ConSol Software GmbH

ConSol CM Process Designer Manual

Version 6.12.0.0

Contents

Contents 2
A - Introduction 8
A.1 ConSol CM for business process management 9
A.2 List of manuals 10
A.3 TecDoc server 11
A.4 About this manual 12
A.4.1 Introduction 12
A.4.2 Structure of this manual 12
A.4.3 ConSol CM configuration covered in this manual 13
A.5 Legal notice 14
A.6 Gender disclaimer 14
A.7 Copyright 14
A.8 Layout explanations 15
B - Workflow design in ConSol CM 16
B.1 Business processes 16
B.2 Process modeling in ConSol CM 17
B.3 The ConSol CM Process Designer at a glance 18
B.3.1 Workflows 19
B.3.2 Tickets and activities 20
B.3.3 Drag-and-drop modeling of workflow components 21
B.3.4 Scopes and nesting of scopes 22
B.3.5 Modeling escalation mechanisms (triggers and wait states) 22
B.3.6 Modeling interrupts and exceptions 23
B.3.7 Scripting capabilities 24
B.3.8 Versioning of workflows 24
B.4 Objects used in processes 26
B.4.1 General objects 27
B.4.2 Data fields 30
B.4.3 Standard ticket fields 31

B.5 Working with the Process Designer 32
B.5.1 Steps to perform to create a new process 32
B.5.2 Starting the Process Designer 32
B.6 Process Designer GUI 34
B.6.1 GUI elements 35
B.6.2 Editing scripts 45
B.6.3 GUI tips and tricks 46
C - Workflow components 47
C.1 Start nodes 48
C.1.1 Properties of a start node 48
C.2 End nodes 50
C.2.1 Possible actions for closed tickets 50
C.2.2 Actions not possible for closed tickets 50
C.2.3 Properties of an end node 51
C.3 Scopes 54
C.3.1 Introduction to scopes 55
C.3.2 Defining a new scope 57
C.3.3 Properties of a scope 58
C.3.4 Scopes and views 59
C.3.5 Scope sort index 59
C.4 Activities 60
C.4.1 Introduction to activities 61
C.4.2 Properties of an activity 64
C.4.3 Process logic of activities 70
C.4.4 Examples of activities 72
C.5 Reopen activities 81
C.5.1 Introduction to reopen activities 82
C.5.2 Properties of a reopen activity 83
C.5.3 Process logic of reopen activities 89
C.5.4 Example for the use of reopen activities 90

CM/ is a registered trademark of ConSol Consulting & Solutions Software GmbH

C.6 Decision nodes 93
C.6.1 Introduction to decision nodes 93
C.6.2 Properties of a decision node 94
C.6.3 Example of a decision node 96
C.7 Adornments (triggers and ACFs) 99
C.7.1 Time triggers 100
C.7.2 Mail triggers 113
C.7.3 Business event triggers 122
C.7.4 Activity Control Forms (ACFs) 137
C.8 Jump-out and jump-in nodes 148
C.8.1 Introduction 148
C.8.2 Jump-out nodes 151
C.8.3 Jump-in nodes 154
D - Workflow programming 157
D.1 Programming CM scripts 159
D.1.1 Java vs. Groovy-style coding 159
D.2 CM API documentation 160
D.3 CM script types in workflows 161
D.4 Script interactions 161
D.5 Scripts in ConSol CM in general 162
D.6 Process logic 163
D.6.1 Introduction 164
D.6.2 Activities 164
D.6.3 Interrupts and exceptions 168
D.6.4 Loops (errors in workflows) 170
D.6.5 Process logic of time triggers 170
D.6.6 Process logic of business event triggers 170
D.7 Important classes and objects 171
D.7.1 Introduction 171
D.7.2 Important objects 171
D.7.3 Convenience classes and methods 172

CM/ is a registered trademark of ConSol Consulting & Solutions Software GmbH

D.8 Working with data fields 176
D.8.1 Introduction to data fields 177
D.8.2 Data types for data fields 177
D.8.3 Details about string fields: use annotations to fine-tune strings 180
D.8.4 Data fields for ticket data: ticket fields 182
D.8.5 Data fields for customer data: customer fields 193
D.8.6 Most important methods for accessing customer data 195
D.8.7 Resource data 200
D.8.8 Using data fields for (invisible) variables 202
D.9 Sending emails 203
D.9.1 Introduction to sending emails 203
D.9.2 Important methods 204
D.9.3 Examples 206
D.9.4 Effects of engineer representation rules for emails sent in scripts 216
D.10 Working with path information 219
D.10.1 Introduction 219
D.10.2 Retrieve path information for a workflow element 220
D.10.3 Example for the use of path information 220
D.11 Working with calendars and times 221
D.11.1 Introduction 221
D.11.2 Calculating with dates and times without a business calendar 222
D.11.3 Calculating with dates and times using a business calendar 222
D.12 Working with object relations 224
D.12.1 Working with ticket relations 225
D.12.2 Working with customer relations 233
D.12.3 Working with resource relations 239
D.13 Working with text classes 242
D.13.1 Introduction 242
D.13.2 Examples 244
D.14 Working with attachments 249
D.14.1 Introduction 249

CM/ is a registered trademark of ConSol Consulting & Solutions Software GmbH

D.14.2 Example: Attaching all attachments of a ticket to the child ticket 249
D.15 Working with overlays 254
D.16 Searching for tickets, customers and resources 255
D.16.1 Introduction 255
D.16.2 Searching for tickets 256
D.16.3 Searching for customers (contacts and companies) 262
D.16.4 Searching for resources 264
D.17 Displaying messages and notifications 266
D.17.1 Messages 266
D.17.2 Notifications 269
D.18 Debug information 271
D.18.1 Introduction 271
D.18.2 Log file configuration 271
D.18.3 Using statements for debug output 272
E - Workflow deployment 274
E.1 Introduction and workflow life cycle 275
E.2 Engineer permissions required for workflow deployment 276
E.3 Actions during workflow deployment 277
F - Best practices 279
F.1 How to use scopes for organizing the workflow 280
F.1.1 Variant A: Using a global scope 281
F.1.2 Variant B: Using three or more main scopes 283
F.2 Where to position the start node 285
F.3 Where to save scripts 286
F.3.1 When to use Admin Tool workflow scripts 287
F.3.2 How to use Admin Tool workflow scripts 288

F.4 How to optimize trigger combinations 289
F.5 How to avoid unneeded ticket update events 292
F.6 How to use the disable auto update parameter 293
F.7 How to avoid self-triggering business event triggers 295
F.8 Which components to use with time triggers 296
G - Appendix 297
G.1 Annotations 298
G.1.1 List of Field Annotations 299
G.1.2 List of Group Annotations 314
G.2 System Properties 318
G.2.1 Alphabetical List of System Properties 319
G.2.2 List of System Properties by Module 436
G.2.3 List of System Properties by Area 541
G.3 Default Java Imports 583
G.4 List of Code Examples 587
G.5 Trademarks 591
G.6 Glossary 593

A - Introduction

This chapter discusses the following:

A.1 ConSol CM for business process management 9
A.2 List of manuals 10
A.3 TecDoc server 11
A.4 About this manual 12
A.4.1 Introduction 12
A.4.2 Structure of this manual 12
A.4.3 ConSol CM configuration covered in this manual 13
A.5 Legal notice 14
A.6 Gender disclaimer 14
A.7 Copyright 14
A.8 Layout explanations 15

A.1 ConSol CM for business process management

ConSol CM is a process management platform, especially suited for use as customer service software.

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

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

Figure 1: *Overview of potential fields of use of the BPM software ConSol CM*

Using ConSol CM, you can handle all components which are relevant in business processes to represent and control your company's processes in an optimal way. ConSol CM is used in various different industries and branches ranging from insurances and banks over fashion designing companies to producers of ticket vending machines or car washes. The flexible process designing mechanism and workflow engine provide a perfect basis for the modeling and controlling of business processes, especially customer service processes, of different kinds.

CM/ is a registered trademark of ConSol Consulting & Solutions Software GmbH

A.2 List of manuals

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

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

CM/ is a registered trademark of ConSol Consulting & Solutions Software GmbH

A.3 TecDoc server

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

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

CM/ is a registered trademark of ConSol Consulting & Solutions Software GmbH

A.4 About this manual

A.4.1 Introduction

When you read this manual, your company is presumably using ConSol CM as a business process management tool and it is your job to manage the system and to implement your company's processes in the application. The manual will help you to understand the principles of ConSol CM workflows and to learn the work with the Process Designer. Numerous tips and tricks provided by our experienced consultants will help you to find the best way to improve your processes.

Before you start work with the Process Designer you should have a profound knowledge of ConSol CM administration, because programming CM workflows requires the usage of several CM components which are configured before (or while) the workflow development takes place. So please read the *ConSol CM Administrator Manual* first.

A.4.2 Structure of this manual

This manual includes several sections which cover the different aspects of designing workflows with the ConSol CM Process Designer.

- The Introduction contains general information about ConSol CM and this manual.
- Workflow design in ConSol CM provides an overview of how business processes are modeled in the Process Designer and how to work with the Process Designer application.
- Workflow components describes the components which are used to create workflows.
- Workflow programming describes how to use the different workflow components in scripts. It includes many real-life examples.
- Workflow deployment describes the workflow life cycle.
- Best practices contains tips and tricks for workflow design.
- The Appendix includes lists of system properties, annotations, code examples, trademarks and a glossary.

A.4.3 ConSol CM configuration covered in this manual

ConSol CM offers various possibilities to implement customer- and system-specific features in order to provide a BPM system which is precisely tailored to your needs as a customer. A considerable part of your CM system's "intelligence" is based on scripting. However, the scripts are not all located in or connected with the Process Designer. In the Process Designer, scripts are stored which are part of a workflow. All other scripts are stored in the Admin Tool. Even some workflow scripts can be stored there.

Therefore, in this manual, you will find explanations about

- the graphical design of workflows using the Process Designer
- scripting for workflow scripts (The respective topics are marked in light blue in the following graphic.)

For detailed explanations of all other scripts, please refer to the *ConSol CM Administrator Manual*.

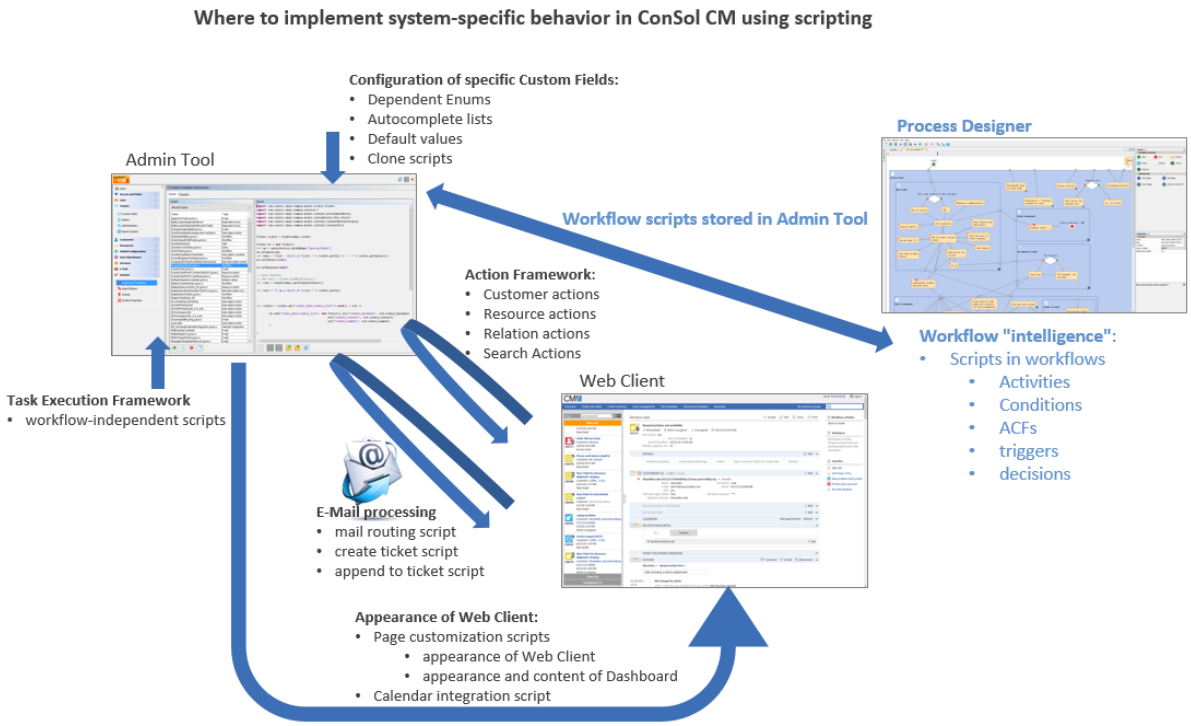


Figure 2: *Scripts which are used to customize ConSol CM and their locations within the system*

A.5 Legal notice

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

A.6 Gender disclaimer

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

A.7 Copyright

© 2019 ConSol Consulting & Solutions Software GmbH - All rights are reserved.

A.8 Layout explanations

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

This is an additional information.

This is an important note. Be careful here!

This is a warning!

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

CM/ is a registered trademark of ConSol Consulting & Solutions Software GmbH

B - Workflow design in ConSol CM

B.1 Business processes

In a business process, a certain number of tasks have to be performed in a defined order to achieve a specific goal.

The following components are (usually) relevant in business processes. Please see section Objects used in processes to gain an overview of the ConSol CM objects which represent those components.

- **Process**
  This is a collection of tasks which have to be performed in a certain order. Tasks might be serialized or performed in a parallel way. In ConSol CM, the process is modeled by one or more workflows. ConSol CM can model single processes and can also manage complex process chains.
  Each process has to have a defined input and a defined output. The object which represents a case and which runs through the process is a *ticket*. For the end user, it can be named *Ticket* or *Case* or any other required term.
- **Roles and responsibilities**
  Usually, the persons who work in a process represent different roles, i.e. different responsibilities. In ConSol CM each engineer, i.e. each person who works with the system, can have one or more roles.
- **Access permissions**
  A business process management system can control various processes in a company. Therefore the assignment and control of access permissions is a core functionality. In ConSol CM, the access permissions are assigned to roles.
- **Customer**
  This is the person who has an interest in the outcome of the process. In ConSol CM, there is always one main customer for a ticket. This can be a person, i.e. a contact, or this can be a company. More customers can be added.
- **Tasks**
  Every task, which might also be called case, request, order or whatever seems to be suitable for the respective use case, is treated as a ticket in ConSol CM. In a business process, there might be several kinds of activities which can be performed for a ticket:
  - manual activities
  - system-aided activities
  - fully automatic activities

  ConSol CM can manage all types of tasks. For manual tasks, there are to-do lists for the engineer and several mechanisms which guarantee that no task will be forgotten or ignored.

B.2 Process modeling in ConSol CM

One of the core components of ConSol CM is a powerful workflow engine. Hence, a process is represented in ***ConSol CM*** by a **workflow**. This is the technical representation of the consecutive steps which are required to fulfill all steps which should be performed during the business process.

Examples:

In an IT helpdesk environment, a workflow could consist of the steps:
New Ticket - Accept Ticket - Work on Solution - Inform Customer - Close Ticket.

In a sales process these steps could be:
First Contact: Lead - Second Contact: Opportunity - Contract Candidate - Contract.

The workflow containing all required steps runs in a workflow engine. In this manual you will get to know the details about all components of a workflow and how to use them to build the workflow which represents your business process.

A workflow ...

- represents a specific process, e.g. the steps that have to be performed to handle a customer request.
- puts activities and decisions in a defined order.
- defines the possible paths a ticket can take.

The case or request which has to be dealt with is represented by a **ticket**, i.e., this is the object which passes through the workflow.

The following picture shows the graphical representation of a simple help desk process.

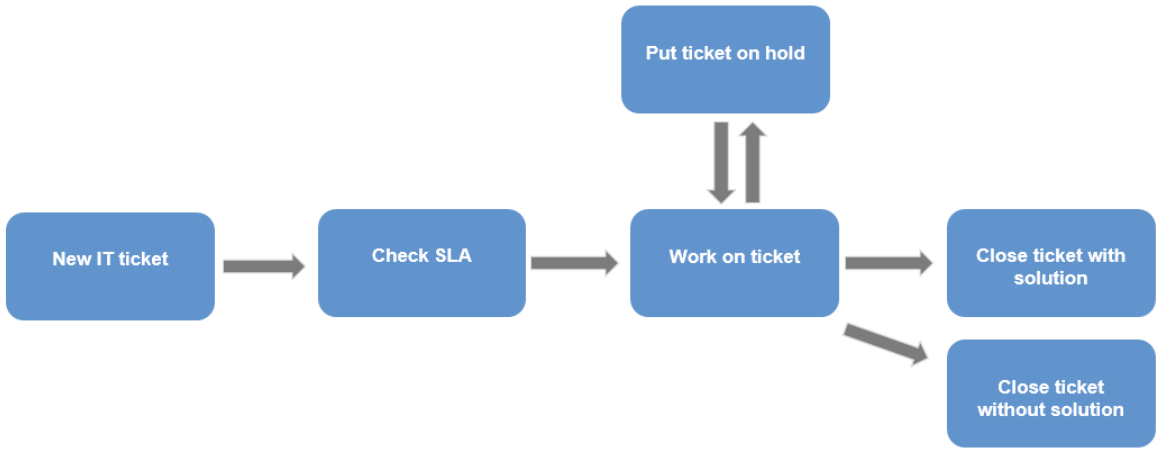


Figure 3: *Graphical representation of a simple business process (IT helpdesk)*

The figure above shows a rather strictly controlled business process. As an engineer, you cannot decide what to do - you cannot react in a dynamic, flexible way when a certain action is required (e.g., you have to start an admin ticket from the existing ticket for some reasons), but you have to follow the process steps. This might be intended to guarantee a unique process and defined result for all cases of a certain kind. On the other hand, there might be teams where a more flexible way of reacting is required and neither the process flow nor the outcome can be completely standardized. For this, ConSol CM in versions 6.11 and higher offers a functionality (see section *Scope Activities*) which covers (adaptive) case management.

B.3 The ConSol CM Process Designer at a glance

This chapter discusses the following:

B.3.1 Workflows 19
B.3.2 Tickets and activities 20
B.3.3 Drag-and-drop modeling of workflow components 21
B.3.4 Scopes and nesting of scopes 22
B.3.5 Modeling escalation mechanisms (triggers and wait states) 22
B.3.6 Modeling interrupts and exceptions 23
B.3.7 Scripting capabilities 24
B.3.8 Versioning of workflows 24

CM/ is a registered trademark of ConSol Consulting & Solutions Software GmbH

B.3.1 Workflows

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, i.e. you use the Process Designer to develop workflows.

> In ConSol CM terminology, a **workflow** always represents the technical entity, whereas a **process** represents the business process from the logical or management point of view.

One of the Process Designer's advantages is that there is no procedural gap between process design and workflow implementation. You can design a workflow for a process using the graphical interface of the Process Designer and as soon as you have assigned the workflow to a queue and have defined roles and users, the process comes alive and engineers can work with it. That means you can use the Process Designer for both steps which are of importance when you want to create IT-supported business processes:

- Model and design the process from a logical point of view
- Implement the process in a technical instance

Due to this flexibility, you can start with a simple version of a workflow, usually in a test environment, and develop the desired functionalities of the process using an iterative approach. In each step of the development and optimization process the team of engineers can test if the use cases are represented as desired.

The graphical representation of a workflow in the Process Designer is very similar to the **Business Process Model and Notation** (BPMN) and can be handled in a very intuitive way.

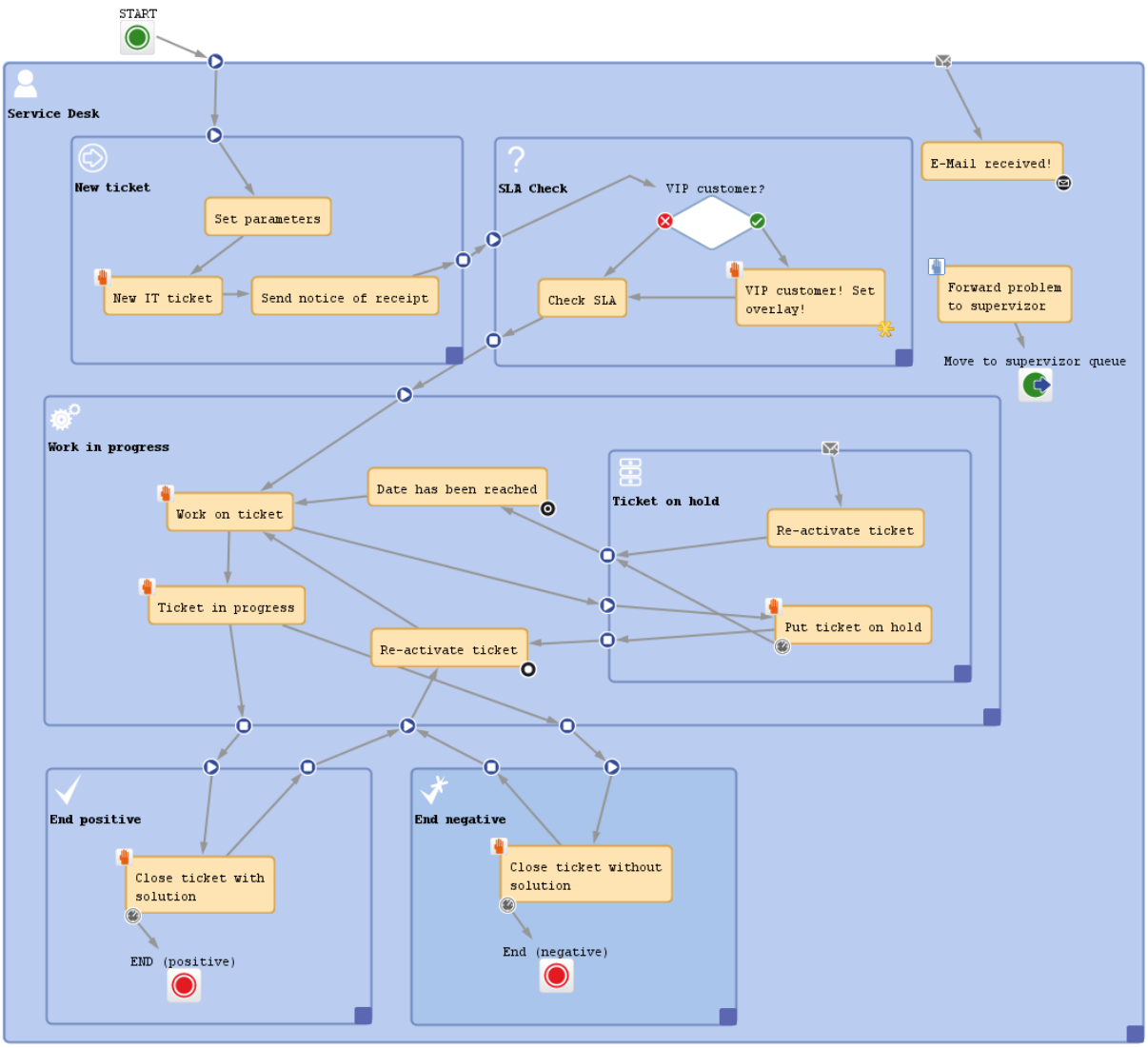


Figure 4: *ConSol CM Process Designer - Workflow modeling, example process*

Read the following sections to get a first impression of the Process Designer's features and functionalities. All topics will be explained in detail in the respective chapters of the manual.

B.3.2 Tickets and activities

Each case, which has to be treated, will be represented by a **ticket**. Thus a ticket is a concrete run through a workflow. This can be a request, an order, or any other task which has to be processed in a business process.

> Please note that the term which is displayed in the Web Client does not necessarily have to be *ticket*. Using labels which are configured in the Admin Tool, every other term is possible, e.g. *case*, *service request*, *opportunity* ... whatever is required in your company's business processes.
>
> In the Process Designer and in the Admin Tool, as well as in the ConSol CM API however, the term *ticket* will still be used!

When a new ticket is created within ConSol CM, it is associated with a workflow (via the queue it belongs to). At first the new ticket is in the START node. During its further life cycle the ticket runs through the various activities of the workflow. An activity is the smallest entity of a workflow and represents a single step within the business process. The life cycle of a ticket ends when it has reached an END node.

You model a process in a workflow by connecting activities in a specific order. The result is a directed flow graph. It shows which activities have to be carried out for a ticket in order to run through the workflow (and thus the business process) successfully. Workflows can have branches so that different flow paths are possible. In this way, you can make sure that, for example, a ticket first has to be accepted, then the problem has to be solved, then the solution has to be documented. Only then the ticket can be closed.

If you apply ConSol CM in case management, there might be (scope) activities which are not placed in a defined order.

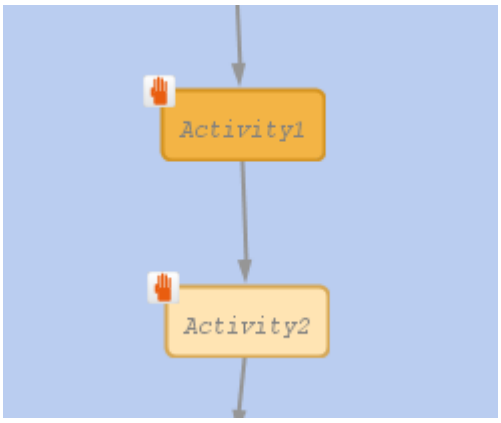


Figure 5: *ConSol CM Process Designer - Two sequential manual activities*

There are manual and automatic activities. Manual activities require engineer interaction and are offered as *Workflow activities* in the Web Client. In contrast, automatic activities are performed without any human input and are kept away from the engineer. This enables ConSol CM to save time for the engineer and to process data from various sources behind the scenes. Only when user interaction is required, the process will come to a halt and wait for engineer input.

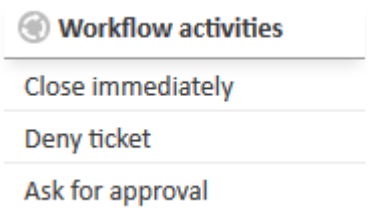


Figure 6: *ConSol CM Web Client - Workflow activities*

B.3.3 Drag-and-drop modeling of workflow components

You can develop your workflow easily and intuitively using drag-and-drop. Drag the required workflow elements, e.g., an activity or a decision node, from the palette to the work space and link them. Then adjust the properties of the elements within the Properties Editor.

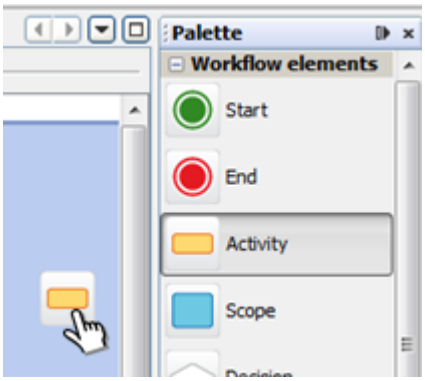


Figure 7: *ConSol CM Process Designer - Drag-and-drop activities*

Using basic elements you build complex workflows step by step. In this way you can model even the most sophisticated business processes.

B.3.4 Scopes and nesting of scopes

During a process, a ticket passes through different status, e.g. new ticket, pre-qualification, active work, and documentation. It might even have to be set on hold for a certain period of time. All those status are represented by scopes. In each scope, there can be one or more activities. In this way, it is easy to develop workflows with a clear structure. Scopes can even be organized in a hierarchical way, e.g. during documentation the ticket has to be set on hold. So, using hierarchical scopes you can even keep track of complicated processes. Choose the level of detail you need any time you want.

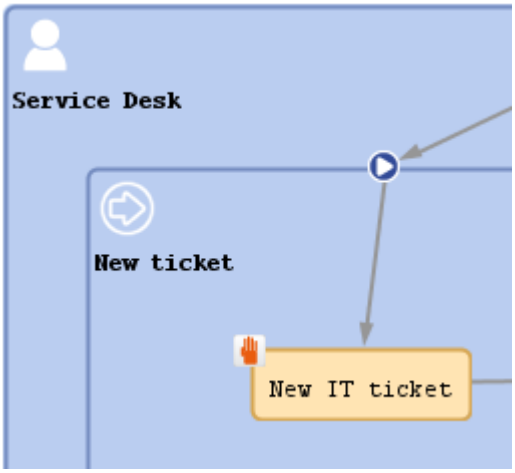


Figure 8: *ConSol CM Process Designer - Nesting scopes*

B.3.5 Modeling escalation mechanisms (triggers and wait states)

In most business processes, adherence to schedules and deadlines is indispensable. ConSol CM helps stick to deadlines and prevents delays by providing automatic timer triggers. These triggers measure for example the reaction time or they initialize reminders.

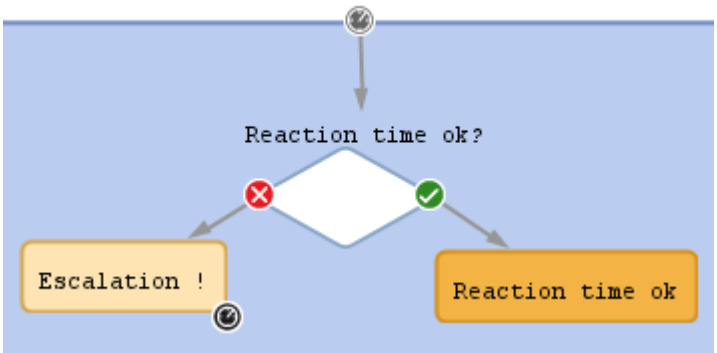


Figure 9: *ConSol CM Process Designer - Triggering processes*

B.3.6 Modeling interrupts and exceptions

In the real world, tasks of a process are not always performed step by step, but may be interrupted by exceptional events. These can be various external incidents. To model such interrupts sequentially is often very complex or even impossible. The Process Designer provides extensive tools to do this.

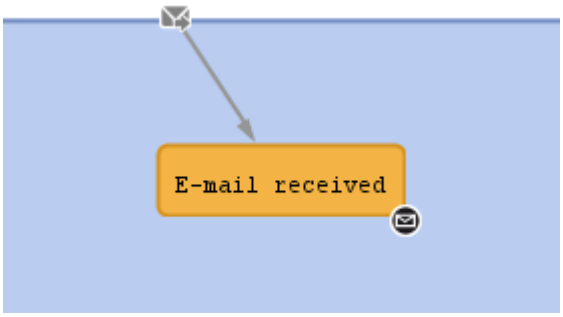


Figure 10: *ConSol CM Process Designer - Modeling interrupts*

B.3.7 Scripting capabilities

The process which has been modeled as a ConSol CM workflow does only consist of basic elements like activities or decision nodes. In every node of the workflow a script can be added to provide the "intelligence" of the process. For example, emails can be sent to customers or to engineers, interactions with other systems can be implemented, tickets can be handed-over. Basically, all operations which can be implemented in Groovy scripts can be performed.

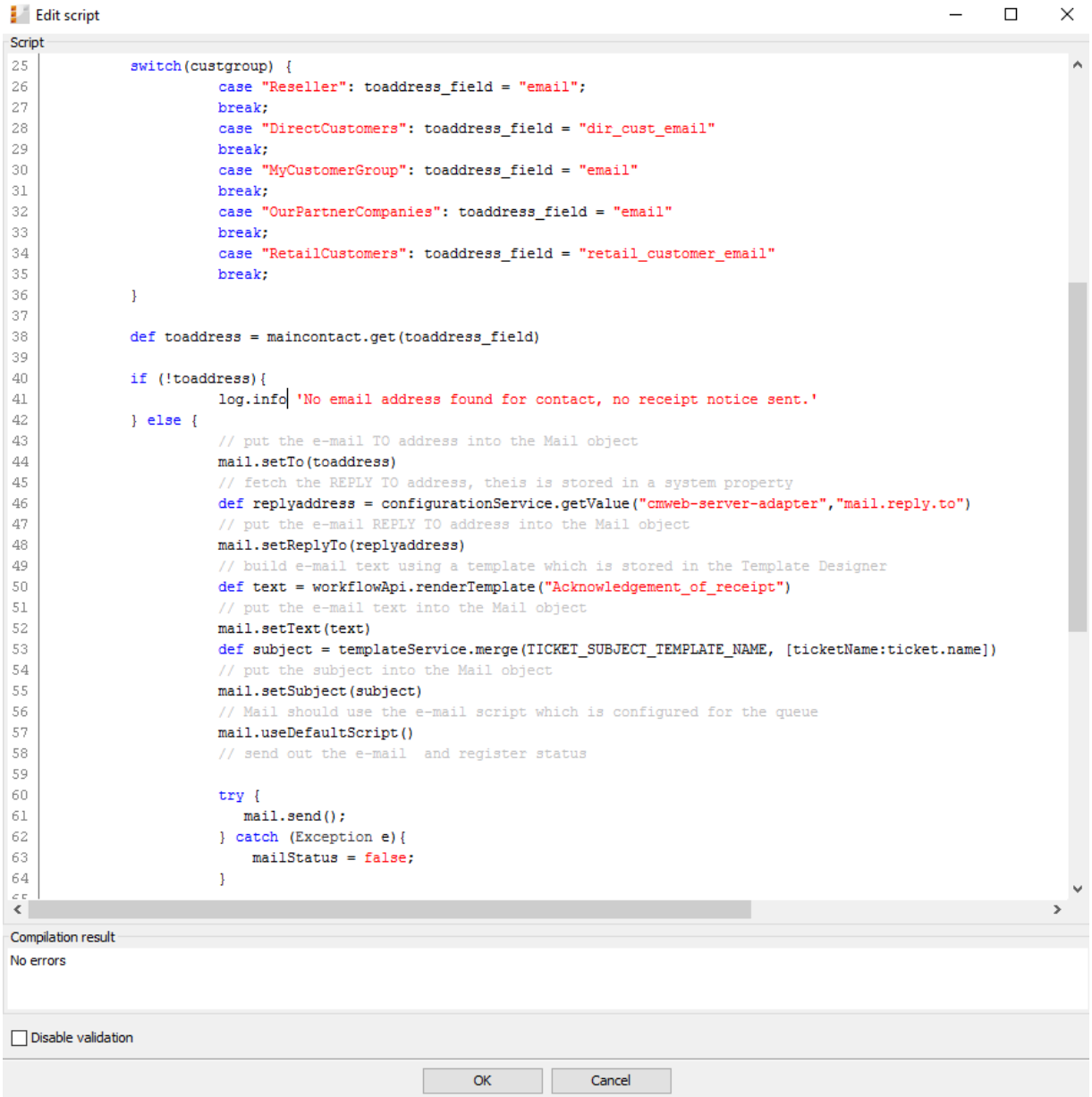
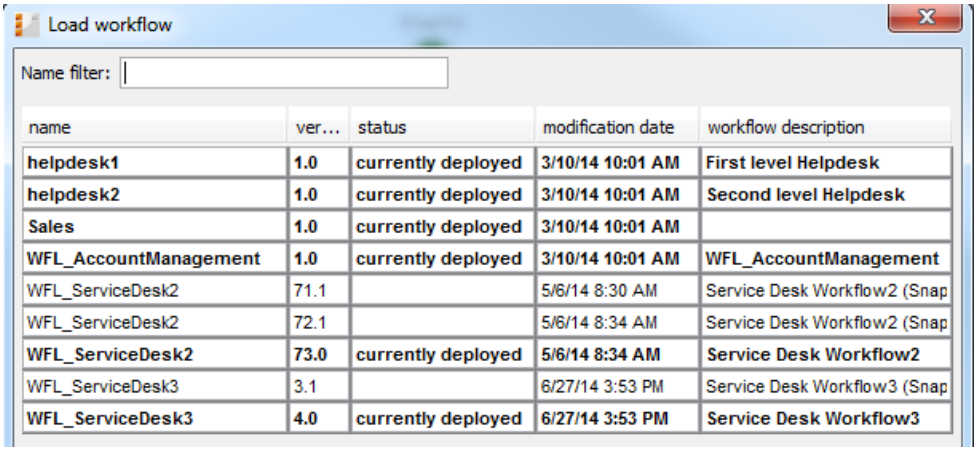


Figure 11: *ConSol CM Process Designer - Script of an activity*

B.3.8 Versioning of workflows

Business processes are changing constantly, following the changing requirements of the economic and technical environment. The Process Designer provides continuous versioning of installed workflows. In this way, you can easily discard a new workflow (e.g. when you have tested a new implementation during system development) and go back to one of the previous versions.



name	ver...	status	modification date	workflow description
helpdesk1	**1.0**	**currently deployed**	**3/10/14 10:01 AM**	**First level Helpdesk**
helpdesk2	**1.0**	**currently deployed**	**3/10/14 10:01 AM**	**Second level Helpdesk**
Sales	**1.0**	**currently deployed**	**3/10/14 10:01 AM**	
WFL_AccountManagement	**1.0**	**currently deployed**	**3/10/14 10:01 AM**	**WFL_AccountManagement**
WFL_ServiceDesk2	71.1		5/6/14 8:30 AM	Service Desk Workflow2 (Snap
WFL_ServiceDesk2	72.1		5/6/14 8:34 AM	Service Desk Workflow2 (Snap
WFL_ServiceDesk2	**73.0**	**currently deployed**	**5/6/14 8:34 AM**	**Service Desk Workflow2**
WFL_ServiceDesk3	3.1		6/27/14 3:53 PM	Service Desk Workflow3 (Snap
WFL_ServiceDesk3	**4.0**	**currently deployed**	**6/27/14 3:53 PM**	**Service Desk Workflow3**

Figure 12: *ConSol CM Process Designer - Workflow versions*

B.4 Objects used in processes

This chapter discusses the following:

B.4.1 General objects 27

B.4.2 Data fields 30

B.4.3 Standard ticket fields 31

B.4.1 General objects

During process design and workflow development you will have to deal mainly with the following objects:

Mandatory objects:

- **Ticket**
  This represents the case. Depending on the use case this can be, for example, a help desk case, a sales opportunity, a direct order, or a service request.

  > Please note that the term which is displayed in the Web Client does not necessarily have to be *ticket*. Using labels which are configured in the Admin Tool, every other term is possible, e.g. *case*, *service request*, *opportunity* ... whatever is required in your company's business processes.
  >
  > In the Process Designer and in the Admin Tool, as well as in the ConSol CM API however, the term *ticket* will still be used!

- **Main customer**
  The customer is the person (contact) or company who has the question or service request. This person or company is the main customer of the ticket. This represents the external side of the CM system.
- **Queue**
  This is the organizing unit within the ConSol CM system which groups tickets of one realm and which is access point for the assignment of access permissions and of the workflow. One queue has exactly one workflow which cannot be changed. For example, in a company, there could be one queue for the sales department, one for the customer service, and one for the internal IT.
- **Engineer**
  This is the person who is responsible for completing the tasks in the ticket. A ConSol CM engineer has a login and password for the Web Client. The main engineer can also be called the ticket owner. It can change during the process. The engineer represents the internal side of the system.

  > Please note that the term which is displayed in the Web Client does not necessarily have to be *engineer*. Using labels which are configured in the Admin Tool, every other term is possible, e.g. *agent*, *service agent*, *assignee* ... whatever is required in your company's business processes.
  >
  > In the Process Designer and in the Admin Tool, as well as in the ConSol CM API however, the term *engineer* will still be used!

- **Resource**
  This represents an object in the CM database section which represents the CM/Resource Pool. The object can represent an asset, a contract, a person, a product or any other entity, depending on the design of the Resource Pool. CM/Resource Pool is a ConSol CM Add-on which is not part of the standard CM distribution.
- **Workflow**
  This is the design or model for the process. A workflow is assigned to a queue (and can be assigned to more than one queues). Hence, all tickets which are in this queue run through the

process defined by this workflow. The workflow elements, e.g. activities, conditions, or decisions, represent the most important means in ConSol CM to configure and control the process flow. One workflow can be assigned to one or to several queues, e.g. the IT service desk team as well as the customer service team, both could work with the workflow *serviceWorkflow*.

- **Ticket fields**
  These are the data fields which are used to define the data model for the ticket data only. They also determine the GUI design of the Web Client. Ticket fields are never defined on a single-field basis, but always in **ticket field groups**.
- **Customer fields**
  These are the data fields which are used to define the data model for the customer data. They also determine the GUI design of the Web Client.
  Customer fields are never defined on a single-field basis, but always in **customer field groups**.
- **Resource fields** (only if CM/Resource Pool is active)
  These are the data fields which are used to define the data model for the resource data. They also determine the GUI design of the Web Client.
  Resource fields are never defined on a single-field basis, but always in **resource field groups**.

Optional objects:

- **One or more additional customer(s)**
  In addition to the main customer, i.e. main contact or main company, more contacts (or companies) can be added to a ticket. For each additional customer a customer role might be assigned. For example, there might be a representative for someone who has opened the ticket or the team manager should also be a contact for a support case. An additional customer can become the main customer during the process and vice versa.
- **One or more additional engineer(s)**
  Additional engineers can be added to a ticket in specific roles which are defined as required. For example, a supervisor might be set as additional engineer to give an approval (role *approver*) or a QA team member can be added to the ticket in the role *QA* to check the result before the ticket is closed.

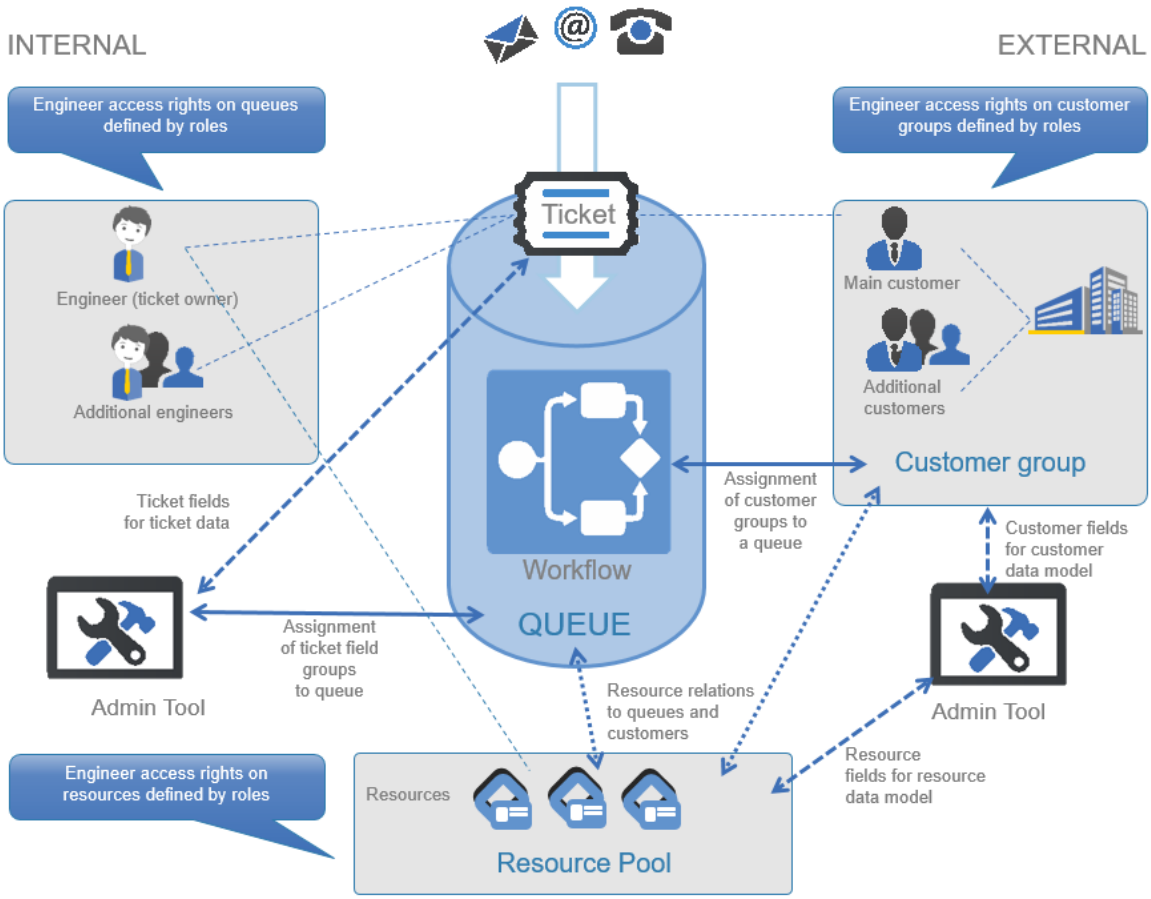


Figure 13: *ConSol CM - Basic principle*

B.4.2 Data fields

For a detailed explanation of this topic, please see section Working with data fields.

There are three types of data fields:

- **Ticket fields**
  Used to define ticket data, managed in ticket field groups, as known from previous CM versions.
- **Customer fields**
  Used to define customer data as part of the FlexCDM, the customer data model. Managed in customer field groups.
- **Resource fields** (if CM/Resource Pool is active)
  Used to define resource data as part of the resource data model. Managed in resource field groups.

You can access the content of a ticket field, customer field or resource field using the following notation:

ticket:

```
ticket.get("<group name>.<field name>")
```

unit, for one field:

```
unit.get("<group name>:<field name>")
```

resource:

```
resource.get("<group name>.<field name>")
```

Code example 1: *Access to content of data fields of the three main CM objects*

B.4.3 Standard ticket fields

Some fields do not have to be defined as ticket fields in the Admin Tool, because they are always present. These are the following fields of a ticket:

- **Ticket ID**
  Invisible for the user, only internal use in the database.
- **Ticket name**
  Visible in the Web Client, usually called ticket number.
- **Ticket subject**
  Must be set.
- **Create date**
  Is set automatically by the system.
- **Engineer/ticket owner**
  Can be null or one of the engineers.
- **Queue**
  The current queue of the ticket.

CM/ is a registered trademark of ConSol Consulting & Solutions Software GmbH

B.5 Working with the Process Designer

This chapter discusses the following:

B.5.1 Steps to perform to create a new process 32

B.5.2 Starting the Process Designer 32

B.5.1 Steps to perform to create a new process

The work with the Process Designer is one of the first steps in the pipeline of steps which you have to perform when you want to create a new process with users, roles etc. Before we start explaining how to work with the Process Designer, we will therefore provide a short list of tasks you have to do:

1. Design and deploy the workflow using the Process Designer.
2. Create a new queue with this workflow. Here, you will also need the definition of all required data fields and customer groups.
3. Create the views for the new users/engineers using the scopes of the new workflow.
4. Create one or more role(s) that have access to the new queue. Keep in mind that the access to the customer group(s) must match that of the queue.
5. Create one or more engineers/users and assign the new role(s) to them.
6. Check the login in the Web Client. Can you create a ticket in the new role?

All you need to know to perform steps two to six is explained in great detail in the *ConSol CM Administrator Manual*.

> Please note that you can modify almost all parameters and configurations of a queue on a continuous basis except for the assignment of a workflow. Once you have assigned a workflow to a queue, this assignment is fix and cannot be changed anymore. Of course, you can modify the workflow itself, but it is not possible to switch to another workflow for the same queue. This is because the changes in a workflow will be active in the business process as soon as they have been deployed and process inconsistencies have to be avoided in any case.

B.5.2 Starting the Process Designer

The following section describes how to start the Process Designer with Java Web Start or as a local Java application.

B.5.2.1 Starting the Process Designer with Java Web Start

If Java Web Start is available on your machine, you can start the Process Designer as follows:

1. Open the start page of the application server where ConSol CM is running.
2. Click the link to the `master.jnlp` file in the *ConSol CM6 Process Designer* section to start the Process Designer application. Alternatively, you can start the application from the command line using the command indicated below.
3. The Process Designer application is downloaded and started.

B.5.2.2 Starting the Process Designer without Java Web Start

If Java Web Start is not available on your machine, you can use the local Java application instead. Please ask the ConSol CM support or your ConSol CM consultant to obtain the application.

You need to perform a couple of additional steps before starting the application:

1. Unzip the file which you obtained from the ConSol CM support or your ConSol CM consultant.
2. Open the start script with a plain text editor, e.g., Notepad++. For Windows, the file is called `start_procdesigner.bat` and for Linux it is called `start_procdesigner.sh`.
3. Make the following settings:
   - Enter the name of the host (without the protocol) in the line containing `Host=`.
   - Enter the port in the line containing `Port=`.

Start the Process Designer:

- On Windows: Double-click the `start_procdesigner.bat` file. A command line window is opened and the Process Designer is started. Do not close the command line window, as this will close the Process Designer.
- On Linux: Execute the `start_procdesigner.sh` file.

> In case the Process Designer cannot be started, the network connection might be the problem. Check the Java parameters for network connections. *Use direct connection* might be required. Also check the proxy settings.

You can only execute one instance of the Process Designer at a time.

Log in with an administrator account or with an account which has the workflow management permissions. Please refer to the *ConSol CM Administrator Manual*, section *Roles*, for details.

B.6 Process Designer GUI

This chapter discusses the following:

- GUI sections
- Main menu
- Workflow editing panel
- Palette for elements and adornments
- Editing the properties of a workflow element
- Editing scripts
- GUI tips and tricks

CM/ is a registered trademark of ConSol Consulting & Solutions Software GmbH

B.6.1 GUI elements

B.6.1.1 GUI sections

The Process Designer GUI contains the following elements, please see the next figure and the list below.

