Template Manager permissions and you can grant and retrieve the Template Manager permission in a very flexible way.

When the permission has been granted, the user has access to the main menu item *Manage templates*.

Fig. 4: ConSol*CM Web Client - Main Menu with Template Manager Access

42.2.3 Working with the Template Manager

Basic Template Manager GUI: The Template Library

When you open the Template Manager, the *Template library* is displayed:

Fig. 5: ConSol*CM Web Client - Template Manager

A list of all existing templates is shown.

Filter

You can filter the displayed list entries by using the filters in the upper part of the page:

- **Active**

  Only active templates are displayed (i.e. deactivated templates will not be displayed).
- **For queue**

  Only templates which have been assigned to the selected queue will be displayed. Only one queue can be selected. The display of templates which have not been assigned to a certain queue will not be affected here. They will be displayed.
- **Used within**

  Only the templates of the selected type will be displayed, i.e. e-mail, workflow, ticket texts (creation or editing). Only one type can be selected.

Context

This provides another kind of filter. Here you can select ticket data. For each selected criterion, one of the columns named *none* in the list will be named according to the selected parameter. For the templates where the selected Custom Fields are used, this will be indicated in the list.

List

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**

  The group of a template does not have any technical or functional implications, it is only used to order the list in a certain way, i.e. to group templates with a common context.
- **Template**

  The template name. This is also used in workflows to indicate the required template and it is displayed in the Ticket E-Mail 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 E-Mail Editor or Ticket Comment Editor and can be used as workflow e-mail 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 every other template. 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 at exactly 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 checked or unchecked during the writing of e-mails, 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 purpose of the e-mail, the engineer uses either one.

- **Script**
  This template type is only available for administrators (i.e. a user who logs into 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*.

- **Context (here *None*)**
  Can be used to define dependencies or conditions (e.g. field values). Only when the condition is met the template is offered in the Web Client GUI, e.g. the template is only offered for tickets with high priority.

- **Usage**
  Indicates how often the template is used.

- **Used within workflow**
  Boolean. A template can be marked as *workflow template*. Then it is not available in the Ticket E-Mail Editor or Ticket Comment Editor but can only be used by the workflow for automatic e-mails.

- **Used within email**
  Boolean. All templates which have been marked as *Available in Email* will be marked *yes*.

- **Used within ticket create**
  Boolean. All templates which have been marked as *Available in Ticket create* will be marked *yes*.

- **Used within ticket edit**
  Boolean. All templates which have been marked as *Available in Ticket edit* will be marked *yes*.

For every template you can select an operation by using the context menu:

Fig. 6: ConSol*CM Web Client - Template Manager: Context Menu of a Template

- **Edit**

    Edit the template. The same functionalities as described for creating a new template are 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 (*letter*s).
- **Clone**

    Create a copy of the template. A new name is required in this case.
- **Use as e-mail standard**

    (or **Unset standard** for the current standard template)

    Only one template can be marked as standard e-mail template. This will be automatically inserted into any e-mail that is opened in the Ticket E-Mail Editor. It can then be removed by the engineer or used in the e-mail. 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 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.

ⓘ **Information:**

   A standard template must not contain text blocks or variables.

Create a New Template

Here, an example for an e-mail 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 *New* link in the 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.

Fig. 7: ConSol*CM Web Client - Create a New Template

- **Title**
  The name of the template.
- **Group**
  The group (see previous section). You can either use an existing group or create a new one.
- **Release**
  If you want to set up a versioning system for the e-mail templates, you can set the release, i.e.
  version, here.
- **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, this can be configured using the Admin
  Tool and in the Process Designer. To make sure the e-mails 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 E-Mail Editor or in the Ticket Comment Editor).

  - **Email**

    Select if the template should be available in e-mails.

  - **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.

- **Body**

  Here you define the content of the template/letter. You can combine any free text and components of the *Library of Markers* (below the body, see section The Library of Markers for details). Write the text and select the desired element from the library by clicking on it and by pressing *Insert*.

- **Binding**

  Here you can define one or more

  - queue(s)
  - context(s)

  where the template should be available, see section Binding Templates to Queues or to Specific Parameters for details.

In the Web Client, i.e. in the Ticket E-Mail Editor, the template *receipt_notice_ServiceDesk* would have the following layout:

Fig. 8: ConSol*CM Web Client - E-Mail Template in Ticket E-Mail Editor

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 forename* or *engineer lastname*.
- **Ticket Custom Fields and/or Data Object Group Fields**
  That have been designed specifically for the system like e.g. *customer service number*.
- **Components of the 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 system. The names displayed in the Library of Markers are the localized names of the Custom Fields resp. of the Data Object Group Fields.

If no localization is provided, the (technical) field name is displayed. If you would like to read the information about Custom Fields, please refer to chapter Custom Field Administration. For Data Object Group Fields (i. e. customer data), see section Setting Up the Customer Data Model.

Field Group or Main Component	Custom Field Resp. Data Object Group 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
Custom Fields for queue	All Custom Fields of Custom Field groups that have been assigned to the queue	
Ticket	ID	The internal ticket ID, not displayed in the Web Client GUI
	Name	The ticket name, the *ID* in the Web Client GUI
	Subject	
	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

Field Group or Main Component	Custom Field Resp. Data Object Group Field (Example)	Explanation
Engineer	Login	The login of the engineer who is currently logged into the system
	Firstname	First name, last name, e-mail 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>	

⚠ **Attention:**

In case 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 Data Object Group Fields differ between customer data models, it might be necessary to define a similar template for several customer groups.

By clicking on *Add parameter* you can define a field where the engineer has to fill in data at run-time. When you have defined the field, it will be available under *Additional parameters*.

Fig. 9: ConSol*CM Web Client - Template Manager: Add Parameter in Library of Markers

Fig. 10: ConSol*CM Web Client - Template Manager: Library of Markers after Adding Parameter

When the engineer opens the Ticket E-Mail Editor in the ticket and enters data in the field (here *CallBackDate*), the (new) data is automatically written into the field in the template.

Fig. 11: ConSol*CM Web Client - Ticket E-Mail Editor: Enter Data for New Parameter

Create a New Include or Workflow Include

An *include* is a template that cannot be selected by the engineer directly (in the Ticket E-Mail or Comment Editor) but a component which is integrated in other e-mail or ticket text templates, mostly 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 *include* and integrate it into the standard company signature which is a *letter*.

Fig. 12: ConSol*CM Web Client - Template Manager: Signature Include

Fig. 13: ConSol*CM Web Client - Template Manager: Standard Signature Letter

If you follow the example, two purposes can be met:

- The *Standard_Signature* can be defined as e-mail standard. In this way it will automatically be displayed for every new e-mail. 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 E-Mail or Comment Editor) but a component which is integrated in other e-mail 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 some initial analysis questions to the customer.

First, the *text blocks* are created:

Fig. 14: ConSol*CM Web Client - Template Manager: Create First Text Block

Fig. 15: ConSol*CM Web Client - Template Manager: Create Second Text Block

Fig. 16: ConSol*CM Web Client - Template Manager: Create Third Text Block

Then the *letter* is created where the *text blocks* should be used:

Fig. 17: ConSol*CM Web Client - 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:

Fig. 18: ConSol*CM Web Client - Ticket E-Mail Editor: All Text Blocks Selected
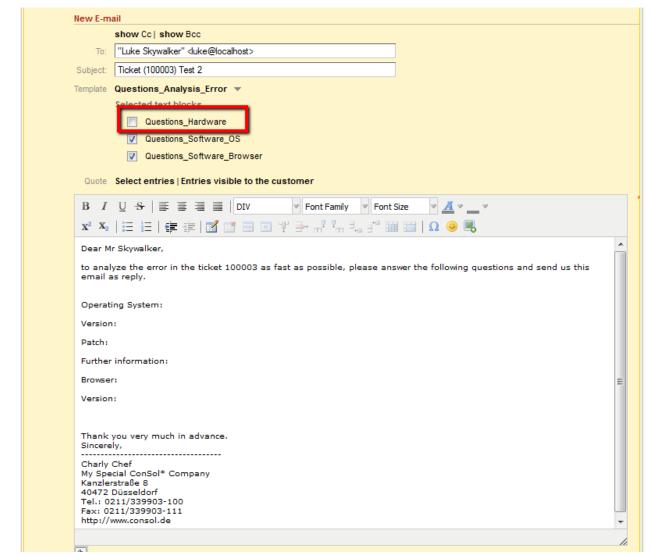
Fig. 19: ConSol*CM Web Client - Ticket E-Mail Editor: One Text Block Deactivated

Create and Use a Script

Sometimes a system has to have a certain *intelligence* regarding the words and phrases used in e-mail 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 Template Manager.

This can only be realized by a ConSol*CM administrator. When you log into 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:

Fig. 20: ConSol*CM Web Client - Template Manager: Example Script for Salutation

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 Custom Fields and Data Object Group Fields managed in the Admin Tool. Please
  refer to sections Custom Field Administration and Setting Up the Customer Data Model for details.

The *script* is then integrated into a *letter* template:

Fig. 21: ConSol*CM Web Client - Template Manager: Insert Script into Letter

In the Web Client, the e-mails are formatted as requested.

Example 1 (for **Mrs**):

Fig. 22: ConSol*CM Web Client - E-Mails Formatted by Script (Customer Data, Example 1)

Fig. 23: ConSol*CM Web Client - E-Mails Formatted by Script (E-Mail, Example 2)

Example 2 (for **Mr**):

Fig. 24: ConSol*CM Web Client - E-Mails Formatted by Script (Customer Data, Example 1)

Fig. 25: ConSol*CM Web Client - E-Mails Formatted by Script (E-Mail, Example 2)

Binding Templates to Queues or to Specific Parameters

The section *Binding* is the last section of the *New Template* page of the 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 when the priority *high* has been set for the ticket)

You can select queues and/or parameters by selecting a *context*, see the following pictures for examples. Use the "+" button to add more binding parameters or the "-" button to remove existing parameters.

The template should only be displayed in the *Helpdesk 1st Level* queue:

Fig. 26: ConSol*CM Web Client - Show Template for Specific Queue

The template should only be displayed in the *Helpdesk 1st Level* queue and only if the priority is *high*:

Fig. 27: ConSol*CM Web Client - Display Template for Specific Queue and Priority

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 binding:

- **Hard binding**
  The check box *soft* is **not** checked:
  The template is only displayed (offered in the Ticket E-Mail 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 check box *soft* is checked:
  As default setting, the template is only displayed (offered in the Ticket E-Mail 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*. Then all softly bound templates are displayed as well.

42.2.4 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 Template Manager have to be checked and modified manually.

All parameters which refer to tickets and queues do not have to be changed. However, due to the new customer model, *FlexCDM*, the syntax for references to Data Object Group Fields has to be adapted.

42.2.5 Page Customization for E-Mail Template Functionalities

Please refer to section Page Customization to learn some details about how to adapt e-mail template parameters.

42.3 CM.Doc

42.3.1 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 MS 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 and belongs to a company), additional customers are NOT relevant in CM.Doc!)
- engineer data (name, phone number, e-mail address of the engineer who works on the case)

When CM.Doc is active in ConSol*CM, the engineer can select the required MS Word template (or OpenOffice template) in the ticket. The document is opened in MS 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 (MS Word or OpenOffice) on their PCs.

With special adaptations implemented by the ConSol*CM consulting team, the 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.

42.3.2 Requirements for Using CM.Doc

On the 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*.
- MS Word or OpenOffice has to be installed. For the supported MS Word and OpenOffice versions, please refer to the current *System Requirements*.

42.3.3 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.

42.3.4 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*.

Fig. 1: ConSol*CM Admin Tool - Configuration of System Property for CM.Doc

42.3.5 Creating an Engineer Role with Permissions for the Doc Template Manager

Only an engineer who has the permission *Write template* (see the following figure) can start the *Doc 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.

Fig. 2: ConSol*CM Admin Tool - Necessary Template Permissions

> ✅ **Consulting Best Practice:**
>
> We recommend to create a role (*e.g. TemplateManager*) that has only the permission *Write template*, no queue permissions or other permissions are granted. Every user who should have access to the Doc Template Manager can be given this role. In this way, there is no merge between regular user permissions and Word /Office template permissions and you can grant and retrieve the Word /Office template permission in a very flexible way.
>
> However, the permission *Write template* also grants access to the Template Manager (for text templates, see section The ConSol CM Template Manager).

42.3.6 Creating MS Word Templates and Making Them Available

Creating MS Word Templates

As a third step, you have to create the MS Word templates. This is done using MS Word. Please create *.doc* or *.docx* files as templates, **not** *.dot* files!

The OpenOffice-specific configuration is explained in section Working with OpenOffice. However, most of the configuration steps are identical or very similar for MS Word and openOffice templates.

Creating a New MS Word Template, Step 1: 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 MS Word template, i.e. you create a CM.Doc template from your regular MS 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 *Manage Word templates* in the main menu to open the *Doc Template Manager*.

Fig. 3: ConSol*CM Web Client - Doc Template Manager

The Doc 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 Doc Template Manager.
- **Language**
  Select the required language. The languages that have been activated in the Admin Tool are offered.
- **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 offered, please see section Selection of the Data Object and Its Consequences for a detailed explanation about which data object you have to select.
- **Word 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 MS Word template, you have to select a data object, i.e. a customer object or a company object from your customer data model. If you are not familiar with the ConSol*CM customer data model concept, please read section The CM Customer Data Model - FlexCDM fist.

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, some 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 Word 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 MS Word template will be offered only in tickets which have main customers from a customer group with this customer data model. This means, if you want to use the same MS Word template text for two customer groups with two different customer data models, you have to create two ConSol*CM MS Word templates, one for each customer data model. Of course, you can use the same MS 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 MS 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 on company level, if for this customer data model the parameter *Company as customer* has been set.

Fig. 4: Selection of Data Object in Doc Template Manager

> ⓘ Please be aware that when you leave the *Data object* field empty, the MS Word template will
> theoretically be available for all customer groups, but there might be runtime errors in case 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 to
> always select a data object to keep the system in order.

How to Control in Which Tickets the MS Word Template Will Be Offered

If the MS Word template is offered in a certain ticket (see section Using MS Word Templates from within the
Web Client below) depends ob two parameters:

- the queue - the template is only offered for ticket in the selected queue(s)
- the customer data model - the template is only offered in tickets which have as main customer a
  customer object which is based on the selected data object

Example (we anticipate the use of MS Word templates in the Web Client, for details see section Using MS
Word Templates from within the Web Client):

- First, the MS Word template is defined for the queue *Service Desk* 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 *Service Desk* and which
  has as main customer an object of the *Reseller Customer* data object type (second figure).
- In the Web Client, the template cannot be used in a ticket which is in the queue *Service Desk* and
  which has as main customer of a data object which is NOT of the *Reseller Customer* data object type
  (third figure).

- In the Web Client, the template cannot be used in all tickets which are NOT in the queue Service Desk (not shown).

Fig. 5: ConSol*CM Web Client - Definition of an MS Word Template for One Queue and for a Certain Data Object

Fig. 6: ConSol*CM Web Client - MS Word Template Offered Because of Queue and Customer Group

Fig. 7: ConSol*CM Web Client - MS Word Template NOT Offered Because of Queue and Customer Group

Creating a New MS Word Template, Step 2: Creating and Editing the Required MS Word Document

When you have filled in all fields and clicked on *Ok*, the new template appears in the *Template library*, *List of templates*. You can perform the following step directly after entering the new template data or you mark the template name (in the *Word template* column) you would like to edit in the list.

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 *Word 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.

Fig. 8: ConSol*CM Web Client - Selection of a Ticket to See All Required Data Fields

All available *MailMerge* fields are displayed:

Fig. 9: 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 Merge Field later on)
- **Group**

  This is set for customer data and for ticket data. For customer data, the data object group is shown (here, only the data object groups from the selected data object are offered), for ticket data, the Custom Field Group is shown.
- **Field**

  This is set for customer data and for ticket data. For customer data, the data object group field name is shown (here, only the data object group fields from the selected data object are offered), for ticket data, the Custom Field name is shown.
- **Value**

  The value of this field in the selected ticket. You do not have to use this in the Word template.

Open the MS Word template from the list in the *Template Library* or click on *Word template* , *Browse ...* to open/download the Word document. 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 Merge Field you need into the respective field (*Field properties*, *Field name*) in MS Word:
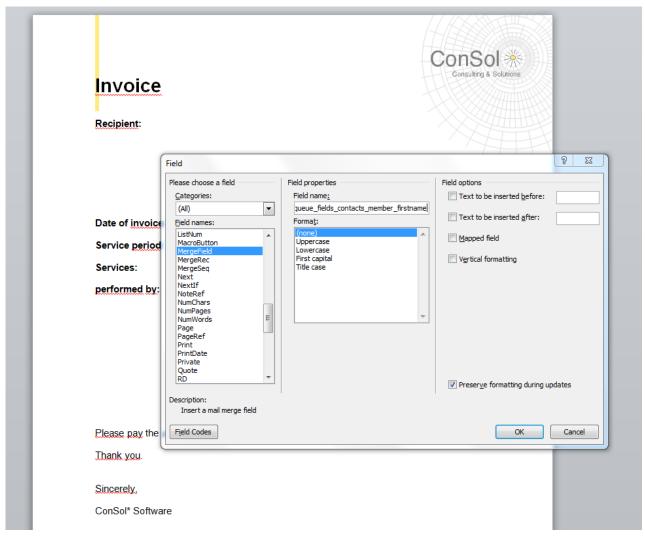
Fig. 10: MS Word - Insert MergeField into Document

Do this with all fields you would like to have pre-filled when the MS 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 (Data Object Group 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 data object group name as Group)
- Contact fields (Data Object Group 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 data object group name as Group)
- Ticket data fields (Custom Fields from the ticket, the respective keys start with *ticket_* and have a Custom 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)
- Engineer data (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.)

Fig. 11: MS Word Example Template

Save the Word document in the local file system and then upload it using the field *Update Word template*. This will store the template in the Doc Template Manager.

Details about ConSol*CM Keys for the Mail Merge 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 data object group 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]_[data object group]_[data object group field], e.g.,
   ticket_queue_fields_contacts_member_companyReferenceField_ResellerCompany_ResellerCompany
   _company_name

42.3.7 Using MS Word Templates from within the Web Client

Creating a New MS Word Attachment Using a Word Template

When MS Word templates are available for a queue, an engineer can use them by clicking on *Attachments* in the *History Section* of a ticket and by selecting the requested template. MS 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.

Fig. 12: ConSol*CM Web Client - MS Word Templates Available as Attachments

Fig. 13: MS Word Example Document

The engineer can then edit the document if requested and save it. This will attach the new version of the document automatically to the ticket.

Fig. 14: ConSol*CM Web Client - Newly Edited Word Document as Attachment at the Ticket

From here on, the work with the Word document is exactly the same as the work with regular Word (.doc, .docx) attachments. See next section.

Working with Existing MS Word Attachments

An engineer can also open an MS Word document which has been attached to the ticket. As an engineer, click on the attachment name. This will open the file in MS Word. Edit the document and save it. A new version of the document will be attached to the ticket automatically.

42.3.8 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 Doc 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 path 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 CM Web Client is identical to the handling of Microsoft Word documents. When preparing a document template with 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 MS 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".

Fig. 15: Creating a Doc Template Using OpenOffice

On this tab a CM field can be added by first selecting the Type *User Field* and the format *Text*. The next step is entering the field name taken from the column *Key* in the document template administration and putting it in the dialog field labeled *Name*. Add a useful value for recognizing the field in the document. Click on the arrow symbol to the right of the field *Value* so that the field shows in the middle list *Selection* and make sure it is selected there. Clicking on the "*Insert*" button on the bottom right side of the dialog will insert the field into the document at the current cursor position. This has to be repeated for every field that should show up in the document.

43 Time Booking Using ConSol*CM

- General Introduction to Time Booking Using ConSol*CM
- Manual Time Bookings
    - Configuration of Manual Time Booking Using the Admin Tool
    - Manual Time Booking from a User's Point of View (Web Client)
    - Manual Time Bookings on the Engineer Profile Page
- Automatic Time Bookings (Available in CM Versions 6.9.4.2 and Up)
    - Introduction to Automatic Time Booking
    - Configuration of Automatic Time Booking
    - Automatic Time Bookings on the Engineer Profile Page
- DWH Reports
- Page Customization for Time Booking

43.1 General Introduction to Time Booking Using ConSol*CM

In ConSol*CM, working hours can be booked on tickets. There are two booking modes:

1. Manual bookings, see section Manual Time Bookings
2. Automatic bookings (available in CM versions 6.9.4.2 and up), see section Automatic Time Bookings

43.2 Manual Time Bookings

In ConSol*CM, an engineer can book working hours on a ticket. Those working hours can then be reported.

Working hours are either booked on projects which have to be assigned to one or more queues or they are booked directly on tickets. For example, if your company plans to perform a migration from Windows 7 to Windows 8 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 completed. Then engineers can book times on the project and can see their own reports for the project. Additionally, a report over all time bookings, of all engineers, might be implemented using the DWH (data warehouse, see section Data Warehouse (DWH) Management).

43.2.1 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.

Fig. 1: ConSol*CM Admin Tool - Global Configuration: Management of Projects

Fig. 2: ConSol*CM Admin Tool - Queue Administration: Assigning Projects to a Queue

43.2.2 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 short 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 to this ticket.

Fig. 3: ConSol*CM Web Client - Time Booking in a Ticket

2. Using the *Time booking* section in the **engineer profile page** to book working hours on a selected ticket (only tickets where the engineer has performed certain activities and tickets owned by the engineer can be selected).

Fig. 4: ConSol*CM Web Client - Time Booking in the Engineer Profile

43.2.3 Manual Time Bookings on the Engineer Profile Page

Engineers can see a list of their time bookings on the engineer profile page. An example is shown in the following figure.

Fig. 5: ConSol*CM Web Client - Time Booking Report in the Engineer Profile

As an engineer, you can select if 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.

43.3 Automatic Time Bookings (Available in CM Versions 6.9.4.2 and Up)

43.3.1 Introduction to Automatic Time Booking

If required, 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 pressed.
- **Duration of creating a ticket**
  Started when *Create ticket* is selected and stopped when the *Create* button is pressed.

Time booking is suspended when a ticket is transferred to the workspace and continued when the ticket is brought back to the active work.

No times are booked on the ticket in case ...

- an operation is canceled
- the engineer logs out manually
- the engineer session is ended automatically

Times are always booked in minutes and always rounded up to the next minute.

43.3.2 Configuration of Automatic Time Booking

In order to enable this functionality in your CM system, set the system property *cmweb-server-adapter, automatic.booking.enabled* to *true*.

Fig. 6: ConSol*CM Admin Tool - Set System Property to Switch On Automatic Time Booking

43.3.3 Automatic Time Bookings on the Engineer Profile Page

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.

Fig. 7: ConSol*CM Web Client - Automatically Booked Time in Time Booking Report on Engineer Profile Page

43.4 DWH Reports

If your company would like to have reports on 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.

43.5 Page Customization for Time Booking

In case the time booking feature is not required, you as an administrator 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

44 ConSol CM Administrator Manual 6.10.2 - Authentication Methods for Engineers in the CM Web Client

44.1 Authentication Methods for Engineers in the Web Client

For the authentication of engineers in the Web Client, you can configure one of the following three methods:

1. **Standard**
   All user (engineer) data is kept in the ConSol*CM database. User name and password are set on the Engineer Administration page, see section Engineer Administration.
2. **LDAP**
   The credentials are kept in an *LDAP* server. ConSol*CM sends a request to this server for the engineer authentication, see section LDAP Authentication.
3. **Kerberos SSO**
   The credentials are taken from a valid session using *Kerberos*, see section Single Sign-On with ConSol*CM Using Kerberos.

44.2 ConSol*CM LDAP Authentication

44.2.1 Introduction to ConSol*CM LDAP Authentication
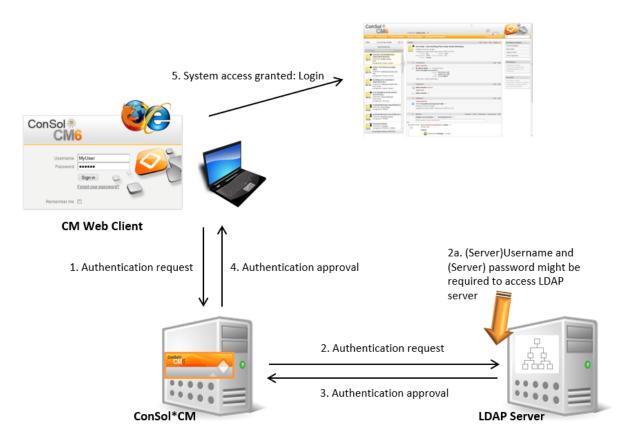
ConSol*CM offers LDAP authentication for the Web Client as a standard feature, i.e. instead of managing the passwords for the ConSol*CM engineers in the CM database, they can be retrieved from an LDAP server (like e.g. a *Microsoft Active Directory* server).

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 CM server sends a request with the engineer's user name and password and checks with the LDAP server if those credentials are correct.

When the credentials are correct, the approval is sent back to the CM server and the engineer is logged into the Web Client.

> ⓘ **Important information:**
>
> 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.

Fig. 1: ConSol*CM LDAP Authentication Process

44.2.2 Configuring the System to Enable LDAP Authentication

There are two ways you can enable the ConSol*CM system to use LDAP authentication:

1. Select *LDAP authentication* during system set-up and enter the requested parameters (system properties) after the set-up.
2. 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 the System During Initial Set-Up

During system set-up you can select LDAP as 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 subsequent section for an explanation.

Fig. 2: ConSol*CM System Set-Up - Authentication Mode LDAP

Switching the Authentication Mode to LDAP in a Running System

cmas-core server	attachment.max.size	100
cmas-core-security	authentication.method	LDAP
cmas-core-security	ldap.authentication	simple
cmas-core-security	ldap.basedn	OU=accounts,DC=consol,DC=de
cmas-core-security	ldap.initialcontextfactory	com.sun.jndi.ldap.LdapCtxFactory
cmas-core-security	ldap.password	
cmas-core-security	ldap.providerurl	ldap://ldap.consol.de:389
cmas-core-security	ldap.searchattr	uid
cmas-core-security	ldap.userdn	

Fig. 3: ConSol*CM Admin Tool - Properties for LDAP Authentication

Required values for LDAP authentication (they are set via *system properties*, please see Appendix C for an explanation):

- **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. Usually should 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 LDAP server to look up users. Only needed if look-up cannot be done anonymously.

> ⚠️ **Attention:**
>
> 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 for a first check.

- **ldap.providerurl**
  The complete URL for the LDAP server:

  ```
  ldap://<HOSTNAME>:<LDAP PORT>
  ```

- **ldap.searchattr**
  Search attribute for looking up LDAP entry connected to CM login, i.e. the attribute which is used as user name for the authentication.

44.2.3 Managing Engineer Accounts for LDAP Authentication

Use the *Engineer Administration* 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.

Fig. 4: ConSol*CM Admin Tool - Engineer Administration

- **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 you can fill in this field. During the LDAP authentication process, this
  LDAP ID is used as the user name which is looked up in the LDAP directory in the *ldap.searchattr*
  node.