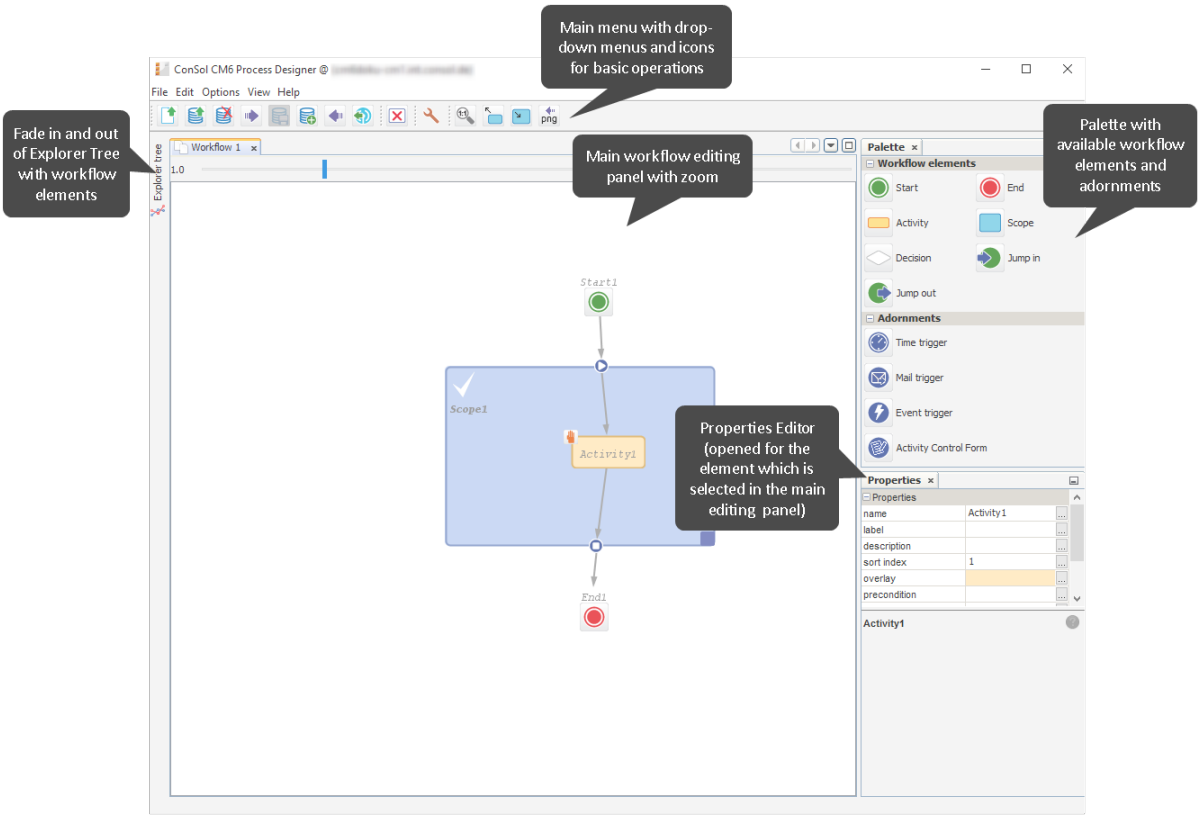


Figure 14: *ConSol CM Process Designer - GUI elements*

B.6.1.2 Main menu

The main menu contains the menu items as text entries and a menu icon list.

Menu main entry	Menu sub entry	Icon	Note
File	New ...		Create a new workflow.
	Load ...		Load a workflow. Opens table with existing workflows, see section Loading a workflow.
	Delete ...		Delete a workflow. Opens table with existing workflows, see section Deleting a workflow.
	Import ...		Import a workflow from a (proprietary workflow format) file.

Menu main entry	Menu sub entry	Icon	Note
	Save ...		Save workflow (existing version).
	Save as new version		Save the workflow as a new version.
	Export ...		Export the workflow to a file. Opens file browser of the operation system. The workflow is saved in a proprietary workflow format (.par).
	Deploy ...		(Save as new version and) deploy the workflow, i.e., install the workflow in the system. The system might prompt you for a decision: • Keep position of the tickets in the process (see section Actions during workflow deployment). • Start at START node again.
	Export to image		Creates an image from the entire workflow as .png file and opens a pop-up menu where you can select the path where to save this png file. Use this functionality for your process documentation.
	Log in		Log in to the Process Designer. Usually the login window is displayed directly after the start of the Process Designer. As login an account with administrator permissions or with the permissions to manage workflows (see *ConSol CM Administrator Manual*, section *Role Administration*) is required.
	Log out		Log out. Does not exit the Process Designer.
	Exit		Exit/stop the Process Designer application.
Edit	Clear current tab		Delete the entire workflow, all elements in the main editing panel.
	Undo		Reverse the last action which has been done in the Process Designer. Can be disabled in the *Local configuration*.
Options	Local configuration		Display pop-up window where you can select the display language of the Process Designer. All languages which have been configured for the system (see section *Global Configuration* in the *ConSol CM Administrator Manual*) are available. The labels in the workflow in the main editing panel will be displayed in the selected language. Using the checkbox *Disable undo tracking*, you can switch off the Undo functionality.
View	Normal zoom		Display workflow in default zoom (like at start of Process Designer).

Menu main entry	Menu sub entry	Icon	Note
	Expand all scopes		Display all scopes in the expanded version. See also GUI tips and tricks section below.
	Collapse all scopes		Display all scopes in the collapsed version. See also GUI tips and tricks section below.
	Hide/Show palette		Do (not) display palette in GUI.
	Hide/Show properties		Do (not) display Properties Editor in GUI.
	Hide/Show explorer		Do (not) display explorer (tree).
	Show ticket transfer history		Opens a pop-up window where the parameters for the ticket transfer during the deployment of a new workflow are displayed: • **Workflow name** Name of the workflow. • **Version** Version of the old workflow. • **Start time** Start of the transfer, will be the start time of the *Deploy* operation. • **End time** End of the transfer, after this time the new workflow will be in full operation. • **Transferred tickets** Number of tickets which have been transferred, i.e., which had to be touched by the system during workflow deployment. Should be identical to the sum of open tickets in all queues which use the workflow. • **Details** Additional information concerning the deployment with ticket transfer.
	IDE log		Opens the Log File Editor in the lower half of the screen and displays the user-specific log file of the Process Designer: `<USER_HOME_DIR>\.cmas\wfeditorR1\var\log`
Help	About		Display version information about the Process Designer and about the Java virtual machine it uses in the current configuration (this is the JVM of the browser plug-in).

B.6.1.3 Workflow editing panel

To design a workflow define the workflow elements using the graphical layout mode of the Process Designer and add the scripts to the elements where required.

A new element can be added to the workflow using drag-and-drop of the element from the palette.

A new element as successor of an existing element can also be created by using the context menu (right mouse click) of an existing element, e.g., for an activity (see the following figure). The new element and the connection to this element will be created.

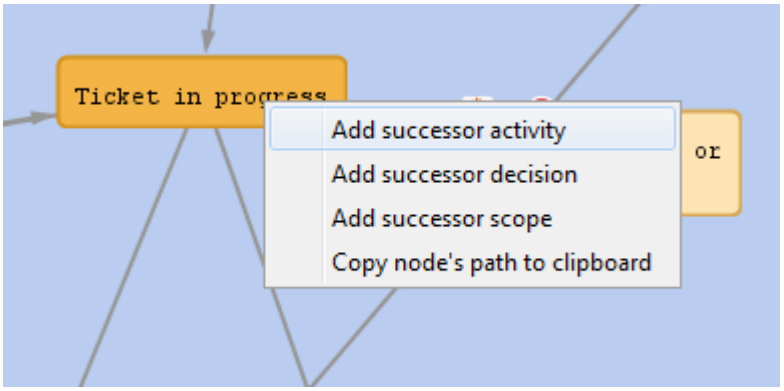


Figure 15: *ConSol CM Process Designer - Context menu for a workflow activity*

A new connection between elements is established using the left mouse button while pressing the CTRL key and just drawing the line. If the connection goes from one scope to another, the scope entry and exit points are added automatically.

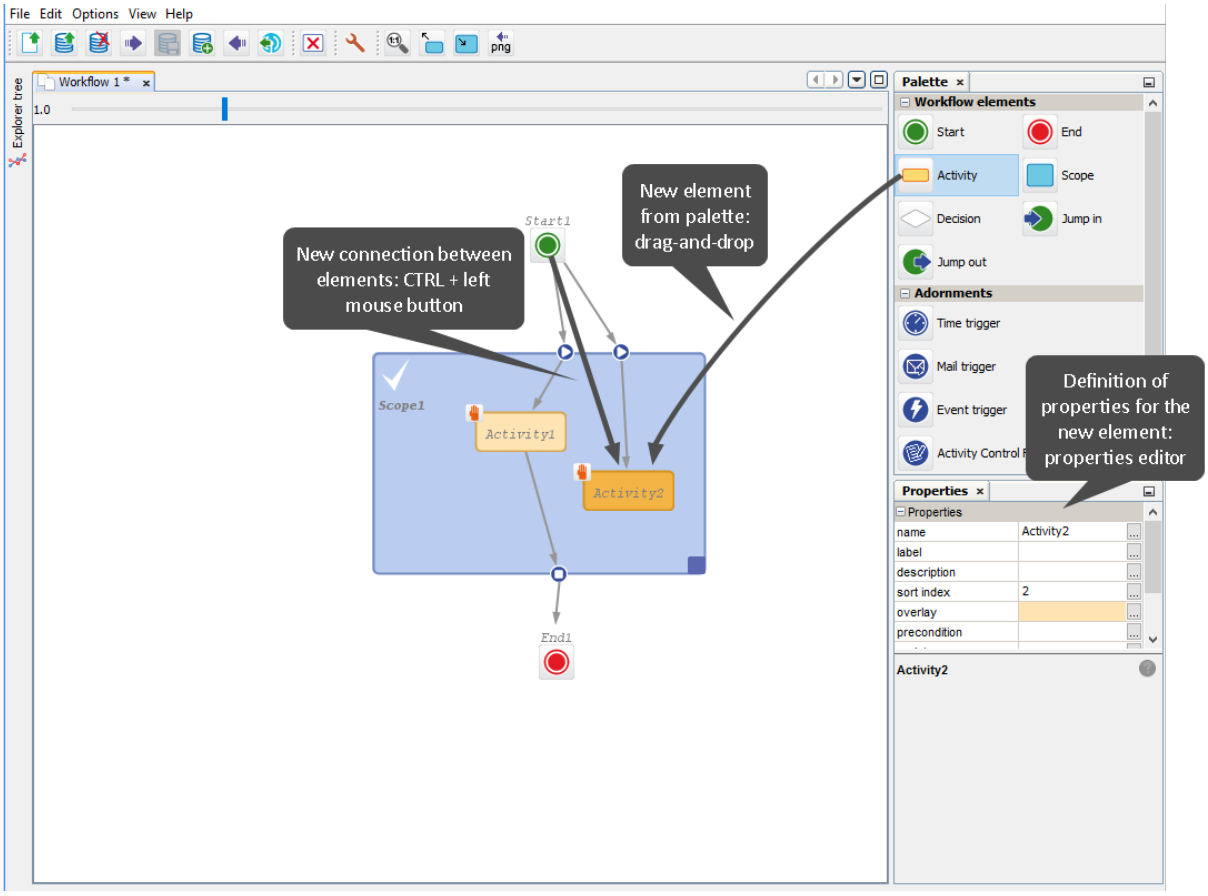
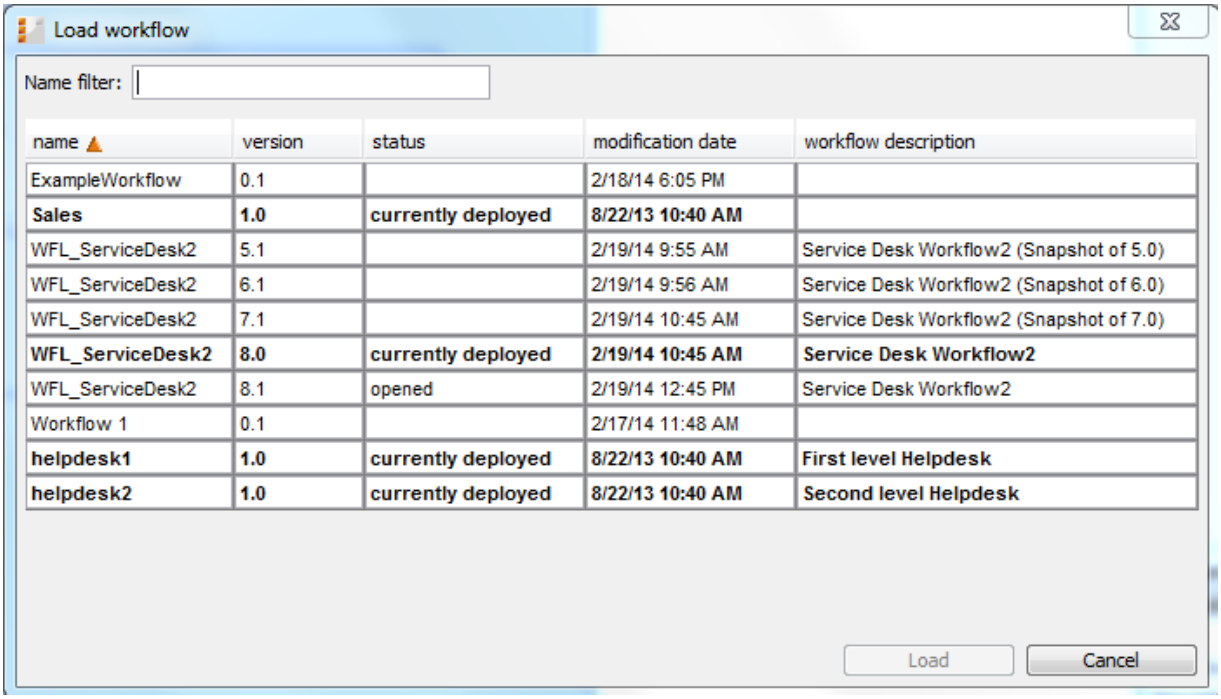


Figure 16: *ConSol CM Process Designer - Adding new elements and connections*

You might consider using a global scope for each workflow. Please refer to the Best practices section for more information about how to design good workflows.

Loading a workflow

When you have selected the icon or menu item *Load*, a table with all available workflows is displayed.



name	version	status	modification date	workflow description
ExampleWorkflow	0.1		2/18/14 6:05 PM	
Sales	**1.0**	**currently deployed**	**8/22/13 10:40 AM**	
WFL_ServiceDesk2	5.1		2/19/14 9:55 AM	Service Desk Workflow2 (Snapshot of 5.0)
WFL_ServiceDesk2	6.1		2/19/14 9:56 AM	Service Desk Workflow2 (Snapshot of 6.0)
WFL_ServiceDesk2	7.1		2/19/14 10:45 AM	Service Desk Workflow2 (Snapshot of 7.0)
WFL_ServiceDesk2	**8.0**	**currently deployed**	**2/19/14 10:45 AM**	**Service Desk Workflow2**
WFL_ServiceDesk2	8.1	opened	2/19/14 12:45 PM	Service Desk Workflow2
Workflow 1	0.1		2/17/14 11:48 AM	
helpdesk1	**1.0**	**currently deployed**	**8/22/13 10:40 AM**	**First level Helpdesk**
helpdesk2	**1.0**	**currently deployed**	**8/22/13 10:40 AM**	**Second level Helpdesk**

Figure 17: *ConSol CM Process Designer - Load a workflow*

The table can be sorted based on a column by clicking on the little triangle icon next to the column header.

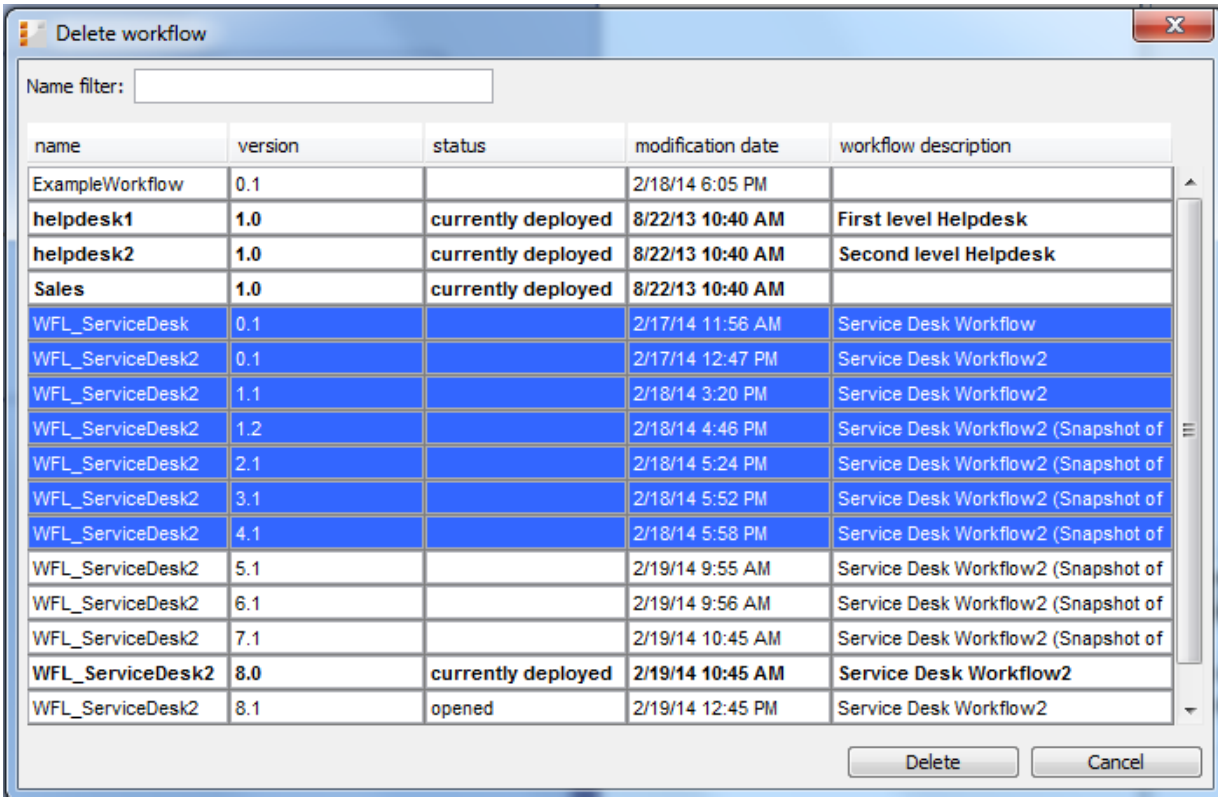
The table contains the following columns:

- **name**
  The name of the workflow as set in the *name* property of the workflow (click into the white space around the global scope to see it for a workflow).
- **version**
  The version of the workflow. This is assigned automatically by the ConSol CM system. When a scenario has been exported and is imported again, the numbering will start with 1.0 anew.
- **status**
  For older workflows this field is empty. The workflows which are deployed are described by *currently deployed*.
- **modification date**
  The date of the last modification (date when the workflow was saved) is indicated.
- **workflow description**
  The description which has been entered into the field *workflow description* (**not** *description*!).

To load a workflow, select it in the list and click *Load*. Only single selection is possible.

Deleting a workflow

When you have selected the icon or menu item *Delete*, a table with all available workflows is displayed.



name	version	status	modification date	workflow description
ExampleWorkflow	0.1		2/18/14 6:05 PM	
helpdesk1	**1.0**	**currently deployed**	**8/22/13 10:40 AM**	**First level Helpdesk**
helpdesk2	**1.0**	**currently deployed**	**8/22/13 10:40 AM**	**Second level Helpdesk**
Sales	**1.0**	**currently deployed**	**8/22/13 10:40 AM**	
WFL_ServiceDesk	0.1		2/17/14 11:56 AM	Service Desk Workflow
WFL_ServiceDesk2	0.1		2/17/14 12:47 PM	Service Desk Workflow2
WFL_ServiceDesk2	1.1		2/18/14 3:20 PM	Service Desk Workflow2
WFL_ServiceDesk2	1.2		2/18/14 4:46 PM	Service Desk Workflow2 (Snapshot of
WFL_ServiceDesk2	2.1		2/18/14 5:24 PM	Service Desk Workflow2 (Snapshot of
WFL_ServiceDesk2	3.1		2/18/14 5:52 PM	Service Desk Workflow2 (Snapshot of
WFL_ServiceDesk2	4.1		2/18/14 5:58 PM	Service Desk Workflow2 (Snapshot of
WFL_ServiceDesk2	5.1		2/19/14 9:55 AM	Service Desk Workflow2 (Snapshot of
WFL_ServiceDesk2	6.1		2/19/14 9:56 AM	Service Desk Workflow2 (Snapshot of
WFL_ServiceDesk2	7.1		2/19/14 10:45 AM	Service Desk Workflow2 (Snapshot of
WFL_ServiceDesk2	**8.0**	**currently deployed**	**2/19/14 10:45 AM**	**Service Desk Workflow2**
WFL_ServiceDesk2	8.1	opened	2/19/14 12:45 PM	Service Desk Workflow2

Figure 18: *ConSol CM Process Designer - Delete a workflow*

The table can be sorted based on a column by clicking on the little triangle icon next to the column header.

The table contains the following columns:

- **name**
  The name of the workflow as set in the *name* property of the workflow (click into the white space around the global scope to see it for a workflow).
- **version**
  The version of the workflow. This is assigned automatically by the ConSol CM system. When a scenario has been exported and is imported again, the numbering will start with 1.0 anew.
- **status**
  For older workflows this field is empty. The workflows which are deployed are described by *currently deployed*.
- **modification date**
  The date of the last modification (date when the workflow was saved) is indicated.
- **workflow description**
  The description which has been entered into the field *workflow description* (**not** *description*!).

To delete one or more workflow(s), select it/them in the list and click *Delete*. For every workflow you are prompted to confirm the deletion, so when you have marked a great number of workflows to delete and then you realize that you would like to keep one of them this is possible without canceling the entire operation.

> You might want to delete all or almost all older workflows before exporting a scenario, because a great number of workflows increases the size of the scenario considerably. For export and import of scenarios, please refer to the respective section in the *ConSol CM Administrator Manual*.
>
> For an export of a scenario (using the Admin Tool), you might also use the option *Only latest/installed versions* for workflows. This can reduce the size of the exported scenario considerably, depending on the number of non-active workflows. Nevertheless, it might be a good idea to clean the CM system regarding workflows from time to time to reduce the old data and a potential data overhead.

B.6.1.4 Palette for elements and adornments

As a default setting the palette is displayed in the top right corner. You can hide (and re-display) the palette using the main menu entry *Hide/Show palette* under *View*.

The palette contains two types of workflow components:

- elements
- adornments

Elements

Elements are basic components which form the workflow and represent the process logic.

Icon	Element	Note	Section
	Start node	Is set automatically, no other start node than the default start node can be added.	Start nodes
	End node	A workflow can contain one or more end nodes.	End nodes
	Activity	The actions in the workflow, manual or automatic.	Activities
	Scope	The highest hierarchy level in workflows	Scopes
	Decision	Decision node which has a `true` and a `false` exit point.	Decision nodes
	Jump-in	Entry point for tickets from other workflows/queues.	Jump-out and jump-in nodes
	Jump-out	Exit point for tickets. A target queue has to be defined. A target node can be defined but is optional.	Jump-out and jump-in nodes
	Reopen	Activity which can be performed for closed tickets.	Reopen activities

Adornments

Adornments are objects which are assigned to a workflow activity or to a scope. Please see indicated sections for detailed explanations.

Icon	Adornment	Note	Section
	Time trigger	Can measure time intervals. Fires when the end of the interval has been reached. Can optionally use a business calendar.	Time triggers
	Mail trigger	Fires when an email for the ticket has come in.	Mail triggers
	Business event trigger	Fires when an event has occurred. The type of event can be specified (e.g., change of engineer, change of priority).	Business event triggers
	ACF (Activity Control Form)	Defines the ACF which should be displayed when the activity is executed. ACFs are defined in the Admin Tool.	Activity Control Forms (ACFs)

Activity types

Activities can have additional icons to indicate if the activity is manual or there are conditions. Please see indicated sections for detailed explanations.

Icon	Name	Description	Section
	Manual	Indicates that the activity is a manual activity, i.e., the engineer has to click it in the Web Client.	Activity type
	Scope activity	Indicates that the activity is a scope activity, i.e., that the activity is not bound to a certain chain of activities but is available in the entire scope	Activities
	Precondition	Indicates that the activity has a condition script attached. The condition script is executed when the previous activity has been performed. The activity with the precondition icon is only displayed if the condition script returns “true”.	Precondition

B.6.1.5 Editing the properties of a workflow element

The properties editor of an element is opened on the right when you click the element in the workflow editing panel. The available settings depend on the selected element.

The following example shows the properties editor of an activity:

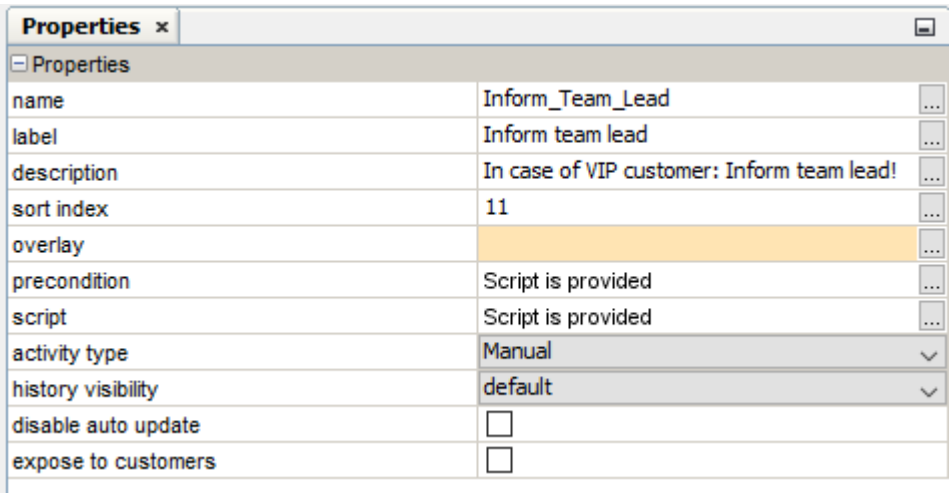


Figure 19: *ConSol CM Process Designer - Properties editor of an activity*

B.6.2 Editing scripts

You use the script editor in the Process Designer to write Groovy scripts (i.e., pure Groovy and Java code is accepted). For explanations, recommendations, and examples concerning workflow programming using scripts, please see section Workflow programming.

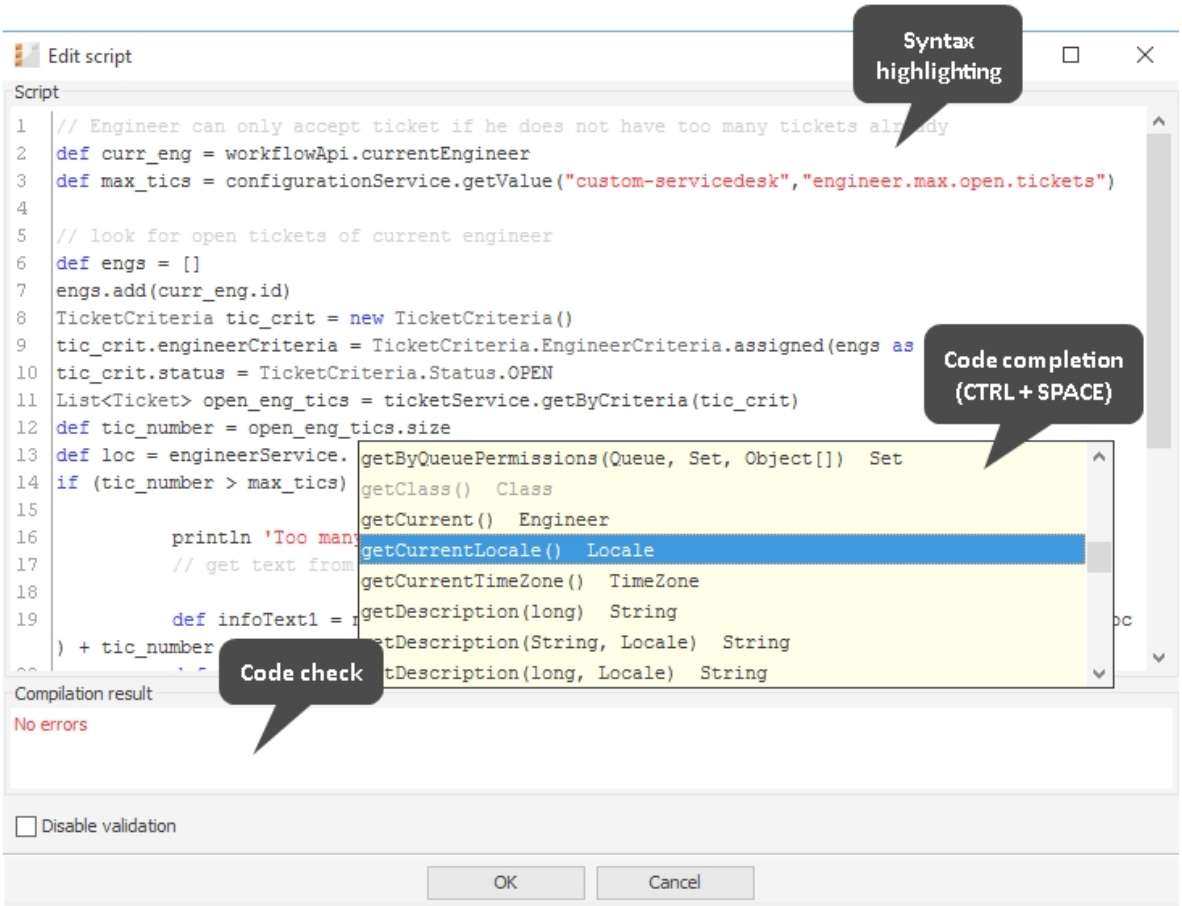


Figure 20: *ConSol CM Process Designer - Script editor*

The script editor provides the following features:

- **Syntax highlighting**
  Groovy code is highlighted according to key words.
- **Code completion**
  When you have entered the name of an object and the dot, the possible methods are suggested. Press CTRL + SPACE to activate code completion.
- **Code check**
  The entered code is controlled according to the correct use of general syntax and methods. The error code is displayed in the *Compilation result* panel.

B.6.3 GUI tips and tricks

To ...	do the following ...
expand a single collapsed scope	Double-click in the scope.
collapse a single expanded scope	Double-click in the scope.
resize a scope	Move the handler of the scope (bottom right of the scope).
add a new email trigger to a scope	Grab the email trigger symbol in the palette with the mouse and drop the newly created email trigger into the respective scope.
add a time trigger to a scope	Grab the time trigger symbol in the palette with the mouse and drop the newly created time trigger into the respective scope.
add a time trigger to an activity	Grab the time trigger symbol in the palette with the mouse and drop the newly created time trigger into the respective activity (used e.g., for resubmissions).
delete a connection between two activities	Mark the connection (not the scope or other surrounding elements!) and press DELETE.
insert a new edge into a connection between two activities	Click on the connection: a small square indicates the new edge which can then be moved around.
copy and paste a scope	Mark the scope, and either press CTRL+C and CTRL+V or right-click and select *Copy element to clipboard* and *Paste element from clipboard* from the context menu. A copy of the scope with all the elements it contains is inserted at the bottom of the workflow.
copy and paste an activity or node	Mark the activity or node, and either press CTRL+C and CTRL+V or right-click and select *Copy element to clipboard* and *Paste element from clipboard* from the context menu. A copy of the activity or node with all its data is inserted at the bottom of the workflow.

C - Workflow components

You can work with various types of workflow components to build the workflows for your ConSol CM system. The palette in the Process Designer offers all elements and adornments, see section Palette for elements and adornments for an overview.

In the following chapters, all workflow elements and adornments will be explained in detail.

Workflow Element	Explanation
Start node	The first node in a workflow, see section Start nodes.
End nodes	One or more end nodes of the process. The ticket is closed. See section End nodes.
Scopes	Realms of a process, see section Scopes.
Activities	The steps of a process. Can be automatic or manual, see section Activities.
Reopen activities	A special kind of activity which can be performed for closed tickets, see section Reopen activities.
Decision nodes	Workflow element which represents a “true” / “false” decision, see section Decision nodes.
Adornments	Elements to control the process flow: triggers and activity control forms. See section Adornments (triggers and ACFs) .
Jump-out and jump-in nodes	Elements which connect workflows, see section Jump-out and jump-in nodes.

C.1 Start nodes

Every workflow contains exactly one start node. When you create a new workflow the start node is added automatically, you do not have to add it yourself.

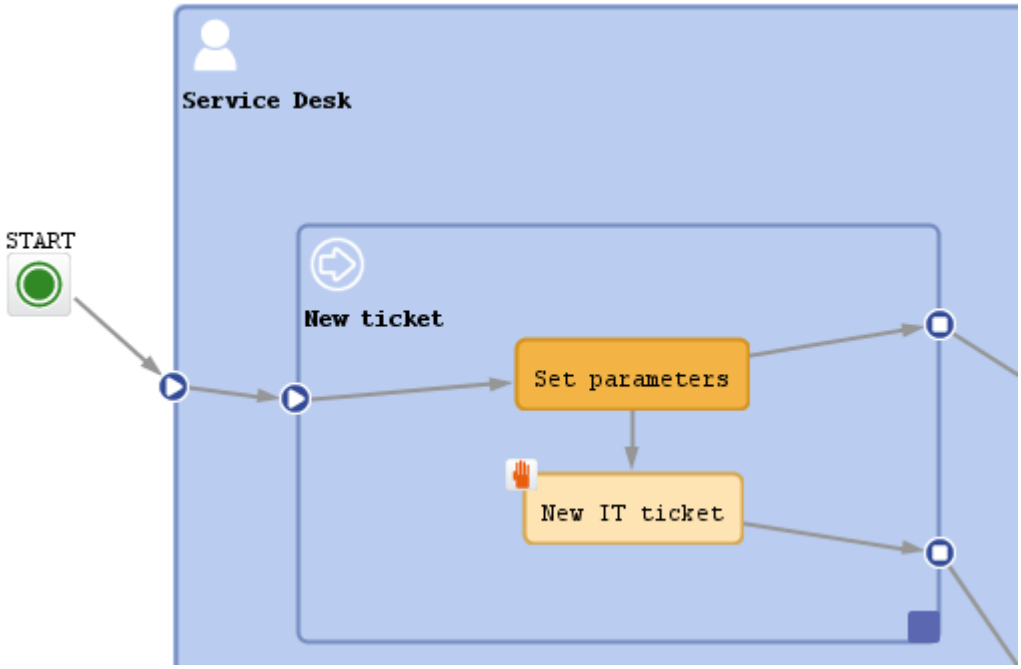


Figure 21: *ConSol CM Process Designer - Start node*

The start node does not have any scripts and cannot be configured in any way.

When a ticket enters the workflow and no specific entry point has been defined, the ticket passes through the start node.

> The start node should not be positioned within the global scope. See also section Best practices.

C.1.1 Properties of a start node

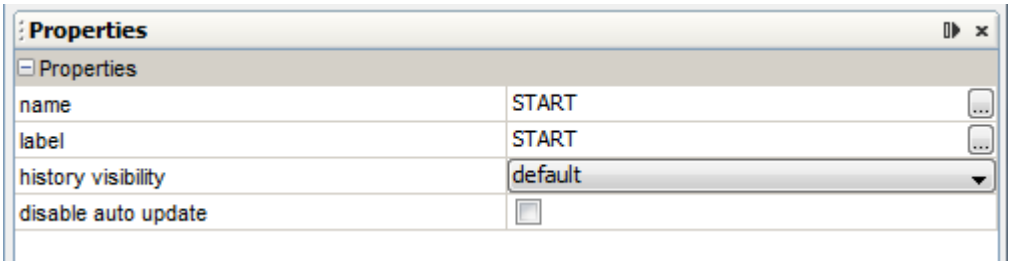


Figure 22: *ConSol CM Process Designer - Start node properties*

Properties:

- **name**

  Technical object name.

- **label**

  Localized name which will be displayed on the GUI.

- **history visibility**

  Mandatory. This property defines on which display levels the execution of the activity is shown in the ticket history in the Web Client. The possible values are:

  - 2nd level and 3rd level
  - only 3rd level
  - on every level
  - default (default value)

    The activity is shown on the display level which is configured in the Admin Tool, navigation group *Tickets*, navigation item *History*. Depending on the type of activity, one of the following settings is used:

    - Manual activity or activity with overlay executed
    - Activity executed after escalation
    - Automatic activity executed
  - hidden on all levels
    The execution of the activity is never displayed in the ticket history of the Web Client.

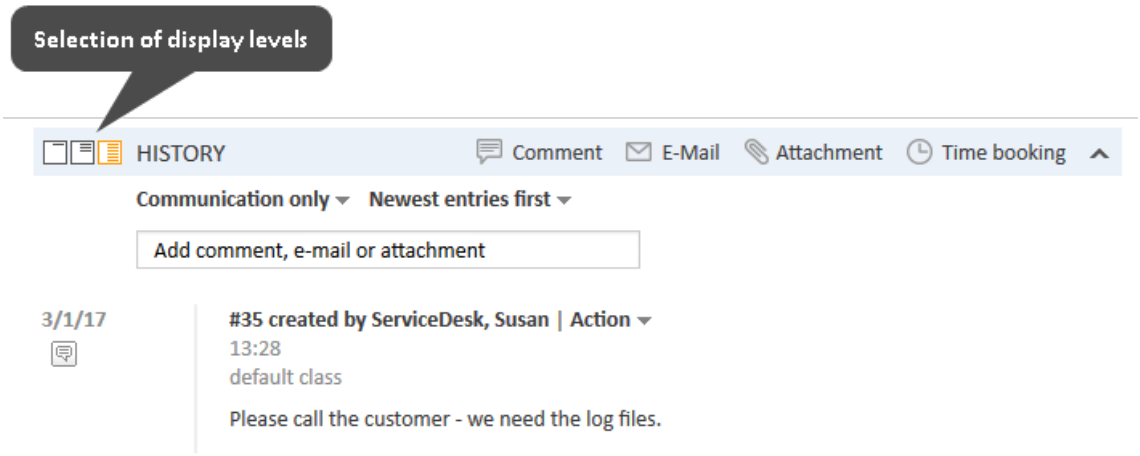


Figure 23: *ConSol CM Web Client - Display levels in the ticket history*

- **disable auto update**

  Defines the behavior of the ticket when an event has been fired or executed. Usually, after an event, a ticket update operation is performed automatically. In case a chain of events is used you should avoid triggering a ticket update operation after every single event. To avoid this, set *disable auto update* to “true” in all events except for the last one. Then, the ticket is only updated once, after the last event.

C.2 End nodes

A workflow in ConSol CM can have one or more end nodes.

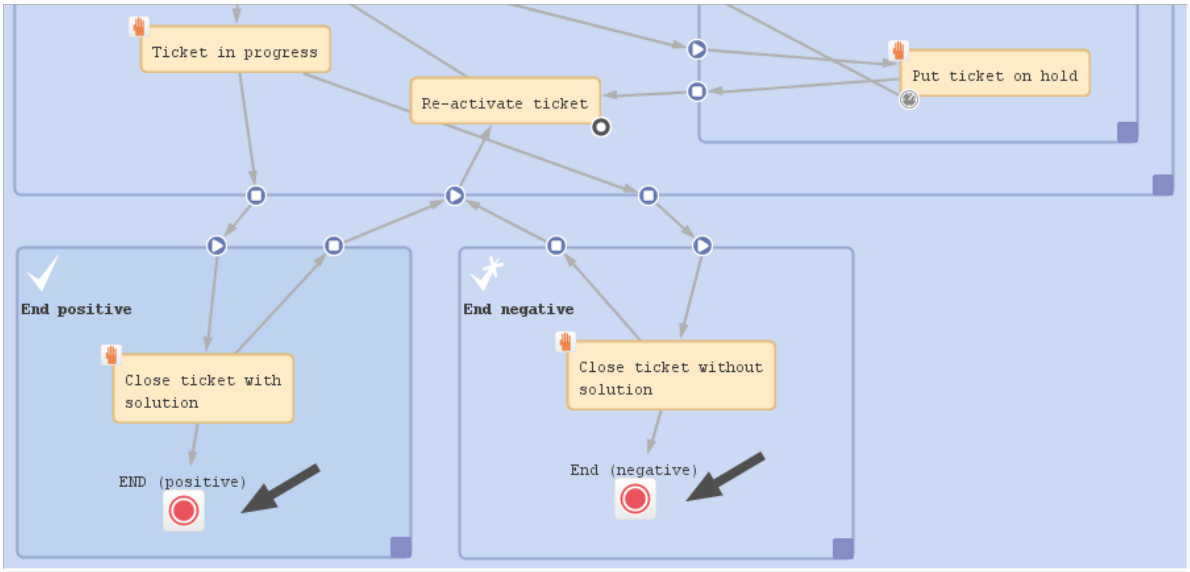


Figure 24: *ConSol CM Process Designer - End nodes*

An end node represents the closing of the ticket, i.e., when a ticket is passed to an end node it is closed in a technical sense.

C.2.1 Possible actions for closed tickets

- The ticket can be reopened by an administrator using the *Ticket Administration* in the Admin Tool, please see the respective section in the *ConSol CM Administrator Manual* for detailed information.
- The ticket can be reopened by an engineer or customer using a reopen activity. Please see Reopen activities for further information.
- Other actions which are run with admin privileges can also be performed, i.e., workflow actions are still possible. This can apply, for example, if a trigger is attached to the scope where the END node is located. The trigger will continue running and measuring the time even if the ticket is located in the END node and is closed. Thus, an action which is performed due to the firing event of the trigger can still be started. To avoid this, either do not use timer trigger in the respective scope or move the END node to a location outside the end scope.
  Another ticket could also perform a linked action with the closed ticket, as well as admin task scripts (from the Task Execution Framework).

C.2.2 Actions not possible for closed tickets

- No engineer can edit the ticket anymore.

However, assuming engineers have the required access permissions, they can still read the ticket. This is an important basis for the use of all ConSol CM tickets of a system as knowledge base.

The passing of the ticket to the end node can be a manual or an automatic action. In the figure above, the end nodes are automatic nodes, i.e., the ticket passes to this node when the previous activity has been performed.

As a minimum a workflow has to contain one end node, because there has to be a way to close the ticket. You might want to create more than one end node. This can be helpful when you create reports, e.g., to distinguish between positive and negative endings.

An end node might have a script, i.e., before the ticket is closed, a script can be executed.

> Sometimes, it might be required to set a ticket to *closed*, *completed*, or *done* from an engineer's point of view, i.e., to set a ticket to a preliminary end. After a while, if there are no more questions or remarks from the customer, the ticket should be closed automatically. You can achieve this by setting a time trigger to an end activity and letting the ticket go to the end node automatically after the defined time (see following figure).

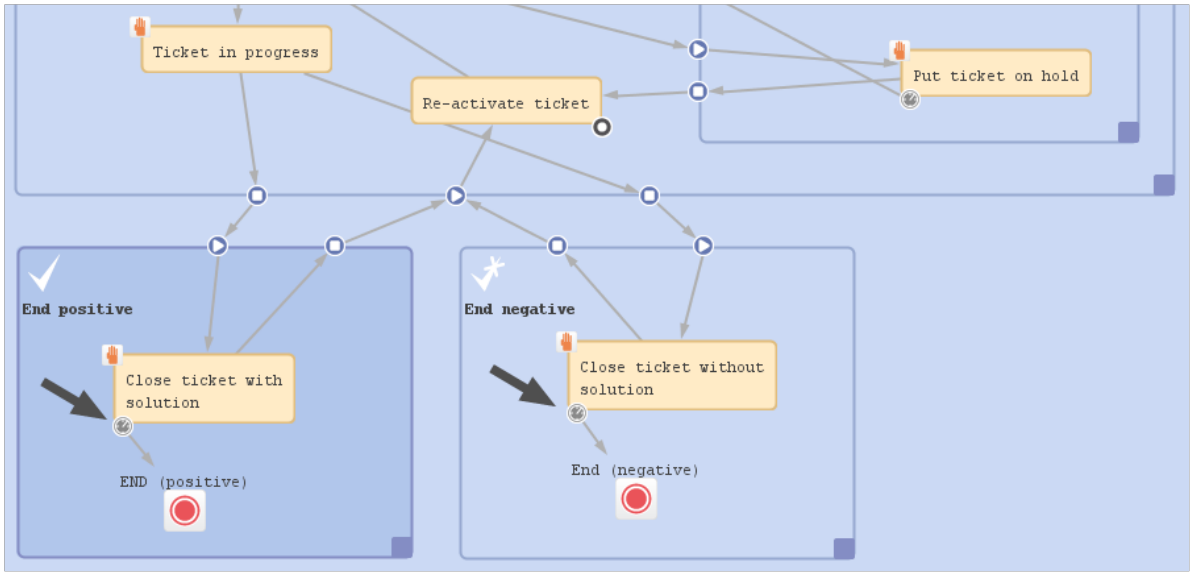
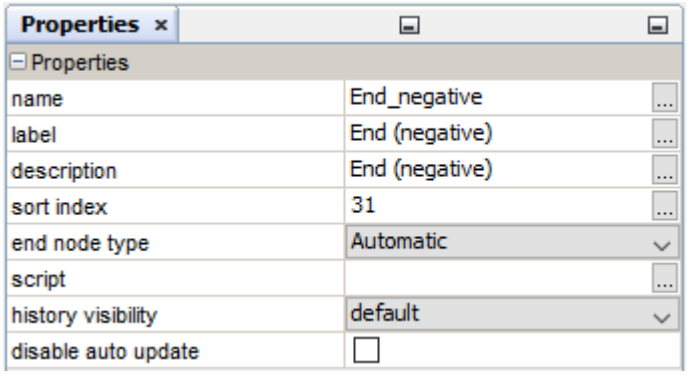


Figure 25: *ConSol CM Process Designer - End nodes reached via time trigger*

C.2.3 Properties of an end node



Properties	
name	End_negative
label	End (negative)
description	End (negative)
sort index	31
end node type	Automatic
script	
history visibility	default
disable auto update	☐

Figure 26: *ConSol CM Process Designer - End node properties*

Properties:

- **name**

  Technical object name.

- **label**

  Localized name which will be displayed on the GUI.

- **description**

  Description which is displayed as mouse-over text.

- **sort index**

  Especially relevant for manual end nodes. Defines the position in the list of workflow activities in the Web Client. You might want to set a high sort index to make sure the end node is displayed at the bottom of the list.

- **end node type**

  - **Automatic**
    Finishing / closing the ticket is an automatic activity which will be executed immediately after the previous activity.
  - **Manual**
    The ticket has to be closed manually. The name or localized label of the end node will appear as workflow activity in the Web Client. The workflow activity has to be executed by an engineer to close the ticket.

- **script**

  Here, a script which should be executed when the ticket enters the end node, i.e. before the ticket is closed, can be edited.

- **history visibility**

  Mandatory. This property defines on which display levels the execution of the activity is shown in the ticket history in the Web Client. The possible values are:

  - 2nd level and 3rd level
  - only 3rd level
  - on every level
  - default (default value)

    The activity is shown on the display level which is configured in the Admin Tool, navigation group *Tickets*, navigation item *History*. Depending on the type of activity, one of the following settings is used:

    - Manual activity or activity with overlay executed
    - Activity executed after escalation
    - Automatic activity executed
  - hidden on all levels
    The execution of the activity is never displayed in the ticket history of the Web Client.

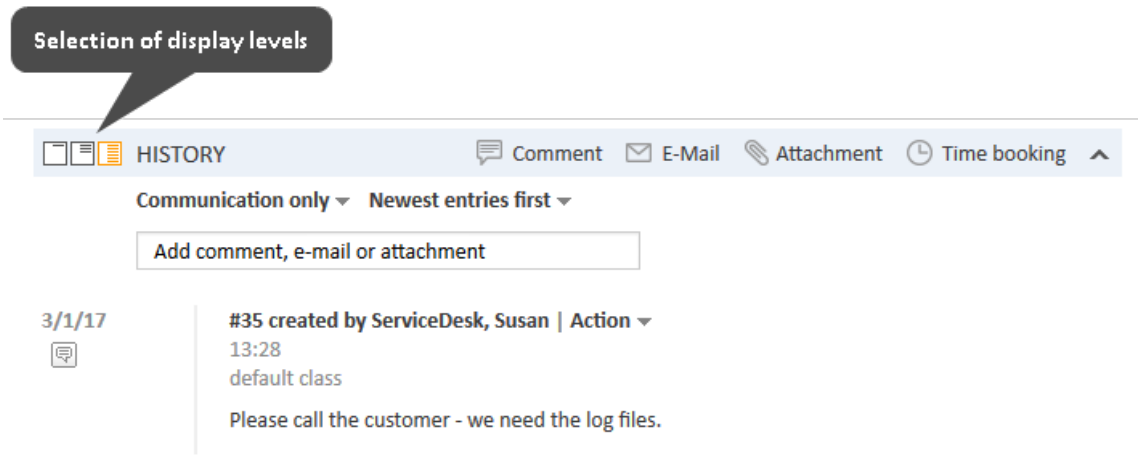


Figure 27: *ConSol CM Web Client - Display levels in the ticket history*

- **disable auto update**

  Defines the behavior of the ticket when an event has been fired or executed. Usually, after an event, a ticket update operation is performed automatically. In case a chain of events is used you should avoid triggering a ticket update operation after every single event. To avoid this, set *disable auto update* to “true” in all events except for the last one. Then, the ticket is only updated once, after the last event.

C.3 Scopes

This chapter discusses the following:

C.3.1 Introduction to scopes 55

C.3.2 Defining a new scope 57

C.3.3 Properties of a scope 58

C.3.4 Scopes and views 59

C.3.5 Scope sort index 59

C.3.1 Introduction to scopes

When a ticket passes through a process there are several positions it has to pass, all in a pre-defined order. For example, in a service desk environment, the ticket comes in as *new ticket*, then it has to be pre-qualified (in our example: are there any SLAs which have to be taken into consideration, is it a VIP customer?). Subsequently, the engineer can work on the ticket and might put it on hold for a while. Then the ticket should be closed, either as *positive, with solution* or *negative, without solution*. Those major steps of the process are represented as scopes in ConSol CM workflows. See the following figure for an example workflow.