44.2.4 Using LDAPS (LDAP over SSL)

Introduction

Per default, when an LDAP client accesses an LDAP server, the information is transferred by default 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 is can use certificates. One way to do this is described in the following section for a Linux machine.

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 <beliebig> -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 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 CM server as shown in the following example:

- cmas-core-security;ldap.authentication = simple
- cmas-core-security;ldap.basedn;DC = myDC,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

44.3 Single Sign-On with ConSol*CM Using Kerberos

44.3.1 Configuration of Kerberos Single Sign-On

Introduction

The *single sign-on* feature relating to ConSol*CM allows users to authenticate against ConSol*CM automatically with 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) is used.

The single sign-on feature is based on the *Kerberos V5* protocol, which is integrated in the *Windows Active Directory*. All information is encrypted using *RC4* and *HMAC*.

The web server works as a *non-Windows Kerberos service* and can be installed on any operating system /application server.

Client and web server use GSSAPI and SPNEGO to exchange authentication information.

This guide shows you how to set up single sign-on in a Windows (Active Directory) environment as this is the most common scenario for our customers.

Requirements

For Kerberos-based single sign-on you need:

- Domain controller for the Windows domain
- ConSol*CM server
- Windows clients

44.3.2 Setting Up the System

Domain Controller

The first step is to configure the domain controller so that it knows the ConSol*CM server. In our example the domain controller is called *win2003srv*, the domain is *CM6SSO.CONSOL.DE*.

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 it is the computer *xp1cm6*.

> ⚠️ **Attention:**
>
> The radio button *Trust this computer for delegation to any service (Kerberos only)* must be activated!

Fig. 1: Registration of 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, in our example the user *tomcat*.

The following account options must be enabled:

- account is trusted for delegation
- no Kerberos pre-authentication needed

Fig. 2: Registration of ConSol*CM Server User

Generating the keytab File

On the domain controller the ConSol*CM server is created as a new Kerberos service, additionally a
Kerberos *keytab* file is generated. This file will be needed later on the ConSol*CM server machine. This
keytab file contains the shared secret key of the service.

```
C:\Programme\Support Tools>ktpass /out tomcat.keytab /ptype KRB5_NT_PRINCIPAL /princ HTTP
/xp1cm6.cm6sso.consol.de@CM6SSO.CONSOL.DE /pass consol.123 /mapuser tomcat /crypto rc4-hmac-nt
```

> ⚠ **Attention:**
>
> If *ktpass* is not available, the *Windows Server 2003 Support Tools* must be installed, available
> here.

ConSol*CM Server

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 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 class path:

- Under JBoss this can be:
  ```
  ../jboss/server/{domain}/conf/cm6-kerberos.properties
  ```
- Under WebLogic this can be:
  ```
  ../{domain}/cm6-kerberos.properties
  ```

In case you have a cluster of more than one ConSol*CM servers in operation, each server has to have a separate properties file.

Properties in this file should contain:

- Reference to a Kerberos config file (e.g. *krb5.ini* or *krb5.conf*)
- One or more service principal(s), i.e. reference to *keytab* file

Example for cm6-kerberos.properties:

```
# path to kerberos configuration
kerberos.config.location=C:\\conf\\krb5.ini

# one or more service principals (principal = path to keytab file)
HTTP/xp1cm6.cm6sso.consol.de@CM6SSO.CONSOL.DE=C:\\conf\\tomcat.keytab
```

Example for krb5.ini:

```
[libdefaults]
   default_realm = CM6SSO.CONSOL.DE
   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]
   CM6SSO.CONSOL.DE = {
      kdc = w2003srvcm6
      admin_server = w2003srvcm6:88
   }
[domain_realm]
   .w2003srvcm6 = CM6SSO.CONSOL.DE
   w2003srvcm6 = CM6SSO.CONSOL.DE
```

File keytab

Copy the *keytab* file you generated on the domain controller to the location you specified in the *cm6-kerberos.properties* config file.

> ⚠ **Attention:**
>
> You have to restart the ConSol*CM server process!

Client Machine

Internet Explorer

The *Internet Explorer* needs to be configured so that 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*.

In detail, the following settings for login behavior are available.

Fig. 3: Internet Explorer Login Configuration

Settings and resulting behavior:

- **Anonymous logon**
  No single sign-on is possible, user will get 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 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

In 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://xp1cm6* if that is the URI).

You can set this property also on the file system. Open the file

C:\Dokumente und Einstellungen\[USER]\Anwendungsdaten\Mozilla\Firefox\Profiles\XYZ.default\prefs.js

and add/replace the following line:

```
user_pref("network.automatic-ntlm-auth.trusted-uris", "http://xp1cm6");
```

> ⚠️ **Attention:**
>
> You have to restart Firefox after this change.

44.3.3 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:

> **You have been automatically logged in and a new session has been created for you.**   ⊠

It is still possible to login as another ConSol*CM user by clicking on 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 you will enable single sign-on, 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/cm6.consol.de@CONSOL.DE=/etc/krb5_consolde.keytab
HTTP/cm6.consol.de@CONSOL.PL=/etc/krb5_consolpl.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 two following 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 *cmas-core-security*, *kerberos.v5.username.regexp* property 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@cm6sso.consol.de* 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 -> page *Configuration* -> tab *CM Services* -> *Kerberos v5 authentication provider*, see section CM Services.

45 CTI with ConSol*CM: CM.Phone

45.1 Introduction to CM.Phone

CM.Phone is a distinct ConSol*CM module which has to be licensed in addition to the core CM system.

CM.Phone is a Windows client application for 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.

Fig. 1: ConSol*CM.Phone - Basic Principle

45.1.1 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 created, displaying the phone number of the caller. The 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. In case no corresponding customer data matches the phone number, just the calling number is displayed and the option *Create customer* is offered.

> ⚠ **Attention:**
>
> 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 CM server. These templates are loaded by the CM.Phone client application during startup. The information displayed in the pop-up window (data object group fields from the customer data model) can be customized by editing the template files (see section Configuring the Client Pop-Up Window).

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 offered, if the caller is unknown in CM).
- **Create ticket**
  Opens the *Create ticket* page for this found (or new) customer in the Web Client.
- **Call back**
  Will be available in case of a missed call.
- **Close**
  Closes the CM.Phone pop-up window.

In case, the customer is not yet present in the CM system, the caller's phone number will be used to fill-in the phone number field in the customer data (data object group 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. In case the user has access to multiple customer groups the respective *dialable* phone number fields of each customer group will be pre-filled.

45.1.2 Outgoing Calls

The engineer can start an outgoing call directly by clicking on a phone number (e.g. in the customer data) in a data object group 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.

45.2 CM.Phone Set-Up

45.2.1 System Requirements

Please refer to the *System Requirements* of the ConSol*CM version which is installed in your environment for detailed information concerning server and client requirements for CM.Phone set-up.

45.2.2 Components Required for CM.Phone Set-Up

For CM.Phone set-up you, as an administrator, need:

- The license for CM.Phone, please ask your ConSol*CM consultant.
- The CM.Phone *.war* file for deployment in the application server.
- A *TAPI 3* driver in the telephone system.

45.2.3 Installing CM.Phone on the Application Server

Basic CM.Phone Server Installation

The CM.Phone module is delivered as a *.war* package. The *.war* package is provided as folder (not as packed file), because in this way you can easily access the configuration files located in this path.

For the server installation, perform the following steps:

1. Copy the *cm-phone.war* folder to *<CM6 path>/server/cmas/deploy*.
2. Adapt the configuration as needed (see sections below).
3. Restart the application server.

To check if the application was deployed correctly, visit the web URI:

http://<CM server>:<CM port>/cm-phone/

You should see the welcome page of the application with the link to the CM.Phone installer download.

Configuring CM.Phone on the Application Server

All application parameters are set in the file **cmphone-config.xml** which is located under *<CM6 path>/server/cmas/deploy/cm-phone.war*.

This file is loaded by the CM.Phone client application during start-up and each time the settings dialog is closed with *OK*.

However, this file should not be edited in order to configure the system. Please make sure that all configuration parameters are set using the Admin Tool.

Configuring the Client Pop-Up Window

The contents of the pop-up window are based on the HTML templates in the following path: *<CM6 path>* */server/cmas/deploy/cm-phone.war/templates*. The main directory contains the templates for the default language (of the client system!):

- CallNotification.html
- ContactData.html
- ContactList.html

For each additional language which should be supported, a folder with the name of the locale has to be created (e.g. *de* for *German*) which contains localized copies of the template files.

The templates are used to render the customer details in the pop-up window. Since every customer may want to see different information in the pop-up, the content can be adapted by editing the HTML files. The templates contain tags which are replaced with actual values by the client application during a call. Those templates are Admin Tool templates which have to be defined for each customer group, please see section Configure the Admin Tool Templates for Customer Data for Each Customer Group for details.

If required, you can change the names of the templates. You can use any file name you want for the three HTML files, just make sure you have entered the correct values in the *config* file.

CallNotification.html

This is the first template which is displayed as soon as an incoming call is detected by the CM.Phone client. This window only displays the calling number since at this time there is no customer data available.

Available tags:

- **[phonenumber]**
  Phone number of the caller.
  Example: <h1>Phonenumber: [phonenumber]</h1>
- **[calltime]**
  Time of the call.
- **[content]**
  This will show additional information within the pop-up window:
  A *Loading* icon during the customer look-up or an error message if something went wrong during the look-up, e.g. wrong user name or password, etc.

> ⚠️ **Attention:**
>
> These tags are case sensitive and must be lower case.

ContactData.html

This will display the actual customer details if the look-up successfully found a matching customer for the phone number. In this template all data object group fields of the customer data model can be used as tags. This way, the displayed customer details can be adapted to any customer's need.

All tags from the *CallNotification* template are available (see above), plus the following tags:

- **[contact.id]**
  CM internal ID of the customer. This ID may be used to create additional links into the CM Web
  Client.
- **[contactContent]**
  Here, the customer data is filled in according to the template which is defined for each customer
  group in the Admin Tool under <CustomerGroupObject> -> Templates -> *CMPhone Customer Details*
  .

ContactList.html

In case the look-up finds more than one customer for a phone number, the *ContactList* template is displayed
in the pop-up window. For each found customer, a row is added in the customer table within the template.

Here, the contact data is filled in according to the template which is defined for each customer group in the
Admin Tool under <CustomerGroupObject> -> Templates -> *CMPhone Customer List*.

Links/Buttons within the Templates

Four buttons are configured in the standard templates:

- **Close**
  Closes the pop-up window.
- **Call**
  Starts an outgoing phone call to the calling number.
- **Open Customer**
  Opens the customer in the Web Client.
- **Create Ticket**
  Opens the Web Client in the *Create ticket* page.

These buttons can also be customized within the templates. Buttons may be removed or additional buttons
or links may be added as required. Each button refers to an HTML link.

For CM.Phone there are two types of links available:

1. **External links**
   These links will open a browser window to display the page.
   For example, the link *http://heldesk/cm-client/contact/[contact.id]* will open the Web Client and display
   the selected customer.
   For these links, all data object group fields of the customer data may be used to create the URL. This
   way, additional functions may be added by creating a link to a customer-specific web application and
   by passing user data from CM as parameters, e.g.: *https://intranet.mycompany.de/index.php?
   id=234&id_person=[customer.personid]*.
2. **CM.Phone internal links**
   These links are only valid within the pop-up window.
   Format: http://cmphone/<command>/?
   The following commands are available:
   a. **contactdata**
      Displays the *ContactData* template for the selected customer.

        Parameter: Contact Id

        This command is used in the *ContactList* template to allow the user to select and display a specific customer:

        http://cmphone/contactdata/?[contact.id]

  b. **contactlist**

        Displays the *ContactList* template.

        This command is used to allow the user to go back to the list of customers from the *Customer Data* page:

        http://cmphone/contactlist

  c. **call**

        Starts an outgoing phone call to the phone number of the customer:

        http://cmphone/call

  d. **runcmd**

        Starts a local application on the client PC.

        Parameter: Command line of the application

        This may be used for instance to start a database application and pass a user ID as a command line parameter, e.g. :

        http://cmphone/runcmd/?dbapp.exe +userid=[customer.userid]

  e. **close**

        Closes the pop-up window:

        http://cmphone/close

Replacing the CM.Phone Pop-Up Window by a Custom Web Application

In case a customer does not want to use the pop-up window from CM.Phone but instead requires a custom web application to be opened for a phone call, this can be done by setting the *OnCallCmd* parameter in the *cmphone-config.xml* file. If it is set to an external URL, a browser window will be opened with this URL for each phone call. The pop-up window will not be displayed.

45.2.4 Installing CM.Phone on Each Windows Client

The CM.Phone client application has to be installed locally on all client PCs that need CTI functionality. The set-up package has to be downloaded from the CM.Phone start page on the CM server.

> ⚠️ **Attention:**
>
> Administrator rights are needed on the client PC to install the CM.Phone application. The reason for this is the registration as *phone: protocol handler* which requires a registry key to be written.

For the client installation, perform the following steps:

1. Open the CM.Phone start page *http://<CM6-URL>/cm-phone*:
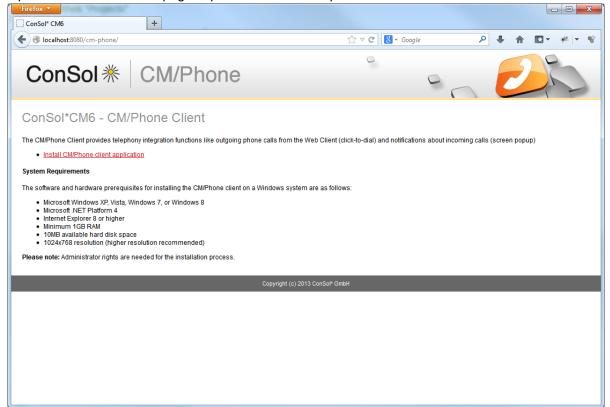
Fig. 2: ConSol*CM.Phone - Client Set-Up (1)

2. Download and run the installation package *CMPhoneSetup.msi*:

Fig. 3: ConSol*CM.Phone - Client Set-Up (2)

3. Start the CM.Phone application. Start *All programs -> ConSol CM6 -> CM Phone.* After the first start it will display the configuration dialog:

Fig. 4: ConSol*CM.Phone - Client Set-Up (3)

Fill in the following fields:

- **Line**

  Select the TAPI line which should be used.

- **CM6 URL**

  Enter the URL of the CM system. The basic URL is required, e.g. *http://myserver:8080*.

- **CM6 Username**

  Your CM user name (the user has to have enough rights to search for data objects through CM REST API !).

- **CM6 Password**

  Your CM password.

> ⓘ **Information:**
>
> The configuration dialog can be opened anytime by opening the context menu of the CM. Phone notification icon in the task bar and selecting *Settings ...* .

Engineer Authentication Modes

The ConSol*CM CTI client will not work together with SSO authentication mode (see section Single Sign-On with ConSol*CM Using Kerberos).

45.3 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 data object group 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 prefix for outgoing calls.

45.3.1 Set Annotations for Data Object Group Fields Containing Phone Numbers

Fig. 5: ConSol*CM Admin Tool - Annotations for Data Object Group Fields with Phone Numbers

Fig. 6: ConSol*CM Web Client - Dialable Number When Using CM.Phone

Two annotations are required for the data object group fields which contain phone numbers:

- **dialable = *true***
  This configures the phone numbers as dialable links in the Web Client.
- **field-indexed = *local***
  This makes the field searchable which is important for the customer look-up.

45.3.2 Configure the Admin Tool Templates for Customer Data for Each Customer Group

The customer data model configuration also allows for two more contexts/types of data object templates now:

- CMPhone customer details
- CMPhone customer list

They are used for defining how CM.Phone should render incoming call information. The first one is used for exactly one data object matching the phone number and the second one is used for multiple matches, so that the engineer has to select the correct customer.

You have to perform two steps:

1. Write the templates and store them in the *Script and Template Administration* of the Admin Tool.
2. Assign the templates to customer data models (*User attributes* - tab *Customer data model*).

Fig. 7: ConSol*CM Admin Tool - Example Template for Rendering Customer Data for Display in CM.Phone

Fig. 8: ConSol*CM Admin Tool - Assignment of CM.Phone Templates for Customer Data to Customer Groups

45.3.3 Configure the Phone Number Format for Each Customer Group

The format defined here is used to transform incoming numbers (from the respective customer group) to a common canonical form exchange format. 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 data object. 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 group *Customers*, navigation item *Customer Groups* 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 which now contains 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 if an outgoing call is going to be performed within the same country or not. Several phone providers do not handle canonical (so theoretically correct) numbers for domestic calls. Therefore the country prefix has to be cut-off 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 if the number provided is a full subscriber number (eventually 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 also a country prefix will be added, so the

result will be: 49600289906. If the engineer enters a full number starting with "+" or "0" then the configuration is skipped - CM assumes no number conversion is required.

Fig. 9: ConSol*CM Admin Tool - 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.

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, if it is already fully qualified. Otherwise 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 for a valid number to be dialed. In all other cases the supplied original number cannot be used for making a connection.

45.3.4 Set the System Properties

There are three new properties with relevance for CM.Phone in CM to be set in the Admin Tool. 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**
  This is the local country code. The value is an international country code like *49* for Germany, for example. Default value is *49*.

- **internal.line.access.prefix**
  This is a prefix that the company's telephony system asks 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.

45.3.5 Change the Prefix for Outgoing Calls

> ℹ️   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 CM consultant for advice.

45.4 Troubleshooting

45.4.1 Logging

For debugging purposes, a log file may be activated on the client. In order to do this, the log configuration file *log4net.xml* in the installation path of the client, usually *<Program Files>\ConSol\CMPhone*, has to be configured.

Since most users do not have *write* access to the *Program Files* directory, the log file path has to be set to a folder that is writable for the user, e.g.:

```
Log4net configuration for CM.Phone logging

<file value="c:\temp\cmphone.log" />
<appender name="RollingFileAppender" type="log4net.Appender.RollingFileAppender">

    <file value=" c:\temp\cmphone.log" />
     <appendToFile value="true" />
     <rollingStyle value="Size" />
     <maxSizeRollBackups value="10" />
     <maximumFileSize value="1MB" />
     <staticLogFileName value="true" />
    <layout type="log4net.Layout.PatternLayout">
        <conversionPattern value="%date %-5level %logger - %message%newline" />
    </layout>
</appender>
```

Notes:

- Special characters and/or whitespace do not have to be masked.
- Use the following to write the log file into the user's home directory:

  ```
  <file value="${USERPROFILE}\phone.log" />
  ```

45.4.2 Registration as phone: protocol handler

If the client application cannot be installed by the users themselves because of insufficient access rights, the application may be distributed by an administrator employing a software distribution system. In that case, the application needs to be registered on the client as *phone: protocol handler* by creating the appropriate registry keys:

- [HKEY_CLASSES_ROOT\phone]
  Please make sure to set the space in *URL Protocol*, otherwise it will not work.

- [HKEY_CLASSES_ROOT\phone\DefaultIcon]
  @="C:\\Program Files (x86)\\ConSol\\CMPhone\\cmphone.ico"
- [HKEY_CLASSES_ROOT\phone\shell]
- [HKEY_CLASSES_ROOT\phone\shell\open]
- [HKEY_CLASSES_ROOT\phone\shell\open\command]
  @="C:\\Program Files (x86)\\ConSol\\CMPhone\\cmphone.exe" (example)

46 ConSol CM Administrator Manual 6.10.2 - The Customer Portal - CM.Track

46.1 The Customer Portal: CM.Track

The portal CM.Track allows customers to log in to the ConSol*CM system. Like the 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 Appendix B (Glossary)) API.
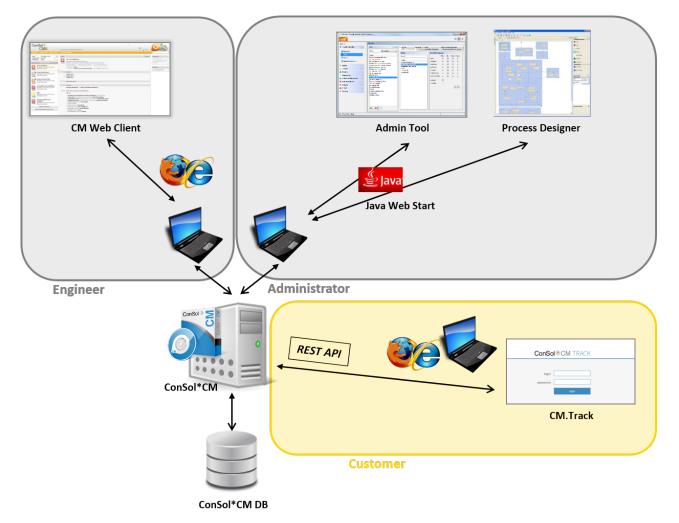
Fig. 1: ConSol*CM System Architecture with CM.Track

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 to a ticket.
- Open/Create a new ticket.
- Search the FAQs for solutions.

See the following sections for topics which concern CM.Track:

- General system access to CM.Track for customers:
  See section System Access for CM.Track Users (Customers).
- Customer authentication modes:
  See section CM.Track: Authentication Modes for the Portal.
- Using the portal for FAQs:
  See section FAQs in CM.Track.

46.2 System Access for CM.Track Users (Customers)

- Precondition
- CM.Track Technical Background
- General Principle of System Access via CM.Track
- Defining the User Profiles/Access Permissions for CM.Track
- Defining the Data Object Group Fields for CM.Track Login and Password
- Granting Access to CM.Track for Customers
- Customer Login to the System
- Extended Customer Permissions to See Company Tickets
- Configure CM.Track for Password Reset by 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.

> **ⓘ Information:**
>
> CM.Track 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 CM license for every customer.

46.2.1 Precondition

Technically, CM.Track is part of every default shipment of ConSol*CM, so there are no new files that have to be deployed. However, the default function set is rather rudimentary and the pages have a rather plain 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) which is 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.

46.2.2 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.

46.2.3 General Principle of System Access via CM.Track

A customer who wants to 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 CM Web Client, or the values can be imported automatically into the database.

The fields for the login and password of customers are data object group fields which are defined like any other data object group field and which have special annotations. If you are not familiar with data object group fields, please refer to section Data Object Group Field Management and GUI Design.

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.

46.2.4 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*.
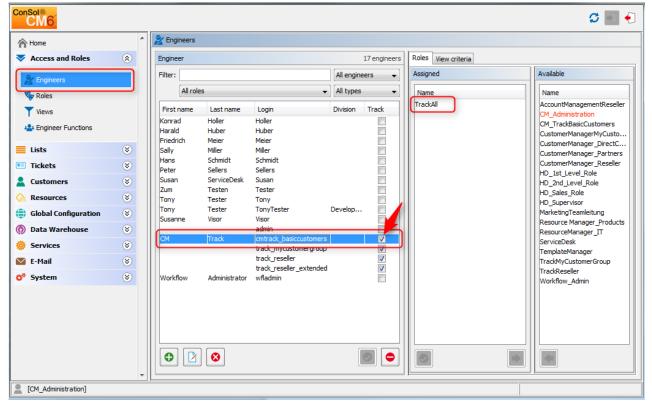
Fig. 1: ConSol*CM Admin Tool - CM.Track: User Profile Name

The user profile is 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) TrackAll that has the role *TrackAll*. This role has read/write/append permissions to the queues *Helpdesk_2nd_Level* and *ServiceDesk*. For a detailed introduction to role administration, please refer to section Role Administration.
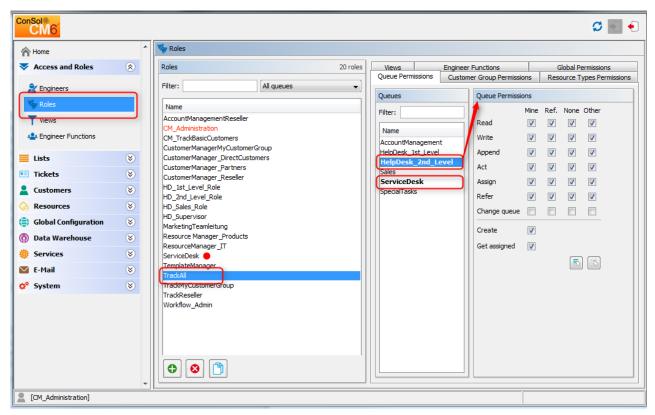
Fig. 2: ConSol*CM Admin Tool - CM.Track: User Profile - Role

In this way, a customer with the CM.Track user profile TrackAll 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.

46.2.5 Defining the Data Object Group Fields for CM.Track Login and Password

The fields for login and password for a customer are regular data object group fields on contact level. Please see section Setting Up the Customer Data Model for an introduction to data object group 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*.

Fig. 3: ConSol*CM Admin Tool - CM.Track: Annotation for Login

- 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 plain text password.

Fig. 4: ConSol*CM Admin Tool - CM.Track: Annotation for Password

46.2.6 Granting Access to CM.Track for Customers

The engineer who works 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.

Fig. 5: ConSol*CM Web Client - CM.Track Users

46.2.7 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.

Technically, there are two ways for the user authentication:

- simple authentication
- LDAP authentication

Please refer to section CM.Track: Authentication Modes for the Portal for details.

Fig. 6: ConSol*CM.Track - Customer Login

Fig. 7: ConSol*CM.Track - Ticket List

46.2.8 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 right *Access tickets of the own company* under *Track User Permissions*. Please refer to the section Role Administration for a detailed explanation.

46.2.9 Configure CM.Track for Password Reset by Customers

Starting with 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 *track-password-reset-template*. Please refer to section Password Reset Template for Customers Using CM.Track 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 CM.Track: Authentication Modes for the Portal for an explanation of all possible authentication modes.

46.3 CM.Track: Authentication Modes for the Portal

46.3.1 Introduction to Authentication Modes in CM.Track

Customers who log in to the ConSol*CM portal (CM.Track) use their login and password. Both are Data Object Group Fields in the contact data.

There are three possible authentication modes:

- Against the ConSol*CM database. This is called *Database Mode*.
- Against an LDAP server. This is called *LDAP Mode*.
- Against an LDAP server and the ConSol*CM database.
  The order can be configured. This is called *Mixed Mode*.

46.3.2 Definition of the CM.Track Authentication Mode

The authentication mode is determined 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 Appendix C (System Properties)) and their respective system behaviors are:

- **DATABASE**
  Attempting a database login, if the unit has a database password, i.e. the login and password are stored in the CM database and are thus managed by the CM engineers or indirectly by the customers themselves when they reset their password. The customer can reset hsi password, see section Admin Tool Templates, Password Reset Template for Customers Using CM.Track.

- **LDAP**

  Trying 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 is authentication using the available LDAP server(s), if an LDAP ID is provided. On failure trying a database login, if the unit (customer) has a database password.

- **DATABASE,LDAP**

  First attempt is a database login, if the unit (customer) has a database password. On failure trying authentication using the available LDAP server(s), if an LDAP ID is provided.

The values are case insensitive, commas and whitespace are ignored.

46.3.3 DATABASE Authentication Mode

System Property for DATABASE Authentication Mode

Set the system property *cmas-core-security*, *contact.authentication.method* to *DATABASE* (this is the default value).

Data Object Group Fields for Contact Login

Two Data Object Group Fields for the contact data are required:

- Login
- Password

Please see section System Access for CM.Track Users (Customers) for a detailed explanation.

46.3.4 LDAP Authentication Mode

System Property for LDAP Authentication Mode

Set the system property *cmas-core-security*, *contact.authentication.method* to *LDAP*.

System Properties Defining the LDAP Server(s)

The LDAP servers can be defined using the following configuration properties from the module *cmas-core-security*.

{name} is a string that you can choose to distinguish LDAP servers. It always has to be set, even if only one LDAP server is configured. You should use a simple string for the {name}, which does not contain any keywords like *internal* or *external* and which does not contain special characters.

- **ldap.initialcontextfactory**
  This is an already existing global property. If it is not set, *com.sun.jndi.ldap.LdapCtxFactory* is being used as a value.
- **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, in case 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, in case 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=consol,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*.

A change of any of the above configuration properties does not require a server restart and is propagated to all cluster nodes. The use of the placeholder {name} allows to define several different LDAP servers.

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).

Data Object Group Field for Contact Login

In case LDAP Mode should be used, besides the annotation *username = true*, the Data Object Group Field which is used for the CM.Track user name (login) has to have an additional annotation:

- **ldapid**

Fig. 1: ConSol*CM Admin Tool - Data Object Group Field for LDAP Authentication of CM.Track users
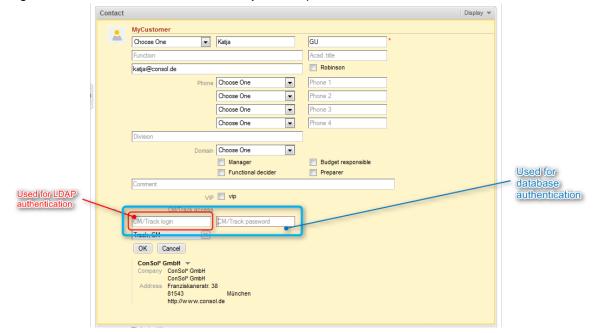
Fig. 2: ConSol*CM Web Client - Field (Red) for LDAP ID in Contact Data

46.3.5 Mixed Authentication Mode

System Property for Mixed Authentication Mode

Set the system property *cmas-core-security*, *contact.authentication.method* depending on the desired order of authentication instances:

- LDAP,DATABASE
- DATABASE,LDAP

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]
```

System Properties Defining the LDAP Server(s)

See respective paragraph in section *LDAP Authentication Mode*: LDAP Servers.

Data Object Group Field for Contact Login

See respective paragraph in sections *LDAP Authentication Mode*: LDAP ID

46.3.6 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.

46.3.7 Using LDAPs for Authentication

The LDAPs authentication for CM.Track follows the same principle as using LDAPs for the authentication in the CM Web Client. Please refer to section LDAP Authentication in the CM Web Client, LDAPS.

46.4 FAQs in CM.Track

- Introduction to FAQs in CM.Track
- Configuring the ConSol*CM System to Allow FAQ Search in CM.Track
- FAQ Search in CM.Track from a Customer's Point of View
- More Complex Solutions for Managing FAQs
    - Using Two FAQ Queues: FAQ Management and Active FAQs
    - Setting Up Two (or More) Parallel FAQ Environments Using Track Users

46.4.1 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 solution already. Only if they do not find any help, they will contact the service desk and/or open a new ticket. This saves time for both customer and service team. It might also be employed in other environments where you would like to offer this service.

According to ConSol*CM standard, every FAQ is treated as a ticket. The queue(s) which should be available as FAQ queue(s) via CM.Track have to be defined as special FAQ queues, because usually customers are 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 queue(s). Here, only read access has to be granted.

46.4.2 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 (check box *FAQ*).

Fig. 1: ConSol*CM Admin Tool - Queue Administration

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.

Fig. 2: ConSol*CM Admin Tool - Role Administration

Then this new role has to be assigned to the user (profile) which is used as CM.Track access user (see section System Access for CM.Track Users (Customers)).

Fig. 3: ConSol*CM Admin Tool - Engineer Administration

46.4.3 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. This might be a solution as in the following example or other service information.

Fig. 4: ConSol*CM.Track - Example for FAQ Search (1)

Fig. 5: ConSol*CM.Track - Example for FAQ Search (2)

46.4.4 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 can 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 offered 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.

47 ConSol CM Administrator Manual 6.10.2 - System Architecture

47.1 System Architecture

ConSol*CM is a Java EE application. It can be operated as

- **CM-only system**, see section System Overview (CM-Only System)

or as

- **CM system with reporting infrastructure**, see section Infrastructure of a CM System with DWH

47.2 System Overview (CM-Only System)

47.2.1 System Architecture

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. For a detailed description of the system, please refer to the *ConSol*CM Operations Manual* and the *ConSol*CM Set-Up Manual*.

A detailed list of supported application servers, database systems, and other systems is given in the current *System Requirements*.

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). For a couple of years now, JWS has been part of every Java edition, so no specific download 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.
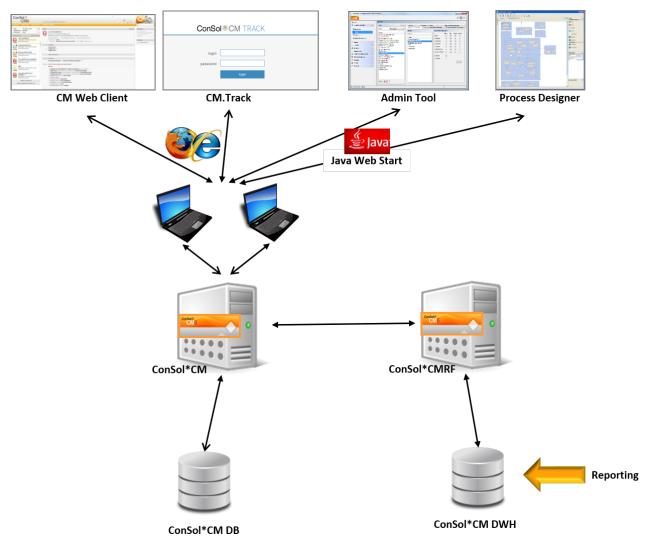
Fig. 1: ConSol*CM - Basic System Architecture

CM Database

The ConSol*CM Database (CM DB) is a relational database which can be operated as Oracle, MSSQL Server or MySQL system (please see *ConSol*CM System Requirements* for details).

- **Oracle**

  One database scheme with one database user is used by CM.
- **MS SQL**

  One database scheme with one database user is used by CM.
- **MySQL**

  One database with one database user is used by CM.

Components for E-Mail Interactions

One of the core functionalities of ConSol*CM is the integration with mail servers. This allows CM to send and to receive e-mails. For the engineer this means, new tickets can easily be opened via e-mail and the entire communication about a case is located in the respective ticket, including all incoming and outgoing e-mails.

In order to receive e-mails, ConSol*CM connects to a mail server and retrieves e-mails from one or more mailbox(es). CM reacts like a regular e-mail client (e.g. Thunderbird, MS-Outlook) and uses standard e-mail protocols like IMAP or POP3. In case you want to use the secure version, IMAPs and POPs are also supported. In this case, the required certificates have to be placed on the server.

In order to send e-mails, ConSol*CM uses an SMTP server.
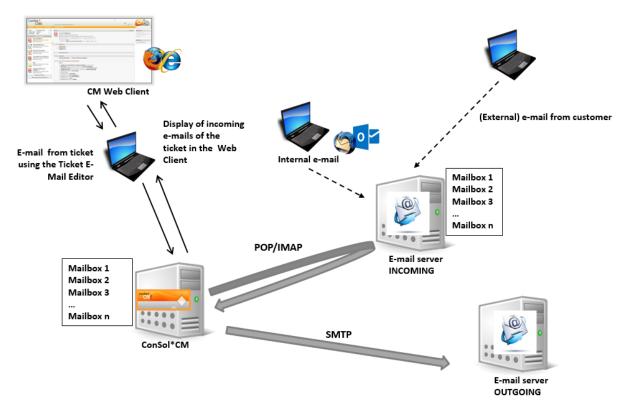
Fig. 2: ConSol*CM - E-Mail Server Interactions

System Architecture with Reporting Infrastructure

This is explained in section Infrastructure of a CM System with DWH.

47.2.2 Short Overview of the File 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 that has been entered during system set-up.

ConSol*CM Data Directory

The following figure and list show examples from a Windows and a Linux system:
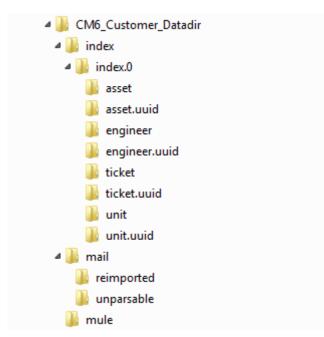
Fig. 3: ConSol*CM - Data Directory (Windows)
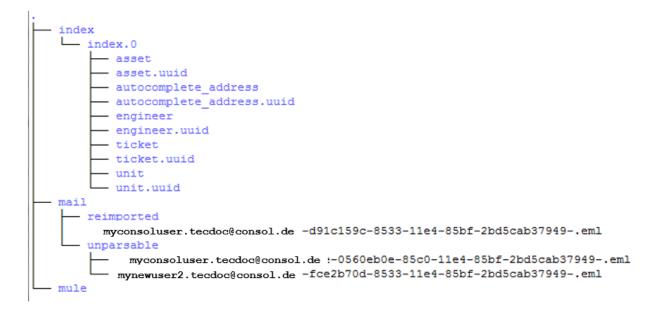
Fig. 4: ConSol*CM - Data Directory (Linux)

Example directories:

- **index**
  This is the directory where all the indexes are stored (see also section Search Configuration and Indexer Management). Be sure to include it into your daily file system back-up.
  - **index.0**
    In this directory, there is a subdirectory for each required index.
- **mail**
  This directory is only relevant when the system is operated in Mule/ESB mode. In case NIMH is used, all data is stored in the database. In this directory, files that are relevant for incoming e-mails are stored.
  - **reimported**
    In this directory, e-mails are stored that had been stored in the *unparsable* directory and could then be re-imported by a manual action of the administrator.
  - **unparsable**
    In this directory, incoming e-mails that cannot be processed by the system are stored. They are listed under *E-Mail Backups* in the Admin Tool, see section E-Mail Backups.
- **mule**
  This is a directory which might be used for *Mule* (internal *ESB*) data.

JBoss 5 Application Server File Structure

The following directories are available in a JBoss 5 installation of ConSol*CM:
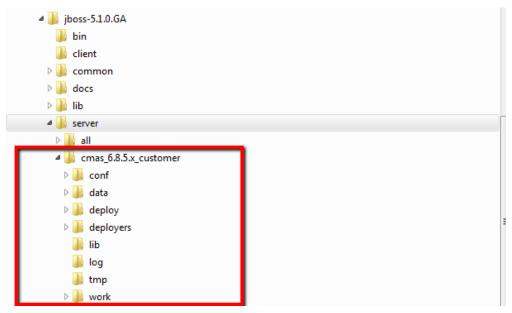
Fig. 5: ConSol*CM - File Structure in a JBoss Application Server

Example directories:

- **conf**

  Configuration data, e.g.:
    - **jboss-log4j**

      Log file configuration
- **data**

  Data for operation, e.g. *tx-operation* keys
- **deploy**

  Deployed data and configuration data:
    - **cm6.ear**

      Core application, *.ear* file
    - **cm-track.war**

      Application file for the portal ConSol*CM.Track
    - **cmDb-ds**

      Database connection configuration
- **deployers**

  Additional deployed application data
- **lib**

  Application-specific libraries, e.g.:
    - **mysql connector**

      In case you use MySQL as a database system.
- **log**

  Log files, see section Log Files.
- **tmp**

  Temporary data
- **work**

  Work directory with a working copy of the application server files. Can be emptied, e.g. for error analysis and/or fixing.

JBoss 7 Application Server File Structure

The following directories are available in a JBoss installation of ConSol*CM:
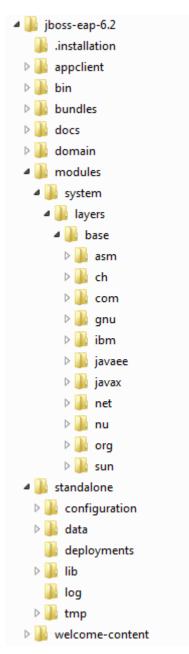
Fig. 6: ConSol*CM - File Structure in a JBoss 7 System

Example directories:

- **modules\system\layers\base**

  Subfolders contain the *JDBC* drivers:
    - com\microsoft\sqlserver\jdbc\main\sqljdbc4.jar

      (MS 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 single-server environments:
    - **configuration**

      Configuration of the DB connection and logging in the file *cm6.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. Can be emptied, e.g. for error analysis and/or fixing.
- **domain**

  Configuration in domain 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 as well as CMRF are *managed servers*. Please see the *ConSol*CM Operations Manual* for details.

Here, only some directories are explained. If you want to administer ConSol*CM as a WebLogic application, please refer to the *ConSol*CM Operations Manual* and to general Weblogic tutorials.
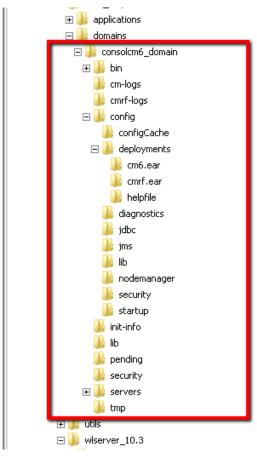
Fig. 7: ConSol*CM - File Structure in an Oracle WebLogic Application Server

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

Log Files

The logging behavior can be configured by editing the file(s)

- **log4j.xml**

  (in JBoss 5, where *Log4J* is used as logging framework)

- **cm6.xml** and/or **cm6-cmrf.xml**

  (in JBoss 7 standalone, where the built-in logging module of JBoss 7 is used)

- **domain.xml**, tag <subsystem xmlns="urn:jboss:domain:logging:1.3">

  (in JBoss 7 in domain mode, where the built-in logging module of JBoss 7 is used)

Log File Types

The following log files are used:

- **boot.log**

  Messages concerning system start-up (e.g. the Java version is indicated).

- **cmrf.log**

  Messages concerning CMRF (ConSol*CM Reporting Framework), i.e. messages that concern the
  data transfer operations from the ConSol*CM database to the CMRF database (DWH). This is done
  using *JMS* (Java Messaging Service).

- **cmweb.log**

  Messages concerning the ConSol*CM Web Client.

- **ctx.log**

  Contains messages of the *Spring Framework*.

- **errors.log**

  Contains only messages that have at least the log level *ERROR*.

- **esb.log**

  Contains messages of the *Mule Framework* (Mule is the internal ESB that is used for the processing
  of incoming e-mails).

- **index.log**

  Messages concerning the Indexer.

- **mail.log**

  Contains messages of the e-mail subsystem.

- **operationtimes.log**

  Only used when it has been enabled. Contains times of requests in order to identify possible
  performance bottlenecks.

- **server.log**

  The general log file that contains all messages, as default setting at least with 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 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.

- **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.

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):