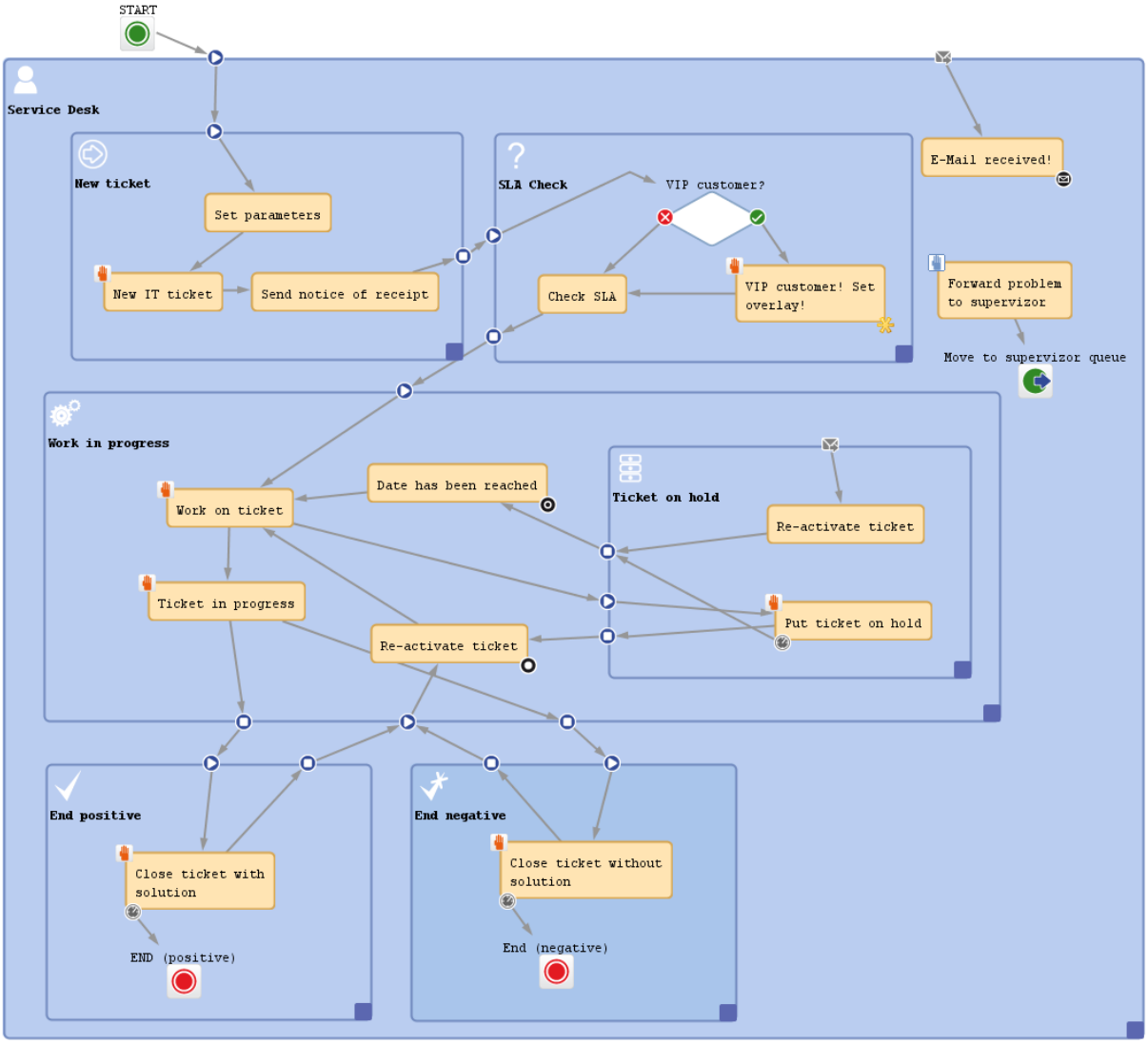


Figure 28: *ConSol CM Process Designer - Workflow with scopes*

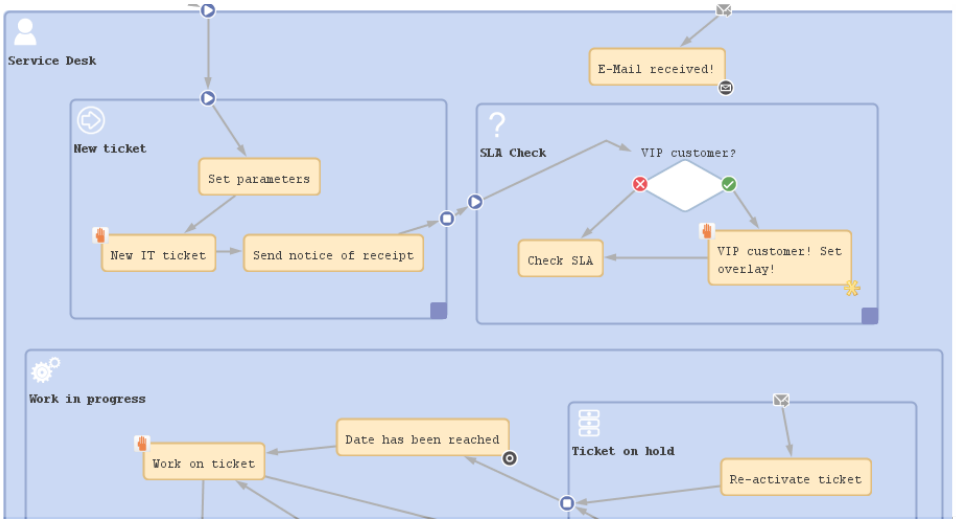
Within each larger process step (i.e., within each scope), there can be one or more activities, e.g., during pre-qualification, first the required parameters are set, the ticket has to be accepted, and a receipt notice is sent. There are regular activities which form a defined sequence of steps which have to be performed, and there might be scope activities which are available as long as the ticket is in the scope, without a strictly defined position in the process. Both types of activities are described in detail in the section Activities. Here, only scopes are explained.

A scope can be part of another scope or - seen from the opposite point of view - a scope can contain sub-scopes.

A scope can have various types of triggers, e.g., a mail trigger fires whenever an email to a ticket, which is currently in the scope, has been received. Please see sections Mail triggers, Time triggers, and Business event triggers for details.

C.3.2 Defining a new scope

In order to define a new scope, i.e., to add a new scope to the workflow, grab the scope icon in the palette and drag-and-drop it to the workflow at the position where you would like to locate it. Activate it with a double-click. Then you can add new activities or other elements or drag existing activities/elements into the scope. When you connect elements by drawing arrows, the entry and exit points of a scope are defined automatically.



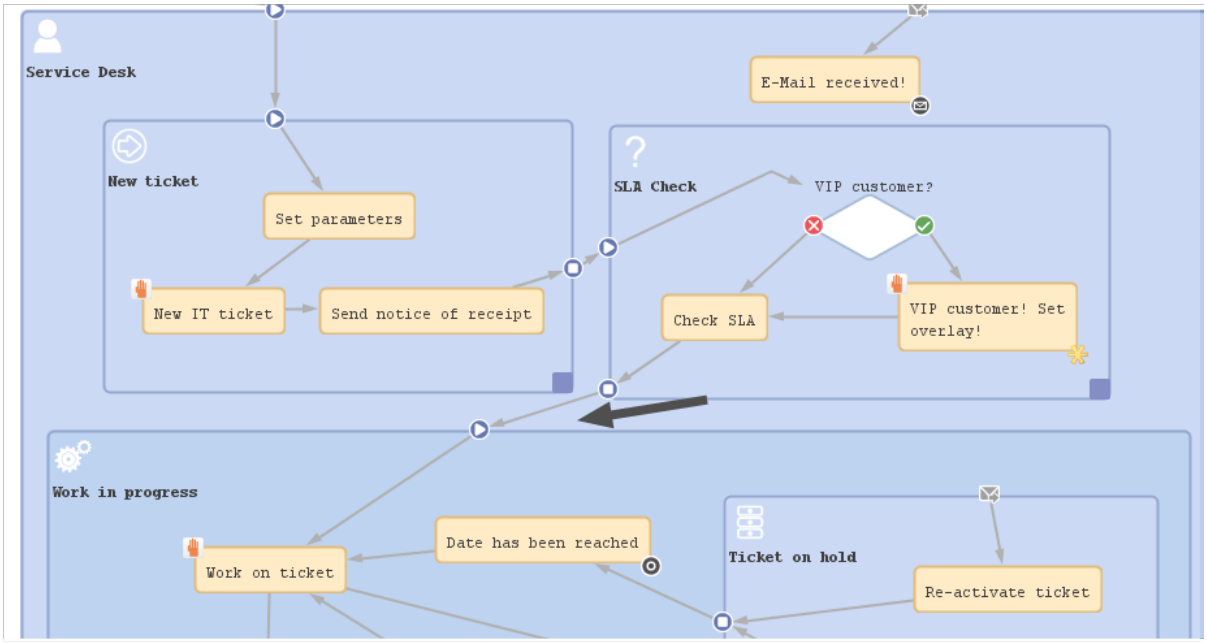


Figure 29: *ConSol CM Process Designer - Automatically generated exit and entry points in scopes*

When you have defined/added the new scope you can define the scope's properties, see next section.

C.3.3 Properties of a scope

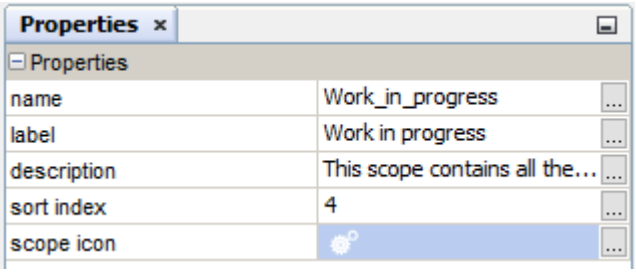


Figure 30: *ConSol CM Process Designer - Scope properties*

The following properties can be defined for a scope:

- **name**

  The technical object name.

- **label**

  The localized name which will be displayed in the Web Client GUI.

- **description**

  The description of the scope. It is not displayed on the GUI of the Web Client. The description is part of the system documentation though.

- **sort index**

  Defines the position of tickets of this scope in a view (in case the view comprises more than one scope) and influences the ordering in CM/Track V2, see paragraph Scope sort index.

- **scope icon**

  The icon which is displayed as scope icon in the Web Client GUI (see following figure). Click into the blue area to pick one of the ConSol CM standard icons or use the file browser (...) to load an icon from the file system.

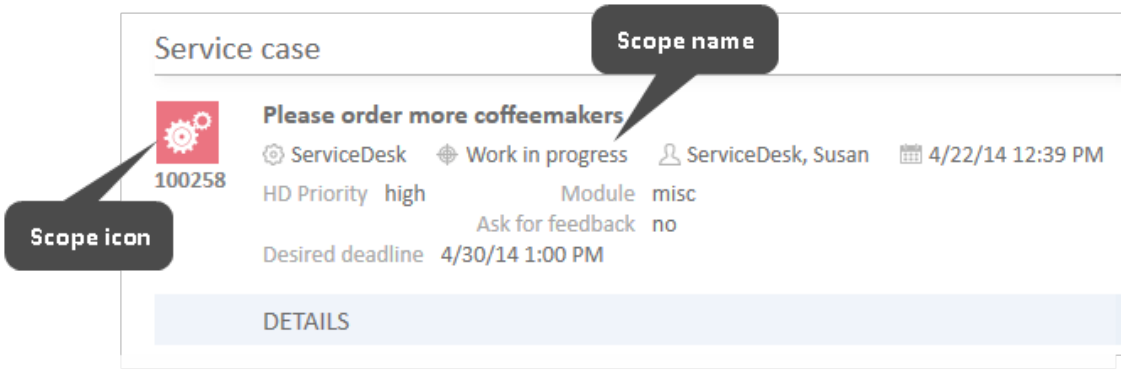


Figure 31: *ConSol CM Web Client - Scope icon*

> ⚠ Please keep in mind that the icon is merged with the background color of the ticket icon. So (in case you would like to upload your own icons) transparent images should be used for ticket icons. Otherwise, the background color might be lost or only be seen in a small border around the icon.

C.3.4 Scopes and views

Views, i.e., the selection criteria for the ticket list(s), are defined based on scopes. For a detailed explanation of views and view definition, please refer to the respective section in the *ConSol CM Administrator Manual*.

In the present context, i.e., when you define scopes in the workflow, it is important to keep in mind which views might be required later on. For example, the mechanism of *new, active*, and *pending* tickets is based entirely on the scope and view definition:

- **View: New**
  All new tickets in the scope *new*.
- **View: Active**
  All active tickets, i.e., tickets which **are not** in a scope *on hold*, *resubmission*, or the like.
- **View: Pending**
  All tickets which **are** in a scope *on hold*, *resubmission*, or the like.

That means, whenever a view is required to display only a certain sort of tickets, a scope has to be defined.

> ⚠ 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 performance of the Web Client 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.

C.3.5 Scope sort index

The sort index defines the order of the tickets in the ticket list in CM/Track V2 if the ticket list is sorted by scope. This setting applies when the user selects *Scope* in the *Sort by* drop-down list above the list of tickets.

C.4 Activities

This chapter discusses the following:

C.4.1 Introduction to activities 61

C.4.2 Properties of an activity 64

C.4.3 Process logic of activities 70

C.4.4 Examples of activities 72

C.4.1 Introduction to activities

An activity represents an action in a workflow. An activity is located within a scope and is of one of the following types:

- manual
- automatic
- scope

Using manual and automatic activities, you can model a rather strict business process where the engineer has to follow one of the provided paths for the ticket. Using scope activities you can model more flexible processes and prepare the CM system for case management where an engineer can react in a rather dynamic way, depending on the required circumstances.

C.4.1.1 Manual activities

A **manual activity** has to be performed by a manual action of the engineer using the Web Client GUI. The activity is displayed as *Workflow activity* in the Web Client, provided at least one of the roles of the engineer has the *Execute* permission (please refer to the *ConSol CM Administrator Manual*, section *Role Administration*, for a detailed explanation). In the Process Designer, the activity is marked by the *red hand/manual* icon.

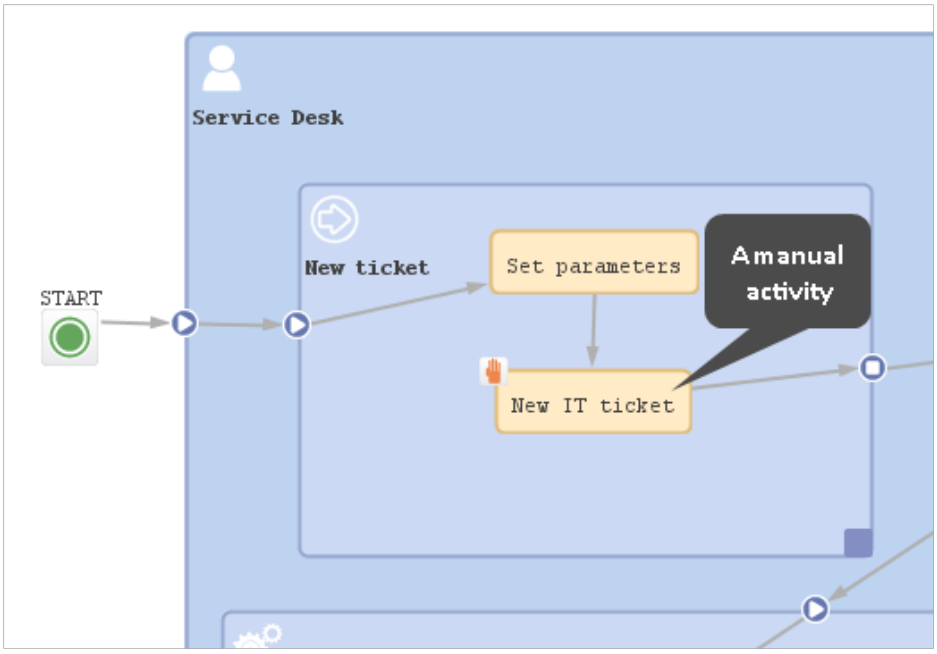


Figure 32: *ConSol CM Process Designer - Manual activity in workflow*

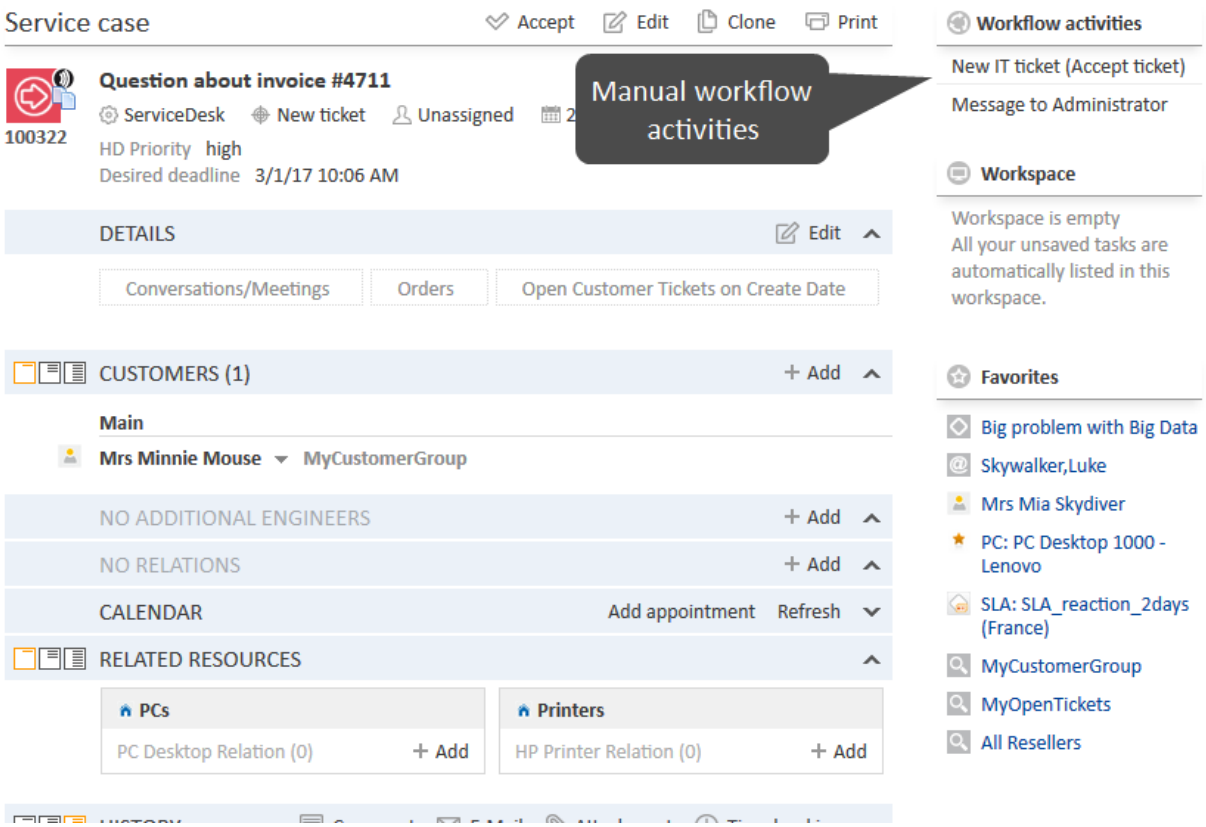


Figure 33: *ConSol CM Web Client - Manual activity*

C.4.1.2 Automatic activities

An **automatic activity** is performed automatically by the system and is not displayed in the Web Client. In the Process Designer, an automatic activity is not marked by any special icon.

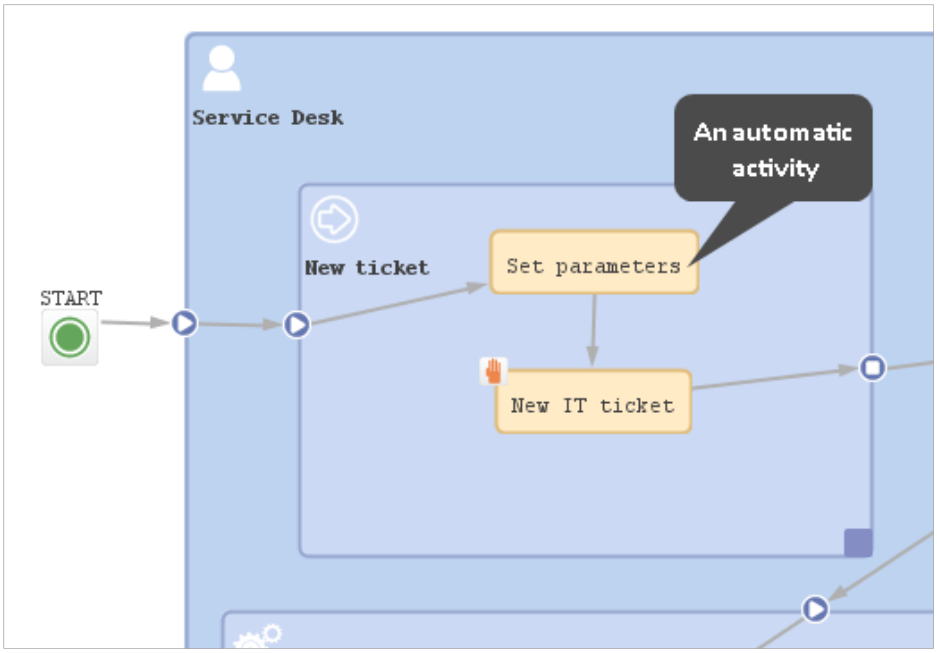


Figure 34: *ConSol CM Process Designer - Automatic activities*

C.4.1.3 Scope activities

A **scope activity** is located within a scope and does not have any incoming connections from other activities. In the Web Client, a scope activity is available as long as the ticket is placed within the scope. This holds true for all subordinate scopes. A scope activity might have subsequent activities (then is causes a process exception) but it can also be an activity without any connections (then it causes an interrupt). See section Interrupts and exceptions to learn more about interrupts and exceptions.

Please note that a ticket does not necessarily leave a scope when the "last" activity within the scope has been performed. See section Activities about this topic.

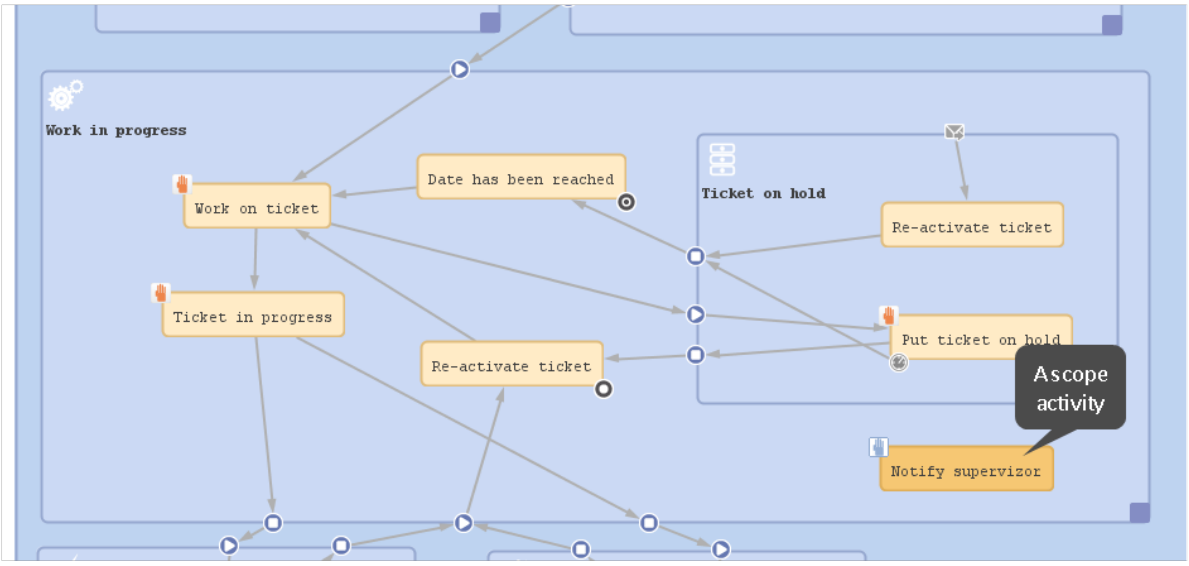


Figure 35: *ConSol CM Process Designer - a scope activity (available in the scopes "Work in Progress" and "Ticket on hold")*

C.4.2 Properties of an activity

Activities have the following properties.

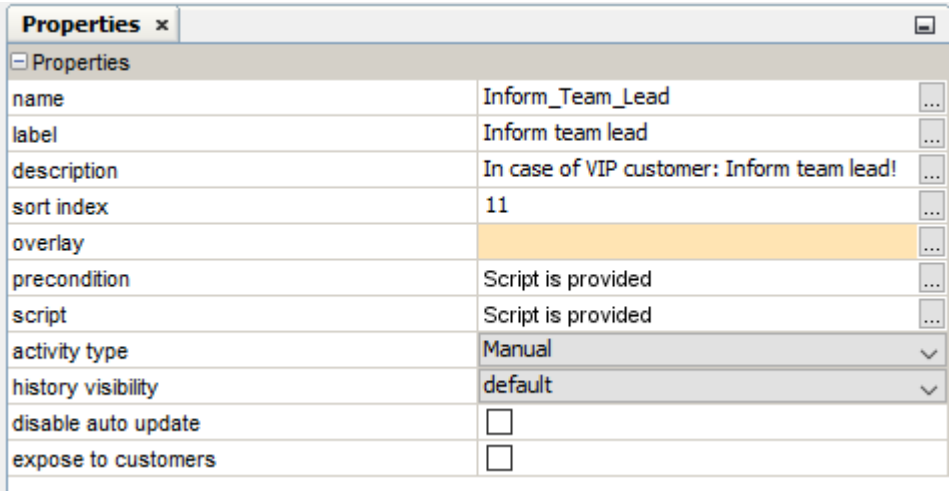


Figure 36: *ConSol CM Process Designer - Properties editor*

Properties:

- **name**

  Mandatory. This is the technical name of the activity. When an activity is newly created, you can edit the label and the activity name will be generated automatically from the label (special characters are omitted). Afterwards, the activity name is never changed automatically but can be edited manually. Allowed characters for names are:

  - letters (small or capital), but no special characters
  - underscores
  - numbers

- **label**

  The localized name of the activity. All languages which have been configured for the system are available and can be filled. In the Web Client, the label is displayed according to the browser locale. If it is not available, the label of the default locale is shown.

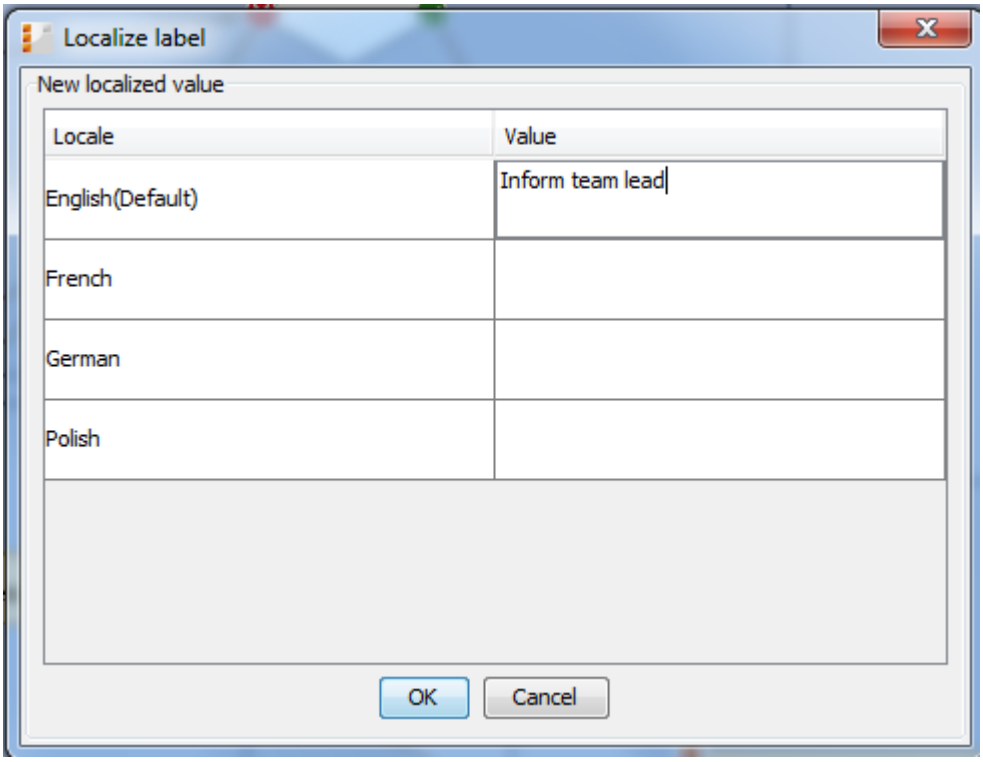


Figure 37: *ConSol CM Process Designer - Localization for labels*

- **description**

  Optional. A localized description which is displayed as a mouse-over in the Web Client. This might help the engineer to understand what will happen when the respective workflow activity is executed.

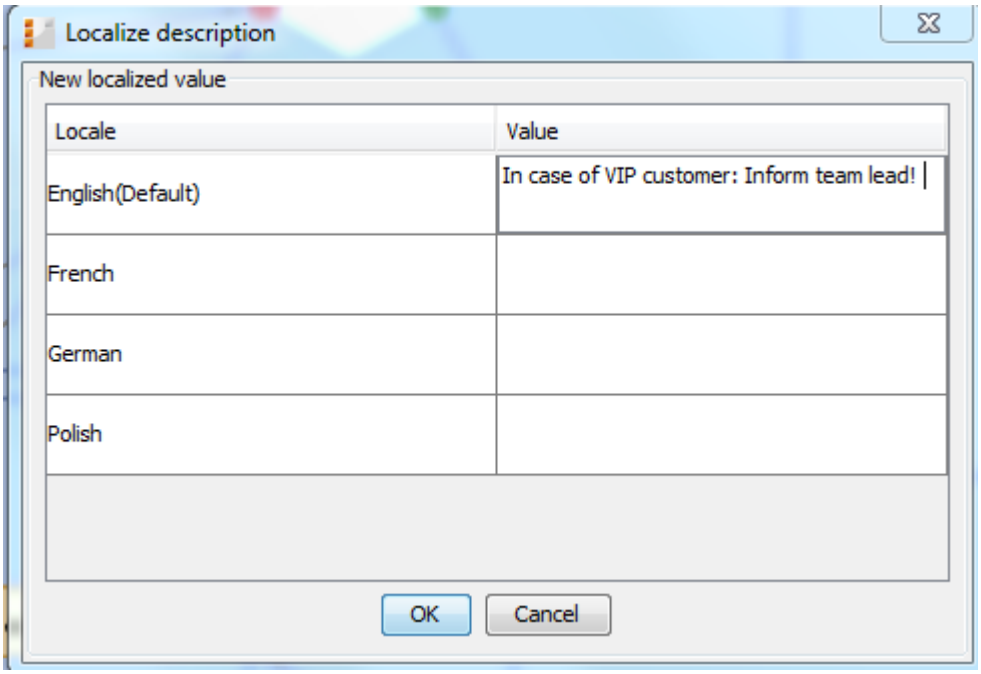


Figure 38: *ConSol CM Process Designer - Localized description of an activity*

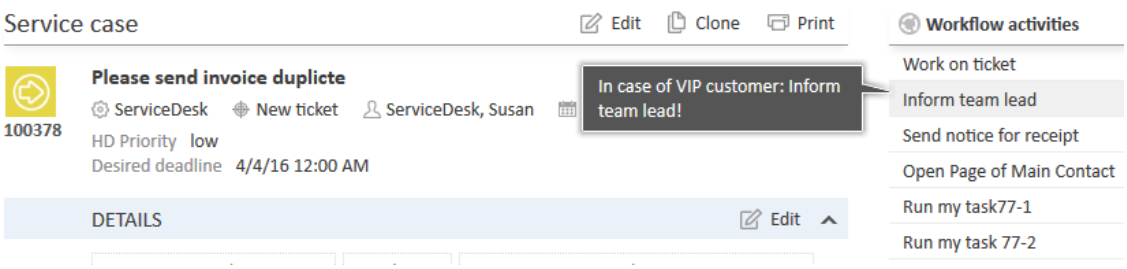


Figure 39: *ConSol CM Web Client - Localized description of an activity as mouse-over*

- **sort index**

  Defines the order of the activities in the list of workflow activities in the Web Client. The activities with a lower number are displayed first in the list.

- **overlay**

  Optional. Click into the orange space to define a standard ConSol CM overlay or one that has already been uploaded. Click the three dots button on the right to open the file explorer of the operation system to upload a new icon. When the ticket passes through an activity the overlay is added to the ticket icon in the Web Client. Up to three overlays can be attached to a ticket icon. This mechanism can be used for several purposes, some examples are:

  - **An escalation:**
    The ticket has been created without any engineer taking care of it.
  - **An email:**
    The ticket has received an email.
  - **A note for the engineer:**
    E.g. another engineer has added a comment to *my* ticket.

Figure 40: *ConSol CM Process Designer - Properties editor: Standard overlays and one customer-defined overlay*

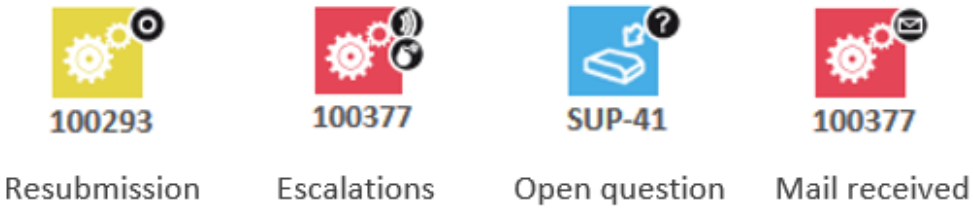


Figure 41: *ConSol CM Web Client - Ticket icons with overlays*

- **overlay range**

  Only displayed when an overlay has been set.

- **Activity**
  The overlay is attached only as long as the ticket stands behind the activity or - if present - behind the last automatic activity which can be passed without manual interaction. As soon as the next manual activity is executed, the overlay is deleted from the ticket icon.
- **Scope**
  The overlay is deleted when the ticket leaves the scope.
- **Process**
  Once the overlay has been attached to the ticket icon, it stays there for the rest of the process.
- **Next overlay**
  The overlay is attached to the ticket icon as long as no new overlay appears. In that case, only the new one is attached, the old one is deleted.

In case you have to cover a use case where it is not sufficient to set or remove an overlay using the overlay range, you can use a workflow script. Please see section Working with overlays for details.

- **precondition**

  Optional. A script can be entered using the script editor (see section Activities). The script has to return “true” or “false”. It is executed when the previous activity has been performed, i.e. when it becomes possible to display the activity with the precondition. In case “true” is returned, the activity is displayed, in case “false” is returned, the activity is not displayed. An activity which has a precondition is marked by the icon *exclamation mark*.

  - Return value is “true”.
    The activity is displayed. If it is a manual activity it can be performed by the engineer in the Web Client.
  - Return value is “false”.
    The activity is not displayed in the Web Client.

  > **Important note about customer fields**
  > When you work with customer fields, i.e. with data fields that contain customer data, please keep in mind that it might be required to consider the data models of different customer groups in case a workflow is used for queues which have been assigned to more than one customer group!

- **script**

  Optional. A script can be entered using the script editor (see section Activities) which is executed when the ticket enters the activity.

- **activity type**
  - **automatic**
    The activity is performed automatically by the system. The action is transparent for the engineer.
  - **manual**
    The activity is marked with the red hand/manual icon in the Process Designer GUI. The

activity is available as workflow activity in the Web Client and is performed when the engineer has clicked on this activity. It is only available in a certain, strictly defined position of the ticket within the process.

- **scope**
  The activity is marked with the blue hand/manual icon in the Process Designer GUI. The activity is available like a workflow activity in the Web Client and is performed when the engineer has clicked on this activity. The scope activity is available as long as the ticket is located within the scope where the activity is placed or in one of the subordinate scopes.

For a detailed explanation of the ConSol CM process logic, please see section Process logic.

- **history visibility**

  Mandatory. This property defines on which display levels the execution of the activity is shown in the ticket history in the Web Client. The possible values are:

  - 2nd level and 3rd level
  - only 3rd level
  - on every level
  - default (default value)

    The activity is shown on the display level which is configured in the Admin Tool, navigation group *Tickets*, navigation item *History*. Depending on the type of activity, one of the following settings is used:

    - Manual activity or activity with overlay executed
    - Activity executed after escalation
    - Automatic activity executed
  - hidden on all levels
    The execution of the activity is never displayed in the ticket history of the Web Client.

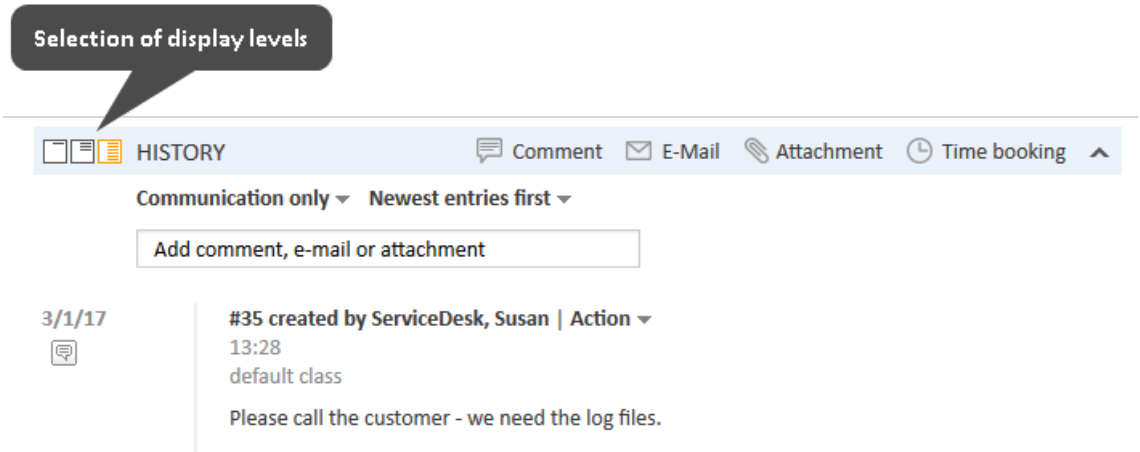


Figure 42: *ConSol CM Web Client - Display levels in the ticket history*

- **disable auto update**

Defines the behavior of the ticket when an event has been fired or executed. Usually, after an event, a ticket update operation is performed automatically. In case a chain of events is used you should avoid triggering a ticket update operation after every single event. To avoid this, set *disable auto update* to “true” in all events except for the last one. Then, the ticket is only updated once, after the last event.

- **expose to customers**

  Boolean. Available for manual activities only. If checked, the activity is displayed as a workflow activity in the ConSol CM portal, CM/Track. The activity is only displayed in CM/Track if no pre-condition script is set or if the precondition script returns “true”. If an ACF (Activity Control Form) is attached to the activity, it is opened when the customer clicks the activity. An example of an ACF in CM/Track is shown in section ACFs in CM/Track.

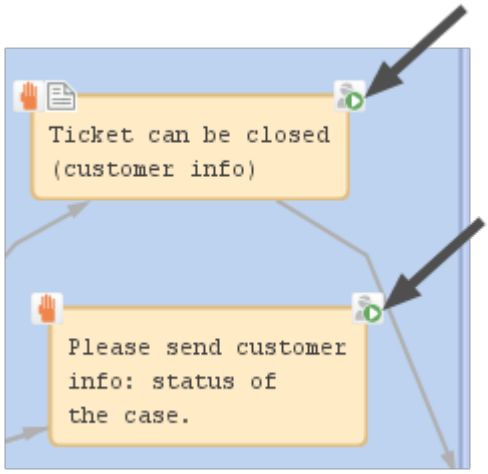


Figure 43: *ConSol CM Process Designer - Two activities which should be displayed for CM/Track customers ("exposed to customers")*

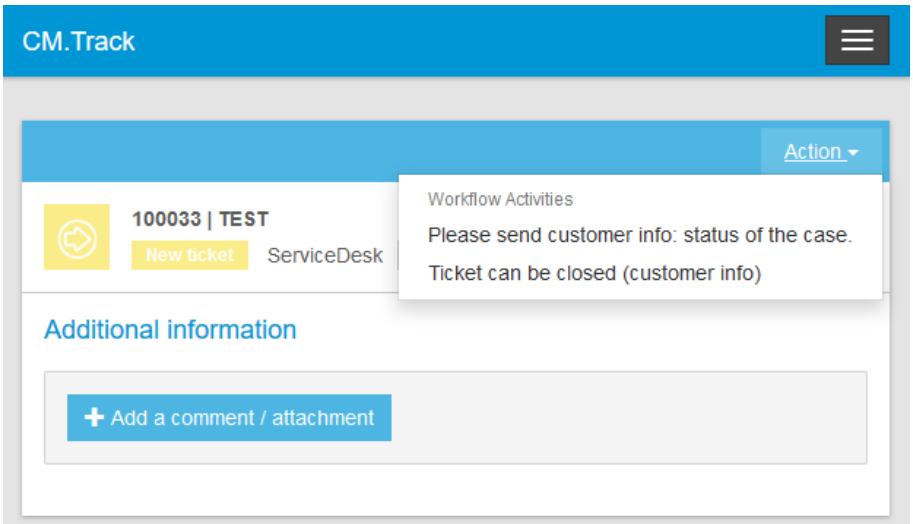


Figure 44: *CM/Track (i.e. customer perspective) - Two activities which are displayed in CM/Track for customers ("exposed to customers")*

C.4.3 Process logic of activities

This is the process logic of activities:

- When a ticket has passed through an activity it always waits behind this activity (and not before the next one!).
- When a ticket has passed through an activity, it checks if one of the potential subsequent activities is an automatic activity. If yes, the ticket passes through this automatic activity as well. This is why there can only be one automatic activity in a process step.
- The ticket passes automatically through (automatic) activities as long as there are new automatic activities. It comes to a halt as soon as there is/are one or more manual activities where engineer interaction is required.
- If one or more of the following manual activities have a precondition script, this script is executed in order to decide if the activity has to be displayed in the Web Client GUI or not.
- If the engineer selects the activity in the Web Client GUI, the script of the activity is executed.
- If there is a `postActivityScript`, this script is executed immediately after the execution of the activity script.
- The ticket waits behind the manual activity. If the following activity is located in a new scope, the ticket will not enter the new scope. It always waits behind the old activity and not before the new one!

In case the activity has an ACF, the Business logic of ACFs also has to be considered.

C.4.3.1 Remark on scope activities

A scope activity does not have any incoming connections, but it can have

- a condition script
- an activity script
- outgoing connections

When a scope activity has been executed, the subsequent position of the ticket depends on the workflow topology:

- If there are no subsequent activities which are directly linked to the scope activity, the ticket executes the scope activity and returns to its previous position within the workflow. This is an interrupt.
- If there are subsequent activities which are linked to the scope activity, the ticket will execute these activities. It will wait after the scope activity, if the subsequent activity is a manual activity. It will pass on through a subsequent automatic activity. This behavior represents an exception. It is even possible that the ticket leaves the workflow (the queue), if a jump-out node is positioned behind a scope activity, or that the ticket is closed, if an end node is positioned behind a scope activity.

> ⚠ A ticket always waits behind the last activity which has been executed and not before the new one!
> This is relevant, for example, when a view is defined: it is important to know that the ticket might not have entered the next scope, because it is still waiting behind the previous activity in the current scope.

If the ticket temporarily leaves its scope due to the execution of a scope activity, triggers are restored when the ticket reenters the scope afterwards.

CM/ is a registered trademark of ConSol Consulting & Solutions Software GmbH

C.4.4 Examples of activities

C.4.4.1 Example 1: Precondition for displaying an activity

In case the ticket has been opened by a VIP contact, i.e., a contact where the boolean field *vip* is "true", the team lead should be informed. If it is no VIP, the activity *Inform team lead* should not be displayed. The customer field *vip* which is part of the customer data model is checked for this purpose.

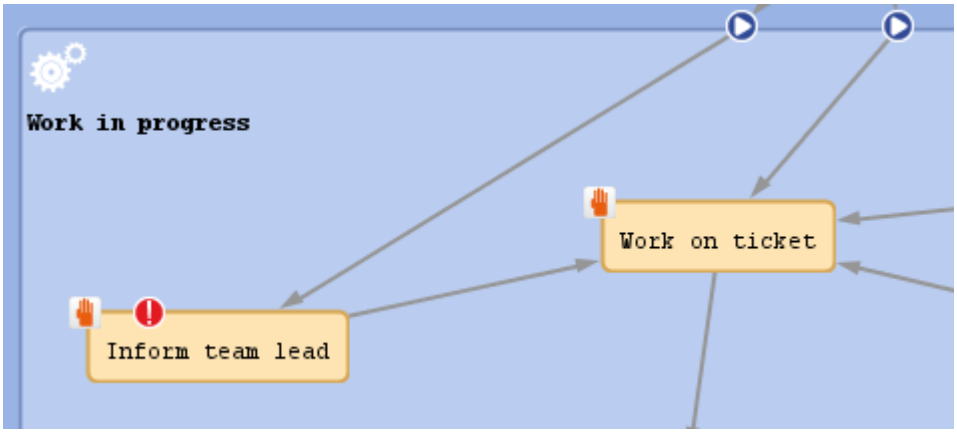


Figure 45: *ConSol CM Process Designer - Workflow activities (one with precondition script)*

```
import com.consol.cmas.core.server.service.*
import com.consol.cmas.common.model.customfield.meta.UnitDefinitionType
def ticket = workflowApi.ticket
// fetch main contatc of the ticket
def maincontact = ticket.getMainContact()
def unit_type = maincontact.definition.type
log.info 'vipCheck: Unittype is ' + unit_type
log.info 'vipCheck: Unittype class is ' + unit_type.getClass()
if(unit_type == UnitDefinitionType.COMPANY) {
   log.info 'No vipCheck for comapnies possible! Returning false ... '
   return false
} else if (unit_type == UnitDefinitionType.CONTACT){

// fetch e-mail address of the man contact. The data object group field has to be
 addressed using data object group name:data object group field name
   def vip_field
   def custgroup = maincontact.customerGroup.name
   println 'vipCheck: Customergroup is now ' + custgroup
   switch(custgroup) {
      case "Reseller": vip_field = "vip_person";
      break;
      case "DirectCustomers": vip_field = "vip_dircust"
      break;
      case "MyCustomerGroup": vip_field = "vip"
      break;
      case "OurPartnerCompanies": vip_field = "vip_partners"
      break;
      case "RetailCustomers": vip_field = "retail_vip"
      break;
```

```
    }
    def vip_value = maincontact.get(vip_field)
    log.info 'VIP is now ' + vip_value

    if (vip_value){
       return true
    } else {
       return false
    }
}
```

Code example 2: *Precondition script: activity should only be displayed for VIP customers*

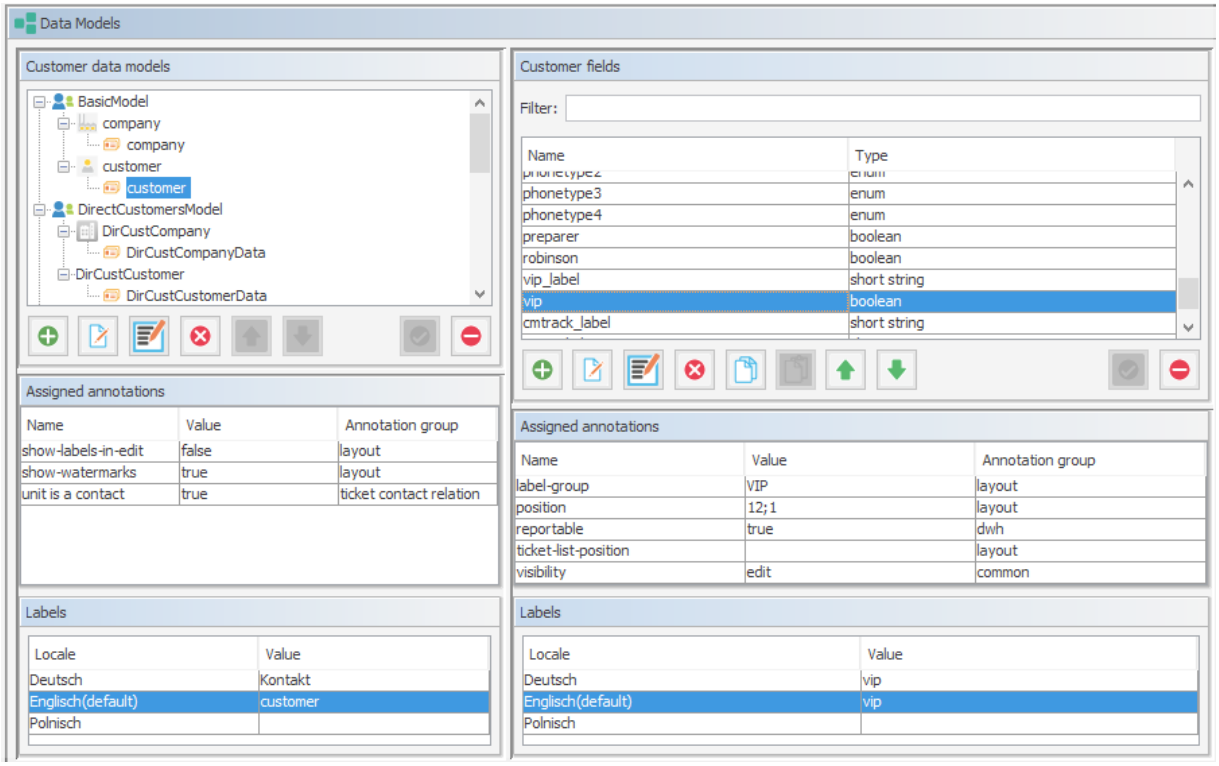


Figure 46: *ConSol CM Admin Tool - Customer field "vip"*

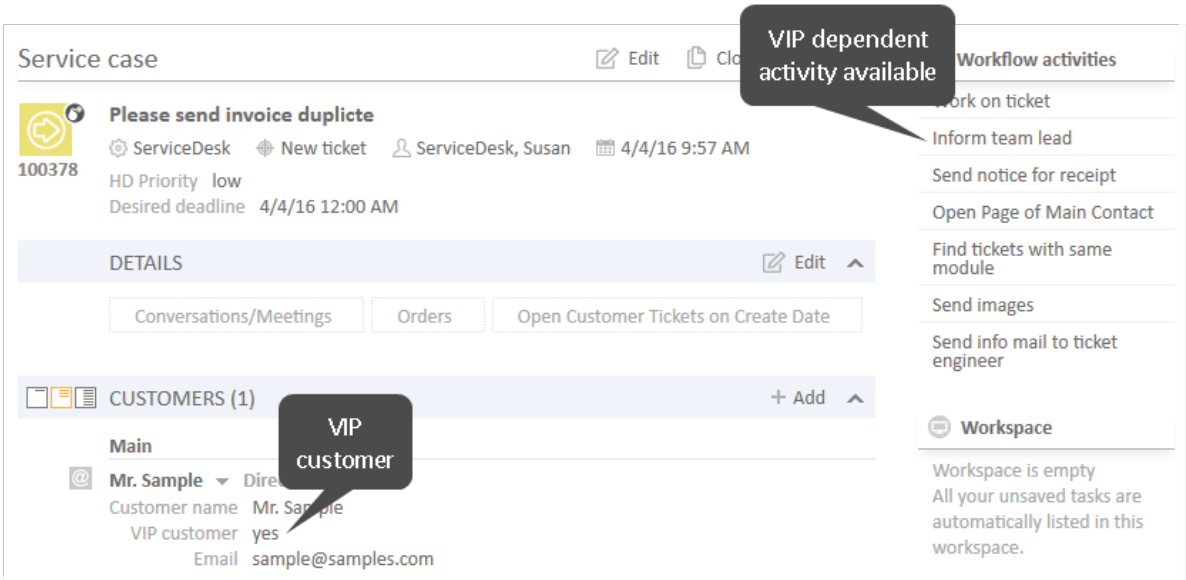


Figure 47: *ConSol CM Web Client - Precondition: Return value TRUE*

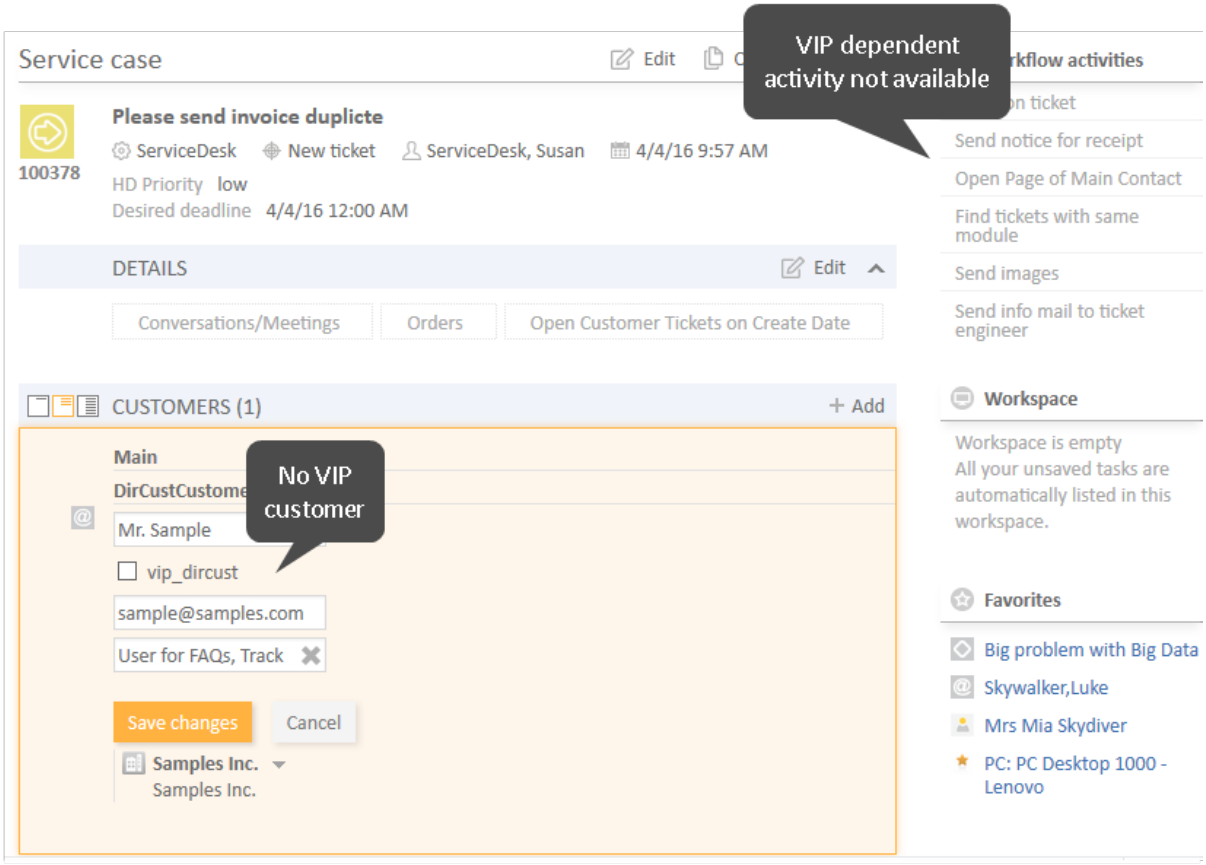


Figure 48: *ConSol CM Web Client - Precondition: Return value FALSE*

C.4.4.2 Example 2: Sending an email to the main contact after opening a ticket

When a ticket has been opened, an email should be sent automatically to the main contact of the ticket.