```
2012-11-06 14:22:12,685 INFO [e.coyote.http11.Http11Protocol] Starting Coyote HTTP/1.1 on http-
0.0.0.0-8080
```

The components of the message:

- **Date:**
  November 6th, 2012
- **Timestamp:**
  14:22:12
- **Loglevel:**
  INFO
- **Logger:**
  e.coyote.http11.Http11Protocol
  Name of a Java class, not complete (only last 30 characters), the real name would be *org.apache. coyote.http11.HttpProtocol*.
- **Message:**
  Starting Coyote HTTP/1.1 on http-0.0.0.0-8080

Simple messages and messages that concern a successful operation often comprise only one line.

When errors occur (log level *ERROR*), you might find stack traces. Please approach one of our ConSol*CM consultants or our ConSol*CM support team for help.

Indexer

In order to perform effective searches in the database, CM builds an index for each Custom Field, Data Object Group Field, and Resource Group Field which should be included in a search. Furthermore, the engineer data, the ticket comments and the attachments are indexed per default. The indexes are stored on the file system! Please refer to the section about the data directory for an explanation of the index directory structure and read the detailed introduction to the entire topic in the section Search Configuration and Indexer Management.

47.2.3 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 MS 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.
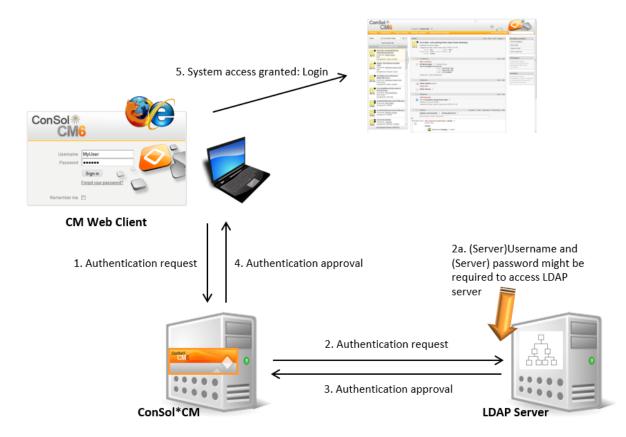
Fig. 8: ConSol*CM - LDAP Authenication (Web Client)

47.3 Infrastructure of a CM System with DWH

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 the 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).
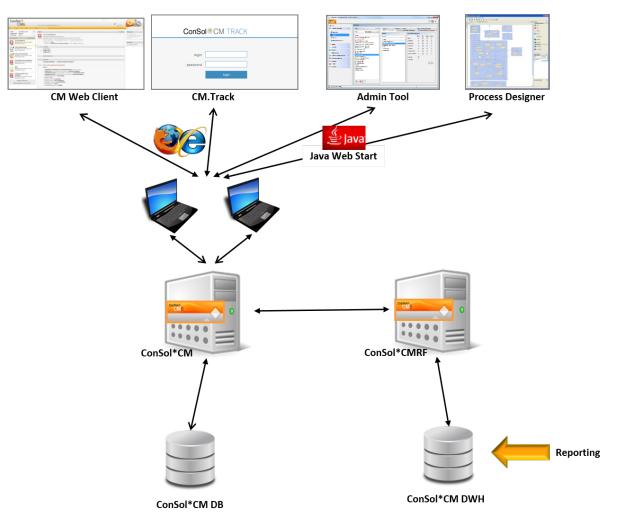
Thus, the ConSol*CM standard function set comprises two components which enable reporting:

- **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 or it can be run on a separate application server. We recommend to use two application servers in case you work with JBoss systems and one application server in case you use Oracle Web Logic. The synchronization of CM data with the DWH can be based on JMS (Java Messaging Service) queues or on direct messaging. For a detailed explanation, please refer to the *ConSol*CM Operations Manual*
  .
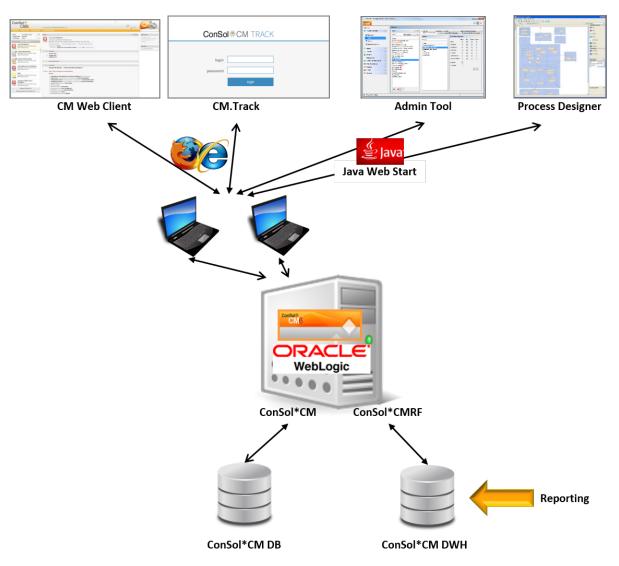
- **DWH** (data warehouse)
  The ConSol*CM DWH is a relational database which can be operated as Oracle, MSSQL Server or MySQL system (please see *ConSol*CM System Requirements* for details). It stores the integrated /pre-processed data from the ConSol*CM database.

Separate application servers for ConSol*CM and CMRF (*standalone mode*):
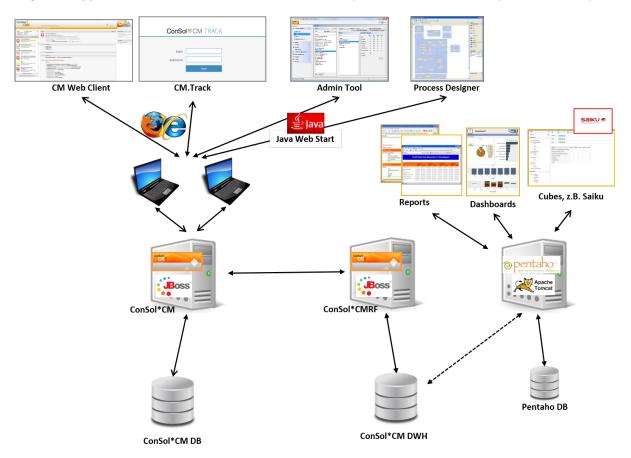
Fig. 1: ConSol*CM - Infrastructure with CMRF and DWH (2 Servers)

One application server for ConSol*CM and CMRF (*overlay mode*, example Oracle WebLogic):
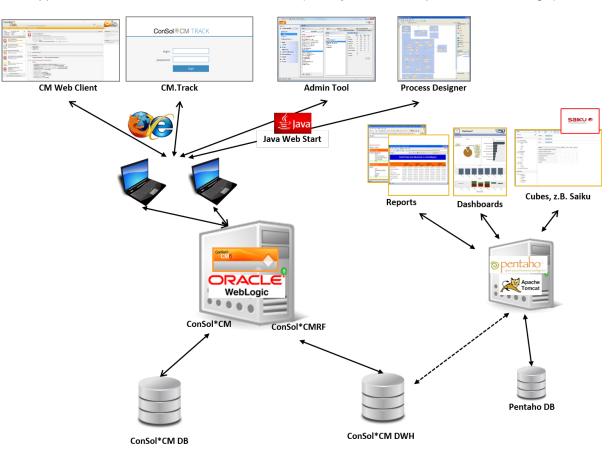
Fig. 2: ConSol*CM - Infrastructure with CMRF and DWH (1 Server)

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 Pentaho[TM] BI Suite.

Separate application servers for ConSol*CM and CMRF (*standalone mode*, example: both JBoss):

Fig. 3: ConSol*CM - Reporting Infrastructure (2 Servers, Both JBoss)

One application server for ConSol*CM and CMRF (*overlay mode*, example: Oracle WebLogic):

Fig. 4: ConSol*CM - Reporting Infrastructure (1 Server, WebLogic)

47.3.1 DWH Database

Oracle

One database scheme with one database user is used by the DWH.

MS 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.

48 Appendix A - List of Annotations (up to Version 6.10.2)

- Alphabetical List of Field Annotations
- Alphabetical List of Group Annotations

48.1 Alphabetical List of Field Annotations

	Name	Annotation Type	Description	Values	Comment
A	accuracy	validation	For date fields, to define the level of detail displayed.	date (default)	Show date without time.
				date-time	Show date with time.
				only-time	Show only time, no date.
B	boolean-type	component-type	Definition of the layout of a boolean field.	check box (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).
C	colspan	layout	Defines how many columns are reserved for the field in the layout.	<number>	Number of columns.
	contact search result column	search-result	Identifies whether the field should be presented in the search result by default.	true	Remove the annotation if the field should not be visible by default. Since CM version 6.1.3.

	Name	**Annotation Type**	**Description**	**Values**	**Comment**
			Deprecated! Do not use!		(Replaced by order-in-result, contact search result column is obsolete!)
	contains contacts	ticket contact relation type	Used only for list field definition, indicates that it can hold unit references to units annotated as contacts.	true / false	Value type is boolean. Necessary to distinguish if the list is shown with the contact (true) or with the ticket (false).
D	dialable	phone commander (CM.Phone)	Defines a field with a phone number.	true	Used with CM. Phone only. Marks a phone number as automatically dialable for outgoing calls for the CTI system.
	dwh-no-history-fi eld		Annotation used to indicate that field will not be historized in DWH	true / false	Since version 6.10.2.0.
E	email	validation	Used for e-mail addresses to check if the format is correct, i.e. if <name>@<domain> has been entered.	true	May be used with *string* Custom Fields. Remove the annotation if the format should not be checked.
	enum field with ticket color	ticket display	Defines the background	true / false	The field has to exist within E

	Name	Annotation Type	Description	Values	Comment
			color of the ticket icon for ticket list and ticket.		*num Administration* where lists, values, and colors are defined.
	enum-in-search-type	component-type	Defines whether an *enum* field used in a search accepts search over multiple values.	single (default) / multiple	Accepts search over multiple values if value *multiple* is set.
	enum-type	component-type	Layout definition of list display.	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 selection where the field is an input field used to filter the list.
F	field-group	layout	Allows grouping of fields in *view* mode. Annotation is ignored in *edit* mode.	<string>	To group fields the same *string* value has to be set in the annotation of each field. Two or more Custom Fields are bound when they share the same value of this

	Name	Annotation Type	Description	Values	Comment
					annotation. The group of coupled Custom Fields is shown only if all of them have values set.
	field indexed	indexing	Used to indicate that a database index will be created for this field.	transitive (default)	All data is displayed (ticket and customer).
				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.
				not indexed	Field is not indexed.
	fieldsize	layout	Displayed field size within ticket layout.	<rows>;<cols>	For *string* Custom Fields with annotation *text-type* and value *textarea*. <rows>

	Name	Annotation Type	Description	Values	Comment
					defines the number of displayed rows and <cols> defines the number of characters displayed per row. Used only for layout purposes.
				<number>	For *enum* Custom Fields. Defines how many values are directly visible in the list box. Used only for layout purposes.
	format	validation	Used to check the correct format of date fields.	<date format>	The pattern for the date is based on *SimpleDateFormat*, e.g. dd.mm.yyyy. Remember to set the proper *colspan* when you want to add hours/minutes. See http://docs.oracle.com/javase/6/docs/api/java/text/SimpleDateFormat.html for date format reference.

	Name	Annotation Type	Description	Values	Comment
G	groupable	cmweb-common	Enables grouping in the ticket list.	true	Used only with *enum* Custom Fields. Remove the annotation if you want to disable grouping.
L	label-group	layout	Indicates a group of fields along with its descriptive label in *view* mode. Annotation is ignored in *edit* mode.	<string>	Indicates a group of Custom Fields along with its descriptive label. The annotation is used in *presentation* mode, ignored in *edit* mode. The group can have exactly one label (a Custom Field of type *string* with assigned additional annotation *text-type* with value *label*). The label is shown when at least one Custom Field from its group has a value set. All fields with the same label value are grouped and displayed under this label.

	Name	Annotation Type	Description	Values	Comment
					The annotation *label-group* has to be assigned to the label, too.
	label-in-view	layout	Shows Custom Field value as a label in *view* mode. Annotation is ignored in *edit* mode.	true	Remove the annotation if the label should not be visible in *view mode*.
	ldapid	contact authentication	Used in a Data Object Group of type *customer*, for the Data Object Group Field which contains the LDAP ID for CM.Track authentication.		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 on 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.
	leave-trailing-zeros	common		true / false	Remaining zeros of the

	Name	Annotation Type	Description	Values	Comment
			Used for the display of fixed point numbers.		fractional part are not cut off when value is *true*.
	list-type	component-type	Disables the *add* and/or *delete* options for Custom Fields of type *list* or *struct*.	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.
M	matches	validation	Checks if input of *string* Custom Fields matches the given RegEx.	<string>	May be used with *string* Custom Fields.
	maxLength	validation	Defines the maximum length of input for *string* Custom Fields.	<number>	May be used with *string* Custom Fields.
	maxValue	validation	Defines the maximum value for *number* Custom Fields.	<number>	May be used with *number* Custom Fields, i.e. *number* and *fixed-point number*.
	minLength	validation	Defines the minimum length of input for *string* Custom Fields.	<number>	May be used with *string* Custom Fields.

	Name	Annotation Type	Description	Values	Comment
	minValue	validation	Defines the minimum value for *number* Custom Fields.	<number>	May be used with *number* Custom Fields, i.e. *number* and *fixed-point number*.
N	no-history-field	performance	Indicates that a single Custom Field should not be historized. Overwrites the group annotation *no-history*.	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.
O	order-in-result	layout	Shows field as a column in the search result list at given position.	<number>	The columns are sorted in ascending order. Since CM version 6.0.1
P	password	contact authentication	Indicates that this field will be used as a	<string>	Used for CM.Track.

	Name	Annotation Type	Description	Values	Comment
			password in the authentication process.		
	position	layout	Defines the position of a field within a grid layout.	<number>; <number>	Values define *row* and *column* (row;column), numbering starts at 0;0. If no values are set, the Custom Field will take the next free grid cell.
			Defines the position of a field within a list (struct).	0;<number>	Only the *column* value is used, the *row* value is ignored.
R	readonly	common	Used to indicate that the Custom Field cannot be modified.	true / false	Field is read only if value is set to *true*. Lack of value or any value except *false* is also treated as *true*.
	reportable	dwh	Indicates that the field is reportable and that it should be transferred to the DWH.	true / false	Field is reportable if value is set to *true*.
	required	validation	Indicates that this is a required field.	true / false	Field is required if value is set to *true*. The user cannot save the ticket

	Name	Annotation Type	Description	Values	Comment
					without having entered a value in a required field. In the Web Client, required fields are marked by a red asterisk.
	rowspan	layout	Indicates how many rows within the layout are occupied by this field.	\<number\>	Number of rows.
S	sortable	cmweb-common	Used to enable sorting of the ticket list.	true	Used for Custom 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.
	show-label-in-edit	layout	Whether Custom Field or Data Object Group Field should be displayed in *edit* mode with label.	true / false	Since version 6.9.4
	show-label-in-view	layout	Whether Custom Field or Data Object	true / false	Since version 6.9.4

	Name	Annotation Type	Description	Values	Comment
			Group Field should be displayed in *view* mode with label		
	show-tooltip	layout	Whether Custom Field or Data Object Group Field should be displayed with tooltip.	true / false	Since version 6.9.4
	show-watermark	layout	Whether Custom Field or Data Object Group Field should be displayed with watermark.	true / false	Since version 6.9.4
T	text-type	component-type	Defines the possible types of a *string* field.	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.
				url	

	Name	Annotation Type	Description	Values	Comment
					Input will be displayed as a hyperlink in *view* mode. String has to match a specific URL pattern: "^((?:mailto\:|(?:(?:ht|f)tps?)\://)1\S+)(?: (?:\\|)?(.*))?$" Example: "http://consol.de ConSol*"
	ticket-list-colspan	layout	Defines how many columns are occupied by the field in the ticket list box.	\<number>	Number of columns.
	ticket-list-position	layout	Defines the position of the field in the ticket list box.	\<number>; \<number>	Values define *row* and *column* (row;column), numbering starts at 0;0.
	ticket-list-rowspan	layout	Defines how many rows are occupied by the field in the ticket list box.	\<number>	Number of rows.
U	username	contact authentification	Indicates that this field will be used as a login name in the authentication process.	true / false	Used for CM. Track.
V	visibility	common		edit	

	Name	Annotation Type	Description	Values	Comment
			Defines when the field is visible.		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 the field will always be visible.
	visibility configuration	visibility	Indicates the visibility of this field in history.	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.

48.2 Alphabetical List of Group Annotations

	Name	Annotation Type	Description	Values	Comment
A	auto-open-group	layout	The group will be opened initially. More than one value can be entered as comma- or semicolon-separated list (can be used for the customer annotation).	ticket:create	Group is opened initially when a new ticket is created.
				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.
D	dwh-no-history		Annotation used to indicate that all fields in group will not be historized in DWH	true / false	Since version 6.10.2.0
G	group-visibility	common	Defines the default visibility of a Custom Field Group.	true / false	The annotation can be overwritten on field level.
N	no-history	performance		true / false	

	Name	Annotation Type	Description	Values	Comment
			Indicates that all Custom Fields belonging to this group will not be historized.		Used to indicate that all Custom Fields that belong to this group should not be historized. Possible values are *true* if this annotation should be active or *false* which is the same like removing the annotation. Use this annotation if you want to prevent 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* on 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

	Name	Annotation Type	Description	Values	Comment
					CM version 6.10.2, use the annotation *dwh -no-history* for this.
R	reportable group	dwh	Indicates that all Custom Fields belonging to this group are reportable and should be transferred to CMRF.	true / false	A value has to be set. Annotation is active if value is set to *true*.
S	show-contact-in-ticket-list		Obsolete! Use page customization! accordionTicke tList. mainCustomer DescriptionVisi ble={true, false}	obsolete	
	show-in-group-section	layout	Defines that a Custom Field Group is displayed in the *Groups* section (as tab).	true / false	Without this annotation the group is shown in the non-tabbed ticket data or contact section.
	show-labels-in-edit	layout	Whether Custom Fields or Data Object Group Fields in this group should be displayed in *edit* mode with labels.	true / false	Since version 6.9.4

	Name	**Annotation Type**	**Description**	**Values**	**Comment**
	show-labels-in-view	layout	Whether Custom Fields or Data Object Group Fields in this group should be displayed in *view* mode with labels.	true / false	Since version 6.9.4
	show-tooltips	layout	Whether Custom Fields or Data Object Group Fields in this group should be displayed with tooltips.	true / false	Since version 6.9.4
	show-watermarks	layout	Whether Custom Fields or Data Object Group Fields in this group should be displayed with watermarks.	true / false	Since version 6.9.4
U	unit is a contact *deprecated*	ticket contact relation		true/false	Removed in version 6.9.0.

49 Appendix B - Glossary

	Term	Explanation
A	Access Permissions	Permissions of an engineer to view or make changes to tickets in the Web Client. Access permissions are always assigned to a group, never to single engineers/users.
	ACIM	Activity item - entry in the history section of a ticket (e.g. comment, e-mail, attachment, time booking entry).
	Action Framework	ConSol*CM module which enables CM to perform actions for specific objects. The Action Framework comprises Search Actions, Unit Actions and Resource Actions.
	AD	Microsoft Active Directory - an LDAP-based directory service for Microsoft Windows domain networks.
	Additional customer	Customer (contact or company) besides the main customer, e.g. an employee of the company. For additional customers, customer roles can be assigned.
	Admin Tool	ConSol*CM component, graphical application to configure and manage a ConSol*CM system. Uses Java Web Start.
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

	Term	Explanation
		methods of the ConSol*CM API to access data in Custom Fields and Data Object Group Fields.
	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.
	CM.Doc	A standard module of ConSol*CM which enables the engineer via ConSol*CM Web Client to work with MS Word documents pre-filled with ConSol*CM ticket or customer parameters.
	CM.Track	ConSol*CM web portal - provides customer access to the ConSol*CM system.
	CM.Resource Pool	A ConSol*CM module which is available in CM versions 6.10.1 and up. Enables CM to store objects of several types in the CM database as resources.
	Company	A data object of type *company*. Often this is a real company or an institution, but it can also be something else, like a machine or a ship.
	Contact	A data object of type *contact*. Often this is the person who has a question or service request, but it can also be something else, like a machine or a product.
	CTI	Computer Telephony Integration - a description for any

	Term	Explanation
		technology that allows interactions on a telephone and a computer to be integrated or coordinated.
	Customer	General term for customer objects in ConSol*CM. A customer can be a contact or a company. Technically, a customer is a data object. The respective Java class is *Unit*.
	Custom Field	A field where ticket data (e.g. priority, software module, etc.) can be stored.
	Custom Field Group	A group of Custom Fields where ticket data can be stored.
D	Data object	A customer (a contact or a company). Former *Unit*.
	Data Object Group	A group of fields where data for customers (contact or company) can be stored. Similar to Custom Field Group for ticket data.
	Data Object Group Field	A field where data for customers (contact or company) can be stored. Similar to Custom Field for ticket data.
	DWH	Data Warehouse - ConSol*CM database used for reporting and data analysis.
E	Engineer	User who has a login to the Web Client and who has to manage the tasks defined in the tickets.
	ESB	Enterprise Service Bus - a software architecture used for communication between mutually interacting software applications in a service-oriented architecture (SOA).
	ERP system	

	Term	Explanation
		Enterprise Resource Planning - often used for this type of enterprise management software.
	ETL	Extract Transform Load - extracts data from one source (this can be a database or another source), transforms it, and loads it into a database, e.g. a data warehouse.
F	FlexCDM	Flexible Customer Data Model - the customer data model which has been 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
I	IMAP	Internet Message Access Protocol - Internet standard protocol to access e-mail on a remote e-mail server. Can be used as plain IMAP or as secure IMAP (IMAPs). In the latter case the 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	KPI	Key Performance Indicator - parameter used for performance measurement for companies, projects etc.
L	LDAP	

	Term	Explanation
		Lightweight Directory Access Protocol - application protocol to access and maintain directory information over an IP network.
	LDAPS	LDAP over SSL
M	Mailbox	Destination to which e-mail messages are delivered. Mailboxes are managed on an e-mail server. ConSol*CM can access one or more separate mailboxes to retrieve e-mails.
	Main customer	The customer who is the main customer of a ticket. Starting with ConSol*CM version 6.9, this can be either a contact or a company.
	Mule	An open source Java-based Enterprise Service Bus (ESB).
N	NIMH	New Incoming Mail Handler - module for retrieving incoming e-mails, new in version 6.9.4.
P	PCDS	Page Customization Definition Section
	Pentaho	Pentaho^{TM} is a business intelligence (BI) suite which is available as open source version and as enterprise edition.
	POP	Post Office Protocol - Internet standard protocol to retrieve e-mails from a remote server via TCP/IP. Can be used as plain POP or as secure POP (POPs). In the latter case the proper certificates are required.
	Portal	CM.Track - provides customer access to ConSol*CM.
	Process Designer	

Term	Explanation
	ConSol*CM component used to design, develop, and deploy workflows.
Q Queue	Comprises tickets from the same domain and makes sure that all tickets of this domain are treated in the same way. A queue always has one workflow. Access rights and other parameters are defined based on queues.
R RDBMS	Relational Database Management System - e.g. Oracle®, MS SQL Server®, MySQL.
Resource	An object in the Resource Pool
Resource Action	Part of the Action Framework. Action which is based on a resource object.
Resource Field	A field where resource data (e.g. inventory number, model, etc.) can be stored.
Resource Field Group	A group of fields where data for resources can be stored. Similar to Custom Field Group for ticket data.
REST	Representational State Transfer - a method to transfer data via a network, based on HTTP.
Role	Defines the access permissions and views of an engineer.
S Script	Program written for a special run-time environment that can interpret and automate the execution of tasks. In ConSol*CM, scripts are stored in

Term	Explanation

	Term	Explanation
		the Admin Tool and are stored as scripts for activities in workflows.
	Search Action	Part of the Action Framework. Action which is based on the result set of a search.
	SMTP	Simple Message Transfer Protocol - standard protocol to send e-mails.
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	Pre-formatted example concerning layout, text, and/or data, e.g. for e-mails or CM.Doc.
	Ticket	Incident, service case, or other request of an internal or external customer. A ticket is the object which runs through the process (defined by the workflow).
U	Unit Action	Part of the Action Framework. Action which is based on a customer object, i.e. on a contact or company object.
V	View	A selection of tickets based on scopes from one or from different workflows, assigned to a role and visible in the ticket list of the ConSol*CM Web Client.
W	Workflow	

	Term	Explanation
		Models a process that should be managed using ConSol*CM step by step.

50 Appendix C - System Properties

The lists provide explanation for all available ConSol*CM system properties. You can define system properties in the Admin Tool, in the navigation group *System*, navigation item System Properties.

- System Properties Ordered by Module
- System Properties Ordered by Property Name

50.1 System Properties Ordered by Module

Module	Property	Explanation
cmas-app-admin-tool	admin.tool.session.check. interval	*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
cmas-app-admin-tool	autocomplete.enabled	*Description:* If the flag is missing or its value is *false*, then the *Auto complete 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
cmas-app-admin-tool	delete.ticket.enabled	*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
cmas-app-admin-tool	start.groovy.task.enabled	*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

Module	Property	Explanation
		property to true *Type:* Boolean *Restart required:* No *System:* No *Optional:* Yes *Example value:* true *Since:* 6.9.4.0
cmas-core-cache	cache-cluster-name	*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
cmas-core-cache	eviction.event.queue.size	*Description:* *Type:* Integer *Restart required:* Yes *System:* Yes *Optional:* No *Example value:* 200000 *Since:* 6.4.0
cmas-core-cache	eviction.max.nodes	*Description:* *Type:* Integer *Restart required:* Yes *System:* Yes *Optional:* No *Example value:* 100000 *Since:* 6.4.0
cmas-core-cache	eviction.wakeup.interval	*Description:* *Type:* Integer *Restart required:* Yes *System:* Yes *Optional:* No *Example value:* 3000 *Since:* 6.4.0
cmas-core-index-common	big.task.minimum.size	*Description:* How many parts task at least should have to be handled by Indexer with low priority.

Module	Property	Explanation
		Type: Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 15 (default) *Since:* 6.8.3
cmas-core-index-common	database.notification.enabled	*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
cmas-core-index-common	database.notification.redelivery.delay.seconds	*Description:* In case of index update database notification channel, indicates notification redelivery delay when exception occurs. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 60 *Since:* 6.8.4.7
cmas-core-index-common	database.notification.redelivery.max.attempts	*Description:* In case of index update database notification channel, indicates maximum redelivery attempts when exception occurs. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 60 *Since:* 6.8.4.7
cmweb-server-adapter	diffTrackingEnabled	*Description:* Defines if parallel editing of a ticket by different engineers should be possible.

Module	Property	Explanation
		Default *true* *Type:* Boolean *Restart required:* No *System:* *Optional:* *Example value:* true *Since:* 6.10.1
cmas-core-index-common	disable.admin.task.auto.commit	*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
cmas-core-index-common	index.attachment	*Description:* Describes if content of attachments is indexed. *Type:* Boolean *Restart required:* No *System:* Yes *Optional:* No *Example value:* true *Since:* 6.4.3
cmas-core-index-common	index.history	*Description:* Describes if unit and ticket history are indexed. *Type:* Boolean *Restart required:* No *System:* Yes *Optional:* No *Example value:* false *Since:* 6.1.0
cmas-core-index-common	index.status	*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

Module	Property	Explanation
		Example value: GREEN *Since:* 6.6.1
cmas-core-index-common	index.task.worker.threads	*Description:* How many threads will be used to execute batch 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
cmas-core-index-common	index.version.current	*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
cmas-core-index-common	index.version.newest	*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
cmas-core-index-common	indexed.assets.per.thread.in. memory	*Description:* How many assets should be loaded into memory at once during indexing per one thread. *Type:* Integer *Restart required:* No *System:* Yes

Module	Property	Explanation
		Optional: No *Example value:* 200 (default) *Since:* 6.8.0
cmas-core-index-common	indexed.engineers.per.thread.in. memory	*Description:* How many engineers should be loaded into memory at once during indexing per one thread. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 300 (default) *Since:* 6.6.14, 6.7.3
cmas-core-index-common	indexed.resources.per.thread.in. memory	*Description:* How many resources should be loaded into memory at once during indexing per one thread. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 200 (default) *Since:* 6.10.0.0
cmas-core-index-common	indexed.tickets.per.thread.in. memory	*Description:* How many tickets should be loaded into memory at once during indexing per one thread. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 100 (default) *Since:* 6.6.14, 6.7.3
cmas-core-index-common	indexed.units.per.thread.in. memory	*Description:* How many units should be loaded into memory at once during indexing per one thread. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No

Module	Property	Explanation
		Example value: 200 (default) *Since:* 6.6.14, 6.7.3
cmas-core-index-common	synchronize.master.address	*Description:* Value of *-Dcmas.http.host.port* informing how to connect to indexing master server. Default null. Since 6.6.17 this value is configurable in set-up to designate initial indexing master server. Please note that changing this value is only allowed when all cluster nodes index changes receivers are stopped. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* 127.0.0.1:80 *Since:* 6.6.0
cmas-core-index-common	synchronize.master.security.token	*Description:* The password for accessing the index snapshot via URL, e.g. for index synchronization or for back-ups. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* token *Since:* 6.6.0
cmas-core-index-common	synchronize.master.security.user	*Description:* The user name for accessing the index snapshot via URL, e.g. for index synchronization or for back-ups. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* user *Since:* 6.6.0
cmas-core-index-common	synchronize.master.timeout.minutes	*Description:* How much time master server may constantly fail

Module	Property	Explanation
		until new master gets elected with index fix procedure. Default 5. Since 6.6.17 this value is configurable in set-up where zero means that master server will never change (failover mechanism is off). *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 5 *Since:* 6.6.0
cmas-core-index-common	synchronize.megabits.per. second	*Description:* How much bandwidth can master server consume to transfer index changes to all slave servers. Default 85. Please do not use all available bandwidth to transfer index changes between hosts. This will most probably partition cluster as some subsystems will not be able to communicate. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 85 *Since:* 6.6.0
cmas-core-index-common	synchronize.sleep.millis	*Description:* How often each slave server polls master server for index changes. Default 1000. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 1000 *Since:* 6.6.0
cmas-core-security	admin.email	*Description:* The e-mail address of the ConSol*CM administrator. The value which you have entered during system set-up is

Module	Property	Explanation
		used initially. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* myuser@consol. de *Since:* 6.0
cmas-core-security	admin.login	*Description:* The name of the ConSol*CM administrator. The value which you have entered during system set-up is used initially. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* admin *Since:* 6.0
cmas-core-security	authentication.method	*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
cmas-core-security	contact.authentication.method	*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
cmas-core-security	contact.inherit.permissions.only. to.own.customer.group	*Description:* Indicates whether authenticated contact inherits all

Module	Property	Explanation
		customer group permissions from representing engineer (false) or only permission to own customer group (true). *Type:* Boolean *Restart required:* No *System:* Yes *Optional:* No *Since:* 6.9.2.3
cmas-core-security	kerberos.v5.enabled	*Description:* Flag which indicates whether SSO via Kerberos is enabled. *Type:* Boolean *Restart required:* No *System:* Yes *Optional:* No *Example value:* false (default if Kerberos has not been enabled during system set-up) *Since:* 6.2.0
cmas-core-security	kerberos.v5.username.regex	*Description:* Regular expression used for mapping Kerberos principal to CM user login. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* (.*)@.* *Since:* 6.2.0
cmas-core-security	ldap.authentication	*Description:* Authentication method used when using LDAP authentication. *Type:* String *Restart required:* Yes *System:* Yes *Optional:* No *Example value:* simple *Since:* 6.0
cmas-core-security	ldap.basedn	*Description:* Base DN used for looking up LDAP user accounts when using LDAP authentication.

Module	Property	Explanation
		Type: String *Restart required:* No *System:* Yes *Optional:* No *Example value:* ou=accounts, dc=consol,dc=de *Since:* 6.0
cmas-core-security	ldap.contact.name.basedn	*Description:* Base path to search for contact DN by LDAP ID (e.g. ou=accounts,dc=consol,dc=de). *Type:* String *Restart required:* No *System:* No *Optional:* Yes *Since:* 6.9.3.0
cmas-core-security	ldap.contact.name.password	*Description:* Password to look up contact DN by LDAP ID. If not set, anonymous account is used. *Type:* String *Restart required:* No *System:* No *Optional:* Yes *Since:* 6.9.3.0
cmas-core-security	ldap.contact.name.providerurl	*Description:* Address of the LDAP server (ldap[s]://host:port). *Type:* String *Restart required:* No *System:* No *Optional:* Yes *Since:* 6.9.3.0
cmas-core-security	ldap.contact.name.searchattr	*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
cmas-core-security	ldap.contact.name.userdn	*Description:* User DN to look up contact DN by LDAP ID. If not set, anonymous account is used.

Module	Property	Explanation
		Type: String *Restart required:* No *System:* No *Optional:* Yes *Since:* 6.9.3.0
cmas-core-security	ldap.initialcontextfactory	*Description:* Class name for initial context factory of LDAP implementation when using LDAP authentication. If it is not set, com.sun.jndi.ldap. LdapCtxFactory is being used as a value. *Type:* String *Restart required:* Yes *System:* Yes *Optional:* No *Example value:* com.sun.jndi. ldap.LdapCtxFactory *Since:* 6.0
cmas-core-security	ldap.password	*Description:* Password for connecting to LDAP to look up users (when using LDAP authentication). Only needed if look-up cannot be done anonymously. *Type:* Password *Restart required:* No *System:* Yes *Optional:* Yes *Since:* 6.1.2
cmas-core-security	ldap.providerurl	*Description:* LDAP provider (when using LDAP authentication). *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* ldap://ldap. consol.de:389 *Since:* 6.0
cmas-core-security	ldap.searchattr	

Module	Property	Explanation
		Description: Search attribute for looking up LDAP entry connected to CM login. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* uid *Since:* 6.0
cmas-core-security	ldap.userdn	*Description:* LDAP user for connecting to LDAP to look up users (when using LDAP authentication). Only needed if look-up cannot be done anonymously. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Since:* 6.1.2
cmas-core-security	policy.password.age	*Description:* Defines (in days) how old the password may be. *Type:* Integer *Restart required:* No *System:* No *Optional:* Yes *Example value:* 5500 (15 years, default) *Since:* 6.10.1.0
cmas-core-security	policy.password.pattern	*Description:* Defines password pattern *Type:* String *Restart required:* No *System:* No *Optional:* Yes *Example value:* "^. 3,$" (default) *Since:* 6.10.1.0
cmas-core-security	policy.rotation.ratio	*Description:* Defines how often password may repeat. E.g. in set to X means that new password

Module	Property	Explanation
		cannot be present among X last user's passwords. *Type:* Integer *Restart required:* No *System:* No *Optional:* Yes *Example value:* 1 (default) *Since:* 6.10.1.0
cmas-core-security	policy.username.case.sensitive	*Description:* Defines whether user name is case-sensitive *Type:* Boolean *Restart required:* No *System:* No *Optional:* Yes *Example value:* true (default) *Since:* 6.10.1.0
cmas-core-server	attachment.allowed.types	*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
cmas-core-server	attachment.max.size	*Description:* Maximum attachment size in MB *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 100 *Since:* 6.4.0
cmas-core-server	config.data.version	*Description:* *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 11 *Since:* 6.0

Module	Property	Explanation
cmas-core-server	defaultCommentClassName	*Description:* Default text class name for comments *Type:* String *Restart required:* No *System:* No *Optional:* Yes *Example value:* *Since:* 6.3.0
cmas-core-server	defaultIncommingMailClassName	*Description:* Default text class name for incoming e-mails *Type:* String *Restart required:* No *System:* No *Optional:* Yes *Example value:* *Since:* 6.3.0
cmas-core-server	defaultOutgoingMailClassName	*Description:* Default text class name for outgoing e-mails *Type:* String *Restart required:* No *System:* No *Optional:* Yes *Example value:* *Since:* 6.3.0
cmas-core-server	fetchSize.strategy	*Description:* Strategy selected to set 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
cmas-core-server	fetchSize.strategy. FetchSizeFixedStrategy.value	*Description:* Sets fetch size value if selected strategy to set fetch size is *FetchSizeFixedStrategy*. *Type:* Integer *Restart required:* No

Module	Property	Explanation
		System: Yes *Optional:* Yes *Example value:* 150 *Since:* 6.8.4.1
cmas-core-server	fetchSize.strategy. FetchSizePageBasedStrategy. limit	*Description:* Sets maximum fetch size value if selected strategy to set fetch size is *FetchSizePageBasedStrategy*. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* 10000 *Since:* 6.8.4.1
cmas-core-server	fetchSize.strategy. FetchSizeThresholdStrategy. value	*Description:* Sets fetch size threshold border values if selected strategy to set fetch size is *FetchSizeThresholdStrategy*. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* 150,300,600,1000 *Since:* 6.8.4.1
cmas-core-server	last.config.change	*Description:* Random UUID created during last change in config. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* 2573c7b7-2bf5-47ff-b5a2-bad31951a266 *Since:* 6.1.0, 6.2.1
cmas-core-server	ldap.certificate.basedn	*Description:* Base DN for certificates location in LDAP tree. If not provided, *cmas-core-security*, *ldap.basedn* is taken. *Type:* String

Module	Property	Explanation
		Restart required: No *System:* Yes *Optional:* Yes *Example value:* ou=accounts, dc=consol,dc=de *Since:* 6.8.4
cmas-core-server	ldap.certificate.content.attribute	*Description:* LDAP attribute name used where certificate data is stored in LDAP tree. Default value is: usercertificate. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* usercertificate *Since:* 6.8.4
cmas-core-server	ldap.certificate.password	*Description:* LDAP Certificates manager password. If not set, *cmas-core-security*, *ldap.password* is taken. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Since:* 6.8.4
cmas-core-server	ldap.certificate.providerurl	*Description:* LDAP Certificates provider URL. If not set, *cmas-core-security*, *ldap.providerurl* is taken. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* ldap://ldap. consol.de:389 *Since:* 6.8.4
cmas-core-server	ldap.certificate.searchattr	*Description:* LDAP attribute name used to search for certificate in LDAP tree. Default value is: mail. *Type:* String

Module	Property	Explanation
		Restart required: No *System:* Yes *Optional:* Yes *Example value:* mail *Since:* 6.8.4
cmas-core-server	ldap.certificate.userdn	*Description:* LDAP Certificates manager DN. If not set, *cmas-core-security*, *ldap.userdn* is taken. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Since:* 6.8.4
cmas-core-server	mail.encryption	*Description:* If property is set to *true*, the encrypt check box in the Ticket E-Mail Editor is checked by default. *Type:* Boolean *Restart required:* No *System:* Yes *Optional:* No *Example value:* true (default = false) *Since:* 6.8.4.0
cmas-core-server	mail.notification.engineerChange	*Description:* Flag if notification e-mail should be sent when engineer of ticket is changed. *Type:* Boolean *Restart required:* No *System:* Yes *Optional:* No *Example value:* true *Since:* 6.1.0
cmas-core-server	mail.notification.sender	*Description: From* address for notification e-mails when engineer of ticket is changed. If not set, *cmas-core-security*, *admin.email* is used instead. *Type:* String *Restart required:* No

Module	Property	Explanation
		System: Yes *Optional:* Yes *Example value:* cm6notification@cm6installation *Since:* 6.6.3
cmas-core-server	mail.smtp.email	*Description:* SMTP e-mail URL for outgoing e-mails *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* smtp://mail.consol.de:25 *Since:* 6.0
cmas-core-server	mail.smtp.envelopesender	*Description:* E-mail address used as sender in SMTP envelope. If not set, the FROM-address of the e-mail is used. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* mysender@mydomain.com *Since:* 6.5.7
cmas-core-server	max.licences.perUser	*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
cmas-core-server	monitoring.engineer.login	*Description:* Login of monitoring engineer. *Type:* String *Restart required:* No *System:* Yes

Module	Property	Explanation

Module	Property	Explanation
		Optional: Yes *Example value:* nagios *Since:* 6.9.3.0
cmas-core-server	monitoring.unit.login	*Description:* Login of monitoring unit. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* nagios *Since:* 6.9.3.0
cmas-core-server	nimh.enabled	*Description:* Enables NIMH service. Must be suffixed with nodeid in cluster e.g. nimh.enabled.NODEID = true *Type:* Boolean *Restart required:* No *System:* No *Optional:* Yes *Example value:* false *Since:* 6.9.4.0
cmas-core-server	resource.replace.batchSize	*Description:* Describes number of objects to be processed in resource replace action. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 5 *Since:* 6.10.0.0
cmas-core-server	resource.replace.timeout	*Description:* Transaction timeout (seconds) of resource replacement action step. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 120 *Since:* 6.10.0.0
cmas-core-server	script.logging.threshold.seconds	*Description:* Describes time in seconds which when exceeded

Module	Property	Explanation
		during script execution, generates warn in logs. *Type:* Integer *Restart required:* No *System:* No *Optional:* Yes *Example value:* 10 (default) *Since:* 6.10.1.0
cmas-core-server	server.session.archive.reaper.interval	*Description:* Server archived sessions' reaper interval (in seconds) *Type:* Integer *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* 60 *Since:* 6.7.1
cmas-core-server	server.session.archive.timeout	*Description:* Server sessions archive validity timeout (in days). After this time session info is removed from db. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 31 *Since:* 6.7.1
cmas-core-server	server.session.reaper.interval	*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
cmas-core-server	server.session.timeout	*Description:* Server session timeout (in seconds) for connected clients. Each client can overwrite this timeout with

Module	Property	Explanation
		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 *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 1800 *Since:* 6.6.1, 6.7.1
cmas-core-server	skip.wfl.transfer.cleanup	*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
cmas-core-server	tickets.delete.size	*Description:* Property that 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
cmas-core-server	ticket.delete.timeout	*Description:* Transaction timeout (in seconds) for deleting tickets. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 60 *Since:* 6.1.3

Module	Property	Explanation
cmas-core-server	unit.replace.batchSize	*Description:* Describes number of objects to be processed in unit replace action. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 5 *Since:* 6.8.2
cmas-core-server	unit.replace.timeout	*Description:* Transaction timeout (seconds) of unit replacement action step. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 120 *Since:* 6.8.2
cmas-core-server	unused.content.remover.cluster.node.id	*Description:* Value of a *cmas.clusternode.id* designating node which 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 -Dcmas.clusternode.id=1 parameter) *Since:* 6.9.0.0
cmas-core-server	unused.content.remover.enabled	*Description:* Flag whether unused ticket attachments and unit content entries removal should take place. *Type:* Boolean *Restart required:* No *System:* Yes *Optional:* No *Example value:* true *Since:* 6.9.0.0

Module	Property	Explanation
cmas-core-server	unused.content.remover.polling. minutes	*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
cmas-core-server	unused.content.remover.ttl. minutes	*Description:* Minimum interval 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
cmas-core-server	warmup.executor.enabled	*Description:* Flag whether server should asynchronously warm up during startup (e.g. fill some of internal caches) *Type:* Boolean *Restart required:* No *System:* Yes *Optional:* No *Example value:* true *Since:* 6.9.4.2
cmas-core-shared	cluster.mode	*Description:* Flag if CMAS is running in cluster. *Type:* Boolean *Restart required:* Yes *System:* Yes *Optional:* No *Example value:* false *Since:* 6.1.0
cmas-core-shared	data.directory	*Description:* Directory for CMAS data (e.g. index) *Type:* String

Module	Property	Explanation
		Restart required: No *System:* Yes *Optional:* No *Example value:* C: \Users\user\cmas *Since:* 6.0
cmas-dwh-server	autocommit.cf.changes	*Description:* *Type:* Boolean *Restart required:* No *System:* Yes *Optional:* No *Example value:* false *Since:* 6.7.0
cmas-dwh-server	batch-commit-interval	*Description:* Number of objects in a JMS message. Higher value means better transfer performance and bigger memory usage. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* 100 *Since:* 6.0.0
cmas-dwh-server	communication.channel	*Description:* Communication channel. Possible values are DIRECT (database communication channel, default value since 6.9.4.1), JMS (default value before 6.9.4.1). Before 6.9.4.1 it has to be manually added. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* DIRECT *Since:* 6.8.5.0
cmas-dwh-server	dwh.mode	*Description:* Current mode of DWH data transfer. Possible values are *OFF*, *ADMIN*, *LIVE*

Module	Property	Explanation
		Type: String *Restart required:* No *System:* Yes *Optional:* No *Example value:* OFF *Since:* 6.0.1
cmas-dwh-server	ignore-queues	*Description:* By adding a comma separated list of queue names it is configured that tickets of these queues are 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
cmas-dwh-server	is.cmrf.alive	*Description:* As a starting point time of sending last message to CMRF should be used. If response from CMRF is not received after value (in seconds) it should create a DWH operation status with error message that CMRF is down. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 1200 *Since:* 6.7.0
cmas-dwh-server	java.naming.factory.initial	*Description:* Factory class for DWH context factory. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* org.jnp. interfaces.NamingContextFactory *Since:* 6.0.1

Module	Property	Explanation
cmas-dwh-server	java.naming.factory.url.pkgs	*Description:* *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* org.jboss. naming:org.jnp.interfaces *Since:* 6.0.1
cmas-dwh-server	java.naming.provider.url	*Description:* URL of naming provider. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* localhost *Since:* 6.0.1
cmas-dwh-server	notification.error.description	*Description:* Text for error e-mails from DWH. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* Error occurred *Since:* 6.0.1
cmas-dwh-server	notification.error.from	*Description:* FROM-address for error e-mails from DWH *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Since:* 6.0.1
cmas-dwh-server	notification.error.subject	*Description:* Subject for error e-mails from DWH *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* Error occurred *Since:* 6.0.1
cmas-dwh-server	notification.error.to	*Description:* TO-address for error e-mails from DWH

Module	Property	Explanation
		Type: String *Restart required:* No *System:* Yes *Optional:* No *Example value:* myuser@consol. de *Since:* 6.0.1
cmas-dwh-server	notification.finished_successfully. description	*Description:* Text for e-mails from DWH when transfer finished successfully. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* Transfer finished successfully *Since:* 6.0.1
cmas-dwh-server	notification.finished_successfully. from	*Description:* FROM-address for e-mails from DWH when transfer finished successfully. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Since:* 6.0.1
cmas-dwh-server	notification.finished_successfully. subject	*Description:* Subject for e-mails from DWH when transfer finished successfully. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* Transfer finished successfully *Since:* 6.0.1
cmas-dwh-server	notification.finished_successfully. to	*Description:* TO-address for e-mails from DWH when transfer finished successfully. *Restart required:* Yes *System:* Yes *Optional:* No

Module	Property	Explanation
		Example value: myuser@consol. de *Since:* 6.0.1
cmas-dwh-server	notification. finished_unsuccessfully. description	*Description:* Text for e-mails from DWH when transfer finished unsuccessfully. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* Transfer finished unsuccessfully *Since:* 6.0.1
cmas-dwh-server	notification. finished_unsuccessfully.from	*Description:* FROM-address for e-mails from DWH when transfer finished unsuccessfully. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Since:* 6.0.1
cmas-dwh-server	notification. finished_unsuccessfully.subject	*Description:* Subject for e-mails from DWH when transfer finished unsuccessfully. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* Transfer finished unsuccessfully *Since:* 6.0.1
cmas-dwh-server	notification. finished_unsuccessfully.to	*Description:* TO-address for e-mails from DWH when transfer finished unsuccessfully. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* myuser@consol. de *Since:* 6.0.1
Module	**Property**	**Explanation**

Module	Property	Explanation
cmas-dwh-server	notification.host	*Description:* E-mail (SMTP) server hostname for sending DWH e-mails. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* mail.consol.de *Since:* 6.1.0
cmas-dwh-server	notification.password	*Description:* Password for sending DWH e-mails (optional). *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Since:* 6.1.0
cmas-dwh-server	notification.port	*Description:* SMTP port for sending DWH e-mails. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* 25 *Since:* 6.1.0
cmas-dwh-server	notification.protocol	*Description:* The protocol used for sending e-mails from DWH. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* pop3\
cmas-dwh-server	notification.username	*Description:* (SMTP) User name for sending DWH e-mails. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* myuser *Since:* 6.1.0
cmas-dwh-server	skip-ticket	*Description:* Tickets are not transferred during transfer

Module	Property	Explanation
		/update. *Type:* Boolean *Restart required:* No *System:* Yes *Optional:* No *Example value:* false *Since:* 6.6.19 *Removed in:* 6.8.1
cmas-dwh-server	skip-ticket-history	*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
cmas-dwh-server	skip-unit	*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
cmas-dwh-server	skip-unit-history	*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
cmas-dwh-server	split.history	*Description:* Changes the SQL that fetches the history for the tickets during DWH transfer not

Module	Property	Explanation
		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
cmas-dwh-server	unit.transfer.order	*Description:* Define in which order Data Object 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
cmas-esb-core	esb.directory	*Description:* Directory used by ESB (Mule) *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* C:\Users\user\cmas\mule *Since:* 6.0
cmas-esb-mail	mail.attachments.validation.info.sender	*Description:* Sets FROM-header of attachments type error notification e-mail. As a default the e-mail address of the administrator which you have entered during system set-up is used. *Type:* String *Restart required:* No *System:* Yes *Optional:* No

Module	Property	Explanation
		Example value: admin@consolcm.com *Since:* 6.7.5
cmas-esb-mail	mail.attachments.validation.info. subject	*Description:* Sets subject of attachments type error notification e-mail. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* E-mail was not processed because its attachments were rejected!!! *Since:* 6.7.5
cmas-esb-mail	mail.callname.pattern	*Description:* Regular expression for subject of incoming e-mails. Available as TICKET_NAME_PATTERN_FORMAT in incoming e-mail scripts. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* .*?Ticket\s+\ ((\S+)\).* *Since:* 6.0
cmas-esb-mail	mail.cluster.node.id	*Description:* Only the node whose mail.cluster.node.id equals cmas.clusternode.id will start the Mule ESB e-mail services. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* unspecified *Since:* 6.6.5
cmas-esb-mail	mail.db.archive	*Description:* If property is set to *true*, incoming e-mails are archived in the database. *Type:* Boolean

Module	Property	Explanation
		Restart required: No *System:* Yes *Optional:* Yes *Example value:* false (default) *Since:* 6.8.5.5
cmas-esb-mail	mail.delete.read	*Description:* Determines whether CM deletes messages fetched via IMAP(S). Setting value to *true* will cause deletion of messages after fetching. Default is to not delete messages fetched via IMAP(S). Note: Messages fetched via POP3(S) will always be deleted. *Type:* Boolean *Restart required:* No *System:* Yes *Optional:* No *Example value:* true *Since:* 6.7.3
cmas-esb-mail	mail.incoming.uri	*Description:* URL for incoming e-mails. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* pop3://cm-incoming-user: password@localhost:10110 *Since:* 6.0 ⊖ This value should not be edited here using the system properties pop-up window, but the mailboxes should be configured using the navigation item E-mail. Using this standard feature all entries are controlled - i.e. for each mailbox which is

Module	Property	Explanation
		added, CM establishes a test connection during mailbox set-up. That way it is not possible to enter wrong values.
cmas-esb-mail	mail.max.restarts	*Description:* Maximum number of e-mail service restarts before giving up *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 3 *Since:* 6.0
cmas-esb-mail	mail.mime.strict	*Description:* If set to *false*, e-mail addresses are not parsed for strict MIME compliance. Default is *true*, which means check for strict MIME compliance. *Type:* Boolean *Restart required:* No *System:* Yes *Optional:* No *Example value:* false *Since:* 6.6.17, 6.7.3
cmas-esb-mail	mail.mule.service	*Description:* FROM-address for e-mails sent by Mule service *Type:* EMail *Restart required:* No *System:* Yes *Optional:* No *Example value:* myuser@consol.de *Since:* 6.0
cmas-esb-mail	mail.polling.interval	*Description:* E-mail polling interval in ms. *Type:* Integer *Restart required:* No *System:* Yes

Module	Property	Explanation
		Optional: No *Example value:* 60000 *Since:* 6.0
cmas-esb-mail	mail.process.error	*Description:* TO-address for error e-mails from Mule. As a default the e-mail address of the administrator which you have entered during system set-up is used. *Type:* EMail *Restart required:* No *System:* Yes *Optional:* No *Example value:* myuser@consol.de *Since:* 6.0
cmas-esb-mail	mail.process.retry.attempts	*Description:* Number of retries when processing e-mail *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 3 *Since:* 6.0.2
cmas-esb-mail	mail.process.timeout	*Description:* E-mail processing timeout in seconds. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 60 *Since:* 6.1.3
cmas-esb-mail	mail.redelivery.retry.count	*Description:* Indicates the number of retries of re-delivering an e-mail from the CM system. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 3 *Since:* 6.1.0

Module	Property	Explanation
cmas-nimh	filesystem.polling.threads. number	*Description:* Number of threads started for db e-mails' queue polling. Default: 1 *Type:* Integer *Restart required:* No *System:* No *Optional:* Yes *Example value:* 10 *Since:* 6.4.0
cmas-nimh	filesystem.polling.threads. shutdown.timeout.seconds	*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
cmas-nimh	filesystem.polling.threads. watchdog.interval.seconds	*Description:* Watchdog thread interval. Default: 30 *Type:* Integer *Restart required:* No *System:* No *Optional:* Yes *Example value:* 60 *Since:* 6.4.0
cmas-nimh	filesystem.task.enabled	*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
cmas-nimh	filesystem.task.interval.seconds	*Description:* Default interval for polling mailboxes. Default: 60 seconds *Type:* Integer

Module	Property	Explanation
		Restart required: No *System:* No *Optional:* Yes *Example value:* 60 *Since:* 6.4.0
cmas-nimh	filesystem.task.polling.folder	*Description:* Polling folder location which will be scanned for e-mails 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
cmas-nimh	filesystem.task.timeout.seconds	*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
cmas-nimh	filesystem.task.transaction. timeout.seconds	*Description:* Default transaction timeout for e-mail 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
cmas-nimh	mailbox.1.connection.host	Host (server) for first configured mailbox. Will overwrite the

Module	Property	Explanation
		default parameter *mailbox. default.connection.host*
cmas-nimh	mailbox.1.connection.password	Password for first configured mailbox. Will overwrite the default parameter *mailbox. default.connection.password*
cmas-nimh	mailbox.1.connection.port	Port for first configured mailbox. Will overwrite the default parameter *mailbox.default. connection.port*
cmas-nimh	mailbox.1.connection.protocol	Protocol (e.g., IMAP or POP3) for first configured mailbox. Will overwrite the default parameter *mailbox.default.connection. protocol*
cmas-nimh	mailbox.1.connection.username	User name for first configured mailbox. Will overwrite the default parameter *mailbox. default.connection.username*
cmas-nimh	mailbox.2.connection.host	Host (server) for second configured mailbox. Will overwrite the default parameter *mailbox.default.connection.host*
cmas-nimh	mailbox.2.connection.password	Password for second configured mailbox. Will overwrite the default parameter *mailbox. default.connection.password*
cmas-nimh	mailbox.2.connection.port	Port for second configured mailbox. Will overwrite the default parameter *mailbox. default.connection.port*
cmas-nimh	mailbox.2.connection.protocol	Protocol (e.g., IMAP or POP3) for second configured mailbox. Will overwrite the default parameter *mailbox.default. connection.protocol*
cmas-nimh	mailbox.2.connection.username	User name for second configured mailbox. Will

Module	Property	Explanation
		overwrite the default parameter *mailbox.default.connection. username*

> ℹ 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.

Module	Property	Explanation
cmas-nimh	mailbox.default.connection.host	*Description:* Host (server name) of a given mailbox from which the poller reads e-mails. *Type:* String *Restart required:* No *System:* No *Optional:* Yes *Example value:* 10.10.1.157 *Since:* 6.4.0
cmas-nimh	mailbox.default.connection. password	*Description:* Password for given mailbox from which the poller reads e-mails. *Type:* String *Restart required:* No *System:* No *Optional:* Yes *Example value:* consol *Since:* 6.4.0
cmas-nimh	mailbox.default.connection.port	*Description:* Port for a given mailbox from which the poller reads e-mails. *Type:* String *Restart required:* No *System:* No *Optional:* Yes *Example value:* 143 *Since:* 6.4.0
cmas-nimh	mailbox.default.connection. protocol	*Description:* Poller's protocol e.g: IMAP or POP3. No default value *Type:* String *Restart required:* No

Module	Property	Explanation
		System: No *Optional:* Yes *Example value:* imap *Since:* 6.4.0
cmas-nimh	mailbox.default.connection.username	*Description:* User name for a given mailbox from which the poller reads e-mails. *Type:* String *Restart required:* No *System:* No *Optional:* Yes *Example value:* username *Since:* 6.4.0
cmas-nimh	mailbox.default.session.mail.debug	*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
cmas-nimh	mailbox.default.session.mail.imap.timeout	*Description:* Example javax.mail property *Type:* Integer *Restart required:* No *System:* No *Optional:* Yes *Example value:* 120 *Since:* 6.4.0
cmas-nimh	mailbox.default.session.mail.mime.address.strict	*Description:* Example javax.mail property - counterpart of the old mule mail.mime.strict, allows to set not so strict e-mail header parsing *Type:* Boolean *Restart required:* No *System:* No

Module	Property	Explanation
		Optional: Yes *Example value:* true *Since:* 6.4.0
cmas-nimh	mailbox.default.session.mail. pop3.timeout	*Description:* Example javax.mail property. *Type:* *Restart required:* *System:* *Optional:* *Example value:* *Since:* 6.4.0
cmas-nimh	mailbox.default.task.delete.read. messages	*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
cmas-nimh	mailbox.default.task.enabled	*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
cmas-nimh	mailbox.default.task.interval. seconds	*Description:* Default interval for polling mailboxes. Default: 60 seconds *Type:* Integer

Module	Property	Explanation
		Restart required: No *System:* No *Optional:* Yes *Example value:* 60 *Since:* 6.4.0
cmas-nimh	mailbox.default.task.max. message.size	*Description:* Maximum size of e-mail messages in bytes. Messages with a size greater than the value of this property can be processed from the Admin Tool. Default set to 10Mb : 10485760 *Type:* Integer *Restart required:* No *System:* No *Optional:* Yes *Example value:* 10485760 *Since:* 6.4.0
cmas-nimh	mailbox.default.task.max. messages.per.run	*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
cmas-nimh	mailbox.default.task.timeout. seconds	*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
cmas-nimh		

Module	Property	Explanation
	mailbox.default.task.transaction. timeout.seconds	*Description:* Default transaction timeout for e-mail 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
cmas-nimh	mailbox.polling.threads.mail.log. enabled	*Description:* Enables e-mail logging which is especially crucial in cluster environment (used as semaphore there) *Type:* String *Restart required:* No *System:* No *Optional:* Yes *Example value:* true (default) *Since:* 6.9.4.1
cmas-nimh	mailbox.polling.threads.number	*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
cmas-nimh	queue.polling.threads.number	*Description:* Number of threads started for e-mails' queue polling. Default: 1 *Type:* Integer *Restart required:* No *System:* No *Optional:* Yes *Example value:* 1 *Since:* 6.4.0
cmas-nimh		

Module	Property	Explanation
	queue.polling.threads.shutdown.timeout.seconds	*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
cmas-nimh	queue.polling.threads.watchdog.interval.seconds	*Description:* Watchdog thread interval. Default: 30 *Type:* Integer *Restart required:* No *System:* No *Optional:* Yes *Example value:* 30 *Since:* 6.4.0
cmas-nimh	queue.task.error.pause.seconds	*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
cmas-nimh	queue.task.interval.seconds	*Description:* Main mails' queue polling thread interval. Default: 15 *Type:* Integer *Restart required:* No *System:* No *Optional:* Yes *Example value:* 15 *Since:* 6.4.0
cmas-nimh	queue.task.max.retries	*Description:* Maximum number of e-mail processing retries after an exception. When reached, the e-mail is moved to the e-mail

Module	Property	Explanation
		archive. This e-mail 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
cmas-nimh	queue.task.timeout.seconds	*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
cmas-nimh	queue.task.transaction.timeout. seconds	*Description:* Transaction timeout for mail processing in the pipe. Default: 60 *Type:* Integer *Restart required:* No *System:* No *Optional:* Yes *Example value:* 60 *Since:* 6.4.0
cmas-nimh-extension	mail.attachments.validation.info. sender	*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 ⓘ This is an equivalent to the old *cmas-esb-mail,*

Module	Property	Explanation
		mail.attachments. validation.info.sender
cmas-nimh-extension	mail.attachments.validation.info. subject	*Description:* Sets subject of attachments type error notification mail. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* Mail was not processed because its attachments were rejected!!! *Since:* 6.7.5 ⓘ This is an equivalent to the old *cmas-esb-mail, mail.attachments. validation.info.subject*
cmas-nimh-extension	mail.db.archive	*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 ⓘ This is an equivalent to the old *cmas-esb-mail, mail.db.archive*
cmas-nimh-extension	mail.error.from.address	ⓘ This is an equivalent to the old *cmas-esb-mail, mail.mule.service*