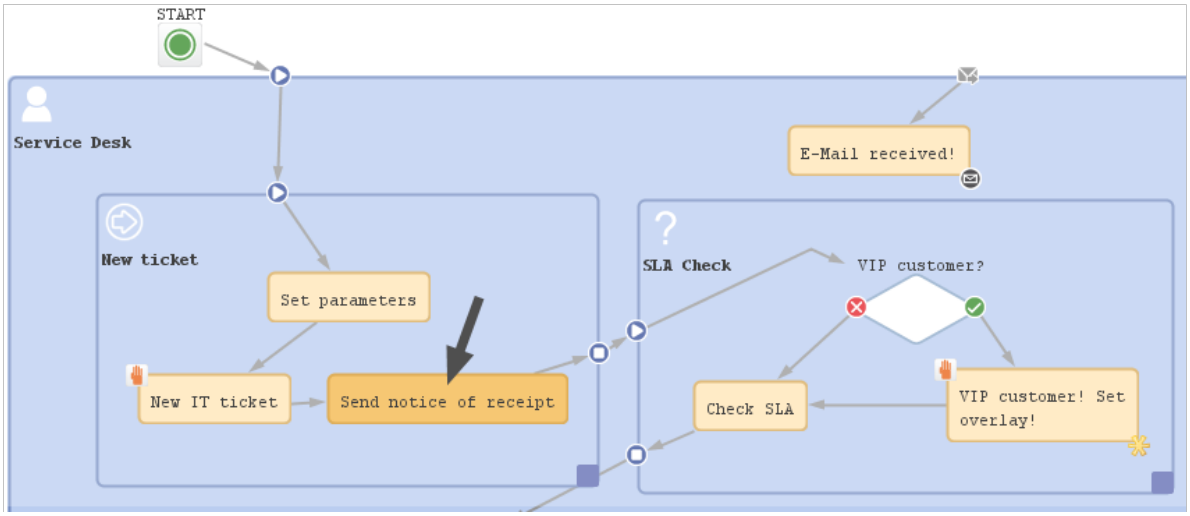


Figure 49: *ConSol CM Process Designer - Automatic activity where receipt note is sent*

Script in workflow activity *Send notice of receipt*:

```
log.info'Calling SendReceiptNotice script ...'
scriptExecutionService.execute("SendReceiptNotice.groovy")
```

Script `SendReceiptNotice.groovy` in Admin Tool:

```
import com.consol.cmas.common.model.mail.Mail
import com.consol.cmas.common.util.MailHeadersUtil;
import com.consol.cmas.core.server.service.*;
import static com.consol.cmas.common.util.TemplateUtil.TICKET_SUBJECT_TEMPLATE_
 NAME;
import com.consol.cmas.common.model.customfield.meta.UnitDefinitionType
// create new mail object
def mail = new Mail()
def ticket = workflowApi.getTicket()

// fetch main contatc of the ticket
def maincontact = ticket.getMainContact()
def unit_type = maincontact.definition.type
log.info 'Mailscript: Unittype is ' + unit_type
log.info 'Mailscript: Unittype class is ' + unit_type.getClass()

if(unit_type == UnitDefinitionType.COMPANY) {
   println 'No email address for company; no receipt notice sent.'
   return
} else if (unit_type == UnitDefinitionType.CONTACT){

   // fetch e-mail address of the man contact. The data object group field has to be
    addressed using data object group name:data object group field name
   def toaddress_field
   def custgroup = maincontact.customerGroup.name

   switch(custgroup) {
      case "Reseller": toaddress_field = "email";
```

```
      break;
      case "DirectCustomers": toaddress_field = "dir_cust_email"
      break;
      case "MyCustomerGroup": toaddress_field = "email"
      break;
      case "OurPartnerCompanies": toaddress_field = "email"
      break;
      case "RetailCustomers": toaddress_field = "retail_customer_email"
      break;
   }
   def toaddress = maincontact.get(toaddress_field)
   if (!toaddress){
      log.info 'No email address found for contact, no receipt notice sent.'
   } else {
      // put the e-mail TO address into the Mail object
      mail.setTo(toaddress)

      // fetch the REPLY TO address, theis is stored in a system property
      def replyaddress = configurationService.getValue("cmweb-server-
       adapter","mail.reply.to")
      // put the e-mail REPLY TO address into the Mail object
      mail.setReplyTo(replyaddress)

      // build e-mail text using a template which is stored in the Template Designer
      def text = workflowApi.renderTemplate("Acknowledgement_of_receipt")
      // put the e-mail text into the Mail object
      mail.setText(text)

      // create the subject of the e-mail, the ticket number with the correct
       Regular Expression
      // has to be set for correct recognition of incoming e-mails for the ticket
      // ****** alternative solutions for PD manual! *****
      // solution 1:
      // def ticketname = ticket.getName()
      // def subject = "Your case has been registered as Ticket (" + ticketname +
       ")"
      // solution 2: (needs import of TemplateUtil.TICKET_SUBJECT_TEMPLATE_NAME
      def subject = templateService.merge(TICKET_SUBJECT_TEMPLATE_NAME,
       [ticketName:ticket.name])
      // put the subject into the Mail object
      mail.setSubject(subject)
      // Mail should use the e-mail script which is configured for the queue
      mail.useDefaultScript()
      // send out the e-mail and register status
      def attList = new ArrayList<AttachmentEntry>()
      def collection = new HashSet<MailEntry>()
      def mailStatus = true;

      try {
         mail.send();
      } catch (Exception e){
         mailStatus = false;
      }
```

```
   } // end if (!toaddress){
} // end of else if (unit_type.equals('COMPANY')){
```

Code example 3: *Scripts for automatic activity where receipt note is sent, variant 1*

```
// lines of code alost identical to variant 1 except for the sending of the mail:

new Mail().setSubject( subj ).setTo( contact_e ).setReplyTo( replyto ).setText(
 text ).setTicketAttachments( null ).send()
```

Code example 4: *Script for automatic activity where receipt note is sent, variant 2*

C.4.4.3 Example 3: Assign the ticket to the current engineer

The ticket should be assigned to the engineer who executes the activity *New IT ticket*.

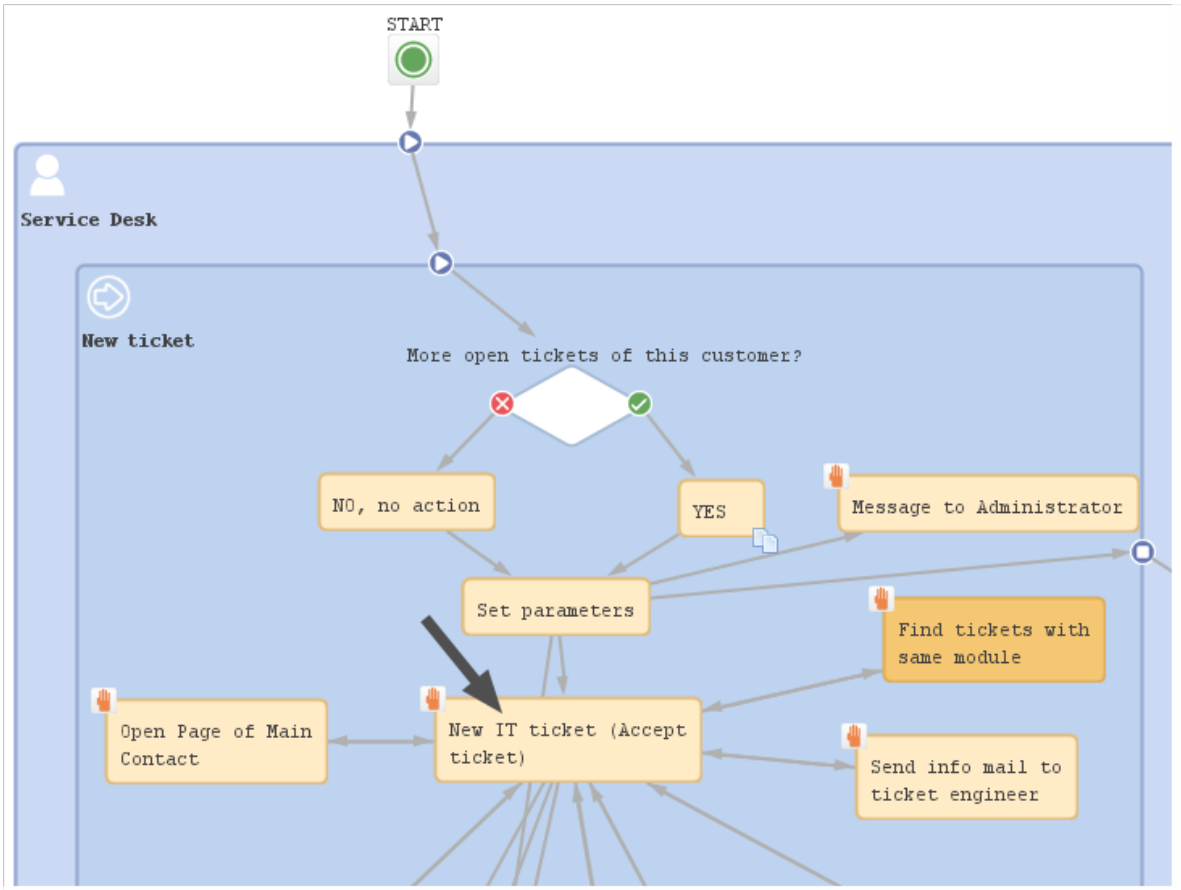


Figure 50: *ConSol CM Process Designer - Workflow activity where engineer should be assigned*

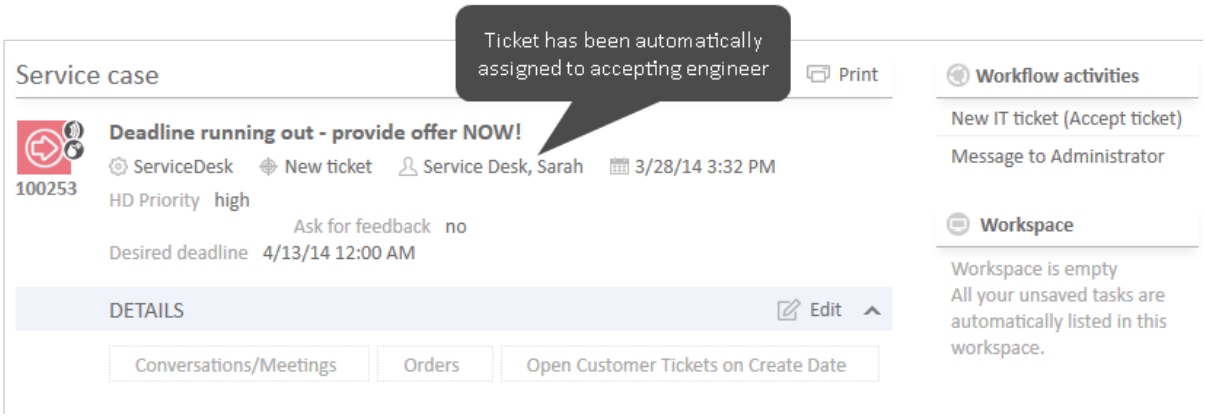


Figure 51: *ConSol CM Web Client - Ticket passed through activity where engineer was assigned*

```
// Get the engineer who is executing the activity:
// Java Style is possible: def curr_eng = workflowApi.getCurrentEngineer()
// we use Groovy style here:
def curr_eng = workflowApi.currentEngineer
// alternative would be:
// def curr_eng = engineerService.current

// Assign the ticket to the current engineer
// Java style is possible:
ticket.setEngineer(curr_eng)
// groovy alternative would be:
// ticket.engineer = curr_eng
```

Code example 5: *Script for assigning ticket to current engineer*

Make sure that you always use the correct engineer object!

The current engineer is the engineer who is logged in, who is executing the current activity. You can get the object by using one of the following methods.

Using workflowApi:

```
// Java notation
def curr_eng = workflowApi.getCurrentEngineer()
```

```
// Groovy notation
def curr_eng = workflowApi.currentEngineer
```

Using engineerService :

```
// Java notation:
def curr_eng = engineerService.getCurrent()
```

```
//Groovy notation
def curr_eng = engineerService.current
```

The ticket engineer is the person who is (at this point of time) the ticket owner and responsible for the ticket. You can get the object by using the following method:

```
//Java notation
def tic_eng = ticket.getEngineer()
```

```
//Groovy notation
def tic_eng = ticket.engineer
```

C.4.4.4 Example 4: Using a scope activity

The scope activity *Forward problem to supervizor* should be present for the entire life cycle of the ticket. For really problematic tickets, the engineer can use a "shortcut": the ticket does not have to be moved through the entire process but can be directly sent to a special management queue where a supervisor takes care of it.

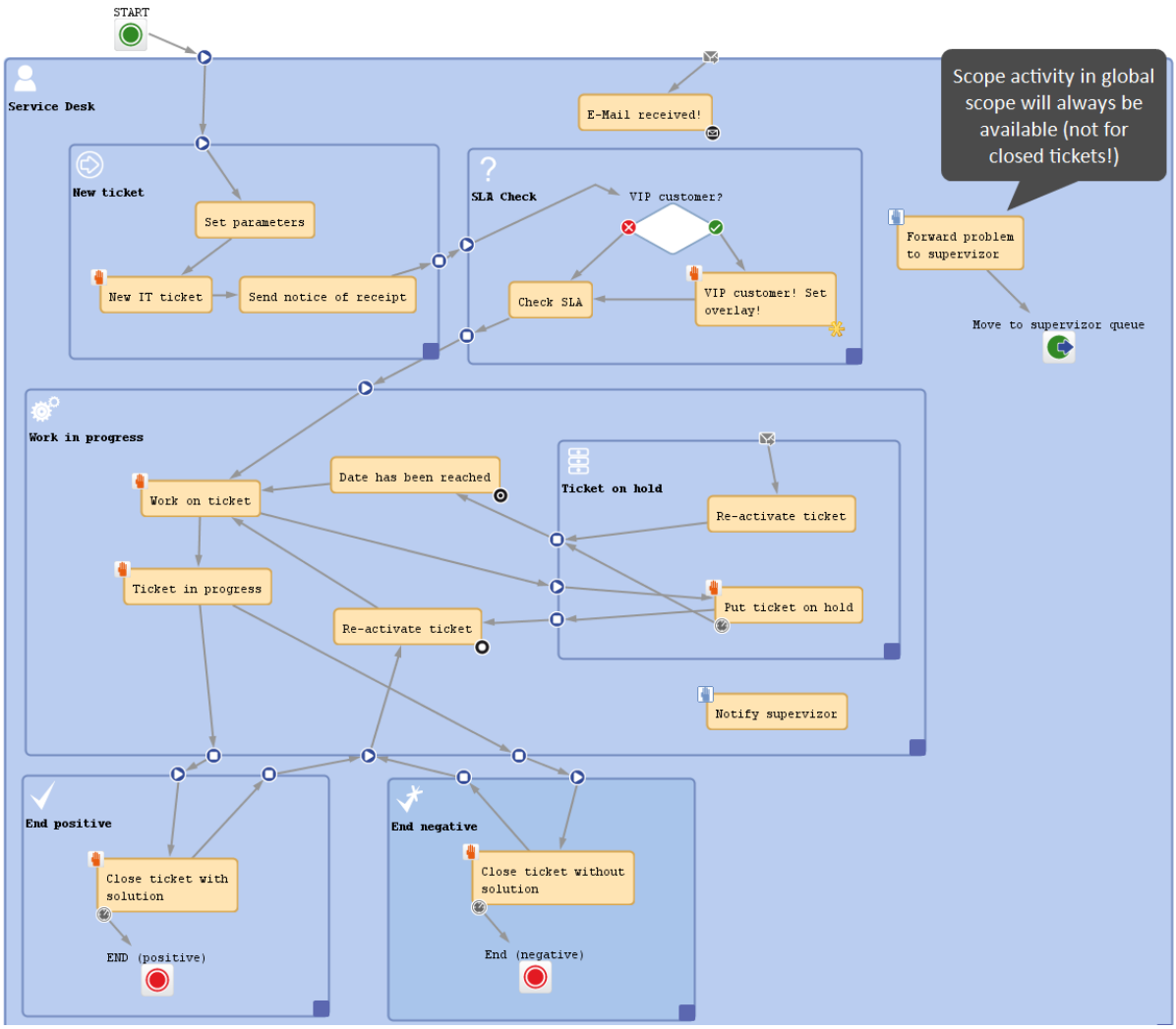


Figure 52: *A scope activity which will be available throughout the entire life cycle of the ticket, because the activity is placed in the global scope*

C.5 Reopen activities

This chapter discusses the following:

C.5.1 Introduction to reopen activities 82

C.5.2 Properties of a reopen activity 83

C.5.3 Process logic of reopen activities 89

C.5.4 Example for the use of reopen activities 90

CM/ is a registered trademark of ConSol Consulting & Solutions Software GmbH

C.5.1 Introduction to reopen activities

Once a ticket is technically closed, i.e., it reaches an end node in the workflow, it cannot be edited in any way and there are no activities for this ticket. Depending on the business process, it might be required to reopen such tickets. This can be modeled using a reopen activity.

Example use cases for reopen activities are:

- The system receives an email related to a closed ticket. Instead of creating a new ticket with a relation to the closed one, the closed ticket should be reopened.
- The customer is not satisfied with the solution given in the ticket. He wants to reopen the ticket in CM/Track.

When using a reopen activity, the ticket does **not** return to the start node, but re-enters the process at the location defined by the outgoing connections of the activity. Reopen activities do not have any incoming connections. They are visible in the scope where they are located and in any subscopes of this scope.

C.5.2 Properties of a reopen activity

Reopen activities have the following properties:

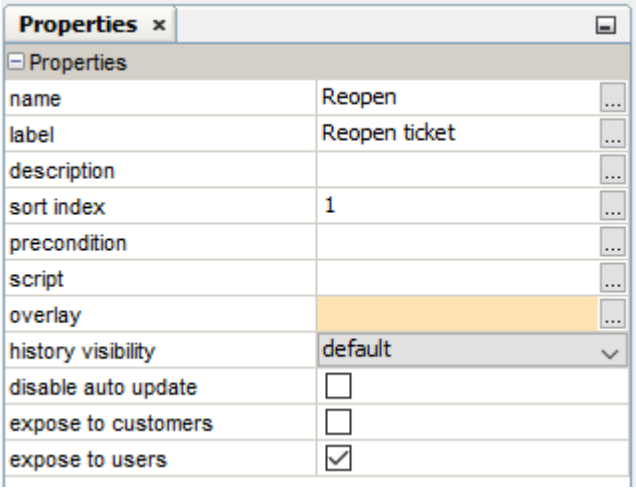


Figure 53: *ConSol CM Process Designer - Properties editor of a reopen activity*

- **name**

  Mandatory. This is the technical name of the reopen activity. When a new reopen activity is created, you can edit the label and the activity name is generated automatically from the label (special characters are omitted). Afterwards, the activity name is never changed automatically but can be edited manually. Allowed characters for names are:

  - letters (small or capital), but no special characters
  - underscores
  - numbers

- **label**

  The localized name of the reopen activity. All languages which have been configured for the system are available and can be filled. In the Web Client, the label is displayed according to the browser locale. If it is not available, the label of the default locale is shown.

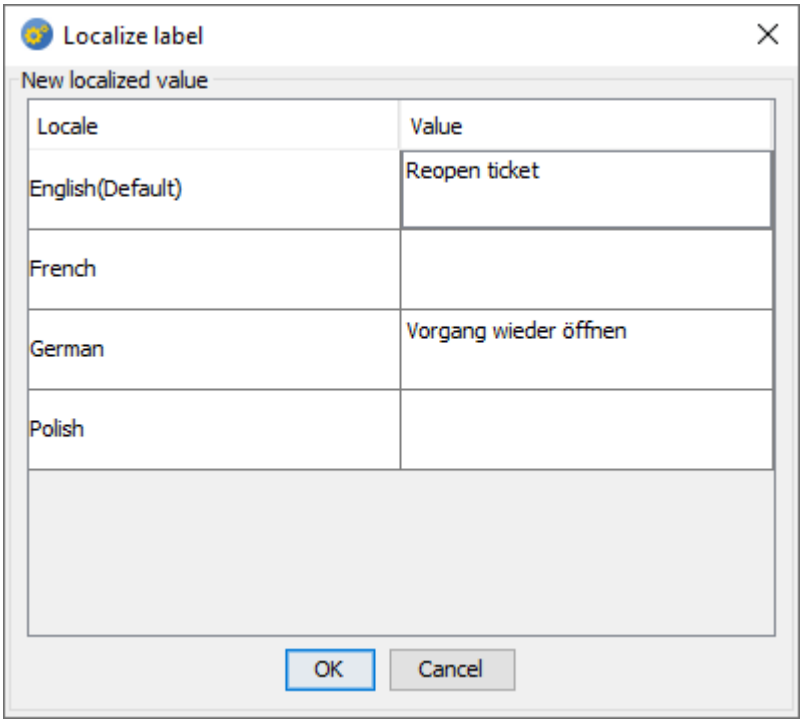


Figure 54: *ConSol CM Process Designer - Localization for labels*

- **description**

  Optional. A localized description which is displayed as a mouse-over in the Web Client. This might help the engineer to understand what will happen when the respective reopen activity is executed.

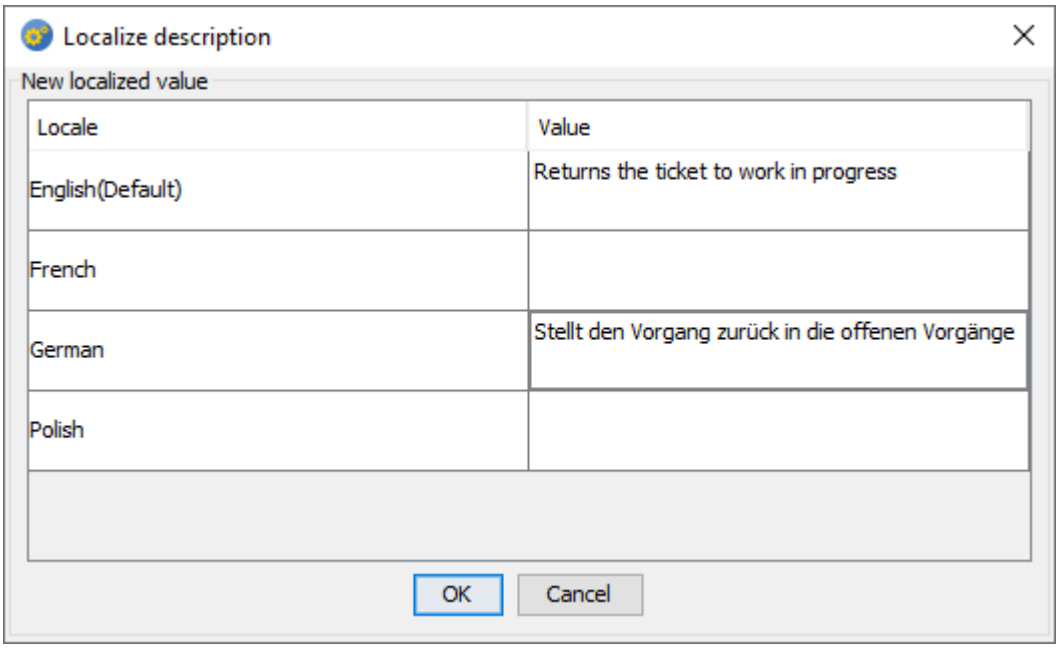


Figure 55: *ConSol CM Process Designer - Localized description of a reopen activity*

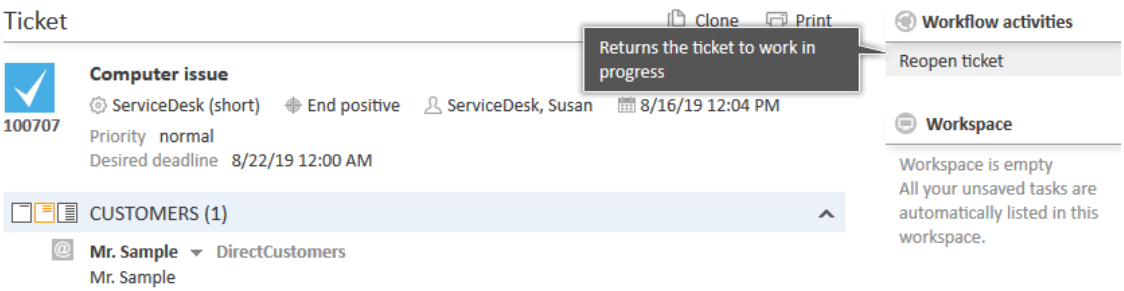


Figure 56: *ConSol CM Web Client - Reopen activity in closed ticket with localized description*

- **sort index**

  Defines the order of the activities in the list of workflow activities in the Web Client. The activities with a lower number are displayed first in the list.

- **precondition**

  Optional. A script can be entered using the script editor (see section Reopen activities). The script has to return “true” or “false”. It is executed when the previous activity has been performed, i.e., when it becomes possible to display the activity with the precondition. In case “true” is returned, the activity is displayed, in case “false” is returned, the activity is not displayed. An activity which has a precondition is marked by the icon *exclamation mark*.

  - Return value is “true”.
    The activity is displayed and can be performed by the engineer in the Web Client.
  - Return value is “false”.
    The activity is not displayed in the Web Client.

- **script**

  Optional. A script can be entered using the script editor (see section Reopen activities) which is executed when the ticket enters the activity.

- **overlay**

  Optional. Click into the orange space to define a standard ConSol CM overlay or one that has already been uploaded. Click the three dots button to open the file explorer of the operation system to upload a new icon. When the ticket passes through an activity the overlay is added to the ticket icon in the Web Client. Up to three overlays can be attached to a ticket icon. This mechanism can be used for several purposes, some examples are:

  - **An escalation:**
    The ticket has been created without any engineer taking care of it.
  - **An email:**
    The ticket has received an email.
  - **A note for the engineer:**
    E.g. another engineer has added a comment to *my* ticket.

Figure 57: *ConSol CM Process Designer - Properties editor: Standard overlays and one customer-defined overlay*

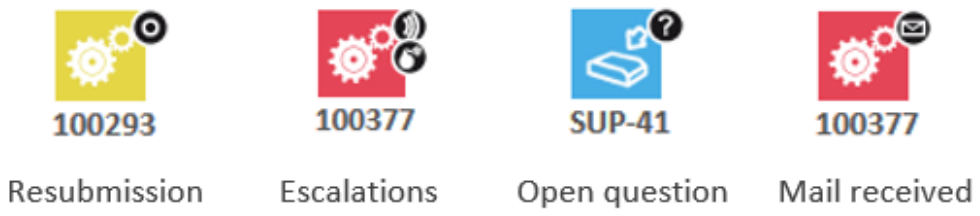


Figure 58: *ConSol CM Web Client - Ticket icons with overlays*

- **overlay range**

  Only displayed when an overlay has been set.

  - **Activity**
    The overlay is attached only as long as the ticket stands behind the activity or - if present - behind the last automatic activity which can be passed without manual interaction. As soon as the next manual activity is executed, the overlay is deleted from the ticket icon.
  - **Scope**
    The overlay is deleted when the ticket leaves the scope.
  - **Process**
    Once the overlay has been attached to the ticket icon, it stays there for the rest of the process.
  - **Next overlay**
    The overlay is attached to the ticket icon as long as no new overlay appears. In that case, only the new one is attached, the old one is deleted.

  In case you have to cover a use case where it is not sufficient to set or remove an overlay using the overlay range, you can use a workflow script. Please see section Working with overlays for details.

- **history visibility**

  Mandatory. This property defines on which display levels the execution of the activity is shown in the ticket history in the Web Client. The possible values are:

  - 2nd level and 3rd level
  - only 3rd level
  - on every level
  - default (default value)

    The activity is shown on the display level which is configured in the Admin Tool, navigation group *Tickets*, navigation item *History*. Depending on the type of activity, one of the following settings is used:

    - Manual activity or activity with overlay executed
    - Activity executed after escalation
    - Automatic activity executed
  - hidden on all levels
    The execution of the activity is never displayed in the ticket history of the Web Client.

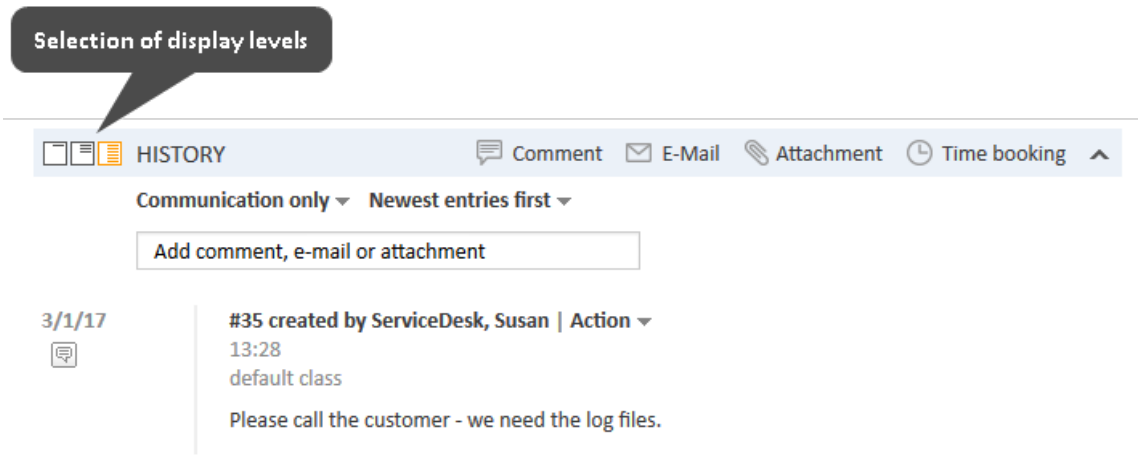


Figure 59: *ConSol CM Web Client - Display levels in the ticket history*

- **disable auto update**

  Defines the behavior of the ticket when an event has been fired or executed. Usually, after an event, a ticket update operation is performed automatically. In case a chain of events is used you should avoid triggering a ticket update operation after every single event. To avoid this, set *disable auto update* to “true” in all events except for the last one. Then, the ticket is only updated once, after the last event.

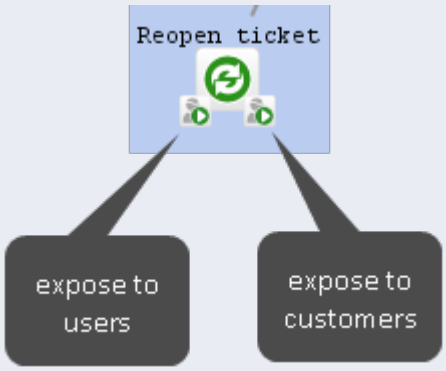
- **expose to customers**

  Boolean. If checked, the activity is displayed as a workflow activity in the ConSol CM portal, CM/Track. The activity is only displayed in CM/Track if no precondition script is set or if the precondition script returns “true”. If an ACF (Activity Control Form) is attached to the activity, it is opened when the customer clicks the activity. An example of an ACF in CM/Track is shown in section ACFs in CM/Track.

- **expose to users**

  Boolean. If checked, the activity is displayed as a workflow activity in the Web Client. The activity is only displayed if no precondition script is set or if the precondition script returns “true”. If an ACF (Activity Control Form) is attached to the activity, it is opened when the user clicks the activity.

The icons for *expose to customers* and *expose to users* are identical. *expose to users* is displayed on the left and *expose to customers* on the right:



If neither *expose to users* nor *expose to customers* is selected, the reopen activity can only be executed via script.

C.5.3 Process logic of reopen activities

The ticket is reopened as soon as the reopen activity is executed. The further processing depends on the outgoing connections of the reopen activity. The process logic of reopen activities is the same as for regular activities (see Process logic of activities).

Please consider the impact of reopening closed tickets on the designed workflow as a whole. The following list shows some examples of areas which might need adaption when introducing reopen activities.

- **Automatic actions on ticket closure**

  If the workflow contains automatic actions which are performed when a ticket is closed, you might need to adapt them for reopened tickets to avoid executing them more than once. It might even be required to undo some of these actions when a ticket is reopened.

  Example: An FAQ ticket is automatically created when a ticket is closed with a solution. Now, the original ticket is reopened because the solution did not work as expected, and the FAQ ticket needs to be updated accordingly.

- **Ticket relations**

  If the workflow implements a specific behavior regarding parent-child and master-slave relations, reopening a ticket could require changes to other tickets as well.

  Example: A parent ticket can only be closed if all its child tickets are closed. Now, a child ticket is reopened and the parent ticket needs to react on this change.

- **Reports/dashboards**

  If the number of closed tickets is analyzed in reports or dashboards, you need to decide how to handle reopened tickets. Otherwise, the report data might be inaccurate for reopened tickets, e.g. because the same ticket is counted more than once.

> Use a script in the reopen activity or place an automatic activity containing a script after the reopen activity in order to add the required logic.

C.5.4 Example for the use of reopen activities

C.5.4.1 Example 1: Reopen a closed ticket when an email is received

The default behavior when an email is received for a closed ticket is to create a new ticket and create a relation between the new ticket and the original ticket for which the email was received. If you would like to reopen the original ticket instead, you can add a reopen activity to the scope where the end node is located and execute this activity in the incoming email processing script.

Please follow the following steps:

1. Add a reopen activity to the scope or scopes containing end nodes.
2. Create a connection from the reopen activity to the activity where the ticket processing should continue.
3. Edit the `NimhIncomingMailRouting.groovy` script in the Admin Tool to add the logic to reopen the ticket.
4. Deploy the workflow and test it.

The following figure shows the part of the workflow with the end scope and reopen activity.

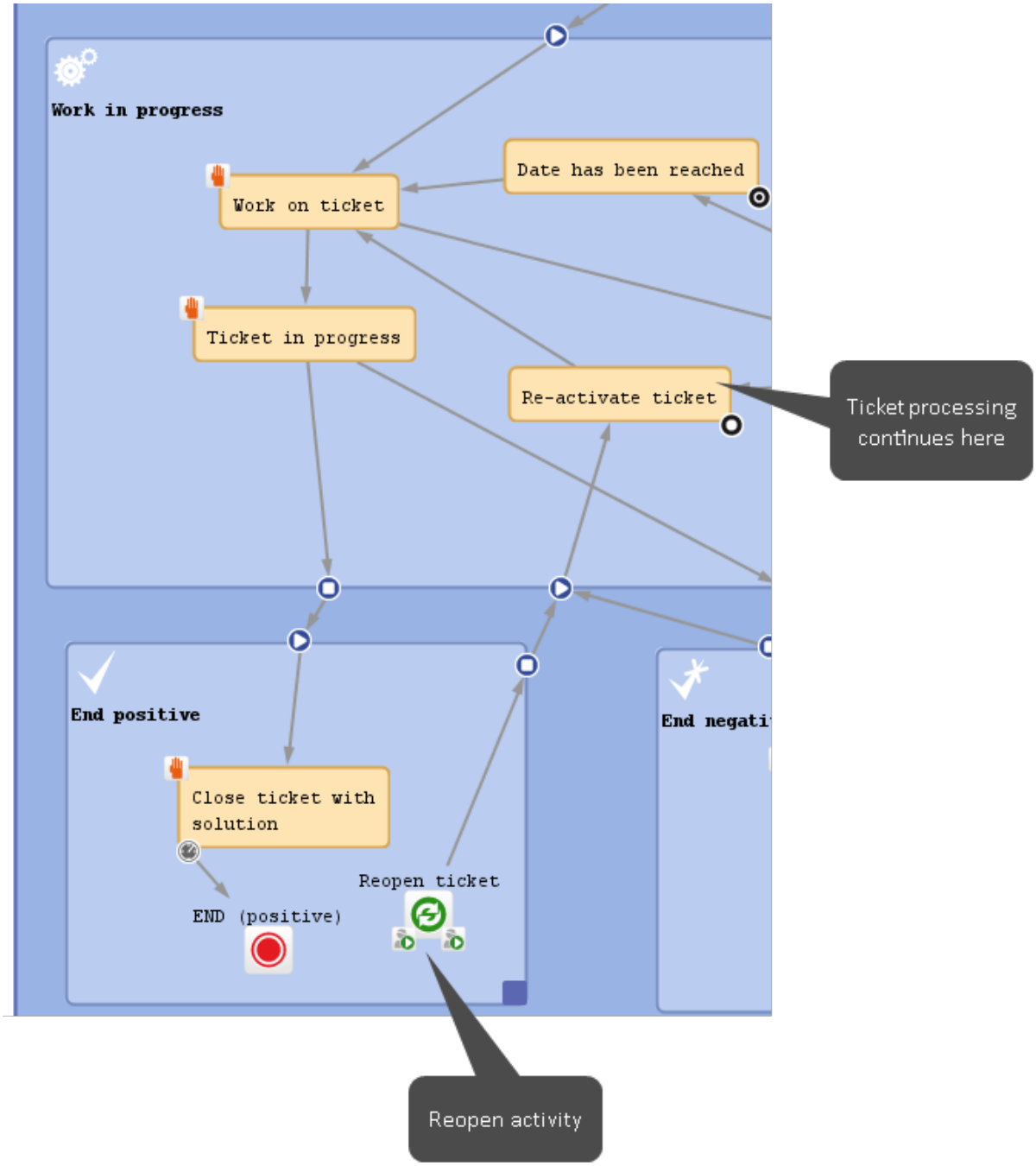


Figure 60: *ConSol CM Process Designer - Workflow with a reopen activity*

The following example shows the `NimhIncomingMailRouting.groovy` script from the Admin Tool. The lines highlighted in red have been added to reopen the ticket if the reopen activity is present.

```
import com.consol.cmas.common.model.ticket.Ticket

mailLog.info("Routing " + mailHolder.getUid())
if (log.isDebugEnabled()) {
   log.debug("Available endpoints for mail routing are " + endpoints.keySet().join
    (", "))
}

// Append to ticket if ticket id can be extracted
def ticketName = mailSupportService.extractTicketNameFromMail(mailHolder,
 pipeContext, TICKET_NAME_PATTERN_FORMAT)
if (ticketName) {
   Ticket ticket = ticketService.getByName(ticketName)
   if (ticket) {
      pipeContext.setAttribute("ticket-id", ticket.getId())
      if (ticket.getScopeInfo().isClosedOrDeleted()) {

         //if ticket is closed, check if it can be reopened
         if (ticketService.getNextActivities(ticket).name.contains
          ("defaultScope/Service_Desk/End_positive/Reopen")) {
            activity = ticketService.getNextActivities(ticket).first()
            ticketService.executeActivity(ticket, activity)
            return endpoints["appendToTicketScript"]
         } else {
            return endpoints["mailToClosedTicketScript"]
         }
      } else {
         return endpoints["appendToTicketScript"]
      }
   }
}

// Default is creating a new ticket
return endpoints["createTicketScript"]
```

Code example 6: *NimhIncomingMailRouting.groovy adapted to reopen a ticket if an email is received*

C.6 Decision nodes

This chapter discusses the following:

C.6.1 Introduction to decision nodes 93

C.6.2 Properties of a decision node 94

C.6.3 Example of a decision node 96

C.6.1 Introduction to decision nodes

A decision node is a node which has one or more entry points and exactly two exit points: “true” and “false”. A decision node always has to have a script which has to return either “true” or “false”.

The ticket enters the decision node, then the script is executed and - depending on the result (“true” or “false”) - the ticket leaves the node via the respective exit point.

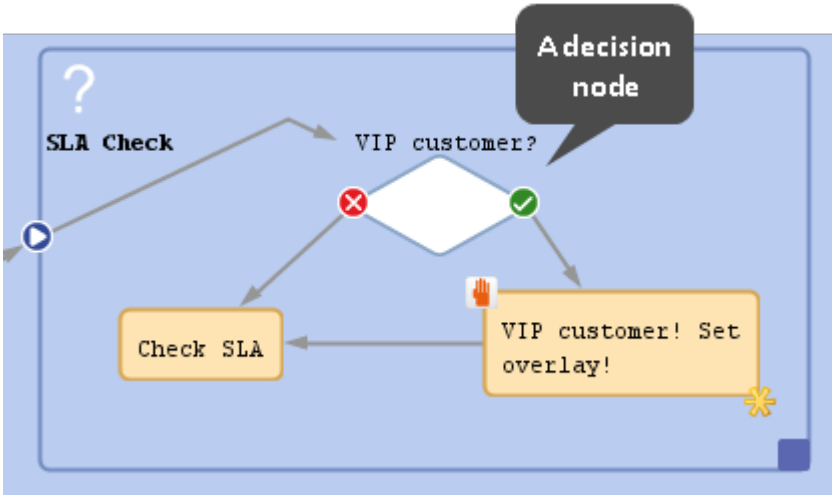


Figure 61: *ConSol CM Process Designer - Decision node*

C.6.2 Properties of a decision node

A decision node has the following properties:

- **name**

  Mandatory, the technical object name.

- **label**

  Optional, the localized name which is displayed in the Web Client GUI.

- **condition**

  Mandatory, a script which returns “true” or “false” has to be provided.

- **history visibility**

  Mandatory. This property defines on which display levels the execution of the activity is shown in the ticket history in the Web Client. The possible values are:

  - 2nd level and 3rd level
  - only 3rd level
  - on every level
  - default (default value)

    The activity is shown on the display level which is configured in the Admin Tool, navigation group *Tickets*, navigation item *History*. Depending on the type of activity, one of the following settings is used:

    - Manual activity or activity with overlay executed
    - Activity executed after escalation
    - Automatic activity executed

  - hidden on all levels
    The execution of the activity is never displayed in the ticket history of the Web Client.

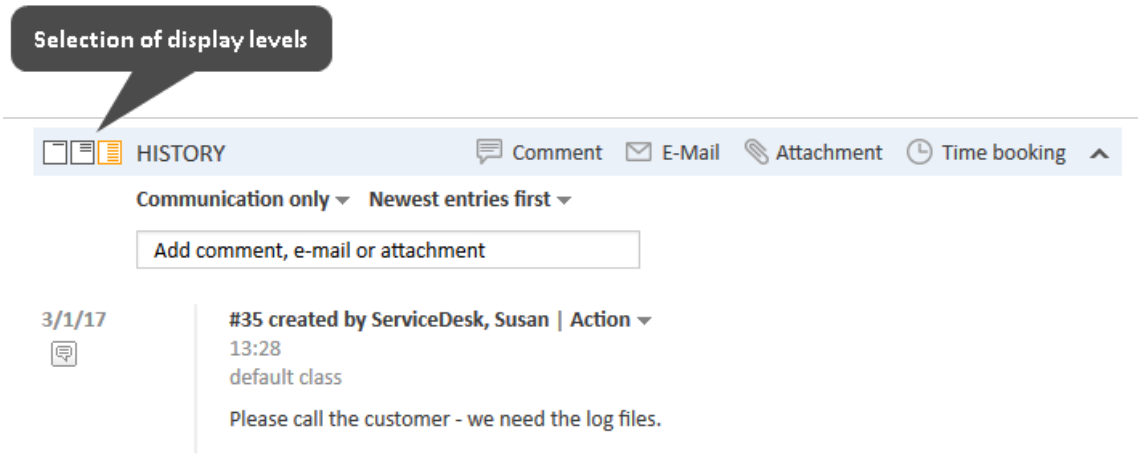


Figure 62: *ConSol CM Web Client - Display levels in the ticket history*

- **disable auto update**

Defines the behavior of the ticket when an event has been fired or executed. Usually, after an event, a ticket update operation is performed automatically. In case a chain of events is used you should avoid triggering a ticket update operation after every single event. To avoid this, set *disable auto update* to “true” in all events except for the last one. Then, the ticket is only updated once, after the last event.

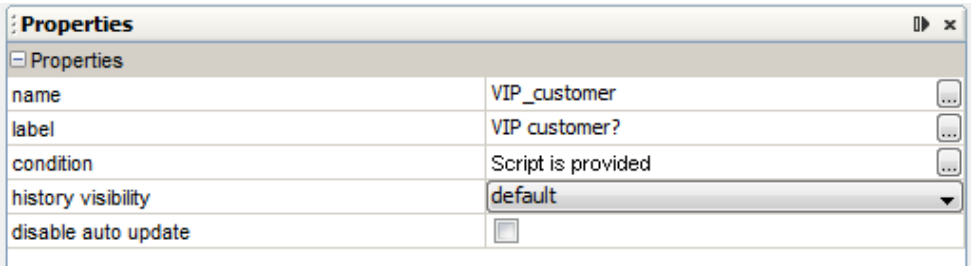


Figure 63: *ConSol CM Process Designer - Decision node: Properties*

C.6.3 Example of a decision node

In the following example, the system should automatically check if the customer (main contact of the ticket) is a *VIP* customer. If yes, the ticket should be marked with the *VIP* overlay (in the example a yellow star).