Module	Property	Explanation
cmas-nimh-extension	mail.error.to.address	ⓘ This is an equivalent to the old *cmas-esb-mail, mail.process.error*
cmas-nimh-extension	mail.mule.service	*Description:* FROM-address for mails sent by Mule service. *Type:* EMail *Restart required:* No *System:* Yes *Optional:* No *Example value:* myuser@consol. de *Since:* 6.4.0
cmas-nimh-extension	mail.on.error	*Description:* When set true an error mail is sent to the above configured address in case email message could not be processed. Default: false *Type:* Boolean *Restart required:* No *System:* No *Optional:* Yes *Example value:* false *Since:* 6.4.0
cmas-nimh-extension	mail.process.error	*Description:* TO-address for error mails from Mule. *Type:* EMail *Restart required:* No *System:* Yes *Optional:* No *Example value:* myuser@consol. de *Since:* 6.4.0
cmas-nimh-extension	mail.ticketname.pattern	*Description:* *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* .*?Ticket\s+\

Module	Property	Explanation
		((\S+)\).* *Since:* 6.4.0
cmas-setup-hibernate	hibernate.dialect	*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
cmas-setup-hibernate	cmas.dropSchemaBeforeSetup	*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
cmas-setup-manager	initialized	*Description:* Flag if CMAS is initialized. If this value is missing or not *true*, set-up will be performed. *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

Module	Property	Explanation
		of the existing system is lost, including system properties!!!
cmas-setup-scene	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
cmas-workflow-engine	jobExecutor.adminMail	*Description:* E-mail 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
cmas-workflow-engine	jobExecutor.idleInterval.seconds	*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:* 5 (default) *Since:* 6.8.0
cmas-workflow-engine	jobExecutor.jobMaxRetries	*Description:* *Type:* Integer

Module	Property	Explanation
		Restart required: No *System:* Yes *Optional:* Yes *Example value:* 5 (default) *Since:* 6.8.0
cmas-workflow-engine	jobExecutor. jobMaxRetriesReachedSubject	*Description:* *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* Job maximum retries reached. Job was removed!!! (default) *Since:* 6.8.0
cmas-workflow-engine	jobExecutor.lockTimeout. seconds	*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
cmas-workflow-engine	jobExecutor.lockingLimit	*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:* 10 (default) *Since:* 6.8.0
cmas-workflow-engine	jobExecutor.mailFrom	*Description:* E-mail which will be set as FROM-header during admin notifications. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes

Module	Property	Explanation
		Example value: jobexecutor@consol.de *Since:* 6.8.0
cmas-workflow-engine	jobExecutor.maxInactivityInterval. minutes	*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
cmas-workflow-engine	jobExecutor.threads	*Description:* Number of job execution threads. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* 1 (default) *Since:* 6.8.0
cmas-workflow-engine	jobExecutor.timerRetryInterval. seconds	*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) *Since:* 6.8.0
cmas-workflow-engine	jobExecutor.txTimeout.seconds	*Description:* Transaction timeout used for job execution. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* 60 (default) *Since:* 6.8.0
cmweb-server-adapter	automatic.booking.enabled	

Module	Property	Explanation
		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
cmweb-server-adapter	checkUserOnlineIntervalInSeconds	*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
cmweb-server-adapter	cmoffice.enabled	*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
cmweb-server-adapter	commentRequiredForTicketCreation	*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
cmweb-server-adapter	customizationVersion	*Description:* *Type:* String *Restart required:* No *System:* Yes

Module	Property	Explanation
		Optional: No *Example value:* cd58453e-f3cc-4538-8030-d15e8796a4a7 *Since:* 6.5.0
cmweb-server-adapter	data.optimization	*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
cmweb-server-adapter	defaultContentEntryClassName	*Description:* Default text class for new ACIMs. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* default_class *Since:* 6.3.0
cmweb-server-adapter	defaultNumberOfCustomFieldsColumns	*Description:* Default number of columns for Custom Fields.

Module	Property	Explanation
		Type: Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 3 *Since:* 6.2.0
cmweb-server-adapter	diffTrackingEnabled	*Description:* Defines if parallel editing of a ticket by different engineers should be possible. Default *true* *Type:* Boolean *Restart required:* No *System:* *Optional:* *Example value:* true *Since:* 6.10.1
cmweb-server-adapter	favoritesSizeLimit	*Description:* Maximum number of items in Favorites list. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 10 *Since:* 6.0
cmweb-server-adapter	globalSearchResultSizeLimit	*Description:* Maximum number of items in Quick Search result. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 10 *Since:* 6.0
cmweb-server-adapter	helpFilePath	*Description:* URL for online help. If not empty, *Help* button is displayed in Web Client. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes

Module	Property	Explanation
		Example value: http://www. consol.de *Since:* 6.2.1
cmweb-server-adapter	hideTicketSubject	*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
cmweb-server-adapter	mail.from	*Description:* Use this address if set instead of engineer e-mail address during e-mail conversation. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Since:* 6.1.2
cmweb-server-adapter	mail.reply.to	*Description:* When set, Web Client will display REPLY-TO-field on e-mail send, prefilled with this value. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Since:* 6.0.1 ⊖ Please see also section Queue Administration. When you set the REPLY-TO-address in the outgoing e-mail script, the *mail.reply.to* system property must not be set (because it would overwrite the configured value)! That

Module	Property	Explanation
		means when you use one outgoing e-mail script for a queue you have to define outgoing e-mail scripts for all queues because the *mail.reply.to* property can no longer be used.
cmweb-server-adapter	mailTemplateAboveQuotedText	*Description:* Indicates behavior of e-mail template in the Ticket E-Mail Editor when another e-mail is quoted, i.e. forwarded or replied to. *Type:* Boolean *Restart required:* No *System:* Yes *Optional:* No *Example value:* false *Since:* 6.2.4
cmweb-server-adapter	maxSizePerPagemapInMegaBytes	*Description:* Maximum size (in MB) for each Wicket pagemap. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 15 *Since:* 6.3.5
cmweb-server-adapter	pagemapLockDurationInSeconds	*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
cmweb-server-adapter	postActivityExecutionScriptName	*Description:* Defines the name for the script which should be executed after every workflow

Module	Property	Explanation
		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
cmweb-server-adapter	queuesExcludedFromGS	*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
cmweb-server-adapter	rememberMeLifetimeInMinutes	*Description:* Lifetime for *remember me* in minutes. *Type:* Integer *Restart required:* Yes *System:* Yes *Optional:* No *Example value:* 1440 *Since:* 6.0
cmweb-server-adapter	request.scope.transaction	*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
cmweb-server-adapter	searchPageSize	

Module	Property	Explanation
		Description: Default page size for search results. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 20 *Since:* 6.0
cmweb-server-adapter	searchPageSizeOptions	*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
cmweb-server-adapter	serverPoolingInterval	*Description:* *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 5 *Since:* 6.1.0
cmweb-server-adapter	supportEmail	*Description:* *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Since:* 6.0
cmweb-server-adapter	themeOverlay	*Description:* Name of used theme overlay *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* kyoEUR *Since:* 6.0
cmweb-server-adapter	ticketListRefreshIntervalInSeconds	*Description:* Refresh interval for ticket list (in seconds). *Type:* Integer

Module	Property	Explanation
		Restart required: No *System:* Yes *Optional:* No *Example value:* 180 *Since:* 6.0
cmweb-server-adapter	ticketListSizeLimit	*Description:* Maximum number of tickets in ticket list. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 100 *Since:* 6.0
cmweb-server-adapter	unitIndexSearchResultSizeLimit	*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
cmweb-server-adapter	urlLogoutPath	*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
cmweb-server-adapter	webSessionTimeoutInMinutes	*Description:* Session timeout in minutes. *Type:* Integer *Restart required:* Yes *System:* Yes *Optional:* No *Example value:* 180 *Removed in:* 6.7.1

Module	Property	Explanation
		Replaced by: cmas-core-server, server.session.timeout
cmweb-server-adapter	wicketAjaxRequestHeaderFilterEnabled	*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
cmas-workflow-jbpm	fetchLock.interval	*Description:* *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 5000 *Removed in:* 6.8.0
cmas-workflow-jbpm	fetchLock.timeout	*Description:* *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 15000 *Removed in:* 6.8.0
cmas-workflow-jbpm	jobExecutor.idleInterval	*Description:* *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 45000 *Removed in:* 6.8.0 *Replaced by:* jobExecutor. idleInterval.seconds
cmas-workflow-jbpm	jobExecutor. jobExecuteRetryNumber	*Description:* *Type:* Integer *Restart required:* No *System:* Yes

Module	Property	Explanation
		Optional: No *Example value:* 5 *Removed in:* 6.8.0 *Replaced by:* jobExecutor. jobMaxRetries
cmas-workflow-jbpm	jobExecutor.timerRetryInterval	*Description:* *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 10000 *Removed in:* 6.8.0 *Replaced by:* jobExecutor. timerRetryInterval.seconds
cmas-workflow-jbpm	mail.sender.address	*Description:* FROM-address for e-mails from the workflow engine. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* myuser@conso. de *Removed in:* 6.8.0 *Replaced by:* jobExecutor. mailFrom
cmas-workflow-jbpm	outdated.lock.age	*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
cmas-workflow-jbpm	refreshTimeInCaseOfConcurrent RememberMeRequests	*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

Module	Property	Explanation
		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

Module	Property	Explanation

50.2 System Properties Ordered by Property Name

Module	Property	Explanation
cmas-core-security	admin.email	*Description:* The e-mail address of the ConSol*CM administrator. The value which you have 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
cmas-core-security	admin.login	*Description:* The name of the ConSol*CM administrator. The value which you have entered during system set-up is used initially. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* admin *Since:* 6.0
cmas-app-admin-tool	admin.tool.session.check.interval	*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
cmas-core-server	attachment.allowed.types	*Description:* Comma-separated list of allowed filename extensions (if no value defined, all file extensions are allowed). *Type:* String *Restart required:* No

Module	Property	Explanation
		System: Yes *Optional:* Yes *Example value:* txt,zip,doc *Since:* 6.5.0
cmas-core-server	attachment.max.size	*Description:* Maximum attachment size in MB *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 100 *Since:* 6.4.0
cmas-core-security	authentication.method	*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
cmas-dwh-server	autocommit.cf.changes	*Description:* *Type:* Boolean *Restart required:* No *System:* Yes *Optional:* No *Example value:* false *Since:* 6.7.0
cmas-app-admin-tool	autocomplete.enabled	*Description:* If the flag is missing or its value is *false*, then the *Auto complete 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
cmweb-server-adapter	automatic.booking.enabled	*Description:* If enabled, time spend on creating comment

Module	Property	Explanation
		/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
cmas-dwh-server	batch-commit-interval	*Description:* Number of objects in a JMS message. Higher value means better transfer performance and bigger memory usage. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* 100 *Since:* 6.0.0
cmas-core-index-common	big.task.minimum.size	*Description:* How many parts task at least should have to be handled by Indexer with low priority. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 15 (default) *Since:* 6.8.3
cmas-core-cache	cache-cluster-name	*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
cmweb-server-adapter	checkUserOnlineIntervalInSeconds	*Description:* The interval in seconds to check which users

Module	Property	Explanation
		are online (default 180sec = 3min). *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 180 *Since:* 6.0
cmas-core-shared	cluster.mode	*Description:* Flag if CMAS is running in cluster. *Type:* Boolean *Restart required:* Yes *System:* Yes *Optional:* No *Example value:* false *Since:* 6.1.0
cmas-setup-hibernate	cmas.dropSchemaBeforeSetup	*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
cmweb-server-adapter	cmoffice.enabled	*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
cmweb-server-adapter	commentRequiredForTicketCreation	*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
cmas-dwh-server	communication.channel	

Module	Property	Explanation
		Description: Communication channel. Possible values are DIRECT (database communication channel, default value since 6.9.4.1), JMS (default value before 6.9.4.1). Before 6.9.4.1 it has to be manually added. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* DIRECT *Since:* 6.8.5.0
cmas-core-server	config.data.version	*Description:* *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 11 *Since:* 6.0
cmas-core-security	contact.authentication.method	*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
cmas-core-security	contact.inherit.permissions.only. to.own.customer.group	*Description:* Indicates whether authenticated contact inherits all customer group permissions from representing engineer (false) or only permission to own customer group (true). *Type:* Boolean *Restart required:* No *System:* Yes *Optional:* No *Since:* 6.9.2.3

Module	Property	Explanation
cmweb-server-adapter	customizationVersion	*Description:* *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* cd58453e-f3cc-4538-8030-d15e8796a4a7 *Since:* 6.5.0
cmas-core-shared	data.directory	*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
cmweb-server-adapter	data.optimization	*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

Module	Property	Explanation
		Example value: MINIFICATION|COMPRESSION
cmas-core-index-common	database.notification.enabled	*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
cmas-core-index-common	database.notification.redelivery. delay.seconds	*Description:* In case of index update database notification channel, indicates notification redelivery delay when exception occurs. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 60 *Since:* 6.8.4.7
cmas-core-index-common	database.notification.redelivery. max.attempts	*Description:* In case of index update database notification channel, indicates maximum redelivery attempts when exception occurs. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 60 *Since:* 6.8.4.7
cmas-core-server	defaultCommentClassName	*Description:* Default text class name for comments *Type:* String *Restart required:* No *System:* No

Module	Property	Explanation
		Optional: Yes *Example value:* *Since:* 6.3.0
cmweb-server-adapter	defaultContentEntryClassName	*Description:* Default text class for new ACIMs.. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* default_class *Since:* 6.3.0
cmas-core-server	defaultIncommingMailClassName	*Description:* Default text class name for incoming e-mails *Type:* String *Restart required:* No *System:* No *Optional:* Yes *Example value:* *Since:* 6.3.0
cmweb-server-adapter	defaultNumberOfCustomFieldsColumns	*Description:* Default number of columns for Custom Fields. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 3 *Since:* 6.2.0
cmas-core-server	defaultOutgoingMailClassName	*Description:* Default text class name for outgoing e-mails *Type:* String *Restart required:* No *System:* No *Optional:* Yes *Example value:* *Since:* 6.3.0
cmas-app-admin-tool	delete.ticket.enabled	*Description:* Controls if the menu entry Delete is displayed in the context menu in the AT for the ticket list in ticket administration *Type:* Boolean *Restart required:* No

Module	Property	Explanation
		System: No *Optional:* Yes *Example value:* true *Since:* 6.9.4.0
cmas-core-index-common	disable.admin.task.auto.commit	*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
cmas-dwh-server	dwh.mode	*Description:* Current mode of 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
cmas-esb-core	esb.directory	*Description:* Directory used by ESB (Mule) *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* C:\Users\user\cmas\mule *Since:* 6.0
cmas-core-cache	eviction.event.queue.size	*Description:* *Type:* Integer *Restart required:* Yes *System:* Yes *Optional:* No *Example value:* 200000 *Since:* 6.4.0
cmas-core-cache	eviction.max.nodes	

Module	Property	Explanation
		Description: *Type:* Integer *Restart required:* Yes *System:* Yes *Optional:* No *Example value:* 100000 *Since:* 6.4.0
cmas-core-cache	eviction.wakeup.interval	*Description:* *Type:* Integer *Restart required:* Yes *System:* Yes *Optional:* No *Example value:* 3000 *Since:* 6.4.0
cmweb-server-adapter	favoritesSizeLimit	*Description:* Maximum number of items in Favorites list *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 10 *Since:* 6.0
cmas-workflow-jbpm	fetchLock.interval	*Description:* *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 5000 *Removed in:* 6.8.0
cmas-workflow-jbpm	fetchLock.timeout	*Description:* *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 15000 *Removed in:* 6.8.0
cmas-core-server	fetchSize.strategy	*Description:* Strategy selected to set fetch size on jdbc result sets. *Type:* String *Restart required:* No *System:* Yes

Module	Property	Explanation
		Optional: Yes *Example value:* FetchSizePageBasedStrategy, FetchSizeThresholdStrategy, FetchSizeFixedStrategy *Since:* 6.8.4.1
cmas-core-server	fetchSize.strategy. FetchSizeFixedStrategy.value	*Description:* Sets fetch size value if selected strategy to set fetch size is *FetchSizeFixedStrategy*. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* 150 *Since:* 6.8.4.1
cmas-core-server	fetchSize.strategy. FetchSizePageBasedStrategy. limit	*Description:* Sets maximum fetch size value if selected strategy to set fetch size is *FetchSizePageBasedStrategy*. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* 10000 *Since:* 6.8.4.1
cmas-core-server	fetchSize.strategy. FetchSizeThresholdStrategy. value	*Description:* Sets fetch size threshold border values if selected strategy to set fetch size is *FetchSizeThresholdStrategy*. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* 150,300,600,1000 *Since:* 6.8.4.1
cmas-nimh	filesystem.polling.threads. number	*Description:* Number of threads started for db mails' queue polling. Default: 1

Module	Property	Explanation

Module	Property	Explanation
		Type: Integer *Restart required:* No *System:* No *Optional:* Yes *Example value:* 10 *Since:* 6.4.0
cmas-nimh	filesystem.polling.threads. shutdown.timeout.seconds	*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
cmas-nimh	filesystem.polling.threads. watchdog.interval.seconds	*Description:* Watchdog thread interval. Default: 30 *Type:* Integer *Restart required:* No *System:* No *Optional:* Yes *Example value:* 60 *Since:* 6.4.0
cmas-nimh	filesystem.task.enabled	*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
cmas-nimh	filesystem.task.interval.seconds	*Description:* Default interval for polling mailboxes. Default: 60 seconds *Type:* Integer *Restart required:* No *System:* No *Optional:* Yes

Module	Property	Explanation
		Example value: 60 *Since:* 6.4.0
cmas-nimh	filesystem.task.polling.folder	*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
cmas-nimh	filesystem.task.timeout.seconds	*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
cmas-nimh	filesystem.task.transaction. timeout.seconds	*Description:* Default transaction timeout for mail 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
cmweb-server-adapter	globalSearchResultSizeLimit	*Description:* Maximum number of items in Quick Search result. *Type:* Integer *Restart required:* No *System:* Yes

Module	Property	Explanation
		Optional: No *Example value:* 10 *Since:* 6.0
cmweb-server-adapter	helpFilePath	*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
cmas-setup-hibernate	hibernate.dialect	*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
cmweb-server-adapter	hideTicketSubject	*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
cmas-dwh-server	ignore-queues	*Description:* By adding a comma separated list of queue names it is configured that tickets of these queues are not transferred to the DWH. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes

Module	Property	Explanation
		Example value: QueueName1, QueueName2,QueueName3 *Since:* 6.6.19 *Removed in:* 6.8.1
cmas-core-index-common	index.attachment	*Description:* Describes if content of attachments is indexed. *Type:* Boolean *Restart required:* No *System:* Yes *Optional:* No *Example value:* true *Since:* 6.4.3
cmas-core-index-common	index.history	*Description:* Describes if unit and ticket history are indexed. *Type:* Boolean *Restart required:* No *System:* Yes *Optional:* No *Example value:* false *Since:* 6.1.0
cmas-core-index-common	index.status	*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
cmas-core-index-common	index.task.worker.threads	*Description:* How many threads will be used to execute batch index tasks (synchronization, administrative, and repair tasks). *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 1 (default) (we

Module	Property	Explanation
		recommend to use a value not larger than 2) *Since:* 6.6.14, 6.7.3
cmas-core-index-common	index.version.current	*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
cmas-core-index-common	index.version.newest	*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
cmas-core-index-common	indexed.assets.per.thread.in.memory	*Description:* How many assets should be loaded into memory at once during indexing per one thread. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 200 (default) *Since:* 6.8.0
cmas-core-index-common	indexed.engineers.per.thread.in.memory	*Description:* How many engineers should be loaded into memory at once during indexing per one thread. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 300 (default) *Since:* 6.6.14, 6.7.3
Module	Property	Explanation

Module	Property	Explanation
cmas-core-index-common	indexed.tickets.per.thread.in. memory	*Description:* How many tickets should be loaded into memory at once during indexing per one thread. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 100 (default) *Since:* 6.6.14, 6.7.3
cmas-core-index-common	indexed.units.per.thread.in. memory	*Description:* How many units should be loaded into memory at once during indexing per one thread. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 200 (default) *Since:* 6.6.14, 6.7.3
cmas-setup-manager	initialized	*Description:* Flag if CMAS is initialized. If this value is missing or not *true*, set-up will be performed. *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!!!

cmas-dwh-server	is.cmrf.alive	*Description:* As a starting point time of sending last message to CMRF should be used. If response from CMRF is not received after value (in seconds) it should create a DWH operation status with error message that CMRF is down. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 1200 *Since:* 6.7.0
cmas-dwh-server	java.naming.factory.initial	*Description:* Factory class for DWH context factory. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* org.jnp. interfaces.NamingContextFactory *Since:* 6.0.1
cmas-dwh-server	java.naming.factory.url.pkgs	*Description:* *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* org.jboss. naming:org.jnp.interfaces *Since:* 6.0.1
cmas-dwh-server	java.naming.provider.url	*Description:* URL of naming provider *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* localhost *Since:* 6.0.1
cmas-workflow-engine	jobExecutor.adminMail	*Description:* E-mail address which will get notified about job

Module	Property	Explanation
		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
cmas-workflow-jbpm	jobExecutor.idleInterval	*Description:* *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 45000 *Removed in:* 6.8.0 *Replaced by:* jobExecutor.idleInterval.seconds
cmas-workflow-engine	jobExecutor.idleInterval.seconds	*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:* 5 (default) *Since:* 6.8.0
cmas-workflow-jbpm	jobExecutor.jobExecuteRetryNumber	*Description:* *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 5 *Removed in:* 6.8.0 *Replaced by:* jobExecutor.jobMaxRetries
cmas-workflow-engine	jobExecutor.jobMaxRetries	*Description:* *Type:* Integer *Restart required:* No *System:* Yes *Optional:* Yes

Module	Property	Explanation
		Example value: 5 (default) *Since:* 6.8.0
cmas-workflow-engine	jobExecutor. jobMaxRetriesReachedSubject	*Description:* *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* Job maximum retries reached. Job was removed!!! (default) *Since:* 6.8.0
cmas-workflow-engine	jobExecutor.lockingLimit	*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:* 10 (default) *Since:* 6.8.0
cmas-workflow-engine	jobExecutor.lockTimeout. seconds	*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
cmas-workflow-engine	jobExecutor.mailFrom	*Description:* E-mail 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
cmas-workflow-engine	jobExecutor.maxInactivityInterval. minutes	*Description:* Number of minutes of allowed job executor inactivity

Module	Property	Explanation
		(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
cmas-workflow-engine	jobExecutor.threads	*Description:* Number of job execution threads *Type:* Integer *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* 1 (default) *Since:* 6.8.0
cmas-workflow-jbpm	jobExecutor.timerRetryInterval	*Description:* *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 10000 *Removed in:* 6.8.0 *Replaced by:* jobExecutor. timerRetryInterval.seconds
cmas-workflow-engine	jobExecutor.timerRetryInterval. seconds	*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) *Since:* 6.8.0
cmas-workflow-engine	jobExecutor.txTimeout.seconds	*Description:* Transaction timeout used for job execution *Type:* Integer *Restart required:* No *System:* Yes

Module	Property	Explanation
		Optional: Yes *Example value:* 60 (default) *Since:* 6.8.0
cmas-core-security	kerberos.v5.enabled	*Description:* Flag which indicates whether SSO via Kerberos is enabled. *Type:* Boolean *Restart required:* No *System:* Yes *Optional:* No *Example value:* false (default if Kerberos has not been enabled during system set-up) *Since:* 6.2.0
cmas-core-security	kerberos.v5.username.regex	*Description:* Regular expression used for mapping Kerberos principal to CM user login. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* (.*)@.* *Since:* 6.2.0
cmas-core-server	last.config.change	*Description:* Random UUID created during last change in config *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* 2573c7b7-2bf5-47ff-b5a2-bad31951a266 *Since:* 6.1.0, 6.2.1
cmas-core-security	ldap.authentication	*Description:* Authentication method used when using LDAP authentication. *Type:* String *Restart required:* Yes *System:* Yes

Module	Property	Explanation

Module	Property	Explanation
		Optional: No *Example value:* simple *Since:* 6.0
cmas-core-security	ldap.basedn	*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
cmas-core-server	ldap.certificate.basedn	*Description:* Base DN for certificates location in LDAP tree. If not provided, *cmas-core-security*, *ldap.basedn* is taken. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* ou=accounts, dc=consol,dc=de *Since:* 6.8.4
cmas-core-server	ldap.certificate.content.attribute	*Description:* LDAP attribute name used where certificate data is stored in LDAP tree. Default value is: usercertificate. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* usercertificate *Since:* 6.8.4
cmas-core-server	ldap.certificate.password	*Description:* LDAP Certificates manager password. If not set, *cmas-core-security*, *ldap.password* is taken. *Type:* String *Restart required:* No *System:* Yes

Module	Property	Explanation
		Optional: Yes *Since:* 6.8.4
cmas-core-server	ldap.certificate.providerurl	*Description:* LDAP Certificates provider URL. If not set, *cmas-core-security*, *ldap.providerurl* is taken. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* ldap://myserver. consol.de:389 *Since:* 6.8.4
cmas-core-server	ldap.certificate.searchattr	*Description:* LDAP attribute name used to search for certificate in LDAP tree. Default value is: mail. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* mail *Since:* 6.8.4
cmas-core-server	ldap.certificate.userdn	*Description:* LDAP Certificates manager DN. If not set, *cmas-core-security*, *ldap.userdn* is taken. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Since:* 6.8.4
cmas-core-security	ldap.contact.name.basedn	*Description:* Base path to search for contact DN by LDAP ID (e.g. ou=accounts,dc=consol,dc=de). *Type:* String *Restart required:* No *System:* No *Optional:* Yes *Since:* 6.9.3.0
cmas-core-security	ldap.contact.name.password	

Module	Property	Explanation
		Description: Password to look up contact DN by LDAP ID. If not set, anonymous account is used. *Type:* String *Restart required:* No *System:* No *Optional:* Yes *Since:* 6.9.3.0
cmas-core-security	ldap.contact.name.providerurl	*Description:* Address of the LDAP server (ldap[s]://host:port). *Type:* String *Restart required:* No *System:* No *Optional:* Yes *Since:* 6.9.3.0
cmas-core-security	ldap.contact.name.searchattr	*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
cmas-core-security	ldap.contact.name.userdn	*Description:* User DN to look up contact DN by LDAP ID. If not set, anonymous account is used. *Type:* String *Restart required:* No *System:* No *Optional:* Yes *Since:* 6.9.3.0
cmas-core-security	ldap.initialcontextfactory	*Description:* Class name for initial context factory of LDAP implementation when using LDAP authentication. If it is not set, com.sun.jndi.ldap. LdapCtxFactory is being used as a value. *Type:* String *Restart required:* Yes *System:* Yes

Module	Property	Explanation
		Optional: No *Example value:* com.sun.jndi. ldap.LdapCtxFactory *Since:* 6.0
cmas-core-security	ldap.password	*Description:* Password for connecting to LDAP to look up users (when using LDAP authentication). Only needed if look-up cannot be done anonymously. *Type:* Password *Restart required:* No *System:* Yes *Optional:* Yes *Since:* 6.1.2
cmas-core-security	ldap.providerurl	*Description:* LDAP provider (when using LDAP authentication). *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* ldap://ldap.consol.de:389 *Since:* 6.0
cmas-core-security	ldap.searchattr	*Description:* Search attribute for looking up LDAP entry connected to CM6 login. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* uid *Since:* 6.0
cmas-core-security	ldap.userdn	*Description:* LDAP user for connecting to LDAP to look up users (when using LDAP authentication). Only needed if look-up cannot be done anonymously. *Type:* String

Module	Property	Explanation
		Restart required: No *System:* Yes *Optional:* Yes *Since:* 6.1.2
cmas-esb-mail	mail.attachments.validation.info.sender	*Description:* Sets FROM-header of attachments type error notification e-mail. As a default the e-mail address of the administrator which you have entered during system set-up is used. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* admin@consolcm.com *Since:* 6.7.5
cmas-nimh-extension	mail.attachments.validation.info.sender	*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 ⓘ This is an equivalent to the old *cmas-esb-mail, mail.attachments. validation.info.sender*
cmas-esb-mail	mail.attachments.validation.info.subject	*Description:* Sets subject of attachments type error notification e-mail. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* E-mail was not

Module	Property	Explanation
		processed because its attachments were rejected!!! *Since:* 6.7.5
cmas-nimh-extension	mail.attachments.validation.info. subject	*Description:* Sets subject of attachments type error notification mail *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* Mail was not processed because its attachments were rejected!!! *Since:* 6.7.5 ⓘ This is an equivalent to the old *cmas-esb-mail, mail.attachments. validation.info.subject*
cmas-esb-mail	mail.callname.pattern	*Description:* Regular expression for subject of incoming e-mails. Available as TICKET_NAME_PATTERN_FO RMAT in incoming e-mail scripts. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* .*?Ticket\s+\ ((\S+)\).* *Since:* 6.0
cmas-esb-mail	mail.cluster.node.id	*Description:* Only the node whose mail.cluster.node.id equals cmas.clusternode.id will start the Mule ESB mail services. *Type:* String *Restart required:* No *System:* Yes

Module	Property	Explanation
		Optional: No *Example value:* unspecified *Since:* 6.6.5
cmas-esb-mail	mail.db.archive	*Description:* If property is set to *true*, incoming e-mails are archived in the database. *Type:* Boolean *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* false (default) *Since:* 6.8.5.5
cmas-nimh-extension	mail.db.archive	*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 ⓘ This is an equivalent to the old *cmas-esb-mail, mail.db.archive*
cmas-esb-mail	mail.delete.read	*Description:* Determines whether CM deletes messages fetched via IMAP(S). Setting value to *true* will cause deletion of messages after fetching. Default is to not delete messages fetched via IMAP(S). Note: Messages fetched via POP3(S) will always be deleted. *Type:* Boolean *Restart required:* No *System:* Yes *Optional:* No *Example value:* true *Since:* 6.7.3

Module	Property	Explanation
cmas-core-server	mail.encryption	*Description:* If property is set to *true*, the encrypt check box in the Ticket E-Mail Editor is checked by default. *Type:* Boolean *Restart required:* No *System:* Yes *Optional:* No *Example value:* true (default = false) *Since:* 6.8.4.0
cmas-nimh-extension	mail.error.from.address	ⓘ This is an equivalent to the old *cmas-esb-mail, mail.mule.service*
cmas-nimh-extension	mail.error.to.address	ⓘ This is an equivalent to the old *cmas-esb-mail, mail.process.error*
cmweb-server-adapter	mail.from	*Description:* Use this address if set instead of engineer e-mail address during e-mail conversation. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Since:* 6.1.2
cmas-esb-mail	mail.incoming.uri	*Description:* URL for incoming e-mails *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* pop3://cm-

Module	Property	Explanation
		incoming-user: password@localhost:10110 _Since:_ 6.0 ⊖ This value should not be edited here using the system properties pop-up window, but the mailboxes should be configured using the navigation item E-mail. Using this standard feature all entries are controlled - i.e. for each mailbox which is added, CM establishes a test connection during mailbox set-up. That way it is not possible to enter wrong values.
cmas-esb-mail	mail.max.restarts	_Description:_ Maximum number of mail service restarts before giving up _Type:_ Integer _Restart required:_ No _System:_ Yes _Optional:_ No _Example value:_ 3 _Since:_ 6.0
cmas-esb-mail	mail.mime.strict	_Description:_ If set to _false_, e-mail addresses are not parsed for strict MIME compliance. Default is _true_, which means check for strict MIME compliance. _Type:_ Boolean _Restart required:_ No _System:_ Yes

Module	Property	Explanation
		Optional: No *Example value:* false *Since:* 6.6.17, 6.7.3
cmas-esb-mail	mail.mule.service	*Description:* FROM-address for e-mails sent by Mule service *Type:* EMail *Restart required:* No *System:* Yes *Optional:* No *Example value:* myuser@consol.de *Since:* 6.0
cmas-nimh-extension	mail.mule.service	*Description:* FROM-address for mails sent by Mule service *Type:* EMail *Restart required:* No *System:* Yes *Optional:* No *Example value:* myuser@consol.de *Since:* 6.4.0
cmas-core-server	mail.notification.engineerChange	*Description:* Flag if notification e-mail should be sent when engineer of ticket is changed. *Type:* Boolean *Restart required:* No *System:* Yes *Optional:* No *Example value:* true *Since:* 6.1.0
cmas-core-server	mail.notification.sender	*Description: From* address for notification e-mails when engineer of ticket is changed. If not set, *cmas-core-security*, *admin.email* is used instead. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes

Module	Property	Explanation

Module	Property	Explanation
		Example value: cm6notification@cm6installation *Since:* 6.6.3
cmas-nimh-extension	mail.on.error	*Description:* When set true an error mail is sent to the above configured address in case email message could not be processed. Default: false *Type:* Boolean *Restart required:* No *System:* No *Optional:* Yes *Example value:* false *Since:* 6.4.0
cmas-esb-mail	mail.polling.interval	*Description:* Mail polling interval in ms *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 60000 *Since:* 6.0
cmas-esb-mail	mail.process.error	*Description:* TO-address for error e-mails from Mule. As a default the e-mail address of the administrator which you have entered during system set-up is used. *Type:* EMail *Restart required:* No *System:* Yes *Optional:* No *Example value:* myuser@consol.de *Since:* 6.0
cmas-nimh-extension	mail.process.error	*Description:* TO-address for error mails from Mule *Type:* EMail *Restart required:* No *System:* Yes *Optional:* No

Module	Property	Explanation
		Example value: myuser@consol. de *Since:* 6.4.0
cmas-esb-mail	mail.process.retry.attempts	*Description:* Number of retries when processing e-mail *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 3 *Since:* 6.0.2
cmas-esb-mail	mail.process.timeout	*Description:* Mail processing timeout in seconds *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 60 *Since:* 6.1.3
cmas-esb-mail	mail.redelivery.retry.count	*Description:* Indicates the number of retries of re-delivering an e-mail from the CM system. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 3 *Since:* 6.1.0
cmweb-server-adapter	mail.reply.to	*Description:* When set, Web Client will display REPLY-TO-field on e-mail send, prefilled with this value. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Since:* 6.0.1 ⊖ Please see also section Queue Administration.

Module	Property	Explanation
		When you set the REPLY-TO-address in the outgoing e-mail script, the *mail.reply.to* system property must not be set (because it would overwrite the configured value)! That means when you use one outgoing e-mail script for a queue you have to define outgoing e-mail scripts for all queues because the *mail.reply.to* property can no longer be used.
cmas-workflow-jbpm	mail.sender.address	*Description:* FROM-address for e-mails 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
cmas-core-server	mail.smtp.email	*Description:* SMTP e-mail URL for outgoing e-mails *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* smtp://myserver.consol.de:25 *Since:* 6.0
cmas-core-server	mail.smtp.envelopesender	*Description:* E-mail address used as sender in SMTP envelope. If not set, the FROM-address of the e-mail is used.

Module	Property	Explanation
		Type: String *Restart required:* No *System:* Yes *Optional:* No *Example value:* mysender@mydomain.com *Since:* 6.5.7
cmas-nimh-extension	mail.ticketname.pattern	*Description:* *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* .*?Ticket\s+\\ ((\S+)\).* *Since:* 6.4.0
cmas-nimh	mailbox.1.connection.host	Host (server) for first configured mailbox. Will overwrite the default parameter *mailbox. default.connection.host*
cmas-nimh	mailbox.1.connection.password	Password for first configured mailbox. Will overwrite the default parameter *mailbox. default.connection.password*
cmas-nimh	mailbox.1.connection.port	Port for first configured mailbox. Will overwrite the default parameter *mailbox.default. connection.port*
cmas-nimh	mailbox.1.connection.protocol	Protocol (e.g., IMAP or POP3) for first configured mailbox. Will overwrite the default parameter *mailbox.default.connection. protocol*
cmas-nimh	mailbox.1.connection.username	User name for first configured mailbox. Will overwrite the default parameter *mailbox. default.connection.username*
cmas-nimh	mailbox.2.connection.host	Host (server) for second configured mailbox. Will overwrite the default parameter *mailbox.default.connection.host*

Module	Property	Explanation
cmas-nimh	mailbox.2.connection.password	Password for second configured mailbox. Will overwrite the default parameter *mailbox.default.connection.password*
cmas-nimh	mailbox.2.connection.port	Port for second configured mailbox. Will overwrite the default parameter *mailbox.default.connection.port*
cmas-nimh	mailbox.2.connection.protocol	Protocol (e.g., IMAP or POP3) for second configured mailbox. Will overwrite the default parameter *mailbox.default.connection.protocol*
cmas-nimh	mailbox.2.connection.username	User name for second configured mailbox. Will overwrite the default parameter *mailbox.default.connection.username*

> ℹ️ 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.

Module	Property	Explanation
cmas-nimh	mailbox.default.connection.host	*Description:* Host (server name) of a given mailbox from which the poller reads e-mails. *Type:* String *Restart required:* No *System:* No *Optional:* Yes *Example value:* 10.10.1.157 *Since:* 6.4.0
cmas-nimh	mailbox.default.connection. password	*Description:* Password for given mailbox from which the poller reads e-mails. *Type:* String *Restart required:* No *System:* No *Optional:* Yes

Module	Property	Explanation
		Example value: consol *Since:* 6.4.0
cmas-nimh	mailbox.default.connection.port	*Description:* Port for a given mailbox from which the poller reads e-mails. *Type:* String *Restart required:* No *System:* No *Optional:* Yes *Example value:* 143 *Since:* 6.4.0
cmas-nimh	mailbox.default.connection.protocol	*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
cmas-nimh	mailbox.default.connection.username	*Description:* User name for a given mailbox from which the poller reads e-mails. *Type:* String *Restart required:* No *System:* No *Optional:* Yes *Example value:* username *Since:* 6.4.0
cmas-nimh	mailbox.default.session.mail.debug	*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
cmas-nimh	mailbox.default.session.mail.imap.timeout	*Description:* Example javax.mail property *Type:* Integer
Module	**Property**	**Explanation**

Module	Property	Explanation
		Restart required: No *System:* No *Optional:* Yes *Example value:* 120 *Since:* 6.4.0
cmas-nimh	mailbox.default.session.mail. mime.address.strict	*Description:* Example javax.mail property - counterpart of the old mule mail.mime.strict, allows to set not so strict mail header parsing *Type:* Boolean *Restart required:* No *System:* No *Optional:* Yes *Example value:* true *Since:* 6.4.0
cmas-nimh	mailbox.default.session.mail. pop3.timeout	*Description:* Example javax.mail property. *Type:* *Restart required:* *System:* *Optional:* *Example value:* *Since:* 6.4.0
cmas-nimh	mailbox.default.task.delete.read. messages	*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
cmas-nimh	mailbox.default.task.enabled	

Module	Property	Explanation
		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
cmas-nimh	mailbox.default.task.interval. seconds	*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
cmas-nimh	mailbox.default.task.max. message.size	*Description:* Maximum size of e-mail messages in bytes. Messages with a size greater than the value of this property can be processed from the Admin Tool. Default set to 10Mb : 10485760 *Type:* Integer *Restart required:* No *System:* No *Optional:* Yes *Example value:* 10485760 *Since:* 6.4.0
cmas-nimh	mailbox.default.task.max. messages.per.run	*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

Module	Property	Explanation
		Optional: Yes *Example value:* 60 *Since:* 6.4.0
cmas-nimh	mailbox.default.task.timeout.seconds	*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
cmas-nimh	mailbox.default.task.transaction.timeout.seconds	*Description:* Default transaction timeout for mail 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
cmas-nimh	mailbox.polling.threads.mail.log.enabled	*Description:* Enables mail logging which is especially crucial in cluster environment (used as semaphore there) *Type:* String *Restart required:* No *System:* No *Optional:* Yes *Example value:* true (default) *Since:* 6.9.4.1
cmas-nimh	mailbox.polling.threads.number	*Description:* Number of threads for accessing mailboxes. Default: 1 *Type:* Integer *Restart required:* No

Module	Property	Explanation
		System: No *Optional:* Yes *Example value:* 1 *Since:* 6.4.0
cmweb-server-adapter	mailTemplateAboveQuotedText	*Description:* Indicates behavior of e-mail template in the Ticket E-Mail Editor when another e-mail is quoted, i.e. forwarded or replied to. *Type:* Boolean *Restart required:* No *System:* Yes *Optional:* No *Example value:* false *Since:* 6.2.4
cmas-core-server	max.licences.perUser	*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
cmweb-server-adapter	maxSizePerPagemapInMegaBytes	*Description:* Maximum size (in MB) for each Wicket pagemap *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 15 *Since:* 6.3.5
cmas-core-server	monitoring.engineer.login	*Description:* Login of monitoring engineer *Type:* String *Restart required:* No *System:* Yes

Module	Property	Explanation
		Optional: Yes *Example value:* nagios *Since:* 6.9.3.0
cmas-core-server	monitoring.unit.login	*Description:* Login of monitoring unit *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* nagios *Since:* 6.9.3.0
cmas-core-server	nimh.enabled	*Description:* Enables NIMH service. Must be suffixed with nodeid in cluster e.g. nimh.enabled.NODEID = true *Type:* Boolean *Restart required:* No *System:* No *Optional:* Yes *Example value:* false *Since:* 6.9.4.0
cmas-dwh-server	notification.error.description	*Description:* Text for error e-mails from DWH *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* Error occurred *Since:* 6.0.1
cmas-dwh-server	notification.error.from	*Description:* FROM-address for error e-mails from DWH *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Since:* 6.0.1
cmas-dwh-server	notification.error.subject	*Description:* Subject for error e-mails from DWH *Type:* String *Restart required:* No *System:* Yes

Module	Property	Explanation
		Optional: No *Example value:* Error occurred *Since:* 6.0.1
cmas-dwh-server	notification.error.to	*Description:* TO-address for error e-mails from DWH *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* myuser@consol.de *Since:* 6.0.1
cmas-dwh-server	notification.finished_successfully.description	*Description:* Text for e-mails from DWH when transfer finished successfully. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* Transfer finished successfully *Since:* 6.0.1
cmas-dwh-server	notification.finished_successfully.from	*Description:* FROM-address for e-mails from DWH when transfer finished successfully. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Since:* 6.0.1
cmas-dwh-server	notification.finished_successfully.subject	*Description:* Subject for e-mails from DWH when transfer finished successfully. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* Transfer finished successfully *Since:* 6.0.1
cmas-dwh-server		

Module	Property	Explanation
	notification.finished_successfully. to	*Description:* TO-address for e-mails from DWH when transfer finished successfully. *Restart required:* Yes *System:* Yes *Optional:* No *Example value:* myuser@consol. de *Since:* 6.0.1
cmas-dwh-server	notification. finished_unsuccessfully. description	*Description:* Text for e-mails from DWH when transfer finished unsuccessfully. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* Transfer finished unsuccessfully *Since:* 6.0.1
cmas-dwh-server	notification. finished_unsuccessfully.from	*Description:* FROM-address for e-mails from DWH when transfer finished unsuccessfully. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Since:* 6.0.1
cmas-dwh-server	notification. finished_unsuccessfully.subject	*Description:* Subject for e-mails from DWH when transfer finished unsuccessfully. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* Transfer finished unsuccessfully *Since:* 6.0.1
cmas-dwh-server	notification. finished_unsuccessfully.to	*Description:* TO-address for e-mails from DWH when transfer finished unsuccessfully. *Type:* String

Module	Property	Explanation
		Restart required: No *System:* Yes *Optional:* No *Example value:* myuser@consol. de *Since:* 6.0.1
cmas-dwh-server	notification.host	*Description:* Mail (SMTP) server hostname for sending DWH e-mails *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* myserver.consol. de *Since:* 6.1.0
cmas-dwh-server	notification.password	*Description:* Password for sending DWH e-mails (optional) *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Since:* 6.1.0
cmas-dwh-server	notification.port	*Description:* SMTP port for sending DWH e-mails *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* 25 *Since:* 6.1.0
cmas-dwh-server	notification.protocol	*Description:* The protocol used for sending e-mails from DWH. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* pop3\
cmas-dwh-server	notification.username	*Description:* (SMTP) User name for sending DWH e-mails *Type:* String

Module	Property	Explanation
		Restart required: No *System:* Yes *Optional:* Yes *Example value:* maz *Since:* 6.1.0
cmas-workflow-jbpm	outdated.lock.age	*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
cmweb-server-adapter	pagemapLockDurationInSeconds	*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
cmweb-server-adapter	postActivityExecutionScriptName	*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
cmas-nimh	queue.polling.threads.number	*Description:* Number of threads started for mails' queue polling.

Module	Property	Explanation
		Default: 1 *Type:* Integer *Restart required:* No *System:* No *Optional:* Yes *Example value:* 1 *Since:* 6.4.0
cmas-nimh	queue.polling.threads.shutdown. timeout.seconds	*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
cmas-nimh	queue.polling.threads.watchdog. interval.seconds	*Description:* Watchdog thread interval. Default: 30 *Type:* Integer *Restart required:* No *System:* No *Optional:* Yes *Example value:* 30 *Since:* 6.4.0
cmas-nimh	queue.task.error.pause.seconds	*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
cmas-nimh	queue.task.interval.seconds	*Description:* Main mails' queue polling thread interval. Default: 15 *Type:* Integer *Restart required:* No *System:* No

Module	Property	Explanation
		Optional: Yes *Example value:* 15 *Since:* 6.4.0
cmas-nimh	queue.task.max.retries	*Description:* Maximum number of e-mail processing retries after an exception. When reached, the e-mail is moved to the e-mail archive. This e-mail 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
cmas-nimh	queue.task.timeout.seconds	*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
cmas-nimh	queue.task.transaction.timeout. seconds	*Description:* Transaction timeout for mail processing in the pipe. Default: 60 *Type:* Integer *Restart required:* No *System:* No *Optional:* Yes *Example value:* 60 *Since:* 6.4.0
cmweb-server-adapter	queuesExcludedFromGS	*Description:* Comma-separated list of queue names which are excluded from Quick Search. *Type:* String *Restart required:* No

Module	Property	Explanation
		System: Yes *Optional:* Yes *Since:* 6.0
cmas-workflow-jbpm	refreshTimeInCaseOfConcurrent RememberMeRequests	*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
cmweb-server-adapter	rememberMeLifetimeInMinutes	*Description:* Lifetime for *remember me* in minutes *Type:* Integer *Restart required:* Yes *System:* Yes *Optional:* No *Example value:* 1440 *Since:* 6.0
cmweb-server-adapter	request.scope.transaction	*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
cmas-setup-scene	scene	*Description:* Scene file which was imported during set-up (can

Module	Property	Explanation
		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
cmweb-server-adapter	searchPageSize	*Description:* Default page size for search results *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 20 *Since:* 6.0
cmweb-server-adapter	searchPageSizeOptions	*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
cmas-core-server	server.session.archive.reaper. interval	*Description:* Server archived sessions' reaper interval (in seconds) *Type:* Integer *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* 60 *Since:* 6.7.1
cmas-core-server	server.session.archive.timeout	*Description:* Server sessions archive validity timeout (in days).

Module	Property	Explanation
		After this time session info is removed from db. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 31 *Since:* 6.7.1
cmas-core-server	server.session.reaper.interval	*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
cmas-core-server	server.session.timeout	*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 *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 1800 *Since:* 6.6.1, 6.7.1
cmweb-server-adapter	serverPoolingInterval	*Description:* *Type:* Integer *Restart required:* No *System:* Yes

Module	Property	Explanation
		Optional: No *Example value:* 5 *Since:* 6.1.0
cmas-dwh-server	skip-ticket	*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
cmas-dwh-server	skip-ticket-history	*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
cmas-dwh-server	skip-unit	*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
cmas-dwh-server	skip-unit-history	*Description:* History of unit is not transferred during transfer /update. *Type:* Boolean *Restart required:* No *System:* Yes *Optional:* No *Example value:* false

Module	Property	Explanation
		Since: 6.6.19 *Removed in:* 6.8.1
cmas-core-server	skip.wfl.transfer.cleanup	*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
cmas-dwh-server	split.history	*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
cmas-app-admin-tool	start.groovy.task.enabled	*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
cmweb-server-adapter	supportEmail	*Description:* *Type:* String *Restart required:* No

Module	Property	Explanation
		System: Yes *Optional:* Yes *Since:* 6.0
cmas-core-index-common	synchronize.master.address	*Description:* Value of *-Dcmas. http.host.port* informing how to connect to indexing master server. Default null. Since 6.6.17 this value is configurable in set-up to designate initial indexing master server. Please note that changing this value is only allowed when all cluster nodes index changes receivers are stopped. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* 127.0.0.1:80 *Since:* 6.6.0
cmas-core-index-common	synchronize.master.security. token	*Description:* The password for accessing the index snapshot via URL, e.g. for index synchronization or for back-ups. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* token *Since:* 6.6.0
cmas-core-index-common	synchronize.master.security.user	*Description:* The user name for accessing the index snapshot via URL, e.g. for index synchronization or for back-ups. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* user *Since:* 6.6.0
cmas-core-index-common		

Module	Property	Explanation
	synchronize.master.timeout. minutes	*Description:* How much time master server may constantly fail until new master gets elected with index fix procedure. Default 5. Since 6.6.17 this value is configurable in set-up where zero means that master server will never change (failover mechanism is off). *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 5 *Since:* 6.6.0
cmas-core-index-common	synchronize.megabits.per. second	*Description:* How much bandwidth can master server consume to transfer index changes to all slave servers. Default 85. Please do not use all available bandwidth to transfer index changes between hosts. This will most probably partition cluster as some subsystems will not be able to communicate. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 85 *Since:* 6.6.0
cmas-core-index-common	synchronize.sleep.millis	*Description:* How often each slave server polls master server for index changes. Default 1000. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 1000 *Since:* 6.6.0
cmweb-server-adapter	themeOverlay	*Description:* Name of used theme overlay

Module	Property	Explanation
		Type: String *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* kyoEUR *Since:* 6.0
cmas-core-server	ticket.delete.timeout	*Description:* Transaction timeout (in seconds) for deleting tickets *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 60 *Since:* 6.1.3
cmweb-server-adapter	ticketListRefreshIntervalInSeconds	*Description:* Refresh interval for ticket list (in seconds) *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 180 *Since:* 6.0
cmweb-server-adapter	ticketListSizeLimit	*Description:* Maximum number of tickets in ticket list *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 100 *Since:* 6.0
cmas-core-server	tickets.delete.size	*Description:* Property that 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

Module	Property	Explanation
cmas-core-server	unit.replace.batchSize	*Description:* Describes number of objects to be processed in unit replace action. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 5 *Since:* 6.8.2
cmas-core-server	unit.replace.timeout	*Description:* Transaction timeout (seconds) of unit replacement action step. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 120 *Since:* 6.8.2
cmas-dwh-server	unit.transfer.order	*Description:* Define in which order Data Object 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
cmweb-server-adapter	unitIndexSearchResultSizeLimit	*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
cmas-core-server	unused.content.remover.cluster.node.id	*Description:* Value of a *cmas. clusternode.id* designating node

Module	Property	Explanation
		which 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 -Dcmas.clusternode.id=1 parameter) *Since:* 6.9.0.0
cmas-core-server	unused.content.remover.enabled	*Description:* Flag whether unused ticket attachments and unit content entries removal should take place. *Type:* Boolean *Restart required:* No *System:* Yes *Optional:* No *Example value:* true *Since:* 6.9.0.0
cmas-core-server	unused.content.remover.polling.minutes	*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
cmas-core-server	unused.content.remover.ttl.minutes	*Description:* Minimum interval 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

Module	Property	Explanation
cmweb-server-adapter	urlLogoutPath	*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
cmas-core-server	warmup.executor.enabled	*Description:* Flag whether server should asynchronously warm up during startup (e.g. fill some of internal caches) *Type:* Boolean *Restart required:* No *System:* Yes *Optional:* No *Example value:* true *Since:* 6.9.4.2
cmweb-server-adapter	webSessionTimeoutInMinutes	*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
cmweb-server-adapter	wicketAjaxRequestHeaderFilterEnabled	*Description:* This enables filter for Wicket AJAX requests, coming from stale pages with Wicket 1.4 scripting (CM6 pre-6.8.0), after update to CM6 post-6.8.0. *Type:* Boolean *Restart required:* Yes *System:* Yes

Module	Property	Explanation

Module	Property	Explanation
		Optional: Yes *Example value:* false *Since:* 6.8.1

51 Appendix D (Important System Properties, Ordered by Area of Application)

- CMRF & DWH Configuration
- Indexer & Search Configuration
  - Indexer
  - Search Results
- LDAP Configuration
  - LDAP Configuration (If LDAP Is Used as Authentication Mode in the CM Web Client)
  - LDAP Configuration (If LDAP Is Used as Authentication Mode in CM.Track)
- E-Mail Configuration
  - Outgoing E-Mail
  - Incoming E-Mail
    - Settings for ESB/Mule
    - Settings for NIMH
    - Mapping of Former Mule and New NIMH Properties
    - Attachments for Incoming E-Mails
  - E-Mail Encryption (Outgoing and Incoming)
- Activity Interval Configuration

51.1 CMRF & DWH Configuration

Module	Property	Explanation
cmas-dwh-server	autocommit.cf.changes	*Description:* *Type:* Boolean *Restart required:* No *System:* Yes *Optional:* No *Example value:* false *Since:* 6.7.0
cmas-dwh-server	batch-commit-interval	*Description:* Number of objects in a JMS message. Higher value means better transfer performance and bigger memory usage. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* 100 *Since:* 6.0.0
cmas-dwh-server	communication.channel	*Description:* Communication channel. Possible values are *DIRECT* (database communication channel, default value since 6.9.4.1), *JMS* (default value before 6.9.4.1). Before 6.9.4.1 it has to be manually added. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* DIRECT *Since:* 6.8.5.0
cmas-dwh-server	dwh.mode	*Description:* Current mode of DWH data transfer. Possible values are *OFF*, *ADMIN*, *LIVE*. *Type:* String *Restart required:* No *System:* Yes *Optional:* No

Module	Property	Explanation
		Example value: OFF *Since:* 6.0.1
cmas-dwh-server	ignore-queues	*Description:* By adding a comma separated list of queue names it is configured that tickets of these queues are 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
cmas-dwh-server	is.cmrf.alive	*Description:* As a starting point time of sending last message to CMRF should be used. If response from CMRF is not received after value (in seconds) it should create a DWH operation status with error message that CMRF is down. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 1200 *Since:* 6.7.0
cmas-dwh-server	java.naming.factory.initial	*Description:* Factory class for DWH context factory. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* org.jnp. interfaces.NamingContextFactory *Since:* 6.0.1
cmas-dwh-server	java.naming.factory.url.pkgs	*Description:* *Type:* String *Restart required:* No

Module	Property	Explanation
		System: Yes *Optional:* No *Example value:* org.jboss. naming:org.jnp.interfaces *Since:* 6.0.1
cmas-dwh-server	java.naming.provider.url	*Description:* URL of naming provider. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* localhost *Since:* 6.0.1
cmas-dwh-server	notification.error.description	*Description:* Text for error e-mails from DWH. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* Error occurred *Since:* 6.0.1
cmas-dwh-server	notification.error.from	*Description:* FROM-address for error e-mails from DWH. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Since:* 6.0.1
cmas-dwh-server	notification.error.subject	*Description:* Subject for error e-mails from DWH. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* Error occurred *Since:* 6.0.1
cmas-dwh-server	notification.error.to	*Description:* TO-address for error e-mails from DWH. *Type:* String *Restart required:* No *System:* Yes

Module	Property	Explanation
		Optional: No *Example value:* myuser@consol. de *Since:* 6.0.1
cmas-dwh-server	notification.finished_successfully. description	*Description:* Text for e-mails from DWH when transfer finished successfully. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* Transfer finished successfully *Since:* 6.0.1
cmas-dwh-server	notification.finished_successfully. from	*Description:* FROM-address for e-mails from DWH when transfer finished successfully. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Since:* 6.0.1
cmas-dwh-server	notification.finished_successfully. subject	*Description:* Subject for e-mails from DWH when transfer finished successfully. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* Transfer finished successfully *Since:* 6.0.1
cmas-dwh-server	notification.finished_successfully. to	*Description:* TO-address for e-mails from DWH when transfer finished successfully. *Restart required:* Yes *System:* Yes *Optional:* No *Example value:* myuser@consol. de *Since:* 6.0.1

Module	Property	Explanation
cmas-dwh-server	notification. finished_unsuccessfully. description	*Description:* Text for e-mails from DWH when transfer finished unsuccessfully. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* Transfer finished unsuccessfully *Since:* 6.0.1
cmas-dwh-server	notification. finished_unsuccessfully.from	*Description:* FROM-address for e-mails from DWH when transfer finished unsuccessfully. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Since:* 6.0.1
cmas-dwh-server	notification. finished_unsuccessfully.subject	*Description:* Subject for e-mails from DWH when transfer finished unsuccessfully. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* Transfer finished unsuccessfully *Since:* 6.0.1