- A customer field of type BOOLEAN has to be defined in the customer data model (FlexCDM) to mark a customer as *VIP* (yes/no). Please refer to the *ConSol CM Administrator Manual*, section *Setting Up the Customer Data Model*.

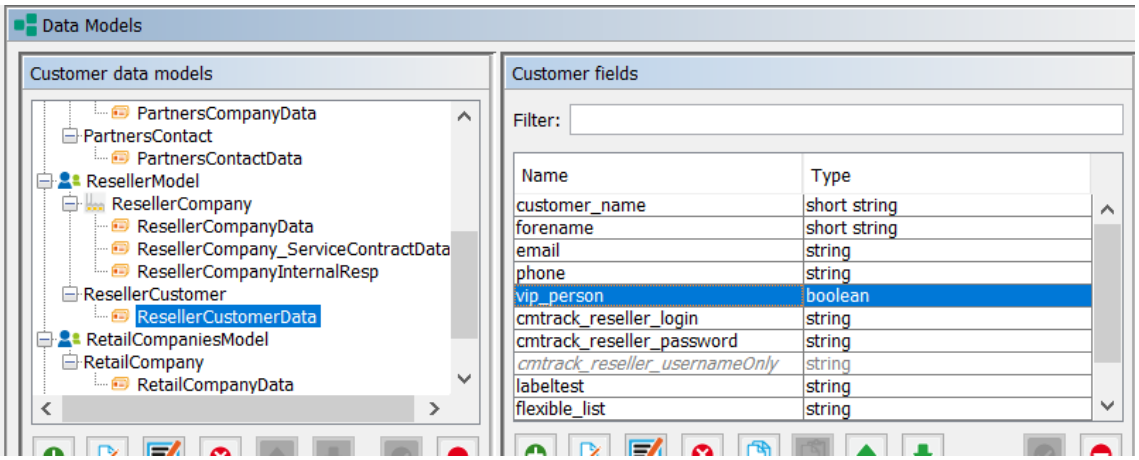


Figure 64: *ConSol CM Admin Tool - Customer field "vip" in customer data*

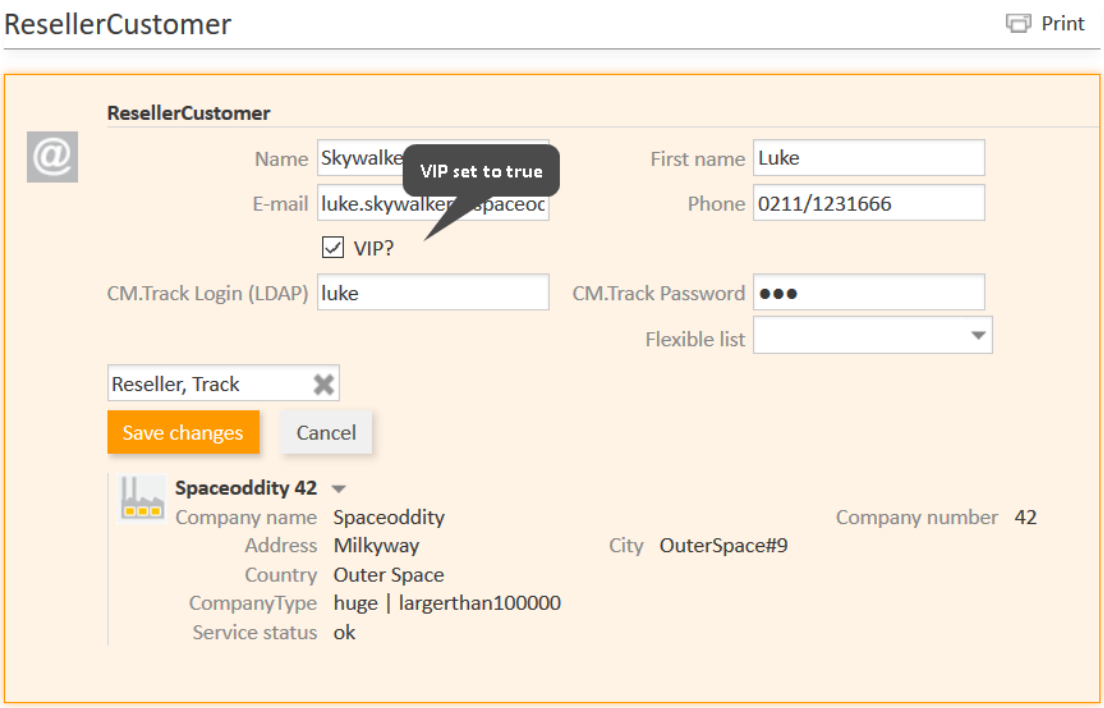


Figure 65: *ConSol CM Web Client - Customer field "VIP" for customer/contact data*

- In the script of the decision node, it has to be checked if the customer is a *VIP* (return value: “true”) or not (return value: “false”).

```
import com.consol.cmas.core.server.service.*
import com.consol.cmas.common.model.customfield.meta.UnitDefinitionType
def ticket = workflowApi.ticket
// fetch main contatc of the ticket
def maincontact = ticket.getMainContact()
def unit_type = maincontact.definition.type
log.info 'vipCheck: Unittype is ' + unit_type
log.info 'vipCheck: Unittype class is ' + unit_type.getClass()
if(unit_type == UnitDefinitionType.COMPANY) {
   log.info 'No vipCheck for comapnies possible! Returning false ... '
   return false
} else if (unit_type == UnitDefinitionType.CONTACT){

// fetch e-mail address of the man contact. The data object group field has
 to be addressed using data object group name:data object group field name
   def vip_field
   def custgroup = maincontact.customerGroup.name
   println 'vipCheck: Customergroup is now ' + custgroup
   switch(custgroup) {
      case "Reseller": vip_field = "vip_person";
      break;
      case "DirectCustomers": vip_field = "vip_dircust"
      break;
      case "MyCustomerGroup": vip_field = "vip"
      break;
      case "OurPartnerCompanies": vip_field = "vip_partners"
      break;
      case "RetailCustomers": vip_field = "retail_vip"
      break;
   }
   def vip_value = maincontact.get(vip_field)
   log.info 'VIP is now ' + vip_value

   if (vip_value){
      return true
   } else {
      return false
   }
}
```

Code example 7: *Precondition script: activity should only be displayed for VIP customers*

- When a ticket has passed automatically through the decision node and the following automatic activity where the *VIP* overlay is added, the ticket icon in the Web Client is marked with the overlay, see following figure.
  It is important that the overlay range is set to *process*. Otherwise the overlay will be removed as soon as the ticket leaves the activity or scope and will, in the example configuration, not be visible in the process.

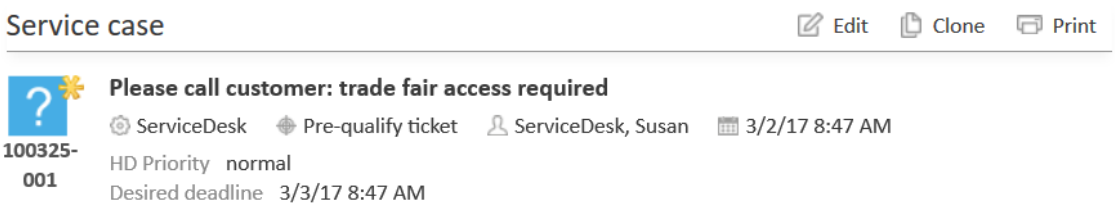


Figure 66: *ConSol CM Web Client - Ticket icon with VIP overlay*

C.7 Adornments (triggers and ACFs)

The ConSol CM workflow engine can react to several kinds of events. This is controlled by triggers. ACFs offer dynamic forms.

Adornment type	Explanation
Time Triggers	Control the time which has elapsed since the ticket has entered a scope or an activity, see section Time triggers.
Mail Triggers	Control if an email has been received by a ticket in the scope, see section Mail triggers.
Business Event Triggers	Control events like the change of the engineer or adding of a comment. See section Business event triggers.
ACF	Using Activity Control Forms (ACFs) you can control the data that have to be entered by the user in a certain step of the process, see section Activity Control Forms (ACFs).

C.7.1 Time triggers

This chapter discusses the following:

- Introduction to time triggers
- Adding a time trigger to a workflow
- Properties of a time trigger
- Business logic and initialization of a time trigger
- Examples for time triggers
- Scripting with time triggers

C.7.1.1 Introduction to time triggers

A workflow can contain several time triggers.

Figure 67: *ConSol CM Process Designer - Time trigger*

A time trigger is a mechanism which reacts when a certain period of time has elapsed. This can be required, for example, in the following situations:

- **Use case 1:**
  An engineer wants to put a ticket on hold for a defined time, because he knows that the customer will not be available until then.
- **Use case 2:**
  The system should automatically control the escalation time, i.e., when a ticket has come in and has not been taken care of, there should be an alert (this can be an overlay at the ticket icon, an email to the team lead, or other actions).
- **Use case 3:**
  A ticket has been solved and the engineer closes it. However, this should be a preliminary end and the ticket should be closed technically after a defined period of time.

Those use cases can be implemented using time triggers.

A time trigger can be configured to use a business calendar, i.e., to take only those times into consideration which are defined as working hours.

A time trigger can be attached to ...

- **a scope**
  Then it controls all tickets which are currently in the scope.
- **an activity**
  Then it controls only the tickets which have just run through this activity.

A time trigger has to be of one of two types:

- manual
- with a defined period of time

> You, as a workflow developer, have to implement everything that should happen as a consequence when a time trigger has fired! There are no automatic actions. All the time trigger does, is to give a signal *time elapsed* - just like an alarm clock.

C.7.1.2 Adding a time trigger to a workflow

Adding a time trigger to a scope

Grab the time trigger icon in the palette and drop it into the desired scope. It is automatically attached to the top of the scope. You can modify the position afterwards (move it to the left or right to change the order of triggers or just to improve the layout).

A time trigger, which has been attached to a scope, cannot be moved to another scope or activity. In case you would like to attach a time trigger to another scope/activity, remove the one you have defined and create a new one for the correct scope/activity.

To configure the properties of the trigger, select it in the editing panel and set the correct values in the Properties Editor. See section Properties of a time trigger.

You can draw connections from the trigger to put activities or decision nodes behind it. The first step which is executed after a time trigger always has to be an automatic activity!

Adding a time trigger to an activity

Grab the time trigger icon in the palette and drop it into the desired activity. It will be attached to the corner of the activity.

A time trigger which has been attached to an activity cannot be moved to another scope or activity. In case you would like to attach a time trigger to another scope/activity, remove the one you have defined and create a new one for the correct scope/activity.

To configure the properties of the trigger, select it in the editing panel and set the correct values in the Properties Editor. See section Properties of a time trigger.

You can draw connections from the trigger to put activities or decision nodes behind it. The first step which is executed after a time trigger always has to be an automatic activity!

C.7.1.3 Properties of a time trigger

A time trigger has the following properties:

- **name**

  Mandatory. The technical name of the trigger. It is set automatically but can be changed manually.

- **minutes/hours/days**

  Here you can enter the time interval after which the trigger should fire. The display mode always refers to a 24-hours-day, i.e., when you have entered 30 hours as reaction time and you re-open the workflow, there will be 1 day, 6 hours.

- **use calendar**

  Optional. Mark this check box when the business calendar should be taken into consideration when the time interval is calculated.

Please keep in mind that there are three steps which are necessary to make sure time intervals are calculated using a business calendar:

1. Define a business calendar (see *ConSol CM Administrator Manual*, section *Business Calendars*).
2. Assign the correct business calendar to a queue (see *ConSol CM Administrator Manual*, section *Queue Administration*).
3. Mark the check box *use calendar* for each trigger which should work with the calendar.

Principle of the use of a business calendar:

1 day means 24 hrs of absolute time, it has nothing to do with the use of a calendar. The calendar only plays a role when the time trigger is activated, then the 24 hrs, i.e., 86400000 milliseconds, will be taken as business calendar input (if the calendar is enabled).

Example:

When we have as trigger time 1 day = 24 hrs without calendar, the 24 hrs are calculated like regular time, so the escalation will fire one day later at the same time.

In contrast: When we use a calendar (with, for example, 7 work hrs per work day), the 24 hrs will be split-up according to the calendar, resulting in the firing event more than 3 days later (24 hrs = 3 x 7 hrs + 3 hrs).

See also section Working with calendars and times.

- **repeatable**

  Optional. Mark this check box to make sure the trigger can fire more than once for one ticket. If a trigger is *repeatable*, it is reset immediately after it has fired, i.e., the time count starts again.

  The script on timer start is executed again. The first firing event is initialized by the (technical) user *admin*, all following firing events are initiated by the Job Executor.

- **script after timer**

  Optional. A script can be defined which is executed when the time interval which is controlled by the trigger has elapsed, i.e., when the time trigger fires. The return value of this script determines if the following automatic activity will be executed or not.

- **script on timer start**

  Optional. A script can be defined when the time trigger starts to measure time, i.e., when the ticket has entered the scope/activity to which the trigger is attached. In this script you can either:

  - return “null ”or “false”, so that the timer won't be initialized and won't fire, or
  - calculate a `Long` value, standing for the milliseconds from now until the date, when the timer should fire. The calculated `Long` value then has to be handed over as parameter of the method `trigger.setDueTime()` to initialize the timer.

- **activate manually**

  Optional, only for time triggers at activities. Mark this check box when the user (the engineer) should select the time when the trigger should fire. For the user, a date-picker (web calendar) is displayed. The fields to set the date and time and the *script on timer start* field are not available for manual time triggers.

- **retry interval**

  The time in seconds after which the trigger execution should be executed again in case a script has run with an error. The time can be configured in the Admin Tool (property `jobExecutor.timerRetryInterval.seconds`).

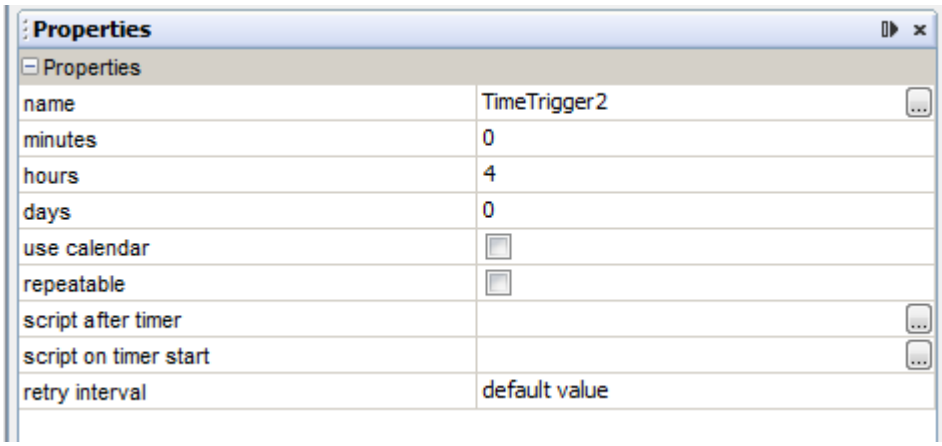


Figure 68: *ConSol CM Process Designer - Properties of a time trigger*

C.7.1.4 Business logic and initialization of a time trigger

The time measuring of a trigger is started (i.e., the trigger is initialized) when the ticket enters the scope/activity. It stops (i.e., the trigger fires) when the defined period of time which has been set as fixed value (minutes/hours/days) or the manually defined time has elapsed.

When you, as a workflow developer, would like to initialize a trigger using other values, this has to be done using scripts. Here, short examples will be provided, please see section Working with calendars and times for a detailed explanation of programming workflow trigger times. In those chapters, the code examples are provided, too.

- **Example 1:**
  The reaction time for a ticket should be calculated based on the priority. In the *script on timer start*, the different reaction times are used (a good way to implement this, would be to use customer-specific system properties) and the reaction time is calculated. Then the trigger is initialized, i.e., the time interval is set.

- **Example 2:**
  When an email to a ticket has come in and after three hours, no engineer has read the email and has taken care of the ticket, an alert should be triggered. To implement this, an incoming email (see section Mail triggers) has an adjacent automatic activity which re-initializes a time trigger with 3 hours.

A time trigger can also be deactivated. In example 2, this would be required to prevent the time trigger from firing initially, because it should not be initialized before any email comes in.

C.7.1.5 Examples for time triggers

The implementations for the use cases mentioned above (see Introduction to time triggers) would be:

- **Use case 1:**
  Put a manual time trigger to the activity *Put ticket on hold*. The engineer can select the desired end date by using the date picker in the Web Client. Usually then the ticket is led back to the active tickets.

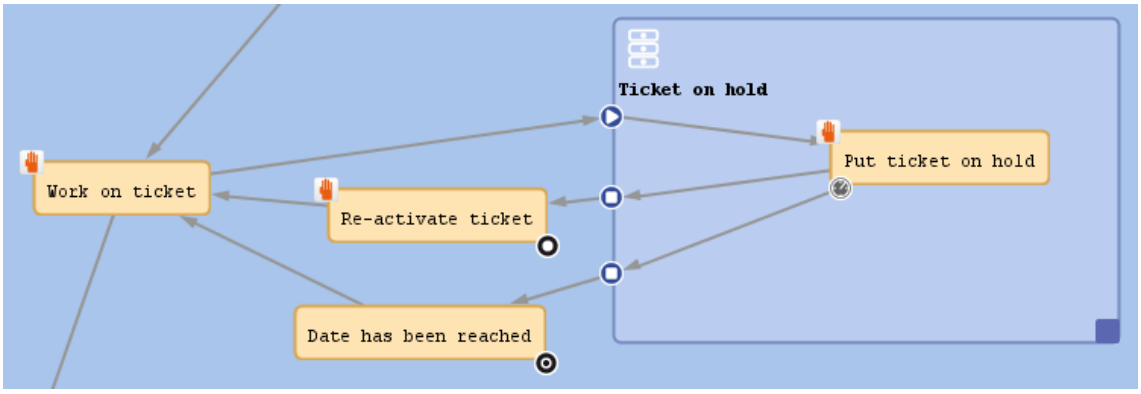


Figure 69: *ConSol CM Process Designer - Use case 1: Workflow*

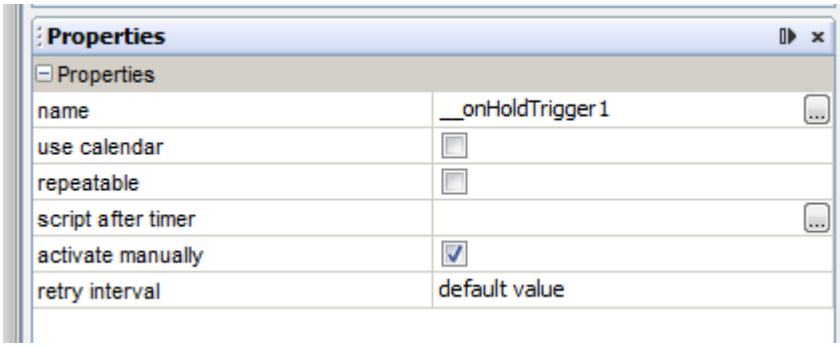


Figure 70: *ConSol CM Process Designer - Use case 1: Properties editor for time trigger*

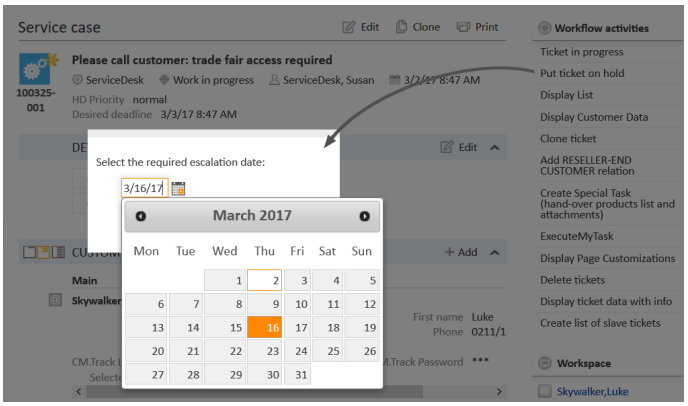


Figure 71: *ConSol CM Web Client - Use case 1: Date picker*

- **Use case 2:**
  Put a time trigger on the scope where the new tickets come in. Define the time for the trigger (this might depend on SLAs), e.g., four hours. Put a decision node behind the trigger if an

engineer has taken care of the ticket or not. If not, an email is sent to the team lead.

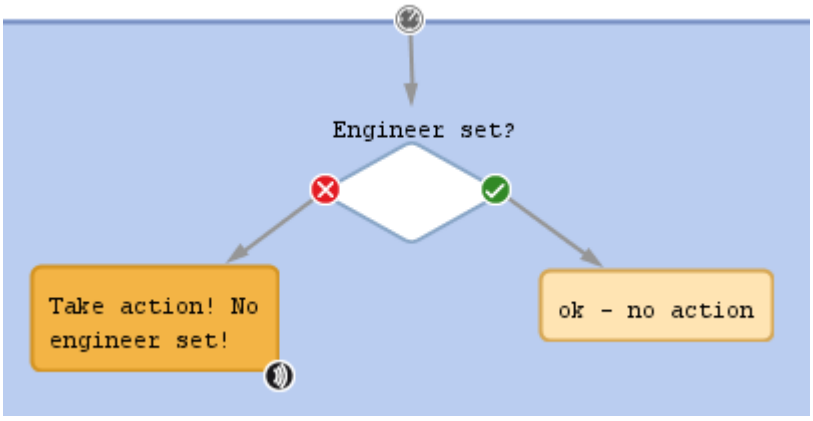
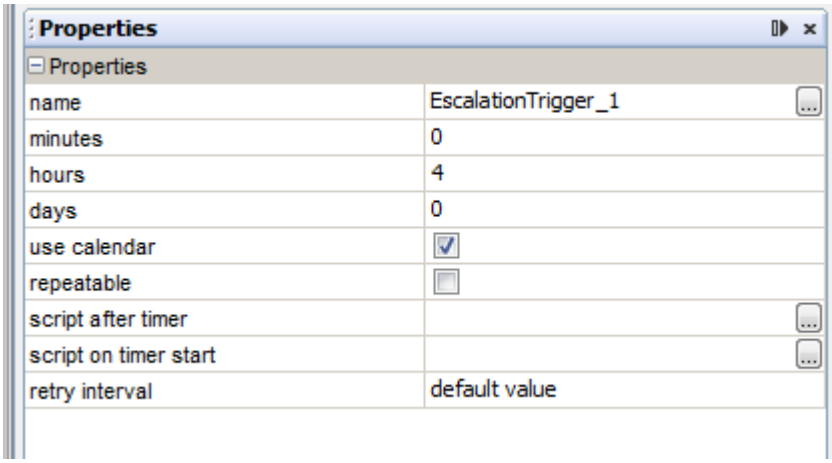


Figure 72: *ConSol CM Process Designer - Use case 2: Workflow*



Properties

Properties

name	EscalationTrigger_1
minutes	0
hours	4
days	0
use calendar	☑
repeatable	☐
script after timer	
script on timer start	
retry interval	default value

Figure 73: *ConSol CM Process Designer - Use case 2: Properties editor for time trigger*

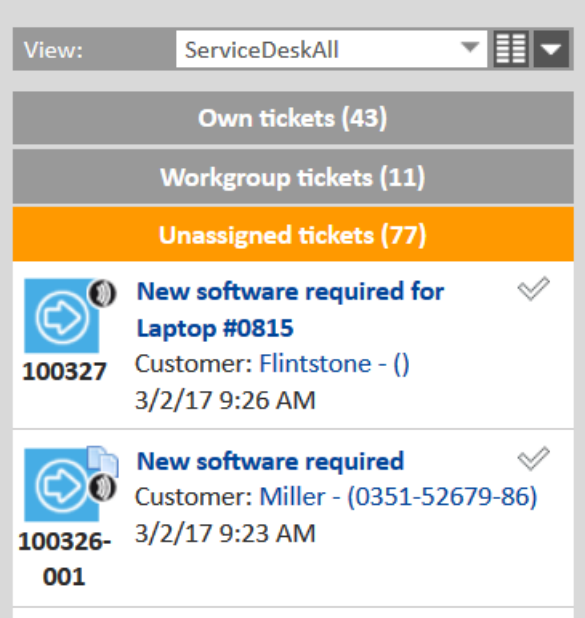


Figure 74: *ConSol CM Web Client - Use case 2: Ticket list*

- **Use case 3:**
  Put a time trigger to the activity *Close ticket with solution* and set a defined period of time for the trigger, e.g., five days. Behind the trigger there is the end node of the process. For five days, the ticket can still be edited, after this time, it is closed automatically.

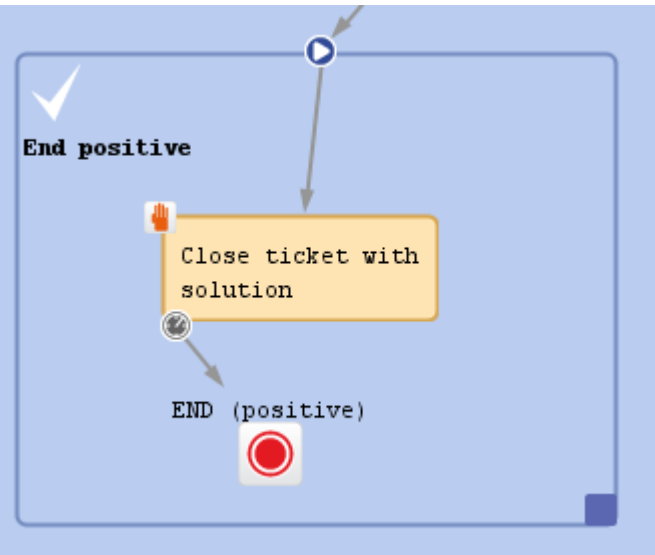


Figure 75: *CM Process Designer - Use case 3: Workflow*

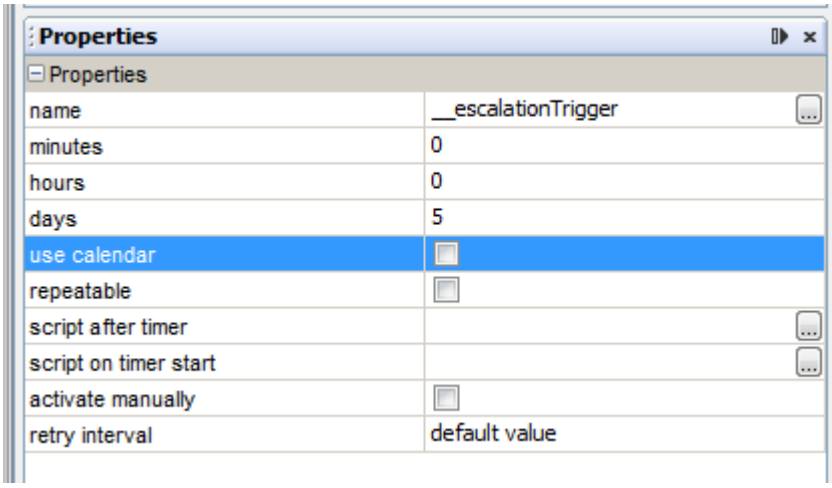


Figure 76: *ConSol CM Process Designer - Use case 3: Properties editor for time trigger*

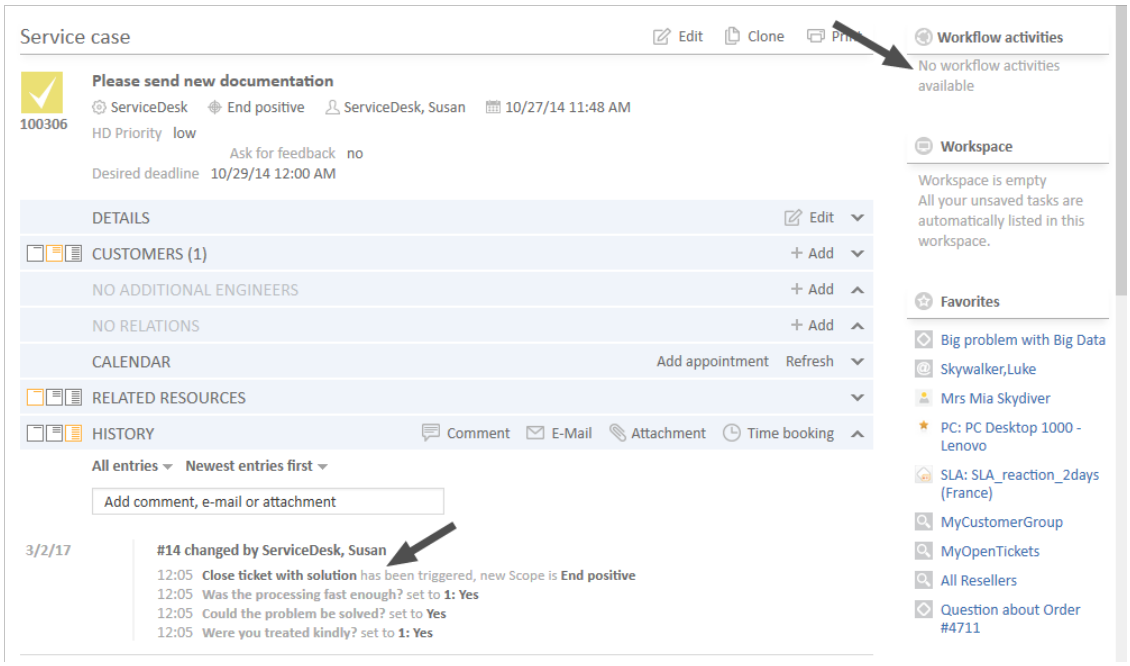


Figure 77: *ConSol CM Web Client - Closed ticket*

C.7.1.6 Scripting with time triggers

The following methods are of major importance when you work with time triggers:

TimerTrigger

The most important object in a script of a trigger is the trigger itself. It is an object of the Java class `TimerTrigger` and it is implicitly available as `trigger` in each trigger-script.

- **TimerTrigger.setDueTime(long pDueTime in millisecs)**
  Sets the time when the trigger should fire, indicated in milliseconds. The time provided as method parameter will be added to the original start time of the trigger, i.e., to the time when the ticket entered the scope or the activity where the trigger is located. So `setDueTime()` defines the time period in milliseconds from the entry time to the desired firing event.

workflowApi

workflowApi (the singleton instance of the `WorkflowContextService`) offers two methods to reinitialize the firing time of a trigger. Reinitialization means the trigger is reset to its original state with no time elapsed. In both methods, the trigger name (pTriggerName) has to be provided as path, explanation see section about working with path information.

- **reinitializeTrigger(String pTriggerName)**
  The trigger is reinitialized with the base date of the ticket entering the respective scope.
- **reinitializeTrigger(String pTriggerName, Date pBaseDate)**
  The trigger is reinitialized with the base date set explicitly. In this way, a trigger can be reinitialized with a data which is different from the date when the ticket has entered the respective scope. pBaseDate is an absolute date, provided as Java DATE object.
- **workflowApi.deactivateTimer()**
  (different method signatures)
  Deactivates the given time trigger, i.e., the trigger will never fire until re-initialized.
  (There is **no** method `activateTimer()`. Use `workflowApi.reinitializeTrigger()` to re-activate the trigger).

Please see also section Working with calendars and times.

Ticket

If you use a manual time trigger in an activity, you can retrieve the escalation/resubmission date set by the engineer using the `getStatusDate()` method of the `Ticket` object.

Overview of the TimerTrigger events

The following table provides an overview of possible events of time triggers. Please take the features of the objects and methods into consideration when working with time triggers!

Event in process	Timer trigger event	Result
Ticket enters scope where timer trigger is attached	Script on timer start is executed	Script on timer start returns NULL or FALSE: Timer trigger is not initialized, will never fire unless it is reinitialized at another point of time in the process Script on timer start uses `trigger.setDueTime(<Long>)` to initialize the trigger. The firing time of the trigger is calculated using the base date (i.e., the timestamp when the ticket enters the scope) and adding the <Long> value milliseconds, i.e., the method `setDueTime()` uses time relative to the time of trigger initialization. In order to set a certain time, you could use a DATE object which contains the target date/deadline and the following line of code: `trigger.setDueTime(target.time - (new Date()).time)` If the checkbox *use calendar* is checked, the calendar will be taken into consideration.

Event in process	Timer trigger event	Result
Ticket is still in the scope where the timer trigger is attached	Some script in the workflow calls the method workflowApi.reinitializeTrigger()	`reinitializeTrigger(<TriggerName>)` is called: The trigger is started again using the timestamp when the ticket entered the scope (where the trigger is attached) `reinitializeTrigger (<TriggerName>,<newBaseDate>)` is called: The trigger is started again, the time measuring starting with the newBaseDate. For example, if you would like to use "now" as newBaseDate, you have to call `workflowApi.reinitializeTrigger("TIMER_NAME", new Date())`. For a and b): If the checkbox *use calendar* is checked, the calendar will be taken into consideration.
Timer trigger has fired	Script after timer is executed	Everything within the script will be executed. Then: Script after timer returns FALSE: Nothing else happens. An automatic activity which is linked to the trigger will not be executed. Script after timer returns TRUE: An automatic activity which is linked to the timer trigger will be executed.

Please see also the section Which components to use with time triggers.

Examples for time trigger scripts

Example 1: Setting a queue-dependent due date

This script could be used as a script on timer start for a time trigger at a scope. It will initialize the trigger for an escalation depending on the queue, i.e., if the ticket is in the *HelpDesk_1st_Level* queue there is less time until the escalation than in the *HelpDesk_2nd_Level* queue.

Within the scripts *scripts on timer start* and *script after timer*, the object `trigger`exists as an implicit initialization of `TimerTrigger`. So you can work using triggers without any steps before. However, in an Admin Tool script you will have to import the `TimerTrigger` class or the respective Java package.

The following script could be used in a service desk and help desk environment and placed in the following `TimerTrigger`.

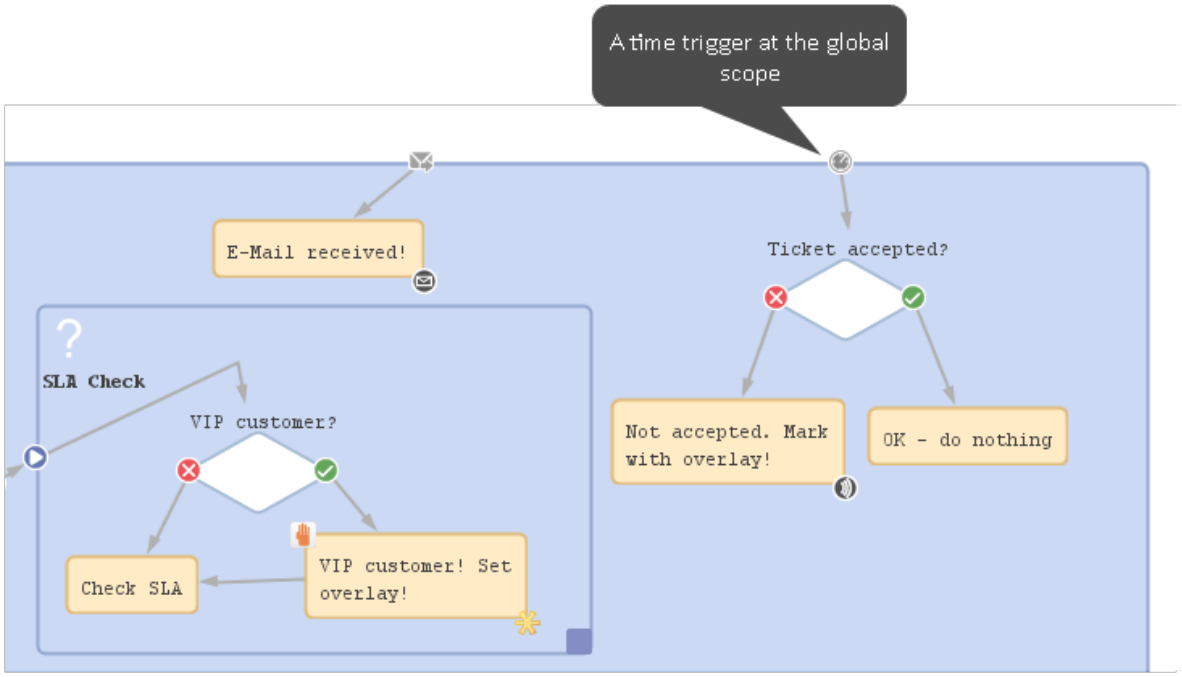


Figure 78: *ConSol CM Process Designer - TimerTrigger in ServiceDesk workflow*

```
def addedEscalMillis = 0

switch (ticket.queue.name) {
   case "HelpDesk_1st_Level":
      addedEscalMillis = 12*60*60*1000L;
      break;
   case "HelpDesk_2nd_Level":
      addedEscalMillis = 24*60*60*1000L;
      break;
   case "ServiceDesk":
      addedEscalMillis = 4*60*60*1000L;
   }

trigger.setDueTime(addedEscalMillis)
```

Code example 8: *Example for a script on timer start*

> ⚠ For this example, it makes sense to use fixed values for the times directly in the script code. In real life environments you might want to store escalation times and the like in system properties and retrieve them using the `configurationService`. in this way, an administrator can easily access and edit the escalation times without any manipulation of the workflow implementation.
>
> In real life, a business calendar might also be used - please see *Example 2*.

In the `server.log` file, you can see the time when the trigger is supposed to fire.

```
2017-03-02 13:22:25,979 INFO [w.DefaultWorkflowEventListener] [Susan-] Ticket's
 100332-001 timer defaultScope/Service_Desk/TimeTriggerDesiredDeadline was
 activated with escalation time Thu Mar 02 17:22:25 CET 2017
```

The ticket was created on March 2, 2017, 1.22 p.m., thus the escalation time is four hours later: 5.22 p.m.

The same principle could be applied to calculate the escalation time depending on the ticket priority, the *VIP* status of a customer, or any other parameter.

Example 2: Calculating an escalation 4 hours before the deadline

A deadline can be selected in the ticket (ticket field of type DATE). Four hours before this deadline will be reached, the ticket should be put into escalation state (marked by an overlay).

```
def deadl = ticket.get("serviceDesk_fields.desiredDeadline")
if (deadl == null){
   log.info("No deadline set!")
} else {
   log.info """DEADL: ${deadl}: ${deadl.time}"""

   // 4hrs before deadline the escalation should be set
   // business calendar should be used
   // ServiceDeskCalendar is assigned to queue ServiceDesk, this is transparent here

   def now = new Date()
   log.info """NOW: ${now}: ${now.time}"""

   // time required in millisecds
   def four_hours = -4*60*60*1000L

   // calculate escalation date
   def escalDate = BusinessCalendarUtil.getBusinessTime(deadl, four_hours,
    ticket.queue.calendar)
   // calculate and set due time
   log.info """escalDate: ${escalDate}: ${escalDate.time}"""
   def dueTime = escalDate.time - now.time

   trigger.setDueTime(dueTime)
   log.info 'DUE TIME is now ' + dueTime
}
```

Code example 9: *Calculate and set time for TimerTrigger using BusinessCalendar*

Example output in the `server.log` file:

The escalation date is four hours before the deadline. Because a business calendar is used, the escalation time is calculated with working hours only!

```
2017-03-02 12:33:22,083 INFO  [ine_25_216_initializing_script] [S        DEADL: Fri Mar 03 09:00:00 CET 2017: 1488528000000
2017-03-02 12:33:22,084 INFO  [ine_25_216_initializing_script] [Susa     NOW: Thu Mar 02 12:33:22 CET 2017: 1488454402084
2017-03-02 12:33:22,088 INFO  [ine_25_216_initializing_script] [Susan-] escalDate: Thu Mar 02 15:00:00 CET 2017: 1488463200000
2017-03-02 12:33:22,088 INFO  [ine_25_216_initializing_script] [Susan-] DUE TIME is now 8797916
2017-03-02 12:33:22,089 INFO  [w.DefaultWorkflowEventListener] [Susan-] Ticket's 100331-001 timer defaultScope/Service_Desk/TimeTriggerDesiredDe
adline was activated with escalation time Thu Mar 02 14:59:59 CET 2017
```

C.7.2 Mail triggers

This chapter discusses the following:

- Introduction to mail triggers
- Adding a mail trigger to a workflow
- Properties of a mail trigger
- Examples for mail triggers
- Process logic with mail triggers

C.7.2.1 Introduction to mail triggers

One of the core functionalities of ConSol CM is its interaction with an email infrastructure. This makes it possible for the engineer to send manual emails and for the system to send automatic emails to customers and to engineers, as required in the respective process step. Obviously, ConSol CM has also to receive emails. This is done by retrieving emails from one or more mailboxes with ConSol CM-owned addresses. For a detailed explanation of all interactions between the mail server and ConSol CM, please refer to the *ConSol CM Administrator Manual* and the *ConSol CM Operations Manual*. In the current manual, only the workflow interactions are explained.

Figure 79: *ConSol CM Process Designer - Mail trigger*

Mail trigger at a scope

When an email is received which belongs to an existing ticket, it might be required to register this action and to perform specific actions subsequently. This can be achieved using one or more mail triggers within a workflow.

Usually, there is a difference as how emails to open tickets are treated versus emails to closed tickets. In the default configuration, ConSol CM behaves as follows:

- If the ticket is open, the email will be integrated into the ticket and all attachments of the email will be attached to the ticket. The focus in the following section will be on this use case.
- If the ticket is closed, a new ticket will be opened, and a reference (ticket-ticket relation) of the new ticket to the old ticket will be set. Alternatively, you can use a reopen activity (see Reopen activities) in the end scope to reopen the ticket when an email is received. If a new ticket is created, mail triggers are not used. Therefore, this case will not be explained in the following section.

For a detailed explanation of the system configuration regarding email processing, please refer to the *ConSol CM Administrator Manual*, section *Scripts of type Email*.

> ⚠ Please keep in mind that (in the default configuration, i.e., without modification of the Admin Tool script `AppendToTicket.groovy`) the **only automatic action**, which is performed by ConSol CM after having received an email in a specific mailbox, is to attach this email to the ticket with the matching ticket tag in the subject, e.g., *Ticket (<TicketNumber>)*. See also *ConSol CM Administrator Manual* section *Scripts of Type Email*.
>
> **All other actions**, which should be executed when an email has been received, have to be programmed **manually** in the workflow (and/or in Admin Tool scripts)!

Examples for the use of mail triggers are:

When an email has been received ...

- the engineer of the ticket (the ticket owner) should also get an email as notification.
- the ticket icon (in the Web Client) should be marked by an overlay.
- the ticket should be transferred to an activity where the engineer has to confirm that he/she has read the email.
- the sender and the subject of the email are checked and parsed. If the email is a confirmation or a denial in an approval process, the ticket is managed according to the defined rules and activities in the workflow. That way, the approval can be performed using the email only, no login of the approver in the Web Client is required.

Mail trigger at an activity

When a mail trigger is attached to an activity, this activity is only executed when an email is received.

Figure 80: *ConSol CM Process Designer - Mail trigger at activity*

C.7.2.2 Adding a mail trigger to a workflow

Adding a mail trigger to a scope

Grab the mail trigger icon in the palette and drop it into the desired scope. It is automatically attached to the top of the scope. You can modify the position afterwards (move it to the left or right in order to improve the layout). Only one mail trigger can be used per scope.

A mail trigger which has been attached to a scope cannot be moved to another scope. In case you would like to attach a mail trigger to another scope, remove the one you have defined and create a new one for the correct scope.

You can draw connections from the trigger to put activities or decision nodes behind it. The first step which is executed after a mail trigger always has to be an automatic activity!

Adding a mail trigger to an activity

In the very rare case that you have to attach a mail trigger to an activity (we do not recommend this!), grab the mail trigger icon in the palette and drop it into the desired activity. It will be attached to the corner of the activity.

A mail trigger which has been attached to an activity cannot be moved to another scope or activity. In case you would like to attach a mail trigger to another scope/activity, remove the one you have defined and create a new one for the correct scope/activity.

C.7.2.3 Properties of a mail trigger

A mail trigger does not have any properties.

C.7.2.4 Examples for mail triggers

Example 1: Overlay for the ticket icon

When an email has been received for a ticket which is currently in the scope, the ticket icon in the Web Client GUI should be marked with the overlay *mail*.

The mail trigger is attached to the scope and the overlay is attached to the automatic activity which is connected to the trigger. The overlay range is *activity*.

In this way, the ticket is marked with the overlay when the email has come in. As soon as an engineer has moved the ticket to another activity, the overlay disappears.

Please note that the ticket does not leave its context. All that happens is the attachment of the overlay to the ticket icon. Then the ticket returns to its original position in the workflow. We call this an interrupt. Please read the section Process logic for a detailed explanation.

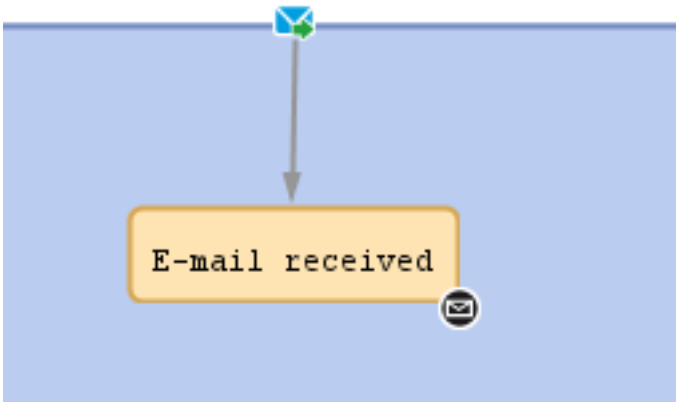


Figure 81: *ConSol CM Process Designer - Use case 1: Scope with mail trigger*

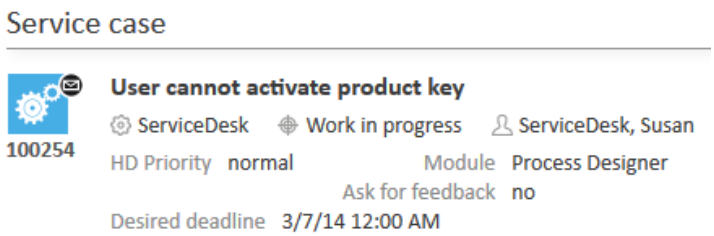


Figure 82: *ConSol CM Web Client - Use case 1: Ticket with overlay icon*

Example 2: Activity to acknowledge the email

When an email has been received for a ticket which is currently in the scope, the ticket icon in the Web Client GUI should be marked with the overlay *mail*. The engineer has to confirm that he/she has read the email. Since the email trigger is attached to the global scope and thus an email can be received during the entire process, the acknowledgment activity can be implemented very well using a scope activity. In this way, the acknowledgment activity (named *Set confirmation: email read* in our example) is also available during the entire process.

The mail trigger is attached to the scope and the overlay is attached to the adjacent automatic activity. The overlay range is *activity*. In this way, the ticket is marked with the overlay when the email has come in.

Within the script which follows the mail trigger, a boolean field *mail_to_read* is set to “true”. In the workflow, an activity *Set confirmation: email read* is offered when required. It is only displayed in case the value of the boolean field *mail_to_read* is “true”. This is a stronger mechanism to remind the engineer of an incoming email than to use only the overlay. The engineer has to confirm the email by executing the scope activity *Set confirmation: email read* explicitly. Within this workflow activity, the value of the boolean field *mail_to_read* is set back to “false” and the mail overlay is removed. Now the ticket is ready to receive another email and to notify the engineer.

Please note that also in this case the ticket does not leave its context as a consequence of the action which is executed after the email has come in. All that happens is the attachment of the overlay to the ticket icon and the modification of a BOOLEAN variable. The ticket returns to its original position in the workflow. So this is also an interrupt. Please read the section Process logic for a detailed explanation.

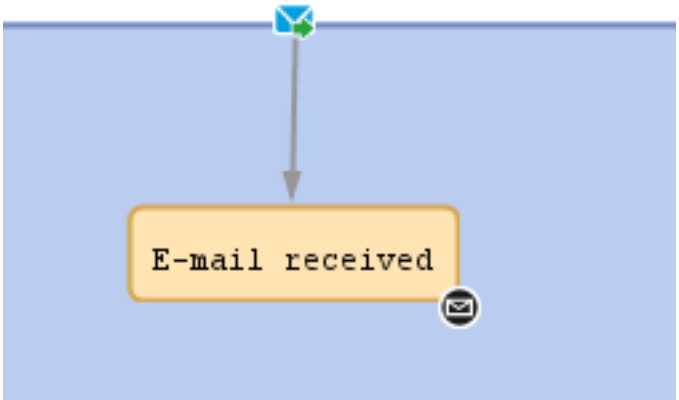


Figure 83: *ConSol CM Process Designer - Use case 2: Scope with mail trigger*

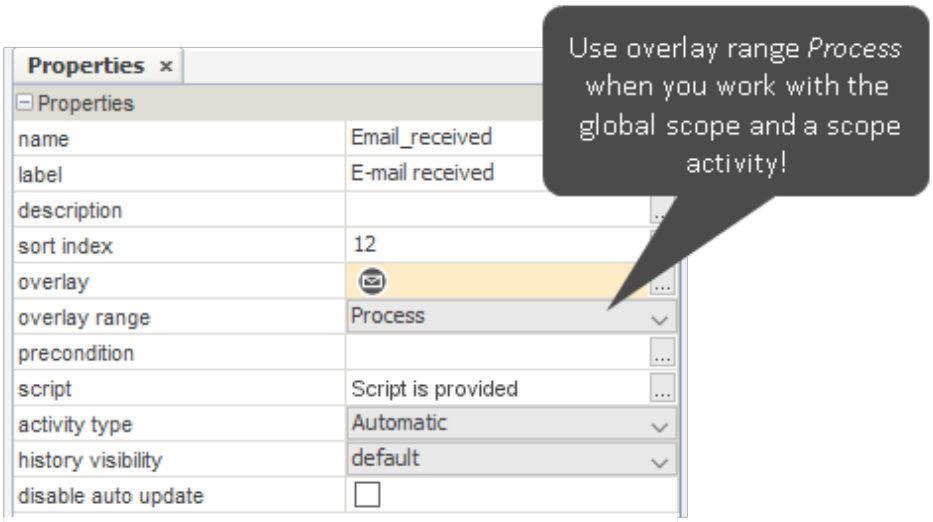


Figure 84: *ConSol CM Process Designer - Use case 2: Properties of activity "Email received"*

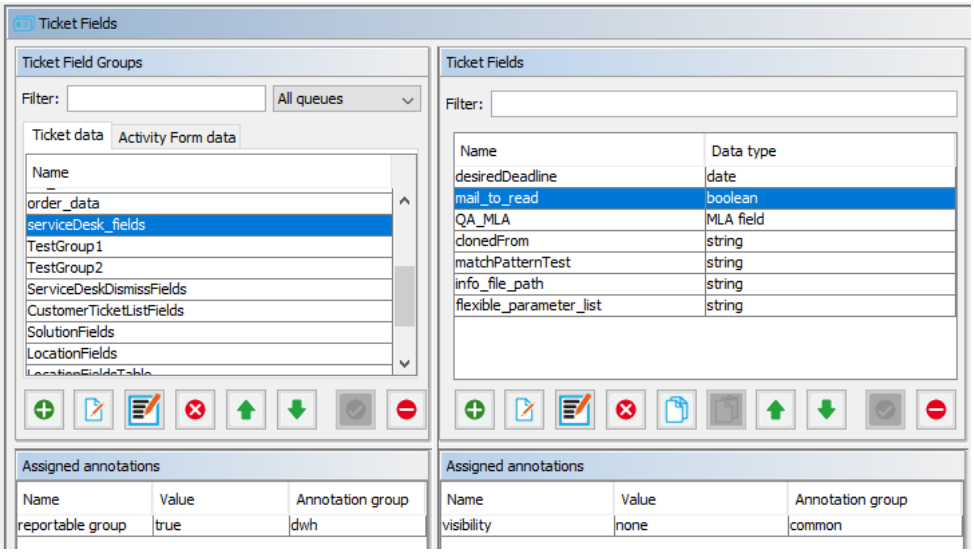


Figure 85: *ConSol CM Admin Tool - Use case 2: New boolean field to register email*

The script of the activity *Email received* contains the following script:

```
ticket.set("serviceDesk_fields.mail_to_read",true)
```

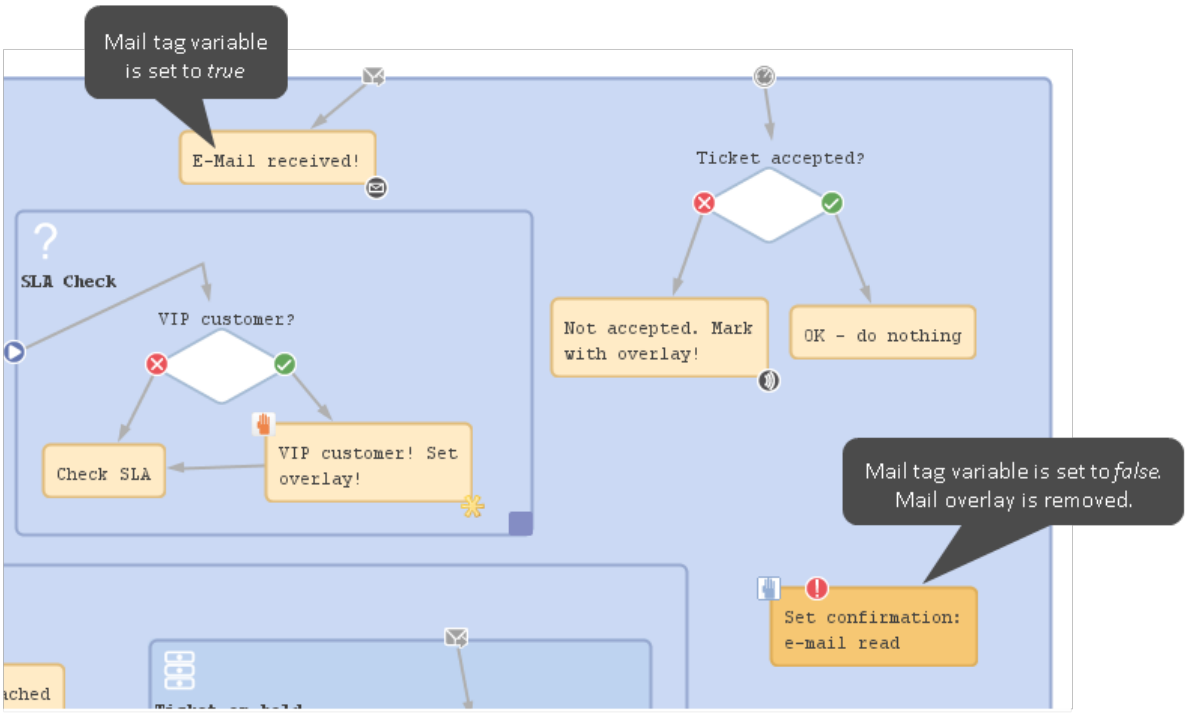
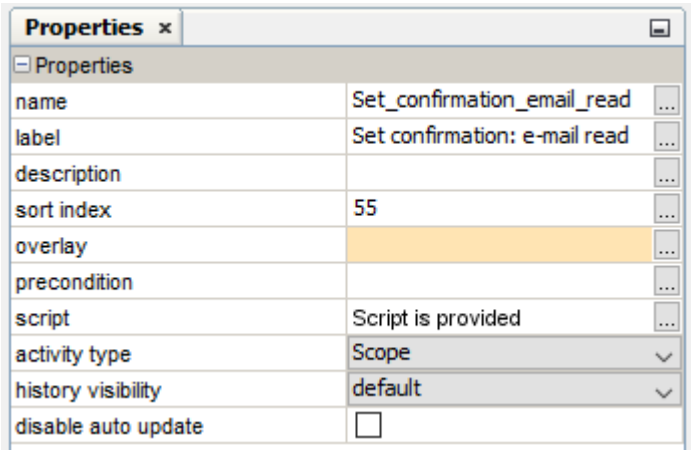


Figure 86: *ConSol CM Process Designer - Use case 2: Activity for email confirmation*



Properties	
name	Set_confirmation_email_read
label	Set confirmation: e-mail read
description	
sort index	55
overlay	
precondition	
script	Script is provided
activity type	Scope
history visibility	default
disable auto update	☐

Figure 87: *ConSol CM Process Designer - Use case 2: Properties of activity "Confirmed: email read!"*

The precondition script for the scope activity *Set confirmation: email read* contains the following script:

```
return ticket.get("serviceDesk_fields.mail_to_read")
```

The script for the scope activity *Set confirmation: email read* contains the following script:

```
ticket.set("serviceDesk_fields.mail_to_read",false)

def si = ticket.scopeInfo

for (ov in si.getActivatedOverlays().toArray() ) {
   if(ov.name.contains("Email_received_overlay")){
      si.removeOverlay(ov)
   }
}
```

Code example 10: *Removing an overlay, here: the email overlay, variant 1*

```
ticket.set("serviceDesk_fields.mail_to_read",false)

def si = ticket.scopeInfo
def ovs = si.activatedOverlays

ovs.each(){ ov ->
   // log.info 'OVERLAY NAME is now ' + ov.name
   if (ov.name == " defaultScope/ServiceDeskTicketInProgressScope/Email_received_
    overlay"){
      si.removeOverlay(ov)
   }
}
```

Code example 11: *Removing an overlay, here: the email overlay, variant 2*

```
ticket.set("serviceDesk_fields.mail_to_read",false)
def si = ticket.scopeInfo
for (ov in si.getActivatedOverlays().toArray() ) { // use the array to avoid
 concurrent modification
   if (ov.parent.name == "defaultScope/ServiceDeskTicketInProgressScope/Email_
    received") { // identifier is the name of the parent activity
      si.removeOverlay(ov)
   }
}
```

Code example 12: *Removing an overlay, here: the email overlay, variant 3*

> Using variant 1 or 2, you have to find out the name of the overlay first. This is usually done by having it printed out into the `server.log` file. Using variant 3, you only have to find out the name of the overlay's parent activity, i.e. the activity where the overlay is attached. This can be done rather comfortably with a right mouse click and by copying the name of the element to the clipboard.

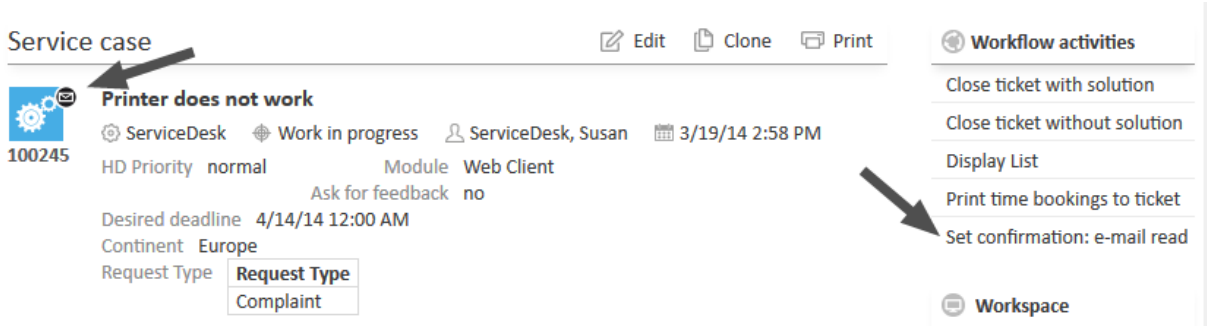


Figure 88: *ConSol CM Web Client - Use case 2: scope activity "Set confirmation: email read"*

C.7.2.5 Process logic with mail triggers

When an email is received, the mail trigger of the innermost possible scope fires.

Example 1:
The ticket is at position **(1)** in the *Ticket on hold* scope. When an email comes in, the mail trigger for this scope fires **(2)** and, as a consequence, the ticket is moved to another scope **(3)**.

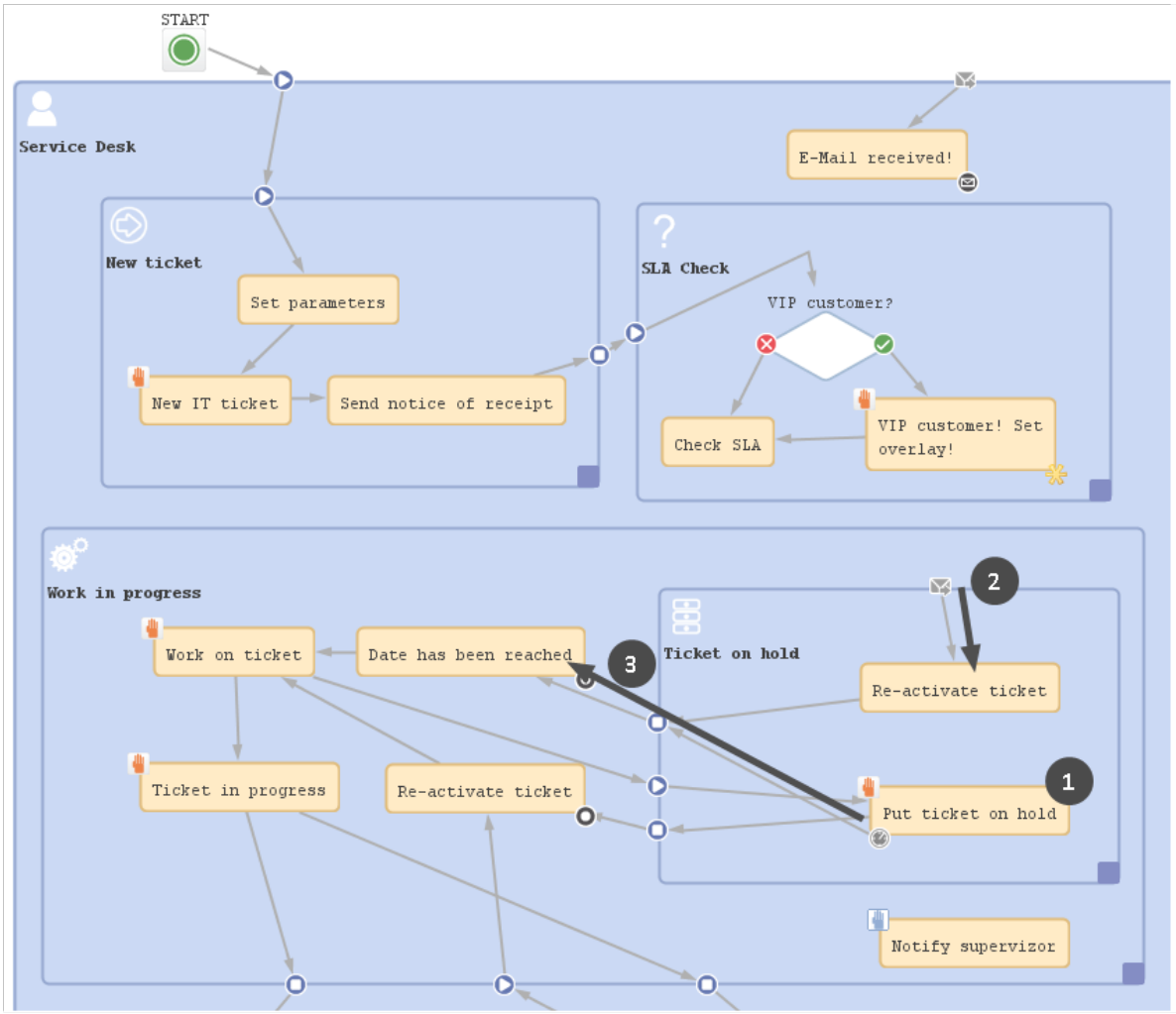


Figure 89: *ConSol CM Process Designer - Example 1: Mail trigger of sub-scope active*

Example 2:
The ticket is at position **(1)** in the *Work in progress* scope. When an email comes in, the mail trigger of the main scope **(2)** fires (because the *Work in progress* scope does not have a mail trigger). So the ticket position is not changed **(3)**.

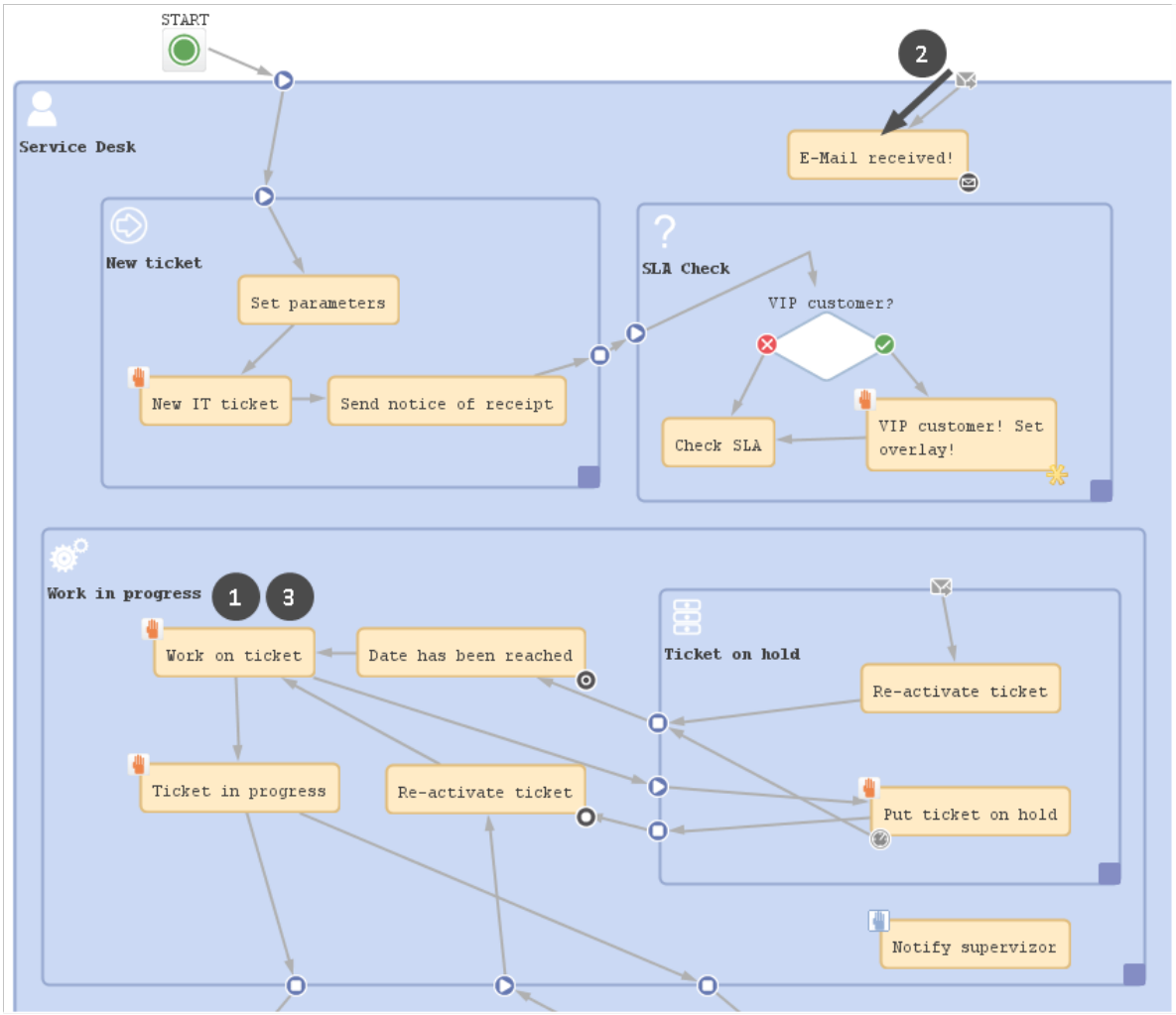


Figure 90: *ConSol CM Process Designer - Example 2: Mail trigger of main scope active*

C.7.3 Business event triggers

This chapter discusses the following:

- Introduction to business event triggers
- Adding a business event trigger to a workflow
- Properties of a business event trigger
- Business logic of business event triggers
- Examples for business event triggers
- Best practices for using business event triggers

C.7.3.1 Introduction to business event triggers

In business processes, there are often events during a regular process which have to be taken care of. For example, it might be required to inform the team lead if someone sets a ticket priority to *Extra High*. Or, after a change of the engineer of a ticket, it might be required to see if the engineer is logged in (if he/she is not in, the ticket has to be transferred to another engineer). There are numerous examples in business life for such events.

Figure 91: *ConSol CM Process Designer - Business event trigger*

ConSol CM can notice events using business event triggers and can react to the following types of events:

- change of engineer
- change of queue
- change of the subject
- change of the referenced engineer(s)
- related resource(s)
- time booking
- change of a comment
  (usually adding a new comment, i.e., a text comment or an email)
- change of any ticket field which has been defined by the system developer
  (this can be, e.g., the priority, a category, the content of a certain text box)

When the event occurs, the business event trigger fires.

> You, as a workflow developer, have to implement everything that should happen as a consequence when a business event trigger has fired! There are no automatic actions. All the business event trigger does, is to give a signal **event has occurred**.

A workflow can contain as many business event triggers as required. However, you have to make sure that in the process it is possible that all business event triggers can fire potentially (and that one does not depend on an action which cannot ever happen, because another business event (or time) trigger has fired before). Please see section Process logic for more information.

C.7.3.2 Adding a business event trigger to a workflow

Business event triggers can only be attached to a scope, never to activities.

Adding a business event trigger to a scope

Grab the business event trigger icon in the palette and drop it into the desired scope. It is automatically attached to the top of the scope. You can modify the position afterwards (move it to the left or right to change the order of triggers or just to improve the layout).

A business event trigger which has been attached to a scope cannot be moved to another scope. In case you would like to attach a business event trigger to another scope, remove the one you have defined and create a new one for the correct scope.

To configure the properties of the trigger, select it in the editing panel and set the correct values in the Properties Editor. See the following section Properties of a business event trigger.

You can draw connections from the trigger to put activities or decision nodes behind it. The first step which is executed after a business event trigger always has to be an automatic activity!

C.7.3.3 Properties of a business event trigger

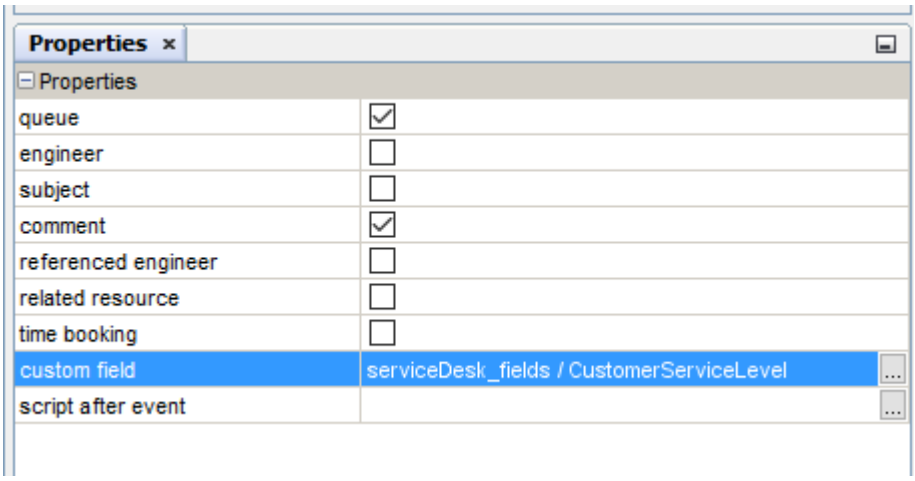


Figure 92: *ConSol CM Process Designer - Properties of a business event trigger*

A business event trigger has the following properties:

- **queue**

  Mark this check box if the business event trigger should react to a change of the queue, i.e., the trigger fires when the ticket is transferred to another queue. It is not relevant if this has been a manual action or has been performed automatically by the system.

- **engineer**

  Mark this check box when the trigger should react to a change of the engineer (owner) of the ticket. This can be a manual or an automatic action. There are three possible constellations:

  - The ticket did not have an engineer and an engineer is set.
  - The ticket has an engineer and the ticket is given to another engineer.
  - The ticket has an engineer and the engineer is set to “null” (no engineer).
- **subject**

  Mark this check box when the trigger should react to a change of the ticket subject.
- **comment**

  Mark this check box when the trigger should react to the change of a comment, i.e.:
  - An engineer has added a new (text) comment.
  - A customer has added a new (text) comment using ConSol CM/Track access.
  - An email has been received for the ticket.
  - An email has been sent out from the ticket.
  - One or more attachment(s) has/have been added to the ticket.
- **referenced engineer**

  Mark this check box when the trigger should react to a change of additional engineers in certain engineer roles of the ticket (ticket section *Additional engineers*). This can be one of the following situations (manually set or automatically by the system):
  - The ticket did not have any additional engineers and one or more additional engineer(s) is/are set.
  - The ticket has one or more additional engineer(s) and one or more of them is/are set to “null” or changed to another name.
  - The ticket has one or more additional engineer(s) and all those engineers are set to “null” (no engineer).
- **related resource**

  Mark this check box when the trigger should react to a change of related resources at the ticket, i.e.:
  - the relation to a resource is deleted
  - a new relation with a resource is established
  - a relation is modified (e.g., the description is changed)
- **time booking**

  Mark this check box when the trigger should react to a change of time bookings at the ticket, i.e.,
  - a new time booking entry is added
- **Custom Field**

  Use the (...) button to open the pop-up window *Event trigger* (see next figure) where you can select the ticket fields which should be monitored. Use the *plus* and *minus* buttons to add more fields or to reduce the number of monitored fields. As in the ticket field definition (see *ConSol CM Administrator Manual*, section *Ticket Field Administration*), you first have to select the ticket field group in the left pull-down menu and then you can choose one of the ticket fields of this group in the right pull-down menu. You can select as many ticket fields as you like.

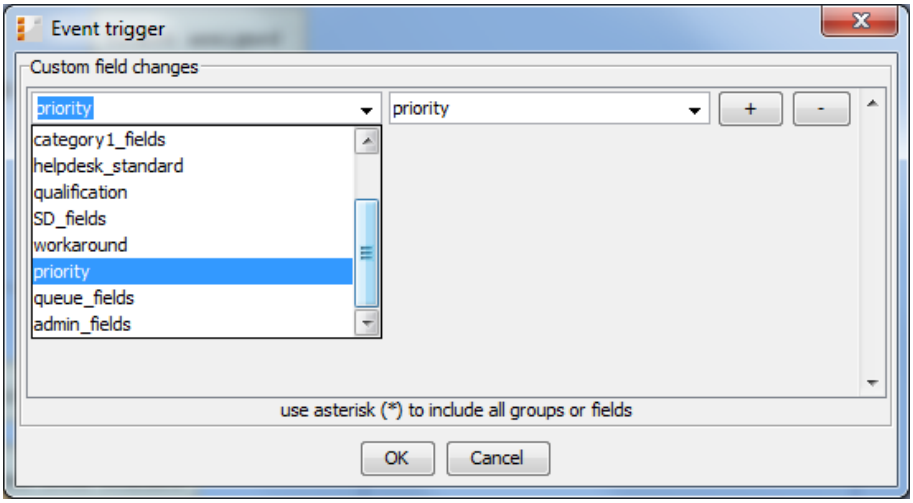


Figure 93: *ConSol CM Process Designer - Property "Custom Field" of a business event trigger*

- **script after event**

  Here you can define a script (using the ConSol CM script editor) which should be executed when the business event trigger has fired. It has to return “true” or “false”. When it returns “true”, the event is really fired, i.e., the automatic activity behind the business event trigger is executed. In case the script returns “false”, the event is blocked and the automatic activity is not executed. That way you can exactly control when the action (activity) should be performed, e.g., the trigger reacts to a change of the priority but should only really fire when the new priority is *Extra High*. Then the script checks the new priority and only when the new value is *Extra High* the script returns “true”, for all other values it returns “false”.

  > ⚠ The *script after event* is only used to control and fine-tune the firing of the business event trigger! Every action which should be performed when the trigger has fired has to be located in an automatic activity behind the trigger! This guarantees a good process logic and helps visualize the process in the Process Designer.

C.7.3.4 Business logic of business event triggers

Firing order of serialized business event triggers

When an event has occurred which is relevant for a business event trigger, this trigger fires. Then the *script after event* is executed. If it returns “true”, the following automatic activity or decision node with two following automatic activities is executed.

If the engineer changes more than one ticket parameter and different business event triggers have been defined for those parameters at the scope, the business event triggers fire according to their order at the scope.

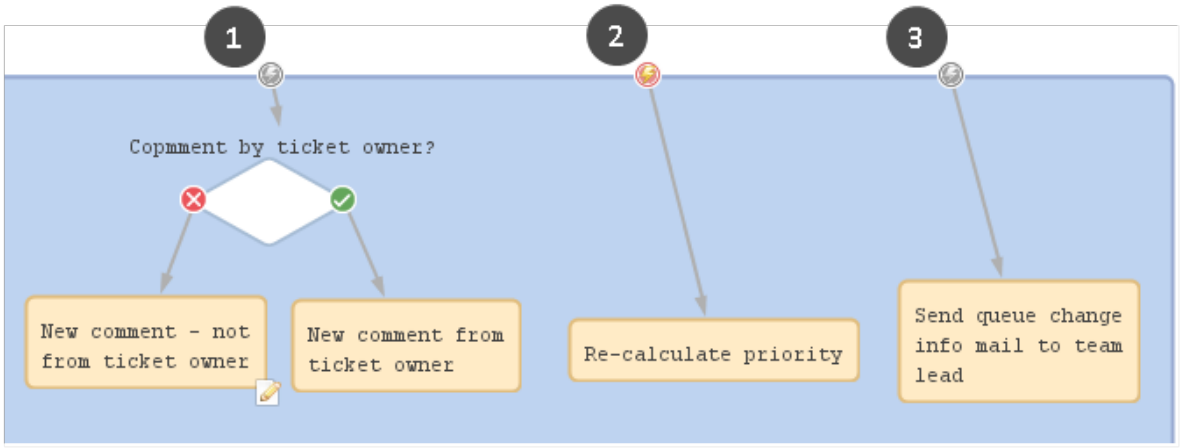


Figure 94: *ConSol CM Process Designer - Firing order of business event triggers (1)*

If one of the business event trigger actions leads the ticket to a new destination (i.e., it is no longer in the scope where the next business event trigger would be located), the following business event trigger is not fired. In the example in the following figure, the business event trigger **(3)** will not be fired, if the *Re-calculate priority* trigger **(2)** has been fired (see *Use Case 2* in section Examples for business event triggers), because the subsequent actions lead the ticket to another queue.

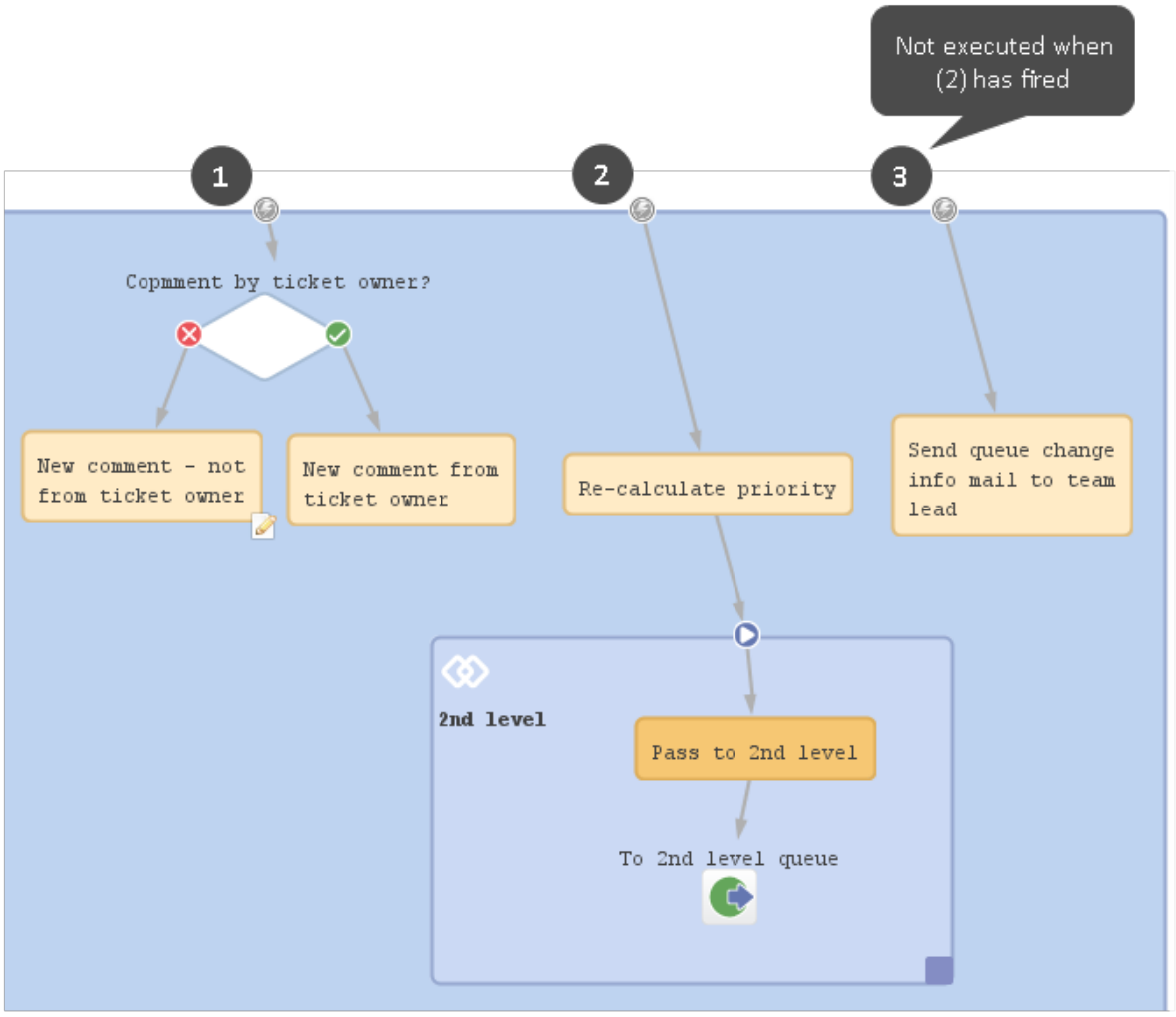


Figure 95: *ConSol CM Process Designer - Firing order of business event triggers (2)*

Firing order of business event triggers in hierarchical scopes

In case there are business event triggers in hierarchical scopes, the event is consumed by the innermost business event trigger, i.e., by the business event trigger of the innermost scope. All events which have not been consumed there, are further processed by the next outer scope, then the next and so on.

Case 1

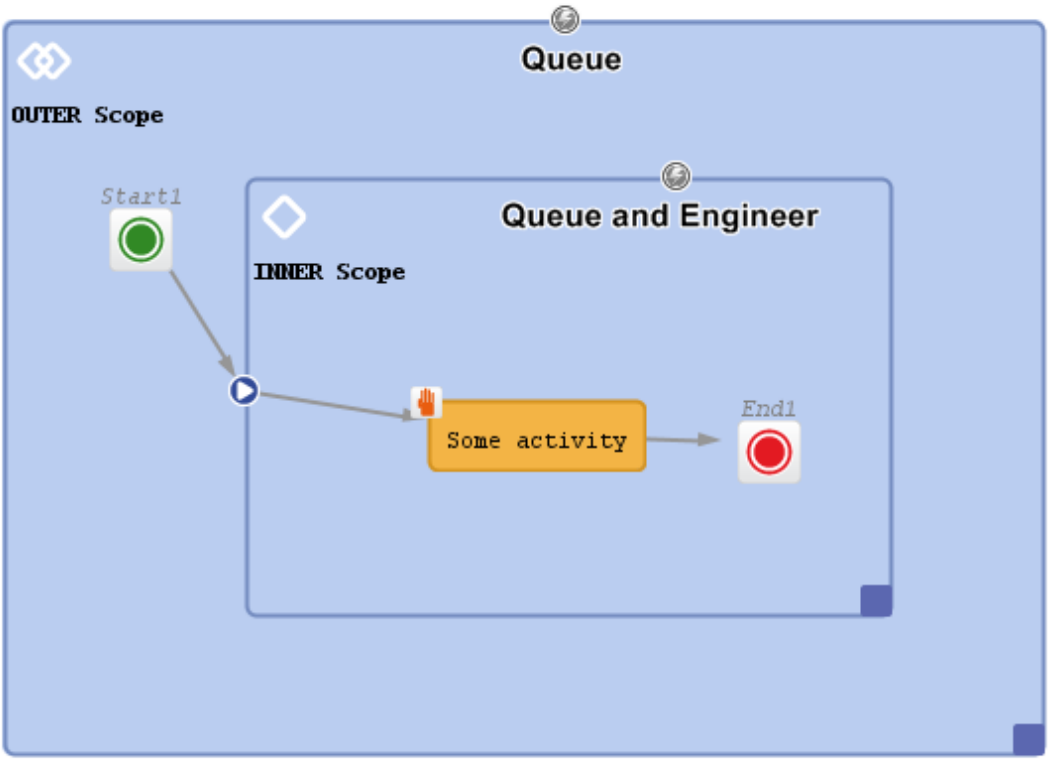


Figure 96: *ConSol CM Process Designer - Hierarchical business event triggers (1)*

Fired events:

Events	Triggers fired
Queue	Inner
Queue and Engineer	Inner for both
Engineer	Inner

Case 2

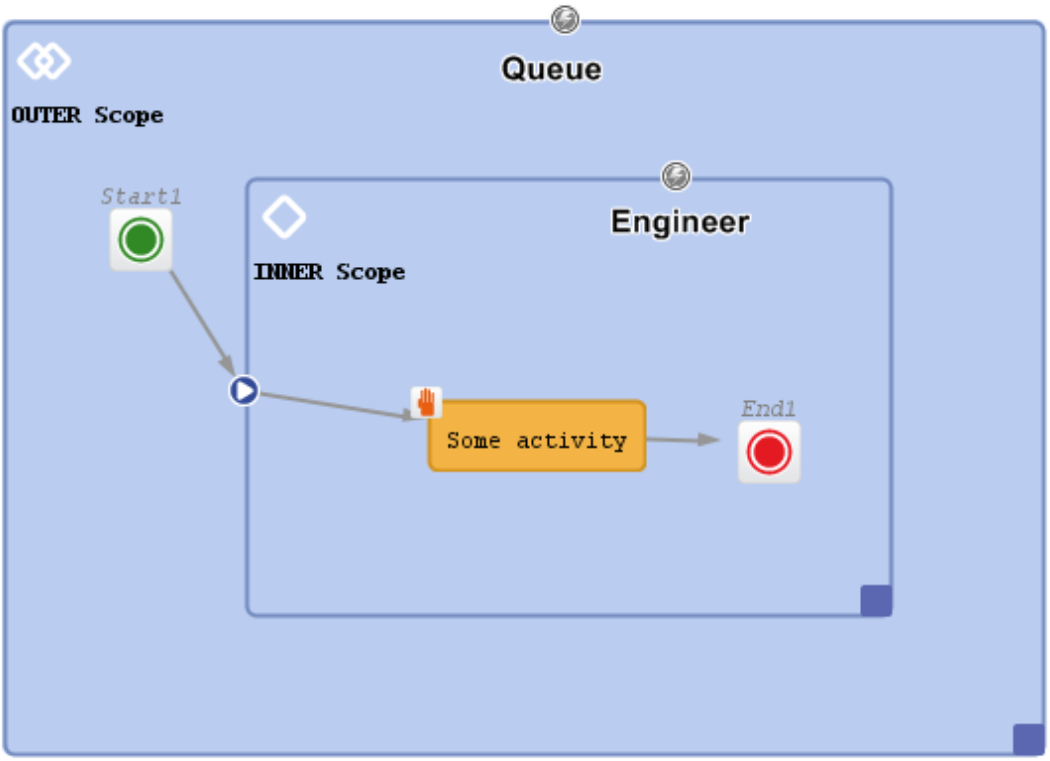


Figure 97: *ConSol CM Process Designer - Hierarchical business event triggers (2)*

Fired events:

Events	Triggers fired
Queue	Outer
Engineer	Inner
Queue and Engineer	Inner and Outer

Case 3

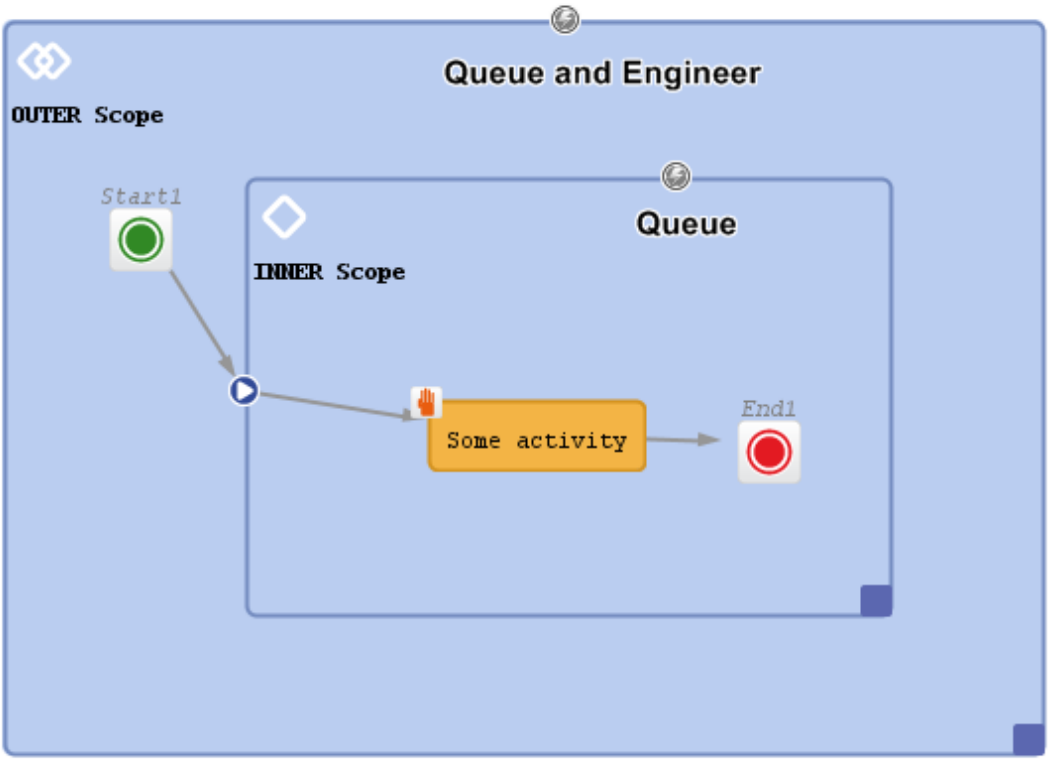


Figure 98: *ConSol CM Process Designer - Hierarchical business event triggers (3)*

Fired events:

Events	Triggers fired
Queue	Inner
Engineer	Outer
Queue and Engineer	Inner (queue) and Outer (engineer only)

C.7.3.5 Examples for business event triggers

Example 1: Reacting on comments

If a new comment has been added to the ticket by someone else, not by the current engineer (the ticket owner), then an overlay should be attached to the ticket icon. In this way, the ticket is marked and the engineer can see in the ticket list that there is a new comment in one of his/her tickets. The comment can be made by another engineer who has writing access to the queue or by a customer who can add comments using ConSol CM/Track access. Or an email might have been received.

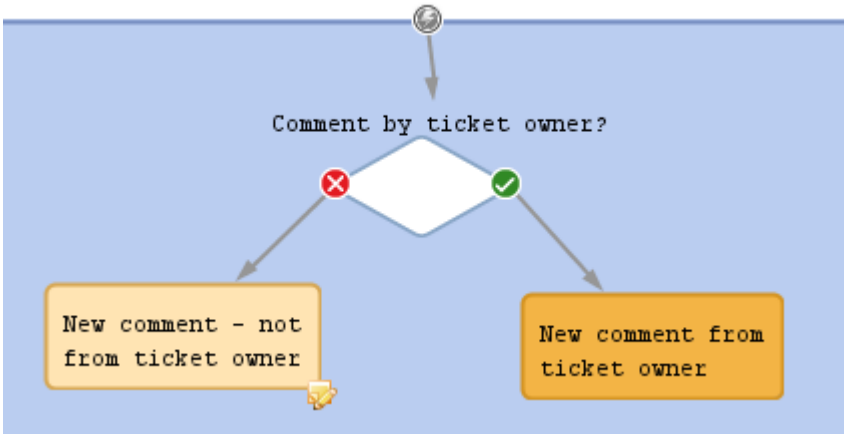


Figure 99: *ConSol CM Process Designer - Business event trigger with following activities*

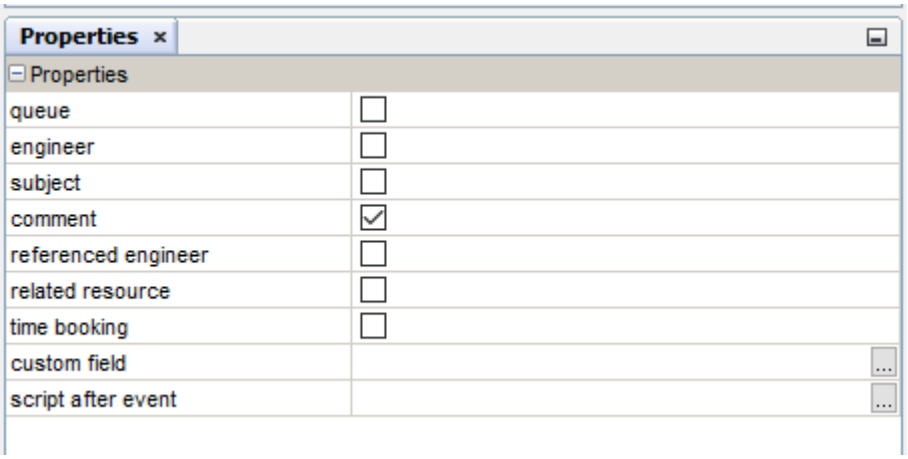


Figure 100: *ConSol CM Process Designer - Properties of a business event trigger (1)*

```
return (engineerService.current == ticket.engineer)
```

Code example 13: *Code of decision node script*

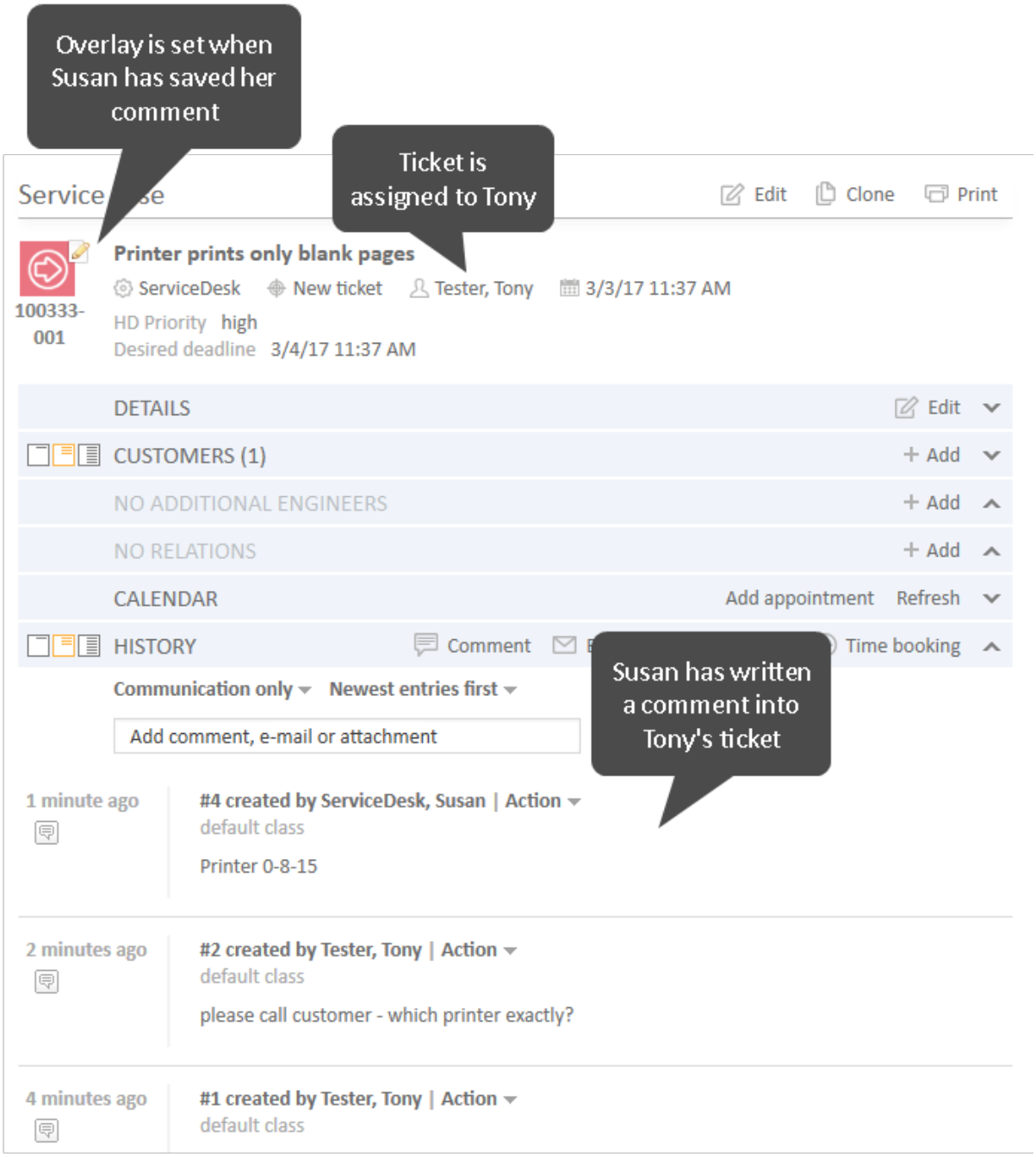


Figure 101: *ConSol CM Web Client- Ticket marked with new overlay*

Example 2: Reacting on changes to ticket fields

This is an example from an ITIL Service Desk environment. According to the ITIL standards, the ticket priority is calculated from two values: *impact* and *urgency*. That means, in the ticket there are two fields which can be modified by the engineer and the priority is calculated automatically from the two values. The priority might then be displayed as ticket color or as selection in a read-only list (or both).

This principle requires a re-calculation of the priority in case at least one of the two fields (*impact*/*urgency*) has been changed. This is achieved using a business event trigger with an adjacent activity where the re-calculation is performed.