cmas-dwh-server	notification. finished_unsuccessfully.to	*Description:* TO-address for e-mails from DWH when transfer finished unsuccessfully. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* myuser@consol. de *Since:* 6.0.1
cmas-dwh-server	notification.host	*Description:* E-mail (SMTP) server hostname for sending DWH e-mails.

Module	Property	Explanation
		Type: String *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* mail.consol.de *Since:* 6.1.0
cmas-dwh-server	notification.password	*Description:* Password for sending DWH e-mails (optional). *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Since:* 6.1.0
cmas-dwh-server	notification.port	*Description:* SMTP port for sending DWH e-mails. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* 25 *Since:* 6.1.0
cmas-dwh-server	notification.protocol	*Description:* The protocol used for sending e-mails from DWH. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* pop3\
cmas-dwh-server	notification.username	*Description:* (SMTP) User name for sending DWH e-mails. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* myuser *Since:* 6.1.0
cmas-dwh-server	skip-ticket	*Description:* Tickets are not transferred during transfer /update. *Type:* Boolean *Restart required:* No

Module	Property	Explanation
		System: Yes *Optional:* No *Example value:* false *Since:* 6.6.19 *Removed in:* 6.8.1
cmas-dwh-server	skip-ticket-history	*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
cmas-dwh-server	skip-unit	*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
cmas-dwh-server	skip-unit-history	*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
cmas-dwh-server	split.history	*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

Module	Property	Explanation
		Restart required: No *System:* Yes *Optional:* Yes *Example value:* false *Since:* 6.8.0
cmas-dwh-server	unit.transfer.order	*Description:* Define in which order Data Object 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

51.2 Indexer & Search Configuration

51.2.1 Indexer

Module	Property	Explanation
cmas-core-index-common	big.task.minimum.size	*Description:* How many parts a task at least should have to be handled by the Indexer with low priority. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 15 (default) *Since:* 6.8.3
cmas-core-index-common	database.notification.enabled	*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
cmas-core-index-common	database.notification.redelivery. delay.seconds	*Description:* In case of index update database notification channel, indicates notification redelivery delay when exception occurs. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 60 *Since:* 6.8.4.7
cmas-core-index-common	database.notification.redelivery. max.attempts	*Description:* In case of index update database notification channel, indicates maximum

Module	Property	Explanation
		redelivery attempts when exception occurs. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 60 *Since:* 6.8.4.7
cmas-core-index-common	disable.admin.task.auto.commit	*Description:* All tasks created for index update will be executed automatically right after creation. *Type:* Boolean *Restart required:* No *System:* Yes *Optional:* No *Example value:* false *Since:* 6.6.1
cmas-core-index-common	index.attachment	*Description:* Describes if content of attachments is indexed. *Type:* Boolean *Restart required:* No *System:* Yes *Optional:* No *Example value:* true *Since:* 6.4.3
cmas-core-index-common	index.history	*Description:* Describes if unit and ticket history are indexed. *Type:* Boolean *Restart required:* No *System:* Yes *Optional:* No *Example value:* false *Since:* 6.1.0
cmas-core-index-common	index.status	*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

Module	Property	Explanation
		Example value: GREEN *Since:* 6.6.1
cmas-core-index-common	index.task.worker.threads	*Description:* How many threads will be used to execute batch 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
cmas-core-index-common	index.version.current	*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
cmas-core-index-common	index.version.newest	*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
cmas-core-index-common	indexed.assets.per.thread.in.memory	*Description:* How many assets should be loaded into memory at once during indexing per one thread. *Type:* Integer *Restart required:* No *System:* Yes

Module	Property	Explanation
		Optional: No *Example value:* 200 (default) *Since:* 6.8.0
cmas-core-index-common	indexed.engineers.per.thread.in.memory	*Description:* How many engineers should be loaded into memory at once during indexing per one thread. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 300 (default) *Since:* 6.6.14, 6.7.3
cmas-core-index-common	indexed.tickets.per.thread.in.memory	*Description:* How many tickets should be loaded into memory at once during indexing per one thread. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 100 (default) *Since:* 6.6.14, 6.7.3
cmas-core-index-common	indexed.units.per.thread.in.memory	*Description:* How many units should be loaded into memory at once during indexing per one thread. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 200 (default) *Since:* 6.6.14, 6.7.3
cmas-core-index-common	synchronize.master.address	*Description:* Value of *-Dcmas.http.host.port* informing 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

Module	Property	Explanation

Module	Property	Explanation
		only allowed when all cluster nodes index changes receivers are stopped. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* 127.0.0.1:80 *Since:* 6.6.0
cmas-core-index-common	synchronize.master.security. token	*Description:* The password for accessing the index snapshot via URL, e.g. for index synchronization or for back-ups. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* token *Since:* 6.6.0
cmas-core-index-common	synchronize.master.security.user	*Description:* The user name for accessing the index snapshot via URL, e.g. for index synchronization or for back-ups. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* user *Since:* 6.6.0
cmas-core-index-common	synchronize.master.timeout. minutes	*Description:* How much time master server may constantly fail until new master gets elected with index fix procedure. Default 5. Since 6.6.17 this value is configurable in set-up where zero means that master server will never change (fail-over mechanism is off). *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No

Module	Property	Explanation
		Example value: 5 *Since:* 6.6.0
cmas-core-index-common	synchronize.megabits.per.second	*Description:* How much bandwidth can master server consume to transfer index changes to all slave servers. Default 85. Please do not use all available bandwidth to transfer index changes between hosts. This will most probably partition the cluster as some subsystems will not be able to communicate. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 85 *Since:* 6.6.0
cmas-core-index-common	synchronize.sleep.millis	*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

51.2.2 Search Results

Module	Property	Explanation
cmweb-server-adapter	globalSearchResultSizeLimit	*Description:* Maximum number of items in Quick Search result. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 10 *Since:* 6.0

Module	Property	Explanation
cmweb-server-adapter	unitIndexSearchResultSizeLimit	*Description:* Maximum number of units in unit search result (e.g. when searching for contacts). *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 5 *Since:* 6.0
cmweb-server-adapter	searchPageSize	*Description:* Default page size for search results. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 20 *Since:* 6.0
cmweb-server-adapter	searchPageSizeOptions	*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

51.3 LDAP Configuration

51.3.1 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*.

Module	Property	Explanation
cmas-core-security	authentication.method	*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

Module	Property	Explanation
cmas-core-security	ldap.authentication	*Description:* Authentication method used when using LDAP authentication. *Type:* String *Restart required:* Yes *System:* Yes *Optional:* No *Example value:* simple *Since:* 6.0
cmas-core-security	ldap.basedn	*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
cmas-core-security	ldap.initialcontextfactory	

Module	Property	Explanation
		Description: Class name for initial context factory of LDAP implementation when using LDAP authentication. If it is not set, *com.sun.jndi.ldap. LdapCtxFactory* is being used as a value. *Type:* String *Restart required:* Yes *System:* Yes *Optional:* No *Example value:* com.sun.jndi. ldap.LdapCtxFactory *Since:* 6.0
cmas-core-security	ldap.password	*Description:* Password for connecting to LDAP to look up users (when using LDAP authentication). Only needed if look-up cannot be done anonymously. *Type:* Password *Restart required:* No *System:* Yes *Optional:* Yes *Since:* 6.1.2
cmas-core-security	ldap.providerurl	*Description:* LDAP provider (when using LDAP authentication). *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* ldap://ldap. consol.de:389 *Since:* 6.0
cmas-core-security	ldap.searchattr	*Description:* Search attribute for looking up LDAP entry connected to CM login. *Type:* String *Restart required:* No *System:* Yes *Optional:* No

Module	Property	Explanation
		Example value: uid *Since:* 6.0
cmas-core-security	ldap.userdn	*Description:* LDAP user for connecting to LDAP to look up users (when using LDAP authentication). Only needed if look-up cannot be done anonymously. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Since:* 6.1.2

51.3.2 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*.

Module	Property	Explanation
cmas-core-security	contact.authentication.method	*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

Module	Property	Explanation
cmas-core-security	ldap.contact.name.basedn	*Description:* Base path to search for contact DN by LDAP ID (e.g. ou=accounts,dc=consol,dc=de). *Type:* String *Restart required:* No

Module	Property	Explanation
		System: No *Optional:* Yes *Since:* 6.9.3.0
cmas-core-security	ldap.contact.name.password	*Description:* Password to look up contact DN by LDAP ID. If not set, anonymous account is used. *Type:* String *Restart required:* No *System:* No *Optional:* Yes *Since:* 6.9.3.0
cmas-core-security	ldap.contact.name.providerurl	*Description:* Address of the LDAP server (ldap[s]://host:port). *Type:* String *Restart required:* No *System:* No *Optional:* Yes *Since:* 6.9.3.0
cmas-core-security	ldap.contact.name.searchattr	*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
cmas-core-security	ldap.contact.name.userdn	*Description:* User DN to look up contact DN by LDAP ID. If not set, anonymous account is used. *Type:* String *Restart required:* No *System:* No *Optional:* Yes *Since:* 6.9.3.0
cmas-core-security	ldap.initialcontextfactory	*Description:* Class name for initial context factory of LDAP implementation when using LDAP authentication. If it is not set, *com.sun.jndi.ldap. LdapCtxFactory* is being used as a value.

Module	Property	Explanation
		Type: String *Restart required:* Yes *System:* Yes *Optional:* No *Example value:* com.sun.jndi.ldap.LdapCtxFactory *Since:* 6.0

51.4 E-Mail Configuration

51.4.1 Outgoing E-Mail

Independent of incoming e-mail mode (ESB/Mule and NIMH).

Module	Property	Explanation
cmas-core-server	mail.smtp.email	*Description:* SMTP e-mail URL for outgoing e-mails. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* smtp://mail.consol.de:25 *Since:* 6.0
cmas-core-server	mail.smtp.envelopesender	*Description:* E-mail address used as sender in SMTP envelope. If not set, the FROM-address of the e-mail is used. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* mysender@mydomain.com *Since:* 6.5.7

Module	Property	Explanation
cmweb-server-adapter	mail.from	*Description:* Use this address if set instead of engineer e-mail address during e-mail conversation. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Since:* 6.1.2
cmweb-server-adapter	mail.reply.to	*Description:* When set, Web Client will display REPLY-TO-

Module	Property	Explanation
		field on e-mail sent, prefilled with this value. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Since:* 6.0.1 ⊖ Please see also section Queue Administration. When you set the REPLY-TO-address in the outgoing e-mail script, the *mail.reply.to* system property must not be set (because it would overwrite the configured value)! That means when you use one outgoing e-mail script for a queue you have to define outgoing e-mail scripts for all queues because the *mail.reply.to* property can no longer be used.
cmweb-server-adapter	mailTemplateAboveQuotedText	*Description:* Indicates behavior of e-mail template in the Ticket E-Mail Editor when another e-mail is quoted, i.e. forwarded or replied to. *Type:* Boolean *Restart required:* No *System:* Yes *Optional:* No *Example value:* false *Since:* 6.2.4

Module	Property	Explanation
cmas-workflow-jbpm	mail.sender.address	*Description:* FROM-address for e-mails 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

51.4.2 Incoming E-Mail

Settings for ESB/Mule

Module	Property	Explanation
cmas-esb-core	esb.directory	*Description:* Directory used by ESB (Mule). *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* C:\Users\user\cmas\mule *Since:* 6.0
cmas-esb-mail	mail.attachments.validation.info.sender	*Description:* Sets FROM-header of attachments type error notification e-mail. As a default the e-mail address of the administrator which you have entered during system set-up is used. *Type:* String *Restart required:* No *System:* Yes *Optional:* No

Module	Property	Explanation
		Example value: admin@consolcm.com *Since:* 6.7.5
cmas-esb-mail	mail.attachments.validation.info. subject	*Description:* Sets subject of attachments type error notification e-mail. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* E-mail was not processed because its attachments were rejected!!! *Since:* 6.7.5
cmas-esb-mail	mail.callname.pattern	*Description:* Regular expression for subject of incoming e-mails. Available as TICKET_NAME_PATTERN_FORMAT in incoming e-mail scripts. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* .*?Ticket\s+\((\S+)\).* *Since:* 6.0
cmas-esb-mail	mail.cluster.node.id	*Description:* Only the node whose mail.cluster.node.id equals cmas.clusternode.id will start the Mule ESB mail services. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* unspecified *Since:* 6.6.5
cmas-esb-mail	mail.db.archive	*Description:* If property is set to *true*, incoming e-mails are archived in the database. *Type:* Boolean *Restart required:* No

Module	Property	Explanation
		System: Yes *Optional:* Yes *Example value:* false (default) *Since:* 6.8.5.5
cmas-esb-mail	mail.delete.read	*Description:* Determines whether CM deletes messages fetched via IMAP(S). Setting value to *true* will cause deletion of messages after fetching. Default is to not delete messages fetched via IMAP(S). Note: Messages fetched via POP3(S) will always be deleted. *Type:* Boolean *Restart required:* No *System:* Yes *Optional:* No *Example value:* true *Since:* 6.7.3
cmas-esb-mail	mail.encryption	*Description:* If property is set to *true*, the encrypt check box in the Ticket E-Mail Editor is checked by default. *Type:* Boolean *Restart required:* No *System:* Yes *Optional:* No *Example value:* true (default = false) *Since:* 6.8.4.0
cmas-esb-mail	mail.incoming.uri	*Description:* URL for incoming e-mails. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* pop3://cm-incoming-user: password@localhost:10110 *Since:* 6.0

Module	Property	Explanation
		This value should not be edited here using the system properties pop-up window, but the mailboxes should be configured using the navigation item E-mail. Using this standard feature all entries are controlled - i.e. for each mailbox which is added, CM establishes a test connection during mailbox set-up. That way it is not possible to enter wrong values.
cmas-esb-mail	mail.max.restarts	*Description:* Maximum number of e-mail service restarts before giving up. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 3 *Since:* 6.0
cmas-esb-mail	mail.mime.strict	*Description:* If set to *false*, e-mail addresses are not parsed for strict MIME compliance. Default is *true*, which means check for strict MIME compliance. *Type:* Boolean *Restart required:* No *System:* Yes *Optional:* No *Example value:* false *Since:* 6.6.17, 6.7.3
cmas-esb-mail	mail.mule.service	*Description:* FROM-address for e-mails sent by Mule service. *Type:* EMail

Module	Property	Explanation
		Restart required: No *System:* Yes *Optional:* No *Example value:* myuser@consol. de *Since:* 6.0
cmas-esb-mail	mail.polling.interval	*Description:* Mail polling interval in ms. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 60000 *Since:* 6.0
cmas-esb-mail	mail.process.error	*Description:* TO-address for error e-mails from Mule. As a default the e-mail address of the administrator which you have entered during system set-up is used. *Type:* EMail *Restart required:* No *System:* Yes *Optional:* No *Example value:* myuser@consol. de *Since:* 6.0
cmas-esb-mail	mail.process.retry.attempts	*Description:* Number of retries when processing e-mail. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 3 *Since:* 6.0.2
cmas-esb-mail	mail.process.timeout	*Description:* Mail processing timeout in seconds. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No

Module	Property	Explanation
		Example value: 60 *Since:* 6.1.3
cmas-esb-mail	mail.redelivery.retry.count	*Description:* Indicates the number of retries of re-delivering an e-mail from the CM system. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 3 *Since:* 6.1.0

Settings for NIMH

Those settings apply if NIMH is enabled (and therefore ESB/Mule is disabled):

Module	Property	Explanation
cmas-core-server	nimh.enabled	*Description:* Enables *NIMH* service. Must be suffixed with nodeid in cluster, e.g. *nimh.enabled.NODEID = true* *Type:* Boolean *Restart required:* No *System:* No *Optional:* Yes *Example value:* true *Since:* 6.9.4.0

Module	Property	Explanation
cmas-nimh	filesystem.polling.threads. number	*Description:* Number of threads started for db e-mails' queue polling. Default: 1 *Type:* Integer *Restart required:* No *System:* No *Optional:* Yes *Example value:* 10 *Since:* 6.4.0
cmas-nimh	filesystem.polling.threads. shutdown.timeout.seconds	*Description:* Waiting time after the shutdown signal. When the

Module	Property	Explanation
		timeout is reached, thread will be terminated. Default: 60 *Type:* Integer *Restart required:* No *System:* No *Optional:* Yes *Example value:* 60 *Since:* 6.4.0
cmas-nimh	filesystem.polling.threads. watchdog.interval.seconds	*Description:* Watchdog thread interval. Default: 30 *Type:* Integer *Restart required:* No *System:* No *Optional:* Yes *Example value:* 60 *Since:* 6.4.0
cmas-nimh	filesystem.task.enabled	*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
cmas-nimh	filesystem.task.interval.seconds	*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
cmas-nimh	filesystem.task.polling.folder	*Description:* Polling folder location which will be scanned for e-mails in the format of *eml* files. Default: "mail" subdir of cmas data directory. *Type:* String

Module	Property	Explanation
		Restart required: No *System:* No *Optional:* Yes *Example value:* c://cmas//mail *Since:* 6.4.0
cmas-nimh	filesystem.task.timeout.seconds	*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
cmas-nimh	filesystem.task.transaction. timeout.seconds	*Description:* Default transaction timeout for e-mail 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
cmas-nimh	mailbox.1.connection.host	Host (server) for first configured mailbox. Will overwrite the default parameter *mailbox. default.connection.host.*
cmas-nimh	mailbox.1.connection.password	Password for first configured mailbox. Will overwrite the default parameter *mailbox. default.connection.password.*
cmas-nimh	mailbox.1.connection.port	Port for first configured mailbox. Will overwrite the default parameter *mailbox.default. connection.port.*

Module	Property	Explanation
cmas-nimh	mailbox.1.connection.protocol	Protocol (e.g. IMAP or POP3) for first configured mailbox. Will overwrite the default parameter *mailbox.default.connection. protocol*.
cmas-nimh	mailbox.1.connection.username	User name for first configured mailbox. Will overwrite the default parameter *mailbox. default.connection.username*.
cmas-nimh	mailbox.2.connection.host	Host (server) for second configured mailbox. Will overwrite the default parameter *mailbox.default.connection.host*.
cmas-nimh	mailbox.2.connection.password	Password for second configured mailbox. Will overwrite the default parameter *mailbox. default.connection.password*.
cmas-nimh	mailbox.2.connection.port	Port for second configured mailbox. Will overwrite the default parameter *mailbox. default.connection.port*.
cmas-nimh	mailbox.2.connection.protocol	Protocol (e.g., IMAP or POP3) for second configured mailbox. Will overwrite the default parameter *mailbox.default. connection.protocol*.
cmas-nimh	mailbox.2.connection.username	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.

Module	Property	Explanation
cmas-nimh	mailbox.default.connection.host	*Description:* Host (server name) of a given mailbox from which

Module	Property	Explanation
		the poller reads e-mails. *Type:* String *Restart required:* No *System:* No *Optional:* Yes *Example value:* 10.10.1.157 *Since:* 6.4.0
cmas-nimh	mailbox.default.connection. password	*Description:* Password for given mailbox from which the poller reads e-mails. *Type:* String *Restart required:* No *System:* No *Optional:* Yes *Example value:* consol *Since:* 6.4.0
cmas-nimh	mailbox.default.connection.port	*Description:* Port for a given mailbox from which the poller reads e-mails. *Type:* String *Restart required:* No *System:* No *Optional:* Yes *Example value:* 143 *Since:* 6.4.0
cmas-nimh	mailbox.default.connection. protocol	*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
cmas-nimh	mailbox.default.connection. username	*Description:* User name for a given mailbox from which the poller reads e-mails. *Type:* String *Restart required:* No *System:* No *Optional:* Yes

Module	Property	Explanation
		Example value: username *Since:* 6.4.0
cmas-nimh	mailbox.default.session.mail. debug	*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
cmas-nimh	mailbox.default.session.mail. imap.timeout	*Description:* Example *javax.mail* property. *Type:* Integer *Restart required:* No *System:* No *Optional:* Yes *Example value:* 120 *Since:* 6.4.0
cmas-nimh	mailbox.default.session.mail. mime.address.strict	*Description:* Example *javax.mail* property. Counterpart of the old Mule *mail.mime.strict*, allows to set not so strict mail header parsing. *Type:* Boolean *Restart required:* No *System:* No *Optional:* Yes *Example value:* true *Since:* 6.4.0
cmas-nimh	mailbox.default.session.mail. pop3.timeout	*Description:* Example *javax.mail* property. *Type:* *Restart required:* *System:* *Optional:* *Example value:* *Since:* 6.4.0
cmas-nimh	mailbox.default.task.delete.read. messages	*Description:* This defines whether messages should be removed from the mailbox after

Module	Property	Explanation
		processing. For IMAP protocol messages are marked as *SEEN* by default. For POP3 protocol, if 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
cmas-nimh	mailbox.default.task.enabled	*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
cmas-nimh	mailbox.default.task.interval. seconds	*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
cmas-nimh	mailbox.default.task.max. message.size	*Description:* Maximum size of e-mail messages in bytes. Messages with a size larger than the value of this property cannot be processed by the Admin Tool. Default set to 10 MB: 10485760 *Type:* Integer *Restart required:* No *System:* No *Optional:* Yes

Module	Property	Explanation
		Example value: 10485760 *Since:* 6.4.0
cmas-nimh	mailbox.default.task.max. messages.per.run	*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
cmas-nimh	mailbox.default.task.timeout. seconds	*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
cmas-nimh	mailbox.default.task.transaction. timeout.seconds	*Description:* Default transaction timeout for e-mail 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
cmas-nimh	mailbox.polling.threads.mail.log. enabled	*Description:* Enables e-mail logging which is especially crucial in cluster environments (used as semaphore there). *Type:* String

Module	Property	Explanation
		Restart required: No *System:* No *Optional:* Yes *Example value:* true (default) *Since:* 6.9.4.1
cmas-nimh	mailbox.polling.threads.number	*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
cmas-nimh	queue.polling.threads.number	*Description:* Number of threads started for e-mails' queue polling. Default: 1 *Type:* Integer *Restart required:* No *System:* No *Optional:* Yes *Example value:* 1 *Since:* 6.4.0
cmas-nimh	queue.polling.threads.shutdown.timeout.seconds	*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
cmas-nimh	queue.polling.threads.watchdog.interval.seconds	*Description:* Watchdog thread interval. Default: 30 *Type:* Integer *Restart required:* No *System:* No *Optional:* Yes *Example value:* 30 *Since:* 6.4.0

Module	Property	Explanation
cmas-nimh	queue.task.error.pause.seconds	*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
cmas-nimh	queue.task.interval.seconds	*Description:* Main e-mails' queue polling thread interval. Default: 15 *Type:* Integer *Restart required:* No *System:* No *Optional:* Yes *Example value:* 15 *Since:* 6.4.0
cmas-nimh	queue.task.max.retries	*Description:* Maximum number of e-mail processing retries after an exception. When reached, the e-mail is moved to the e-mail archive. This e-mail can be rescheduled again using the NIMH API (or the Admin Tool). *Type:* Integer *Restart required:* No *System:* No *Optional:* Yes *Example value:* 10 *Since:* 6.4.0
cmas-nimh	queue.task.timeout.seconds	*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

Module	Property	Explanation
		Optional: Yes *Example value:* 600 *Since:* 6.4.0
cmas-nimh	queue.task.transaction.timeout.seconds	*Description:* Transaction timeout for e-mail processing in the pipe. Default: 60 *Type:* Integer *Restart required:* No *System:* No *Optional:* Yes *Example value:* 60 *Since:* 6.4.0
cmas-nimh-extension	mail.attachments.validation.info.sender	*Description:* Sets FROM-header of attachments type error notification e-mail. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* admin@mail.com *Since:* 6.7.5 ⓘ This is an equivalent to the old *cmas-esb-mail*, *mail.attachments.validation.info.sender*
cmas-nimh-extension	mail.attachments.validation.info.subject	*Description:* Sets subject of attachments type error notification e-mail. *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* Mail was not processed because its attachments were rejected!!! *Since:* 6.7.5

Module	Property	Explanation

Module	Property	Explanation
		This is an equivalent to the old *cmas-esb-mail*, *mail.attachments. validation.info.subject*
cmas-nimh-extension	mail.db.archive	*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 This is an equivalent to the old *cmas-esb-mail*, *mail.db.archive*
cmas-nimh-extension	mail.error.from.address	This is an equivalent to the old *cmas-esb-mail*, *mail.mule.service*
cmas-nimh-extension	mail.error.to.address	This is an equivalent to the old *cmas-esb-mail*, *mail.process.error*
cmas-nimh-extension	mail.mule.service	*Description:* FROM-address for e-mails sent by Mule service. *Type:* EMail *Restart required:* No *System:* Yes *Optional:* No *Example value:* myuser@consol.

Module	Property	Explanation
		de *Since:* 6.4.0
cmas-nimh-extension	mail.on.error	*Description:* If set to *true* an error e-mail is sent to the above configured address in case the e-mail message could not be processed. Default: false *Type:* Boolean *Restart required:* No *System:* No *Optional:* Yes *Example value:* false *Since:* 6.4.0
cmas-nimh-extension	mail.process.error	*Description:* TO-address for error e-mails from Mule. *Type:* EMail *Restart required:* No *System:* Yes *Optional:* No *Example value:* myuser@consol.de *Since:* 6.4.0
cmas-nimh-extension	mail.ticketname.pattern	*Description:* *Type:* String *Restart required:* No *System:* Yes *Optional:* No *Example value:* .*?Ticket\s+\((\S+)\).* *Since:* 6.4.0

Mapping of Former Mule and New NIMH Properties

Mule property	NIMH property
cmas-esb-mail, mail.delete.read	cmas-nimh, mailbox.default.task.delete.read.messages
cmas-esb-mail, mail.polling.interval	cmas-nimh, mailbox.default.task.interval.seconds
cmas-esb-mail, mail.process.retry.attempts	cmas-nimh, queue.task.max.retries

Mule property	NIMH property
cmas-esb-mail, mail.mime.strict	cmas-nimh, mailbox.default.session.mail.mime.address.strict
cmas-esb-mail, mail.encryption	cmas-core-server, mail.encryption (moved to core server properties)
cmas-esb-mail, mail.callname.pattern	cmas-nimh-extension, mail.ticketname.pattern
cmas-esb-mail, mail.attachments.validation.info.sender	cmas-nimh-extension, mail.attachments.validation.info.sender
cmas-esb-mail, mail.attachments.validation.info.subject	cmas-nimh-extension, mail.attachments.validation.info.subject
cmas-esb-mail, mail.db.archive	cmas-nimh-extension, mail.db.archive
cmas-esb-mail, mail.mule.service	cmas-nimh-extension, mail.error.from.address
cmas-esb-mail, mail.process.error	cmas-nimh-extension, mail.error.to.address

Attachments for Incoming E-Mails

These settings apply to Mule/ESB and NIMH.

Module	Property	Explanation
cmas-core-server	attachment.allowed.types	*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
cmas-core-server	attachment.max.size	*Description:* Maximum attachment size in MB. *Type:* Integer *Restart required:* No *System:* Yes *Optional:* No *Example value:* 100 *Since:* 6.4.0

51.4.3 E-Mail Encryption (Outgoing and Incoming)

These settings only apply if e-mail encryption is active (*true*). This is valid for ESB Mail and NIMH.

Module	Property	Explanation
cmas-core-server	mail.encryption	*Description:* If property is set to *true*, the encrypt check box in the Ticket E-Mail 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:

Module	Property	Explanation
cmas-core-server	ldap.certificate.basedn	*Description:* Base DN for certificates location in LDAP tree. If not provided, *cmas-core-security*, *ldap.basedn* is taken. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* ou=accounts, dc=consol,dc=de *Since:* 6.8.4
cmas-core-server	ldap.certificate.content.attribute	*Description:* LDAP attribute name used where certificate data is stored in LDAP tree. Default value is *usercertificate*. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* usercertificate *Since:* 6.8.4
cmas-core-server	ldap.certificate.password	*Description:* LDAP certificates manager password. If not set, *cm*

Module	Property	Explanation
		as-core-security, *ldap.password* is taken. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Since:* 6.8.4
cmas-core-server	ldap.certificate.providerurl	*Description:* LDAP certificates provider URL. If not set, *cmas-core-security*, *ldap.providerurl* is taken. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* ldap://ldap.consol.de:389 *Since:* 6.8.4
cmas-core-server	ldap.certificate.searchattr	*Description:* LDAP attribute name used to search for certificate in LDAP tree. Default value is *mail*. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Example value:* mail *Since:* 6.8.4
cmas-core-server	ldap.certificate.userdn	*Description:* LDAP certificates manager DN. If not set, *cmas-core-security*, *ldap.userdn* is taken. *Type:* String *Restart required:* No *System:* Yes *Optional:* Yes *Since:* 6.8.4

51.5 Activity Interval Configuration

Module	Property	Explanation
cmas-app-admin-tool	admin.tool.session.check. interval	*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
cmas-core-server	server.session.timeout	*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 *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.

52 Appendix E - Trademarks

- 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® 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® 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® 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

- 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

- MuleSoft[TM] and Mule ESB[TM] are among the trademarks of MuleSoft, Inc. See Mule Soft web 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

Index

D

E

F

G

U

V

W