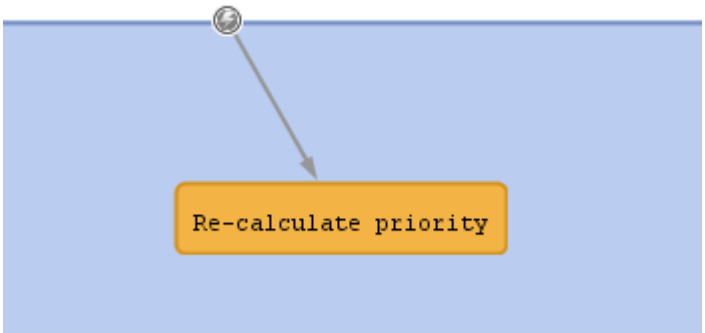


Figure 102: *ConSol CM Process Designer - Business event trigger with following automatic activity*

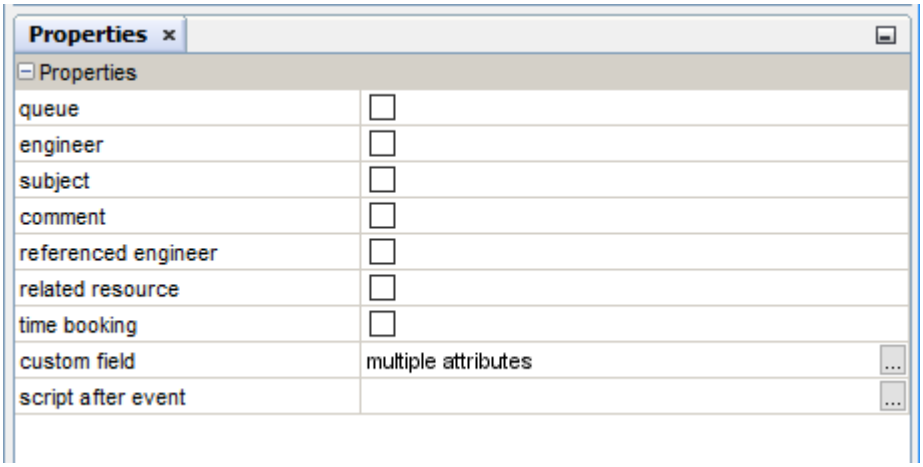


Figure 103: *ConSol CM Process Designer - Properties of a business event trigger (2)*

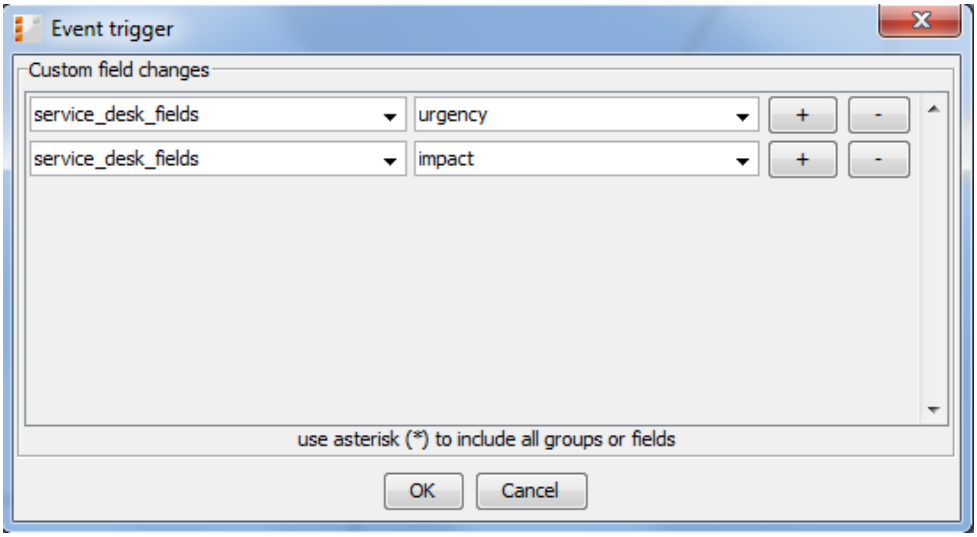


Figure 104: *ConSol CM Process Designer - Property "Custom Field" of a business event trigger (2)*

```
// Re-calculate priority:
String imp_value = ticket.get("service_desk_fields.impact").getName()
String urg_value = ticket.get("service_desk_fields.urgency").getName();

ScriptProvider scriptProvider = scriptProviderService.createDatabaseProvider
 ("calculatePriority.groovy")
//content of calculatePriority.groovy is omitted here, because it is not relevant
 for the current context
```

Code example 14: *Code of automatic activity script Re-calculate priority*

Example 3: Continuing the business process when a ticket field changes

This is an example taken from a shipment and delivery process: new components (e.g., hardware) are ordered. The ticket waits in the scope *Order: Waiting for shipment*. When the shipment has arrived, an engineer of another team registers this shipment and sets the *Shipment received* tag. This change of ticket data (*Shipment received* from “false” to “true”) is registered by the business event trigger which listens to the respective BOOLEAN value (the check box). After the business event trigger has fired, the check box is checked (in the decision node), and when the value is set to “true”, the ticket is forwarded to the next scope *Deliver components*. The engineers who are responsible for the delivery now see the ticket in their view *Components ready for delivery* and can acknowledge the delivery when they are done with *All components delivered*.

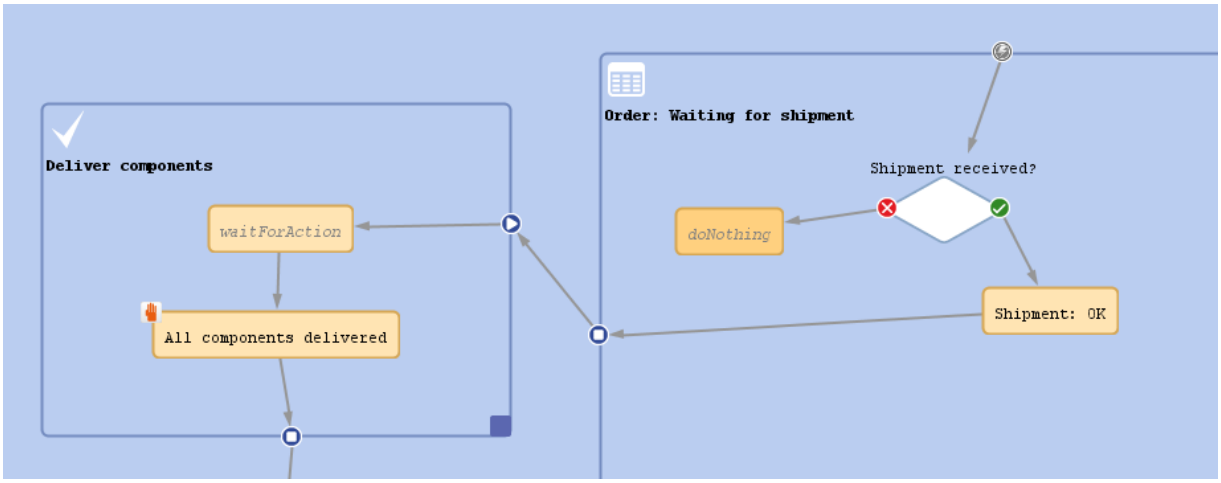


Figure 105: *ConSol CM Process Designer - Workflow for use case 3*

Example 4: Sending emails when a ticket field has changed

A Business Event Trigger being fired means something has been changed within the ticket. So maybe you want to report what has been changed. The object which provides the required information is an object of the class `TicketChanges`.

The following example shows how to report if the deadline in a task ticket has been modified. If yes, and if the ticket is currently owned by an engineer, this engineer should be informed via email about the date being changed.

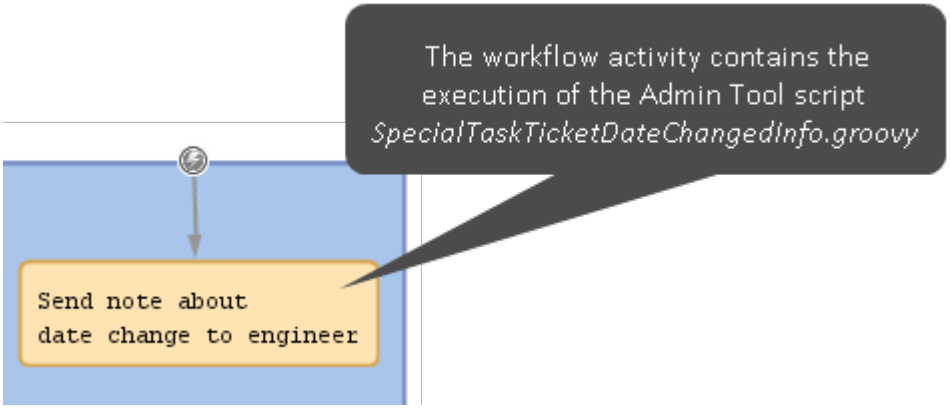


Figure 106: *Business Event Trigger with linked automatic activity. In the example the trigger listens to the change of the ticket field 'Deadline' in the ticket (not to be seen)*

```
import com.consol.cmas.common.model.customfield.meta.FieldKey
import com.consol.cmas.common.model.event.ticket.support.TicketChanges
import com.consol.cmas.common.model.mail.Mail
import com.consol.cmas.core.server.service.*
import static com.consol.cmas.common.util.TemplateUtil.TICKET_SUBJECT_TEMPLATE_NAME
log.info 'SpecialTaskTicketChangedInfo started ...'
// grep ticket changes
TicketChanges changes = workflowApi.getTicketUpdateEvent().getModifications()
def ticket = workflowApi.getTicket()
def eng = ticket.engineer
def new_date
def old_date
if (changes) {
   log.info 'Special task ticket ' + ticket.id + ' has been changed.'
   // if deadline field has been modified, send mail to current engineer if present
   if (eng) {
      // field key for requested field: deadline
      fk_deadl = new FieldKey("SpecialTasks_Fields","Deadline")
      mod_deadl = changes.getCustomFieldChangeInfo(fk_deadl)
      if (mod_deadl){
            // send email to engineer
            def mail = new Mail()
            mail.setTargetEngineer(eng)
            def replyaddress = configurationService.getValue("cmweb-server-
             adapter","mail.reply.to")
            mail.setReplyTo(replyaddress)
            def text = workflowApi.renderTemplate
             ("SpecialTasksTicketDeadlineChangedInfo")
            def text2 = 'The old date was ' + mod_deadl.previousValue.value + ' -- '
            def text3 = ' The new date is ' + mod_deadl.value.value
            mail.setText(text + text2 + text3)
            // def ticketName = ticket.getName()
            def subject = templateService.merge(TICKET_SUBJECT_TEMPLATE_NAME,
             [ticketName:ticket.name])
            def subject2 = " Deadline changed in Special Tasks ticket!"
            mail.setSubject(subject + subject2)

            try {
               mail.send();
            } catch (Exception e){
               mailStatus = false;
            }

      } // end if (mod_deadl)
   } // end if (eng)
// end if (changes)
```

Code example 15: *Sending an email to the current engineer of the ticket if the 'Deadline' field in the ticket has been changed (Dates not formatted)*

C.7.3.6 Best practices for using business event triggers

See section How to avoid self-triggering business event triggers.

C.7.4 Activity Control Forms (ACFs)

This chapter discusses the following:

- Introduction to ACFs
- Adding an ACF to a workflow
- Properties of an ACF
- Business logic of ACFs
- Examples for the use of ACFs
- ACFs in CM/Track

C.7.4.1 Introduction to ACFs

An **Activity Control Form (ACF)** is a web form which is offered to the engineer at one or more process steps. In this way, the data input can be controlled in a very strict way.

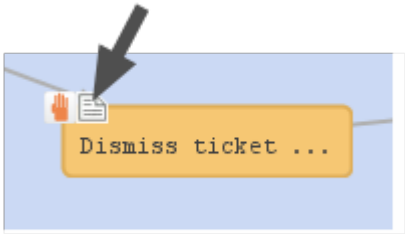


Figure 107: *ConSol CM Process Designer - Activity Control Form (ACF)*

For example, when a help desk agent wants to dismiss a complaint, this cannot be performed without giving a reason. In the process this is implemented using an ACF which is displayed when the engineer has clicked on the workflow activity *Dismiss ticket*. A form is opened where the engineer has to select a category for the dismissal and a text box where he/she can enter a note. Or, using the example of a sales process, when an engineer (a sales agent in this case) clicks on *Make appointment with potential customer*, a form is displayed, where the budget, the size of the customer's company, and the products of interest have to be entered.

An ACF can offer optional and mandatory fields.

> We recommend to set a "..." behind the name of every activity which will automatically open an ACF. This helps the user to distinguish between ACF-loaded activities and simple activities.

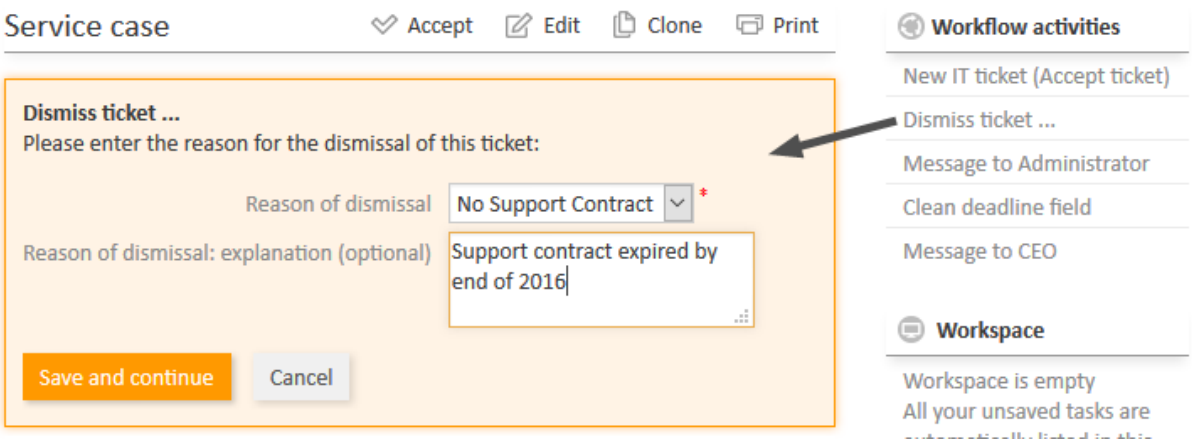


Figure 108: *ConSol CM Web Client - Opened ACF*

C.7.4.2 Adding an ACF to a workflow

Variant A: Starting the ACF definition in the Admin Tool

Before you can add an ACF to the workflow, it has to be defined using the Admin Tool. Please refer to the *ConSol CM Administrator Manual*, chapter *Ticket Field Administration* for a detailed explanation. In the current manual, we assume you have already defined an ACF and want to add it to the workflow.

An ACF is always added to a manual or scope activity. To add an ACF to the target activity, grab the ACF icon in the palette and attach it to the activity using drag-and-drop. Then you can configure the ACF properties. In case you add an ACF to an automatic activity, this activity is changed to type *Manual*.

In the Web Client, the ACF will be opened when the user clicks on the workflow activity to which the ACF is attached in the workflow. See figure above.

Variant B: Starting the ACF definition in the Process Designer

You can also add an empty ACF to a workflow activity and define the name during this operation. Then an empty ACF will be created in the Admin Tool and you have to assign the ticket fields to this ACF in a later step.

> ⚠ Do not forget to reload the Admin Tool data! When you have defined the ACF in the Process Designer, there is no automatic data transfer to the Admin Tool.

C.7.4.3 Properties of an ACF

These are the properties of an ACF:

- **name**

  String. The name of the ACF. Select the name from the drop-down menu. All ACFs which have been defined in the Admin Tool are available.

- **required fields**

This opens a pop-up window (see figure below) where you can define mandatory fields. As a default, all ACF fields are optional, i.e. when the form is opened in the Web Client, the engineer can enter data but can also continue the process without doing so. For mandatory fields, the process can only be continued when the field has been filled in.

- **Initializing script**

  Here, you can define a script which will be executed before the ACF is loaded. Usually, this kind of script is used to set default values in ACF ticket fields.

- **Precondition script**

  Here, you can define a script which is executed to determine if the ACF will be displayed (return value “true”) or not (return value “false”).

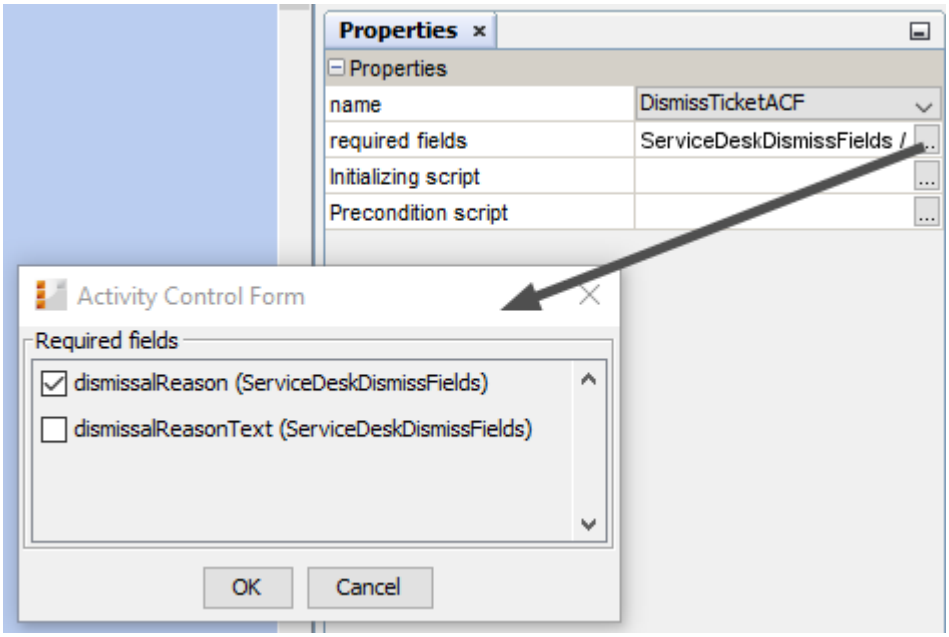


Figure 109: *ConSol CM Process Designer - Properties of an ACF*

All ticket fields which are part of an ACF have to be available in the target queue, i.e. the respective ticket field group has to be assigned to the queue where the workflow is used! There are two possibilities to achieve that:

- You assign the ticket field group to a queue manually.
- You just create the ACF and use it in a workflow. When you deploy the workflow, ConSol CM will automatically assign the required ticket field groups to the queues where the workflow is used.

For a detailed explanation of queue management, please see the *ConSol CM Administrator Manual*.

C.7.4.4 Business logic of ACFs

ACF at a manual, scope or reopen activity

ACFs are only possible for manually executed activities, i.e., for manual, for scope and for reopen activities. When a user selects a workflow activity which has an ACF in the Web Client, the following steps are performed:

1. If an **ACF precondition script** is present, this precondition script is executed.

   If the ACF precondition script returns “true”:

   a. If an **ACF Init Script** is present: the ACF Init Script is executed.

   b. The **ACF** is displayed and the engineer fills in the form, with optional and mandatory fields. If fields, which are part of the ACF, are also available in the regular ticket data fields, those fields might have been edited/filled-in by an engineer before the ACF is used. Thus those fields might be already filled-in in the ACF. The engineer can leave them as-is (and use the ACF as control only) or can modify the content of the fields.

   c. The **workflow activity** is executed as soon as the engineer has clicked on *OK* in the ACF.

   If the ACF precondition script returns “false”:

   a. The **ACF** is not displayed.

   b. The **workflow activity** is executed as soon as the engineer has clicked on the workflow activity (name) in the Web Client.

When an ACF is **canceled**, it returns to the scope of the last activity, because the ticket always waits **behind** the last activity (and **not** before the next).

If the data of the ACF should not be shown before a certain step in the process has been reached, the data can be put into one (or more) separate ticket field group(s) which are **invisible** at the start of the process (ticket field group annotation `group-visibility` = “false”). In the step after the activity with the ACF, the ticket field groups are faded in using the script of a workflow activity. Please refer also to the Best practices section in this manual for more recommendations concerning the use of ACFs.

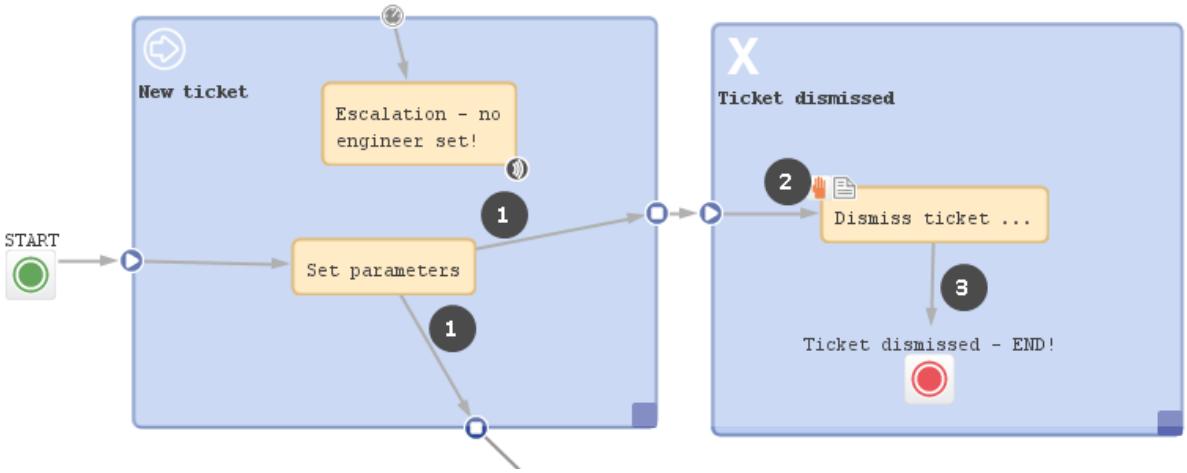


Figure 110: *ConSol CM Process Designer - ACF process logic*

Example (ACF with init script, without precondition script):

- A ticket is created and runs through the automatic activity *Set parameters*.
- It waits behind this activity, at position **(1)** in the scope *New ticket*. The next activities *Dismiss ticket ...* and *New IT ticket* (not shown here) are displayed in the Web Client.
- The engineer selects *Dismiss ticket ... .*
- The init script for the ACF at *Dismiss ticket ...* is executed **(2)**.
- The ACF is shown in the GUI.
  - **Variant 1:**
    1. The ACF is canceled.
    2. The ticket goes back to **(1)**.
  - **Variant 2:**
    1. The ACF is filled-in and confirmed.
    2. The activity *Dismiss ticket ...* is executed (in case there is a script in this activity, the script is executed), the ticket passes through the node and continues on its way **(3)**. In the example above, it is closed.

ACF at manual activity with condition

In case a manual activity has a condition, the activity is only displayed if the condition script returns “true”, i.e. also the ACF is only displayed if the condition script returns “true”.

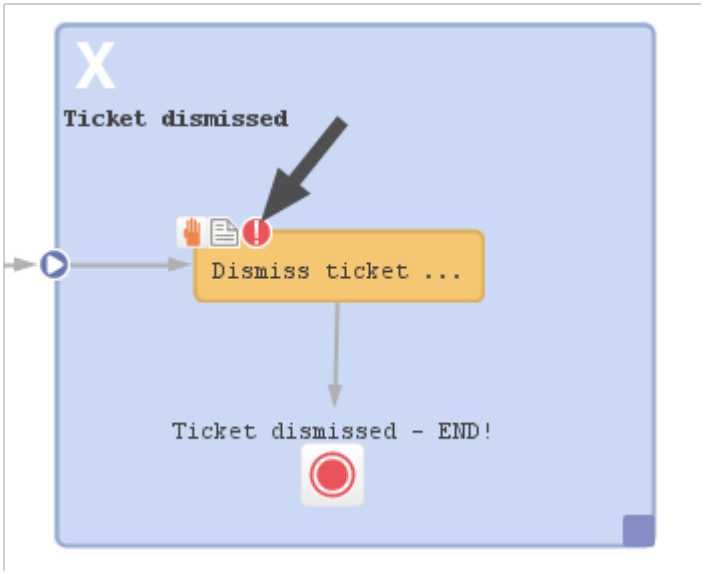


Figure 111: *ConSol CM Process Designer - Manual activity with ACF and condition*

C.7.4.5 Examples for the use of ACFs

Example 1: ACF for the dismissal of a customer request

This example was used in the previous sections. The engineer can only dismiss a customer request when a reason has been given. This is selected from a drop-down menu. Additionally, the engineer can add a note in a text field.

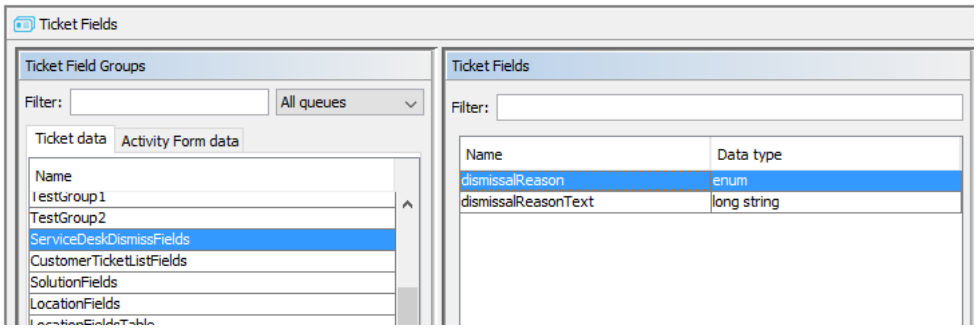


Figure 112: *ConSol CM Admin Tool - definition of ticket fields which will later be used in an ACF*

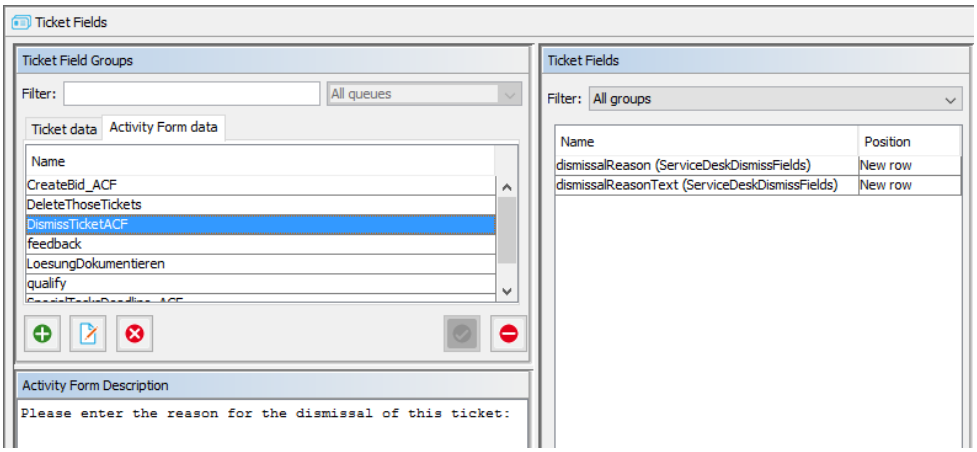


Figure 113: *ConSol CM Admin Tool - ACF definition*

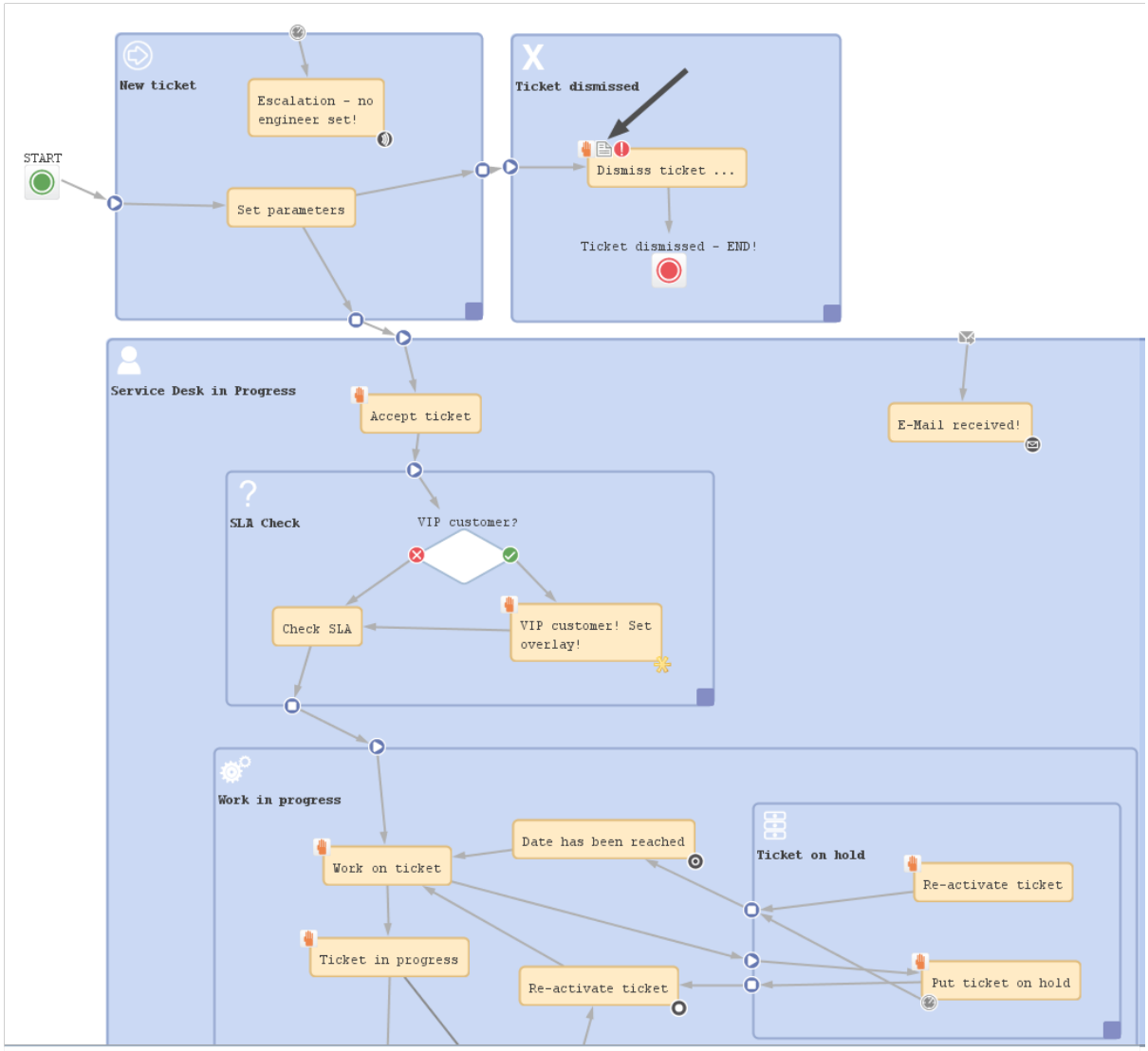


Figure 114: *ConSol CM Process Designer - ACF in workflow*

The Web Client GUI and the ACF properties are shown in the figures of the previous paragraphs.

Example 2: Fill-in sales information when a bid is created

When a sales representative selects the workflow activity *Create bid* in the Web Client, an ACF is opened where several fields are offered. One field is a drop-down menu and a default value is set via script. The other fields are optional. The field *Product* has been filled-in for the ticket in previous process steps, so this field is offered with the selected value. It can either be left unchanged or it can be modified.

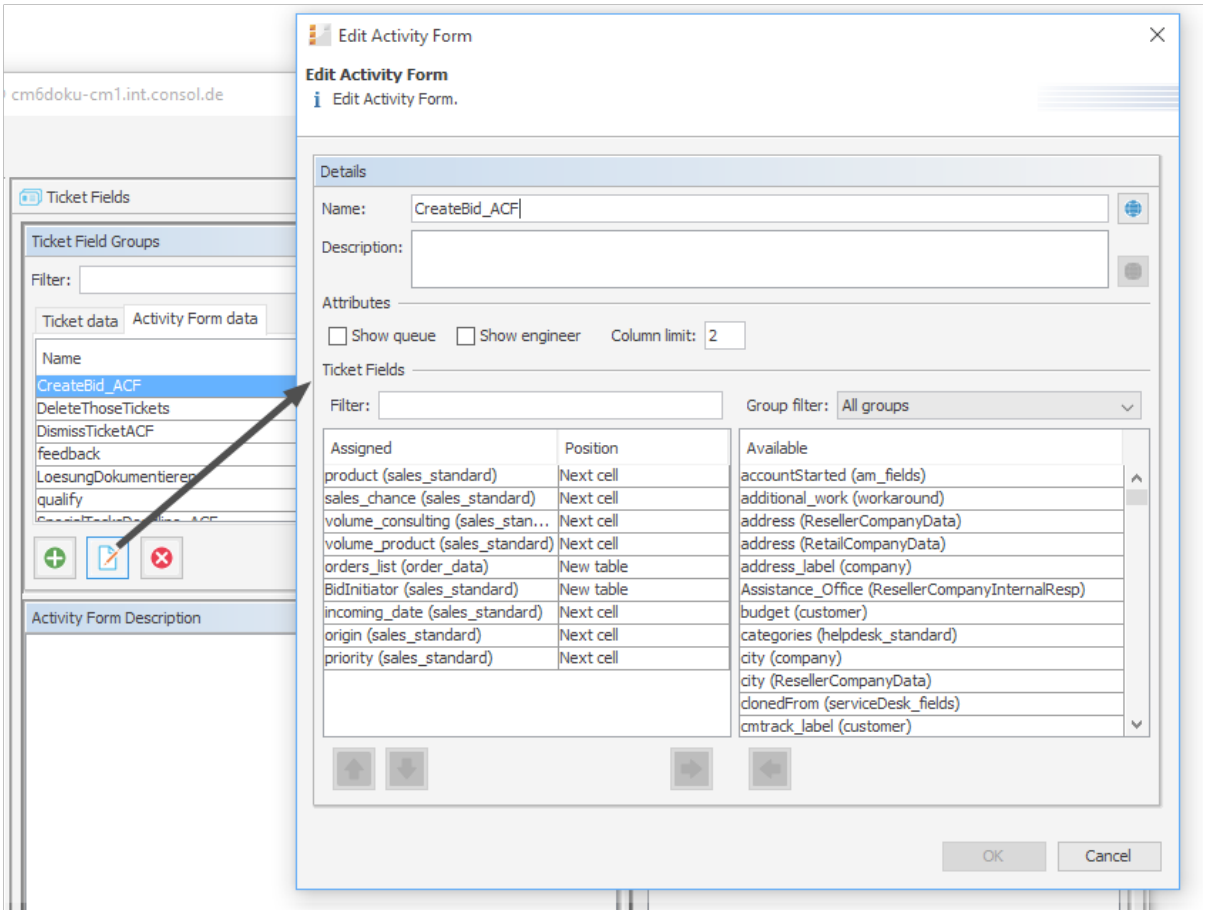


Figure 115: *ConSol CM Admin Tool - ACF for Sales workflow*

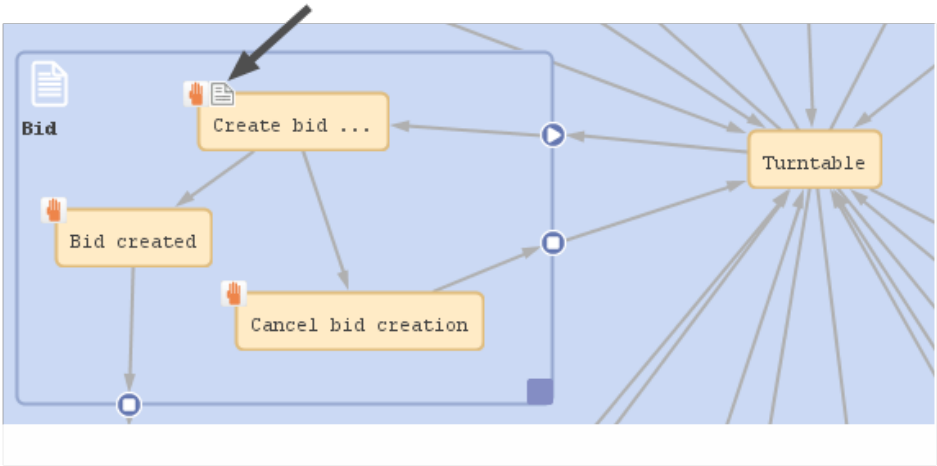


Figure 116: *ConSol CM Process Designer - ACF in Sales workflow*

```
ticket.set("sales_standard.BidInitiator","Mr. Miller")
```

Code example 16: *Process Designer: Initializing script for Create bid ACF*

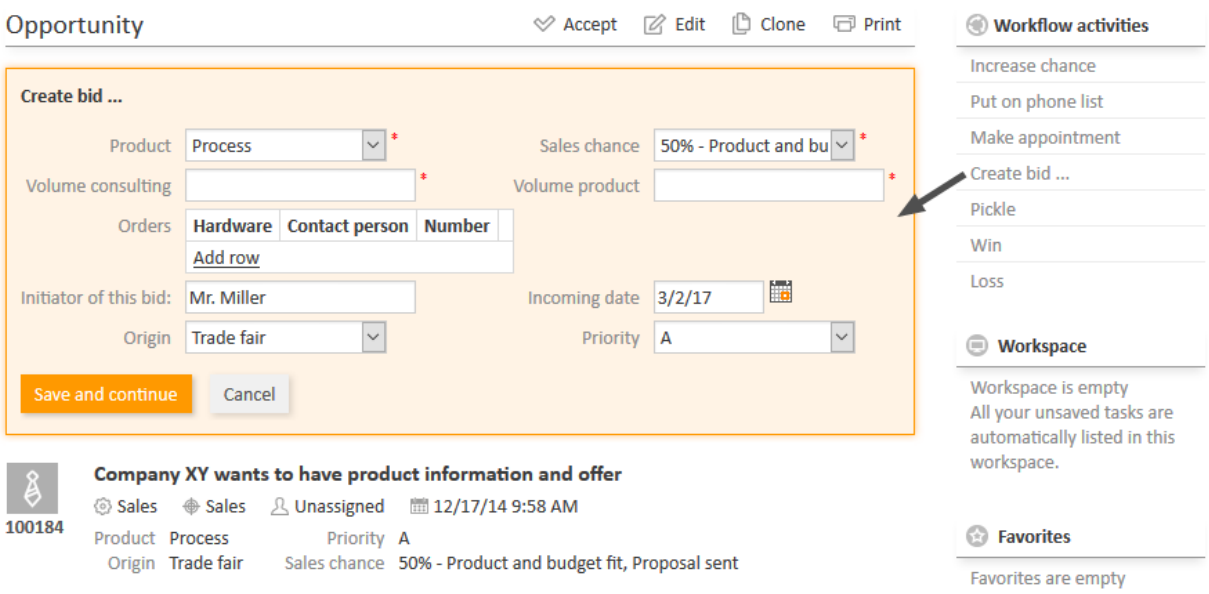


Figure 117: *ConSol CM Web Client - Sales process ACF*

C.7.4.6 ACFs in CM/Track

If an activity with an ACF is configured as expose to customers (see section Editing the properties of a workflow element), the ACF will be available in CM/Track.

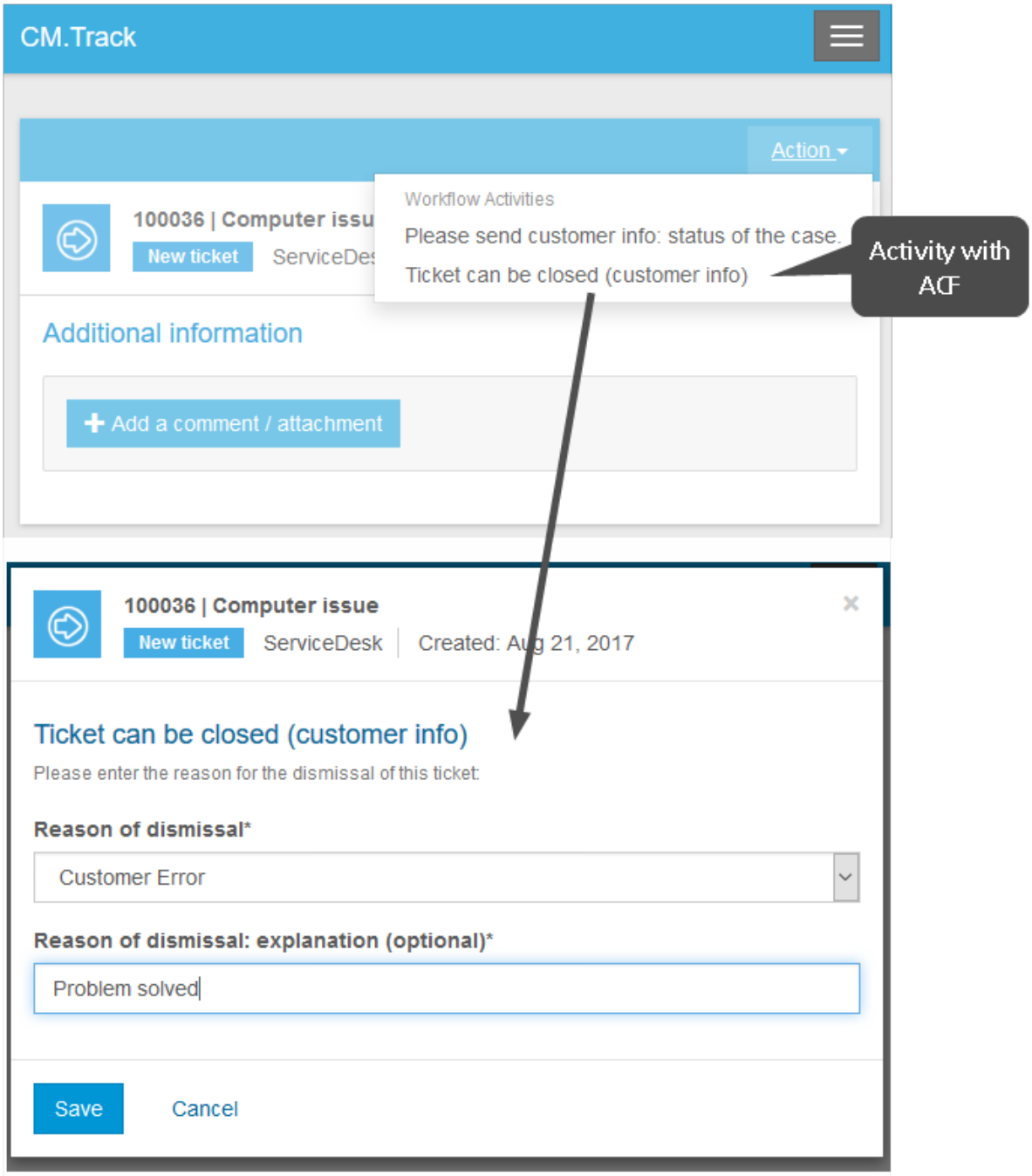


Figure 118: *CM/Track - Activity with ACF*

> Please note that in order to expose data fields (ticket fields, customer fields) in CM/Track, either the CM system property cmas-restapi-core, security.fields.customer.exposure.check.enabled has to be set to “false” (which will expose all fields to customers!) or the property has to be set to “true” and the data fields which should be available ("exposed") have to be annotated with `customer exposure` = “true”. This also applies to all data fields which are used in an ACF!

The following features and functionalities are or are not (yet) available for ACFs in CM/Track.

Supported:

- Ticket fields as well as customer fields can be used
- Mandatory fields (red asterisk)
- Add/remove rows in lists (single field lists or lists of structs)
- Pre-filled fields
- Layout definition using ACF design in the Admin Tool

Not supported (as of CM version 6.11.0.4):

- Queue fields
- Engineer fields
- Dependent enums
- Autocomplete features
- Post activity execution mechanism (e.g. postActivityExecutionHandler)
- Web Client annotations (annotations in the group `cmweb-common`)

However, the features which are not (yet) supported are usually not in operation for customer interactions in CM/Track.

C.8 Jump-out and jump-in nodes

This chapter discusses the following:

C.8.1 Introduction 148

C.8.2 Jump-out nodes 151

C.8.3 Jump-in nodes 154

C.8.1 Introduction

A process often consists of one or more sub-processes, e.g., in an IT help desk, there might be a first level team who accepts and qualifies the tickets, a second level team who can solve several problems, and some third level team with specialists. When you want to represent this process, you have to build a workflow for each special sub-process (1st level, 2nd level, 3rd level). Then the sub-processes have to be linked to make sure the handover of the ticket from one team to the next uses the correct way in the process.

A ticket might pass from the first level to the second level, on to a third level team, back to the second level team with another question, back to another third level team, and then back to the first level team who contacts the customer. So we need connections from one sub-process to the next one, i.e., nodes where a ticket leaves the present workflow, a **jump-out node**, and the counterpart in the following workflow, the **jump-in node**. If the ticket should start at the start node of the new process, no jump-in node is required.

In the Process Designer, jump-out and jump-in nodes are inserted into the workflow by drag-and-drop from the palette and are linked to other workflow elements depending on the desired process.

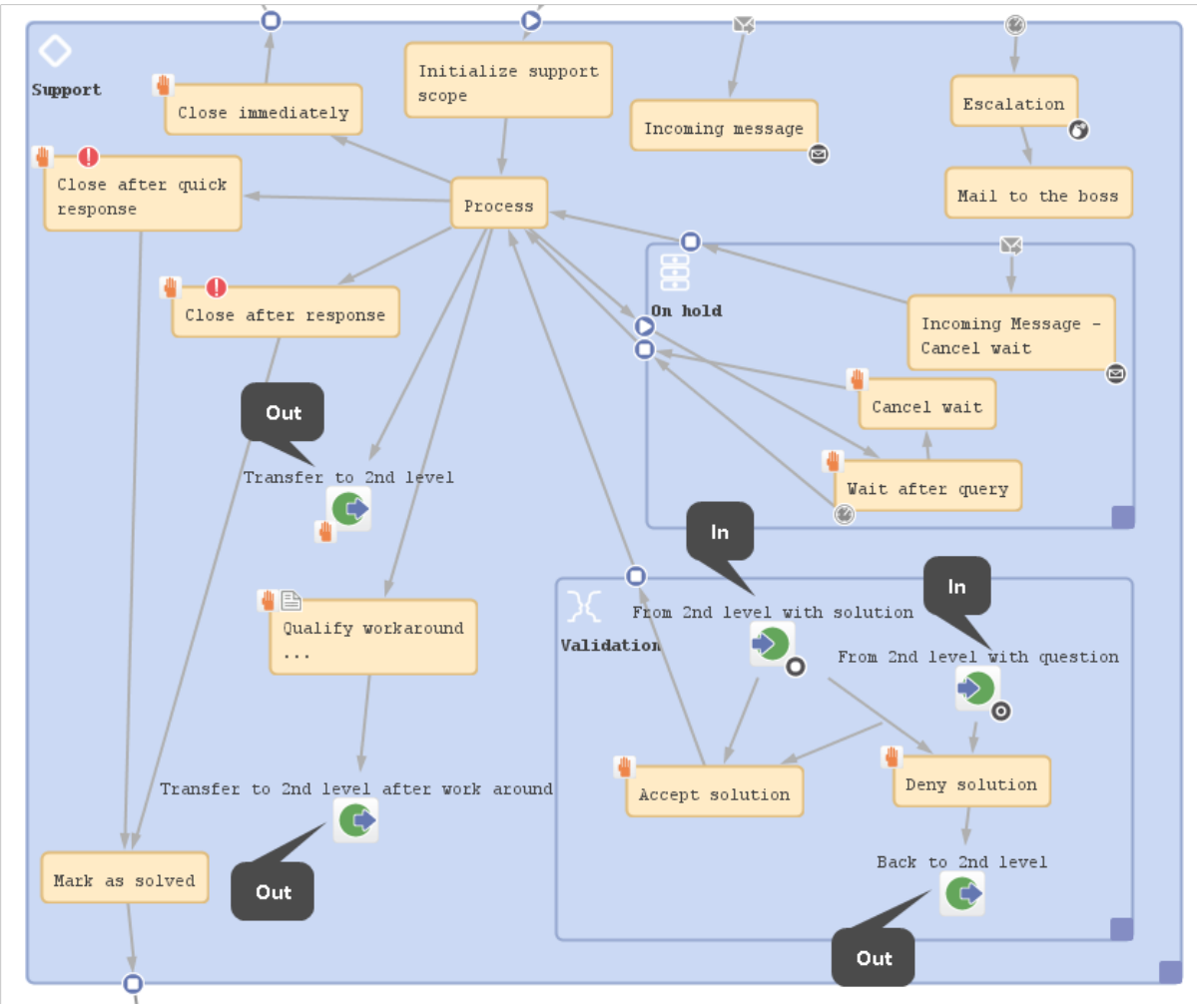


Figure 119: *ConSol CM Process Designer - Example for jump-out and jump-in nodes*

The following question might have come up during the design and development of your business processes:

Why do I have to use more than one workflow and use jump-out and jump-in nodes? Can't I just use one workflow with different scopes?

Answer based on our best practices:

We presume that when different teams are involved that these teams have different access permissions to the system and a team-specific process has to be used for tickets. Therefore, as one workflow is assigned to one queue and as access permissions are also based on queues, each of the involved teams needs at least one queue with one workflow and team-specific access permissions. This implies that a hand-over from one (sub-)process to another is implemented. The hand-over is modeled using jump-out and jump-in nodes. It is possible to use one and the same workflow for several queues or to implement a specific workflow for each team. In any case it is far easier to maintain several small workflows compared to one huge-and-complex workflow.

It is possible to check the roles of an engineer and to display some activities only if a certain role is present. However, this is a mechanism for single activities or a single scope within a workflow. The mechanism should definitely not be used to model the cooperation of entire teams.

C.8.2 Jump-out nodes

A jump-out node defines a position where the ticket is to leave the (sub-)process and to enter the next (sub-)process.

Figure 120: *ConSol CM Process Designer - Jump-out node*

C.8.2.1 Properties of a jump-out node

For a jump-out node the following properties can be defined:

- **name**

  String. Mandatory. Technical object name.

- **label**

  String. Optional. Localized name (if not set, the technical name is used) that will be displayed in the Web Client GUI.

- **description**

  String. Optional. It will be displayed as mouse-over in the Web Client GUI.

- **sort index**

  Selection. Mandatory. Defines the order of the activities in the Web Client GUI.

- **jump out node type**

  Selection. Mandatory. Either *Automatic* or *Manual* has to be selected. In case it is a manual node, the node is marked with the *hand/manual* icon in the Process Designer GUI.

- **script**

  Optional. A script can be defined which is executed when the ticket enters the node.

- **target queue name**

  Selection. Mandatory. Select the queue name to which the ticket should be passed.

- **target jump in node**

  Selection. Optional. Select the jump-in node from the drop-down menu. All jump-in nodes from the workflow of the selected queue are offered. If no jump-in node is selected, the ticket will enter the other process, i.e., the target queue, at the start node.

> ⓘ When you start designing workflows you might have a chicken-and-egg problem when you start to define jump-out and jump-in nodes, because obviously you will have to start with one workflow when the other workflow is not yet present. We recommend to work with dummy queues without specific jump-in node. Then add the correct target queue name and the name of the jump-in node later.

- **history visibility**

  Mandatory. This property defines on which display levels the execution of the activity is shown in the ticket history in the Web Client. The possible values are:

  - 2nd level and 3rd level
  - only 3rd level
  - on every level
  - default (default value)

    The activity is shown on the display level which is configured in the Admin Tool, navigation group *Tickets*, navigation item *History*. Depending on the type of activity, one of the following settings is used:

    - Manual activity or activity with overlay executed
    - Activity executed after escalation
    - Automatic activity executed
  - hidden on all levels
    The execution of the activity is never displayed in the ticket history of the Web Client.

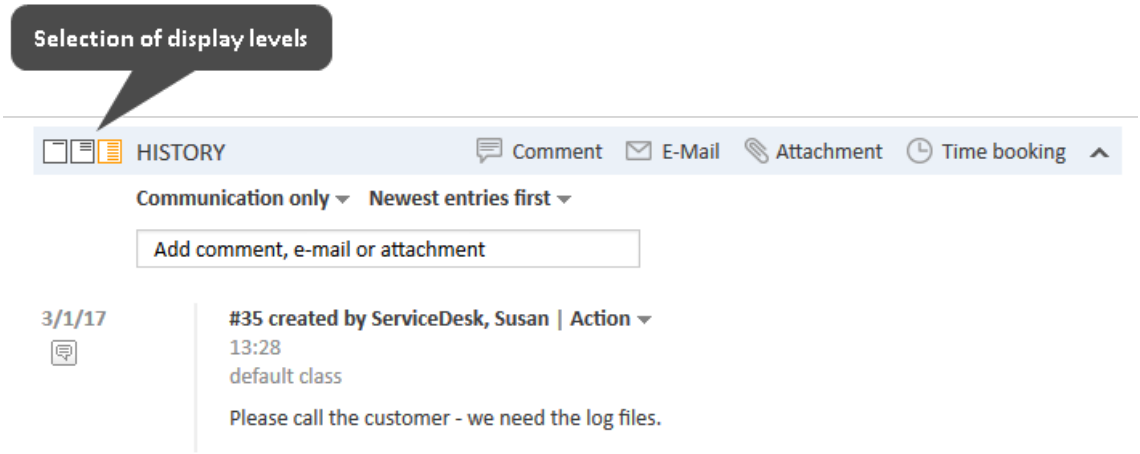


Figure 121: *ConSol CM Web Client - Display levels in the ticket history*

- **disable auto update**

  Defines the behavior of the ticket when an event has been fired or executed. Usually, after an event, a ticket update operation is performed automatically. In case a chain of events is used you should avoid triggering a ticket update operation after every single event. To avoid this, set *disable auto update* to “true” in all events except for the last one. Then, the ticket is only updated once, after the last event.

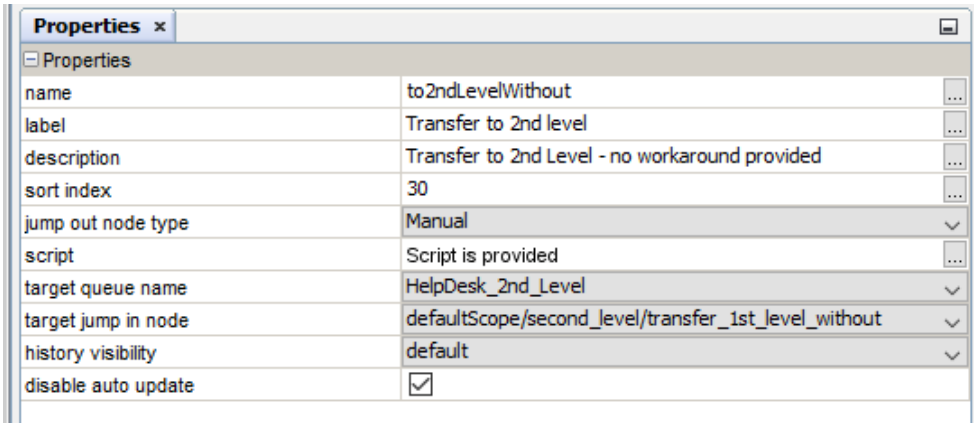


Figure 122: *ConSol CM Process Designer - Jump-out node: Properties editor*

C.8.3 Jump-in nodes

A jump-in node is a node which defines the position where a ticket from another process (queue) can enter a queue with the current workflow. All jump-in nodes of a workflow are offered as target jump-in nodes when the queue with the respective workflow has been selected as target queue for a jump-out node.

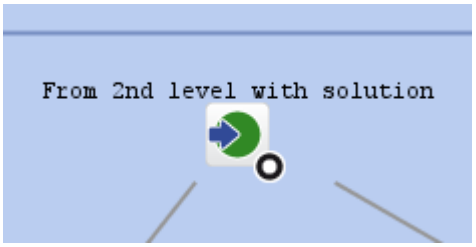


Figure 123: *ConSol CM Process Designer - Jump-in node*

C.8.3.1 Properties of a jump-in node

For a jump-in node the following properties can be defined:

- **name**

  String. Mandatory. Technical object name.

- **label**

  String. Optional. Localized name (if not set, the technical name is used) that will be displayed in the Web Client GUI.

- **description**

  String. Optional. It will be displayed as mouse-over in the Web Client GUI.

- **script**

  Optional. A script can be defined which is executed when the ticket enters the node.

- **overlay**

  Selection. Optional. Click into the orange space to load a standard ConSol CM overlay or use the file explorer (...) for an upload of another icon from the file system.

- **overlay range**

  Selection. Only displayed when overlay has been set.

  - **Activity**
    The overlay is attached only as long as the ticket stands behind the activity. As soon as the next activity is executed, the overlay is deleted from the ticket icon.
  - **Scope**
    The overlay is deleted when the ticket leaves the scope.
  - **Process**
    Once the overlay has been attached to the ticket icon, it stays there for the rest of the process.
  - **Next overlay**
    The overlay is attached to the ticket icon as long as no new overlay appears. In that case, only the new one is attached, the old one is deleted.

- **history visibility**

  Mandatory. This property defines on which display levels the execution of the activity is shown in the ticket history in the Web Client. The possible values are:

  - 2nd level and 3rd level
  - only 3rd level
  - on every level
  - default (default value)

    The activity is shown on the display level which is configured in the Admin Tool, navigation group *Tickets*, navigation item *History*. Depending on the type of activity, one of the following settings is used:

    - Manual activity or activity with overlay executed
    - Activity executed after escalation
    - Automatic activity executed
  - hidden on all levels
    The execution of the activity is never displayed in the ticket history of the Web Client.

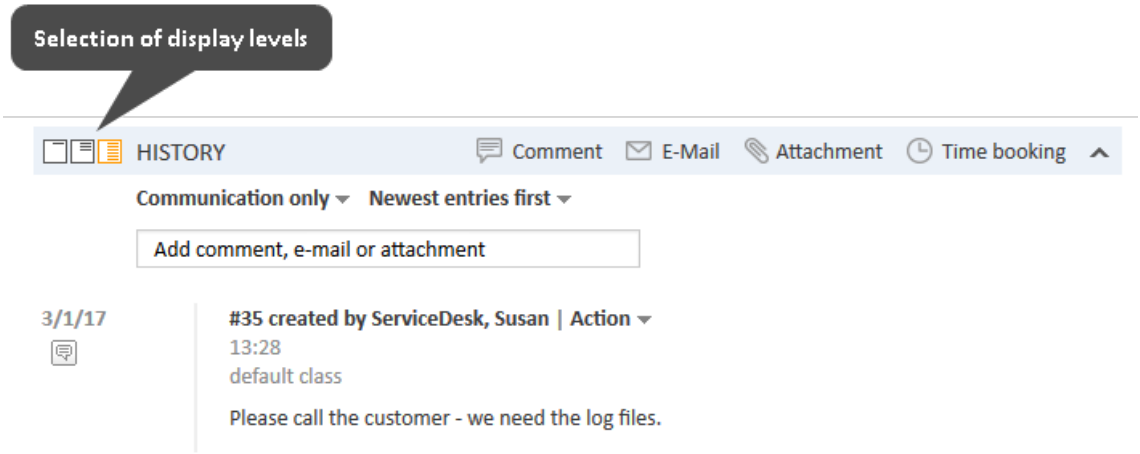


Figure 124: *ConSol CM Web Client - Display levels in the ticket history*

- **disable auto update**

  Defines the behavior of the ticket when an event has been fired or executed. Usually, after an event, a ticket update operation is performed automatically. In case a chain of events is used you should avoid triggering a ticket update operation after every single event. To avoid this, set *disable auto update* to “true” in all events except for the last one. Then, the ticket is only updated once, after the last event.

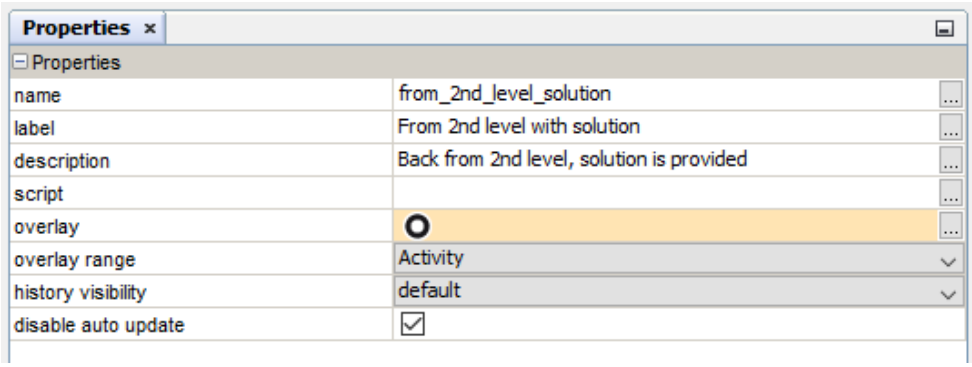


Figure 125: *ConSol CM Process Designer - Jump-in node: Properties editor*

D - Workflow programming

This chapter discusses the following:

D.1 Programming CM scripts 159
D.1.1 Java vs. Groovy-style coding 159
D.2 CM API documentation 160
D.3 CM script types in workflows 161
D.4 Script interactions 161
D.5 Scripts in ConSol CM in general 162
D.6 Process logic 163
D.6.1 Introduction 164
D.6.2 Activities 164
D.6.3 Interrupts and exceptions 168
D.6.4 Loops (errors in workflows) 170
D.6.5 Process logic of time triggers 170
D.6.6 Process logic of business event triggers 170
D.7 Important classes and objects 171
D.7.1 Introduction 171
D.7.2 Important objects 171
D.7.3 Convenience classes and methods 172
D.8 Working with data fields 176
D.8.1 Introduction to data fields 177
D.8.2 Data types for data fields 177
D.8.3 Details about string fields: use annotations to fine-tune strings 180
D.8.4 Data fields for ticket data: ticket fields 182
D.8.5 Data fields for customer data: customer fields 193
D.8.6 Most important methods for accessing customer data 195
D.8.7 Resource data 200
D.8.8 Using data fields for (invisible) variables 202
D.9 Sending emails 203
D.9.1 Introduction to sending emails 203
D.9.2 Important methods 204
D.9.3 Examples 206
D.9.4 Effects of engineer representation rules for emails sent in scripts 216

D.10 Working with path information 219
D.10.1 Introduction 219
D.10.2 Retrieve path information for a workflow element 220
D.10.3 Example for the use of path information 220
D.11 Working with calendars and times 221
D.11.1 Introduction 221
D.11.2 Calculating with dates and times without a business calendar 222
D.11.3 Calculating with dates and times using a business calendar 222
D.12 Working with object relations 224
D.12.1 Working with ticket relations 225
D.12.2 Working with customer relations 233
D.12.3 Working with resource relations 239
D.13 Working with text classes 242
D.13.1 Introduction 242
D.13.2 Examples 244
D.14 Working with attachments 249
D.14.1 Introduction 249
D.14.2 Example: Attaching all attachments of a ticket to the child ticket 249
D.15 Working with overlays 254
D.16 Searching for tickets, customers and resources 255
D.16.1 Introduction 255
D.16.2 Searching for tickets 256
D.16.3 Searching for customers (contacts and companies) 262
D.16.4 Searching for resources 264
D.17 Displaying messages and notifications 266
D.17.1 Messages 266
D.17.2 Notifications 269
D.18 Debug information 271
D.18.1 Introduction 271
D.18.2 Log file configuration 271
D.18.3 Using statements for debug output 272

D.1 Programming CM scripts

As you have seen in the previous sections, ConSol workflows can be set up rather easily using the **Process Designer's graphical interface**. However, in order to bring "real intelligence" into workflows, programming, i.e., writing **ConSol CM workflow scripts** which are used in the workflow activities and preconditions, is required.

ConSol CM scripts are written in Groovy, so you should have at least basic knowledge of this programming language. Since Groovy code runs in the Java Virtual Machine, you can also write Java code. Thus, if you are a Java or Groovy developer, it will be easy for you to learn how to build sophisticated workflows using the ConSol CM Groovy API.

> If you are interested in the Groovy training **Groovy in a Nutshell** provided by ConSol, please ask your CM sales representative.

In the current manual, we use Java style and Groovy style. You can decide which way to follow.

> **Important note for CM systems which are updated from a version prior to CM 6.11 to a version 6.11 or higher**
>
> In ConSol CM version 6.11, the database and therefore the API has been modified considerably. Thus, a great number of classes and methods have become deprecated. In case you update a system to version 6.11, you will have to replace the deprecated classes and methods by the new ones. Please ask your ConSol CM consultant for the API documentation of your CM version! All changes are documented in the CM 6.11 Release Notes.

D.1.1 Java vs. Groovy-style coding

As mentioned above, you have to use Groovy for ConSol CM scripts. There might be different possibilities to express or code the same content. In the following paragraphs, we will give you some hints and provide some examples how to work with the Groovy API.

D.1.1.1 Omitting getter methods in Groovy

Most Groovy objects possess numerous **getter** methods to retrieve values from object attributes. You can either use the complete **getter** methods, or you can use the short (convenience) form. Please see the following examples for workflow scripts.

Use Case	Java-like syntax (extended version)	Groovy syntax (short version)
Get the subject of a ticket.	`String mysubject = ticket.getSubject()`	`def mysubject = ticket.subject`
Get the engineer of a ticket.	`Engineer myeng = ticket.getEngineer()`	`def myeng = ticket.engineer`
Get the main contact of a ticket.	`Unit mymaincontact = ticket.getMainContact()`	`def mymaincontact = ticket.mainContact`

Use Case	Java-like syntax (extended version)	Groovy syntax (short version)
Get the value of a certain ticket field from a ticket.	`String myprio = ticket.get ("helpdesk_fields", "prio")`	`def myprio = ticket.get ("helpdesk_fields.prio")`

Get the unit type for the primary contact.	`Unit mycustomer = workflowApi.getPrimaryContact()`
	`UnitDefinition myunitdef = mycustomer.getDefinition()`
	`UnitDefinitionType mydeftype = myunitdef.getType()`
		`def mycustomer = workflowApi.primaryContact`
		`def myunitdef = mycustomer.definition`
		`def mydeftype = mycustomer.definition.type`

Access to ticket fields cannot be shortened, because there are no getter methods for those fields. Please read the section Working with data fields for details about working with data from ticket fields.

D.1.1.2 Omitting setter methods in Groovy

Most Groovy objects possess numerous **setter** methods to set values for object attributes. You can either use the complete **setter** methods, or you can use the short (convenience) form. Please see the following examples for workflow scripts.

Use case	Java-like syntax (extended version)	Groovy syntax (short version)
Set the subject of a ticket.	`ticket.setSubject("asd")`	`ticket.subject = "asd"`

D.2 CM API documentation

A Groovy API Doc is provided for the ConSol CM API. Please ask your ConSol CM consultant or sales rep if you would like to receive the respective .jar file.

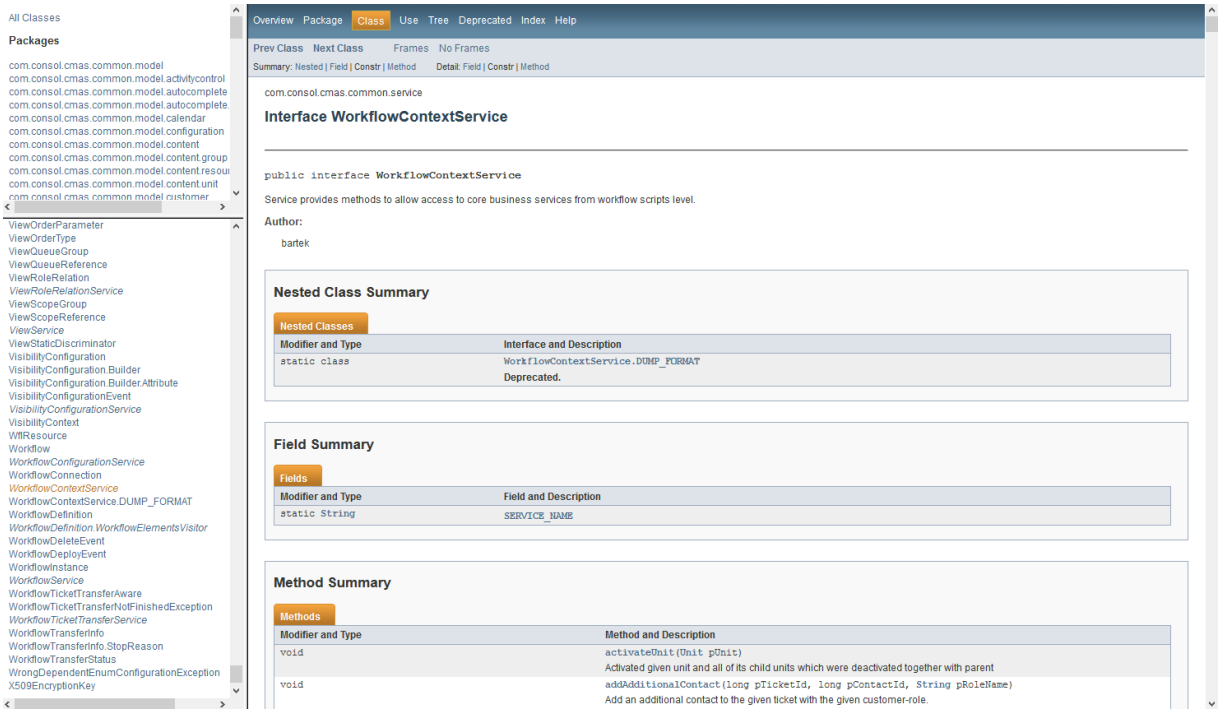


Figure 126: *ConSol CM Groovy API Doc*

D.3 CM script types in workflows

In ConSol CM workflows, scripts are used in the following contexts:

- As activity script for an activity.
- As precondition script for an activity which has to return “true” or “false”.
- As script for a decision node which has to return “true” or “false”.
- As script for a business event trigger which is executed before the trigger fires.
- As script for a time trigger
    - which is executed when the time trigger is initialized, i.e., when the ticket enters the scope where the time trigger is attached.
    - which is executed when the time trigger fires, i.e., when the defined time has elapsed.
- As script for end nodes.
- As script for jump-in or jump-out nodes.
- As precondition scripts for ACFs which have to return “true” or “false”.
- As initializing scripts for ACFs.

Please refer to the respective sections in this manual for an explanation how to insert the scripts.

D.4 Script interactions

For each workflow script, you can decide if the code should run directly in the workflow or if the script should be stored in the Admin Tool, section *Scripts* and should be called from the workflow script. See section Where to save scripts for details.

D.5 Scripts in ConSol CM in general

Please keep in mind that the configuration and programming using the Process Designer make up only "half-the-intelligence" of your ConSol CM system! Numerous configurations and scripts are managed using the Admin Tool! As far as scripts are concerned, please read the section about Admin Tool Scripts in the *ConSol CM Administrator Manual*.

CM/ is a registered trademark of ConSol Consulting & Solutions Software GmbH

D.6 Process logic

This chapter discusses the following:

D.6.1 Introduction 164

D.6.2 Activities 164

D.6.3 Interrupts and exceptions 168

D.6.4 Loops (errors in workflows) 170

D.6.5 Process logic of time triggers 170

D.6.6 Process logic of business event triggers 170

CM/ is a registered trademark of ConSol Consulting & Solutions Software GmbH

D.6.1 Introduction

When you create and modify workflows, it is important to know the basic principles of the workflow engine which result in the behavior of the ticket during the process. Therefore, we will give you a short overview of the basic rules of ConSol CM ticket processing.

D.6.2 Activities

Basic rules:

- Passing through a workflow, a ticket always waits **behind** the last activity, **not** before the next!
- Then it looks for the next activity which can be executed/passed.
- If the next possible activity is a manual activity, the ticket stays at the position behind the previous activity.
- If the next possible activity is an automatic activity or a decision node, the activity or the decision node script is executed, i.e. the ticket passes through this activity or node.
- An activity can have **one or more manual** activities as successor activities **or** an activity can have (only) **one automatic** activity as successor activity.
- When you save a workflow, the Process Designer automatically executes a consistency check. If there are any inconsistencies (e.g. two automatic activities originating at the same predecessor activity), an error message is displayed and the workflow cannot be saved.

Example 1:

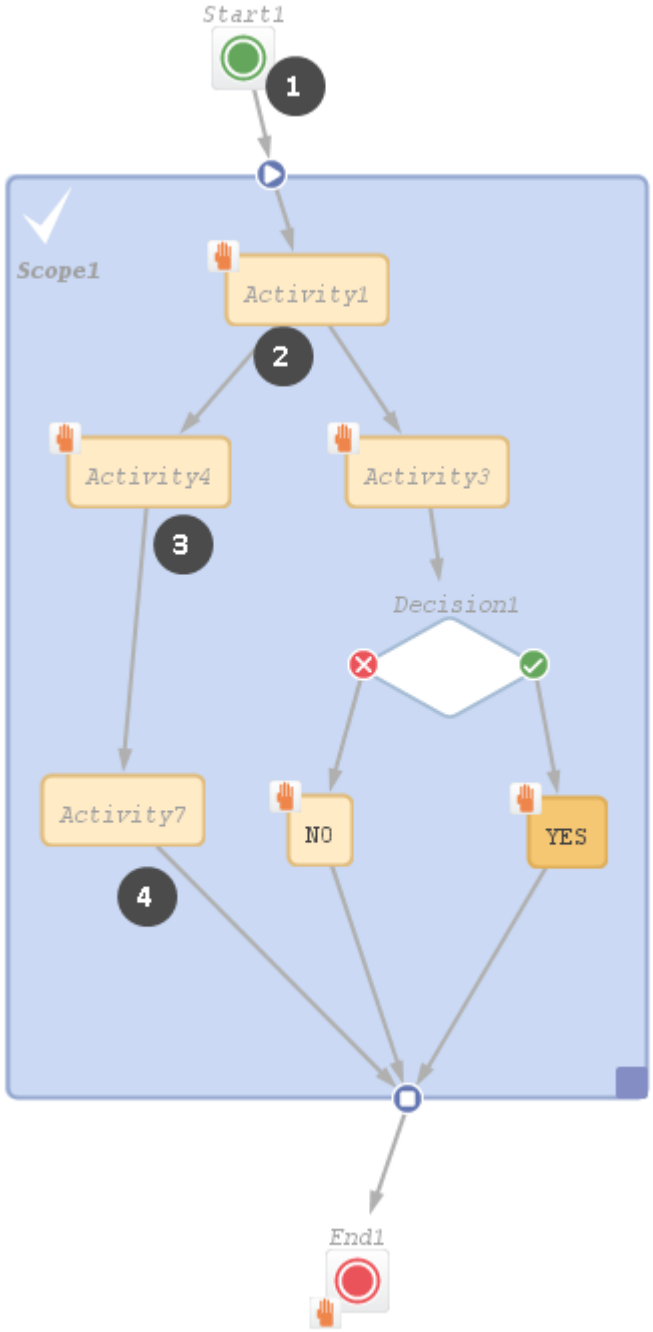


Figure 127: *ConSol CM Process Designer - Process logic example 1*

- The ticket is created and waits behind the START node. (1)
- Activity1 is executed manually. The ticket waits behind Activity1. (2)
- Activity4 is executed manually. The ticket waits behind Activity4. (3)
- End1 is executed manually. The ticket is closed.

Example 2:

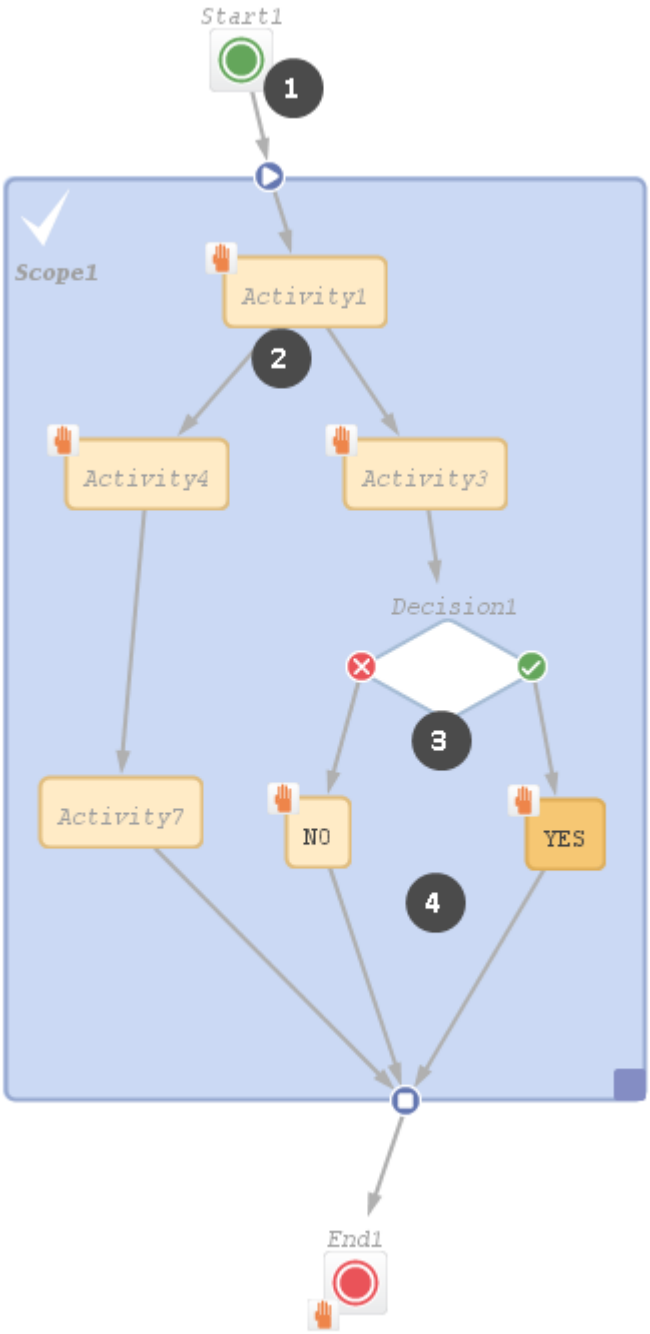


Figure 128: *ConSol CM Process Designer - Process logic example 2*

- The ticket is created and waits behind the START node. (1)
- Activity1 is executed manually. The ticket waits behind Activity1. (2)
- Activity3 is executed manually. The ticket runs automatically through the decision node and takes the FALSE or TRUE exit, depending on the result of the script. Then it waits (3). Only one activity is offered, either NO or YES .

- Either the NO or the YES is executed manually. The ticket waits behind either the NO or the YES activity . (4)
- End1 is executed manually. The ticket is closed.

CM/ is a registered trademark of ConSol Consulting & Solutions Software GmbH

D.6.3 Interrupts and exceptions

In the course of a process, i.e during the time when the ticket is open and engineers work on it, there might be events which have to be taken care of. For example, when an email is received by the ticket or when a time range for an SLA has run out, it is important to register the event and to react accordingly.

There are two ways to define the reaction and behavior of the tickets. You can implement an ...

- **interrupt**
  This is a workflow architecture where the event is registered, one or more automatic activities are executed, and the ticket returns to its previous position in the workflow.
- **exception**
  This is a workflow architecture where the event is registered and, due to the following manual or automatic activities, the ticket leaves its previous position and is taken to a new position within the workflow or in another workflow.

D.6.3.1 Interrupts

Interrupts ...

- are activated by triggers.
- cause a short interruption of the process to react to the trigger event.
- use automatic activities (one or more subsequent automatic actions).
- put the ticket back to its previous position in the workflow, i.e. back to the position where it was when the interrupt event has fired.
- are often used to mark the ticket icon with an overlay, e.g. when an email has been received (see figure below) or when an escalation time has been reached.

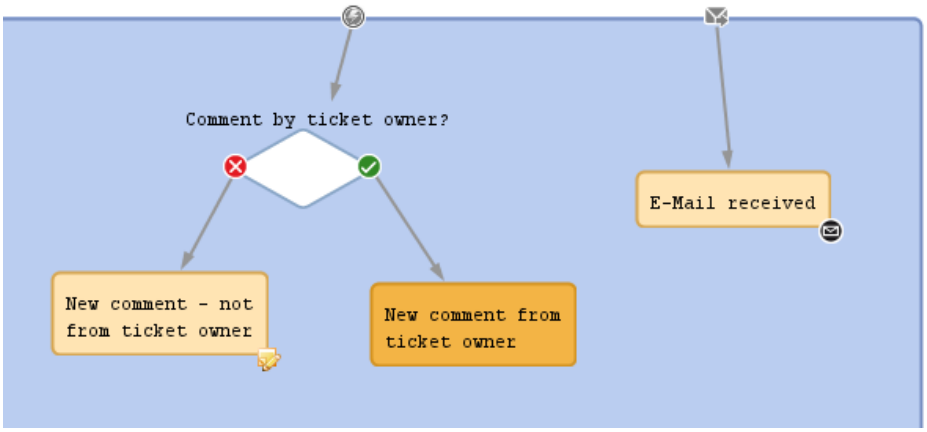


Figure 129: *ConSol CM Process Designer - Two interrupts*

D.6.3.2 Exceptions

Exceptions ...

- are activated by triggers.
- move the ticket from its old position in the workflow to a new position. The latter can be in the

same or in another workflow.

- cause the process to continue at the new position.

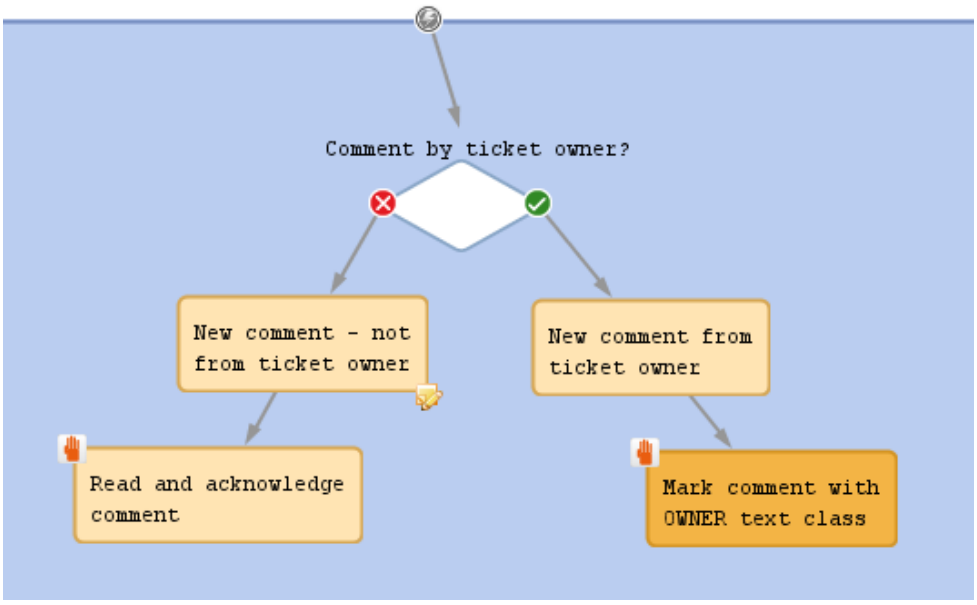


Figure 130: *ConSol CM Process Designer - Exception*

D.6.4 Loops (errors in workflows)

(Infinite) Loops will cause errors in a process. They cannot be detected by the Process Designer, so you could deploy a workflow which contains a loop as shown in the figure below.

However, the process engine detects such loops at run-time and throws an `InfiniteWorkflowLoopException` to prevent the complete system failure. You can of course see the exception in the `server.log` file. In the Web Client, an error message is displayed.

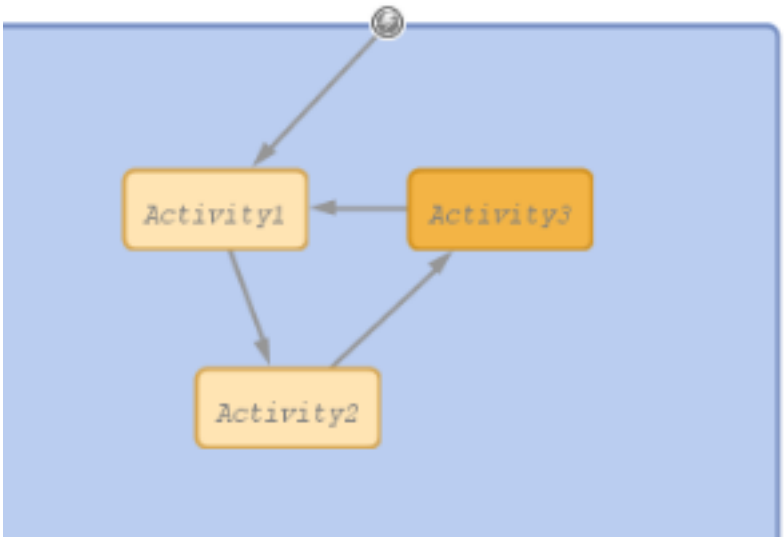


Figure 131: *ConSol CM Process Designer - Loop in workflow*

In the Web Client, an error message will be displayed shortly. It is not possible to save the ticket when an infinite loop exception would be caused.

The exception which you can see in the server.log file will contain lines like the following:

```
2017-08-02 10:30:46,684 ERROR [flow.engine.WorkflowEngineImpl] [Susan-917e766e-
 775c-11e7-a9f8-c5c4447a9199] Error during firing ticket change event:
 com.consol.cmas.workflow.common.InfiniteWorkflowLoopException: Path:
 defaultScope/Service_Desk/Forwarding_Activity-defaultScope/Service_Desk/Do_
 something was already executed
```

Business event triggers can also cause loops when the automatic activity which is attached to the trigger changes the parameter to which the trigger reacts. See section How to avoid self-triggering business event triggers.

D.6.5 Process logic of time triggers

See section Business logic and initialization of a time trigger.

D.6.6 Process logic of business event triggers

See section Business logic of business event triggers.

D.7 Important classes and objects

This chapter discusses the following:

D.7.1 Introduction 171

D.7.2 Important objects 171

D.7.3 Convenience classes and methods 172

D.7.1 Introduction

To make ConSol CM script programming easier, the CM Workflow API provides easy access to objects which are frequently used. Furthermore, convenience classes and methods provide a short way to various objects and methods.

D.7.2 Important objects

Some objects are implicitly present in workflow scripts.

> The same objects are not implicitly present in Admin Tool scripts, i.e. within Admin Tool scripts you will have to use `import` statements for the respective classes or packages!

D.7.2.1 Ticket

In every workflow script, the current ticket can be easily accessed by the object `ticket`. It is derived from the class `Ticket` and is implicitly present. No import and no instantiation is required.

Example:

```
def myId = ticket.getId()
//or shorter, Groovy-like:
def myId = ticket.id
```

Code example 17: *Using the ticket object*

D.7.2.2 workflowAPI

The object `workflowApi` is also implicitly present. It provides easy access to the interface `WorkflowContextService` which is used for numerous operations.

Examples:

```
workflowApi.createChildTicket(myticket, pTicketText, pCustomer)
```

Code example 18: *Using workflowApi to create a child ticket*

```
workflowApi.deactivateTimer("defaultScope/Service_Desk/TimeTrigger1")
```

Code example 19: *Using workflowApi to deactivate a trigger*

```
workflowApi.addValidationError("1", "The ticket cannot be closed before a solution
 is provided. Please fill-in solution and mark it with text class SOLUTION first.")
```

Code example 20: *Using workflowAppi to display a GUI message for the engineer/user*

D.7.2.3 trigger

This object is implicitly available in the scripts of time triggers (*script on timer start*, *script after timer*).

```
def addedEscalMillis = 0

switch (ticket.queue.name) {
   case "HelpDesk_1st_Level":
      addedEscalMillis = 12*60*60*1000L;
      break;
   case "HelpDesk_2nd_Level":
      addedEscalMillis = 24*60*60*1000L;
      break;
   case "ServiceDesk":
      addedEscalMillis = 4*60*60*1000L;
   }

trigger.setDueTime(addedEscalMillis)
```

Code example 21: *Example for a script on timer start*

D.7.3 Convenience classes and methods

The ConSol CM API provides various convenience interfaces and methods which make access to most objects of every-day CM programming a lot easier. Most of those convenience interfaces are part of the package `com.consol.cmas.common.service` and its sub-packages. Please refer to the *ConSol CM Java API documentation* for details. Here, we will show you some examples which might prove useful for most CM programmers.

The implementing instance of the interface is always available by replacing the first letter, which is a capital letter, in the class name by a lower case one, e.g. the object (singleton) with the interface `EngineerService` is available with the object `engineerService`, see *Example 2*.

D.7.3.1 Example 1: Using ConfigurationService to retrieve system properties

```
def tic_nr = configurationService.getValue("custom-mycompany-properties","engineer_
 management.ticket.nr")

// then: ... do something with the engineer management ticket,
// e.g. find out the name of the next engineer a service ticket
//should be assigned
```

Code example 22: *Using the ConfigurationService to retrieve the number of the engineer management ticket*

```
def baseUrl = configurationService.getValue("custom-mycompany-
 properties","base.url.mycompany")
def url = baseUrl + "/cm-client/ticket/ticket_name/" + ticket.getName()
def itComplete = url + " " + ticket.getName()
//alternative: def itComplete = "${url} ${ticket.name}"

// ... do something with the ticket url, e.g. place a link to a child ticket in a
 table of the parent ticket
```

Code example 23: *Using the ConfigurationService to retrieve base URL of the system*

D.7.3.2 Example 2: Using EngineerService to assign the ticket to the current engineer

In numerous processes, it is required to assign the ticket to the engineer who executes the workflow activity *Accept ticket* (or something similar). You can use the following script code for this.

```
// retrieve the engineer who is currently logged in and who is executing the
 workflow activity
def eng = ticketService.current
//assign the current ticket to the current engineer:
ticket.engineer = eng
```

Code example 24: *Assigning the ticket to the current engineer*

D.7.3.3 Example 3: Using EngineerService to assign the ticket to an approver

```
// Script does the following:
// Hand-over ticket to approver only when approver has been set in ticket as
 additional engineer
// Import package, because classes are not available in workflow otherwise:
import com.consol.cmas.common.model.ticket.user.function.*

// Get the name of the approver which has been written/stored in a Custom Field, //
 namely the field with the name
// CF_ApproverName in the Custom Field Group CF_GroupApproverData. The value could
 be for example Mr. Miller:
def gen = ticket.get("CF_GroupApproverData.CF_ApproverName").getName()

// Get the engineer object where the name Mr. Miller is set, i.e.
// the engineer object of the desired approver:
def gen_eng = engineerService.getByName(gen)

// Get the ticketFunction object which represents the ticketFunction (engineer
 role) Approver:
TicketFunction tf = ticketFunctionService.getByName("Approver")

// Add the engineer object of Mr. Miller as Approver. i.e.
// in the ticketFunction (engineer role) Approver to the ticket.
// One of the paramaters is ticket. This does not have to be instantiated,
// because it is implicitly present in workflow
// scripts:
def tu = ticketUserService.addTicketUser(ticket, gen_eng, tf, "Approver")

// Assign the ticket to the engineer, i.e. set the engineer Mr. Miller also as
 ticket owner.
def tic2 = workflowApi.assignEngineer(ticket, gen_eng)
```

Code example 25: *Use of EngineerService*

We have two assignments here:

1. Mr. Miller is set as additional engineer in the engineer role *Approver*.
2. Mr. Miller is set as ticket owner.

D.7.3.4 Example 4: Using EnumService to retrieve an enum value by name

```
def enumValueMLA = enumService.getValueByName( "priority", "REGULAR" )
ticket.set( "helpdesk_fields.prio", enumValueMLA )
```

Code example 26: *Using EnumService to retrieve an enum value by name*

D.7.3.5 Example 5: Using TicketService to retrieve all tickets of a certain view

```
List<Ticket> mylist = ticketService.getByView(new ViewCriteria(
   viewService.getByName("helpdesk_active_tickets"),
   ViewAssignmentParameter.allAssignedTickets(),
   ViewGroupParameter.allTickets(),
   viewOrderParameter.addByName(true)))
```

Code example 27: *Using TicketService to find ticket of a view*

D.7.3.6 Example 6: Using EngineerRoleRelationService to send an email to all engineers of a role

```
// Send e-mail to all engineers of a regular role

def mail = new Mail()
mail.setTo(engineerRoleRelationService.getEngineersWithRoles(roleService.getByName
 ("Supervisor"))*.email.join(","))
mail.setSubject("Ticket (${ticket.name}) -- Escalation!")
mail.setText(workflowApi.renderTemplate("Ticket escalation note to supervisor"))
mail.send()
```

Code example 28: *Using the EngineerRoleRelationService to send an email to all engineers of a role*

D.8 Working with data fields

This chapter discusses the following:

D.8.1 Introduction to data fields 177

D.8.2 Data types for data fields 177

D.8.3 Details about string fields: use annotations to fine-tune strings 180

D.8.4 Data fields for ticket data: ticket fields 182

D.8.5 Data fields for customer data: customer fields 193

D.8.6 Most important methods for accessing customer data 195

D.8.7 Resource data 200

D.8.8 Using data fields for (invisible) variables 202

CM/ is a registered trademark of ConSol Consulting & Solutions Software GmbH

D.8.1 Introduction to data fields

The access to data fields is an essential part of ConSol CM programming. It is potentially required in all scripts of the system, workflow as well as Admin Tool scripts, no matter of which type. Here, we will set the focus on workflow programming, but the access to data fields is basically the same in all scripts.

There are three types of data fields:

- **Ticket fields**
  Used to define ticket data, managed in ticket field groups
- **Customer fields**
  Used to define customer data as part of the FlexCDM, the customer data model. Managed in customer field groups.
- **Resource fields**
  Only if CM/Resource Pool is in operation. Used to define resource data as part of the resource data model. Managed in resource field groups.

> **Rules for work with data fields**
>
> When you work with ticket fields, customer fields or resource fields, there are two main rules you have to keep in mind:
>
> 1. All data fields are always managed and referenced in data field groups, e.g. when you want to retrieve the value of a ticket field, you use e.g., *<ticket field group name>.<ticket field name>*
> 2. You always use the technical unique name to reference a data field group or a data field, not the localized value!

D.8.2 Data types for data fields

A data field is always of a certain data type. As for any variable in programming, it depends on the data type how you have to handle the value of the field, e.g. a STRING field cannot be used for calculating numbers, an ENUM field needs a specific access method.

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

- **autocomplete**
  A data field which contains a scripted autocomplete list. This is a dynamic list which is based on a script of type *Text Autocomplete*. A detailed explanation of scripted autocomplete lists is given in the *ConSol CM Administrator Manual,* section *Admin Tool Scripts, Scripted Auto-complete Lists*.
- **boolean**
  Values: true/false. Depending on the annotation `boolean-type`, the value is displayed as check-box, radio buttons, or drop-down list.

  > If a boolean field has not been touched, its value is “NULL”. If it is checked, its value is “true”, and if it is unchecked again, its value is “false”.

- **date**
  Format and accuracy can be set by annotations.

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

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

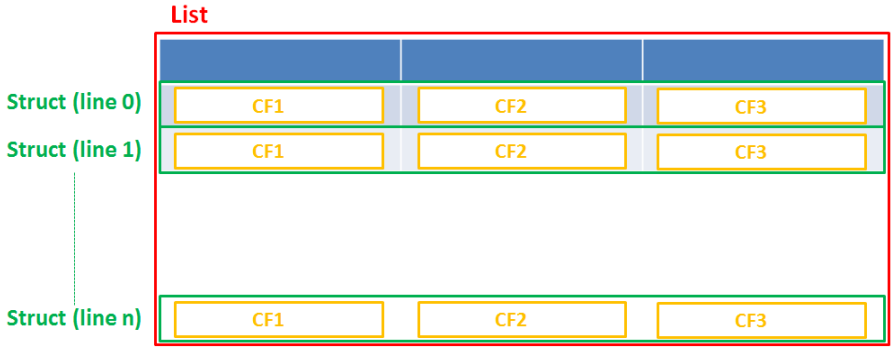


Figure 132: *Scheme: List of structs*

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

Figure 133: *List of structs, technical principle*

- **number**
  For integer values.

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

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

- **long string**
  For large objects.

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

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

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

- **contact data reference**
  Special data type used internally for referencing the contacts associated with a ticket. This data type is not displayed but only used internally in the CM system.
- **MLA field**
  This data type is used for fields that contain hierarchical lists with a tree structure called *MLA* (Multi Level Attributes). The name of the field is the name of the new MLA that has to be defined within the MLA Administration in the Admin Tool. The group of the field has to be referenced when the MLA is created.

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

D.8.3 Details about string fields: use annotations to fine-tune strings

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

How can I ...

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

The link will also be displayed as tooltip.

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

- It starts with `file:` followed by regular slashes:
  - three slashes "///" for files on the same computer as the browser (alternatively "//localhost/") or
  - two slashes followed by the server name followed by another slash for files on file servers accessible from the computer running the browser.
- These are followed by the full path to the file ending with the file name.
- The path on Microsoft Windows systems is also written with forward slashes instead of backslashes.
- The drive letter of a local path on Microsoft Windows systems is noted as usual, for example `C:`.
- Paths with spaces and special characters like "{, }, ^, #, ?" need to be percent encoded ("%20" for a space for example) for Microsoft Windows systems.

Example URLs:

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

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

... define a **label**?

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

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

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

Value for annotation `email`: “true”

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

... define a scripted autocomplete list?

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

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

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

A detailed description of scripted autocomplete lists is provided in section *Scripted Autocomplete Lists* in the *Administrator Manual*.

... define a field for the **CM/Track login**?

Value for annotation `username`: “true”

Will be used for authentication against CM/Track server. Only for customer fields in a contact object.

... define a field for the **CM/Track password**?

Value for annotation `password`: “true”

Will be used for authentication against CM/Track server (in DATABASE mode). Only for customer fields in a contact object.

... define a field containing personal data?

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

This annotation can be assigned to ticket and contact fields. Contact fields with this annotation will be deleted when a contact is anonymized. Ticket fields with this annotation will be deleted when the main customer of the ticket is anonymized.

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

D.8.4 Data fields for ticket data: ticket fields

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

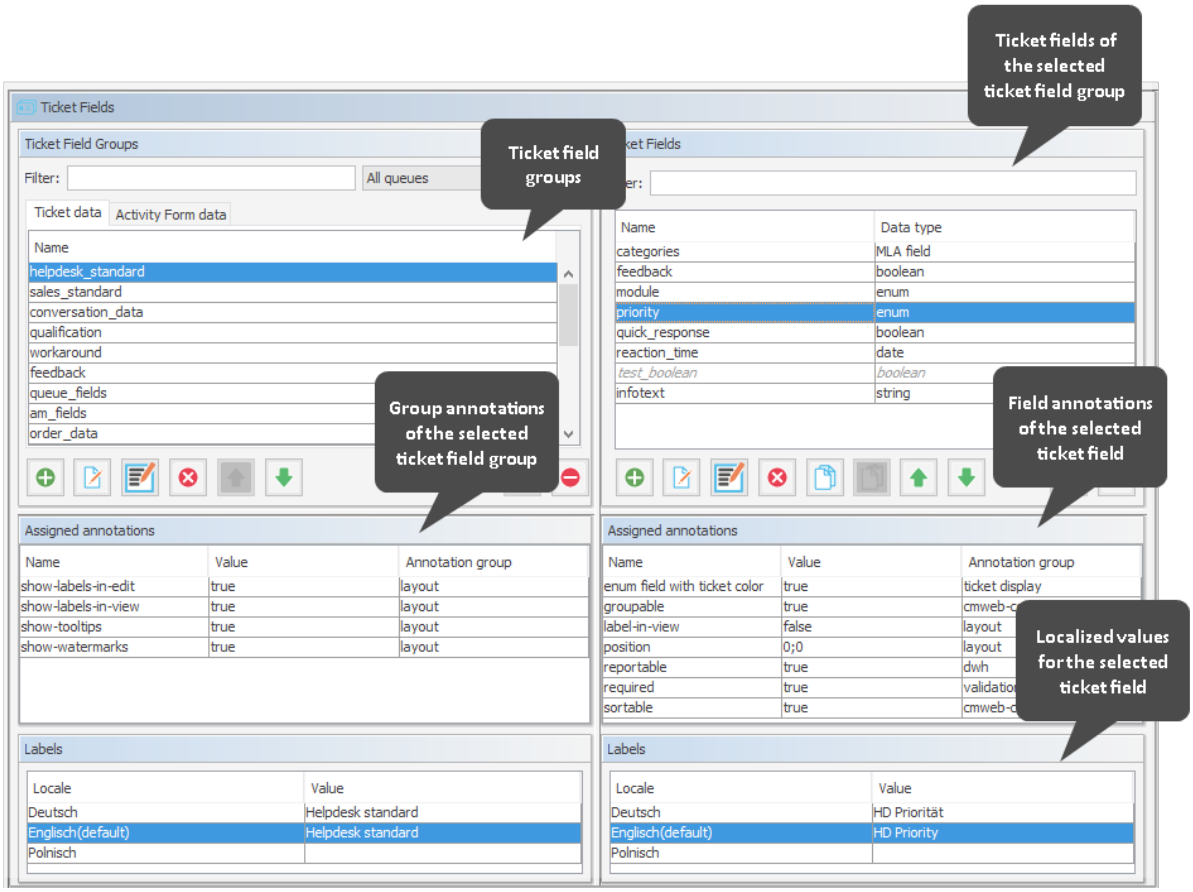


Figure 134: *ConSol CM Admin Tool - Ticket field administration for ticket data*

D.8.4.1 Most important methods for accessing ticket fields

Three methods are of major importance for programming ticket field access in CM scripts. They all are methods of the class `Ticket`:

- **Ticket.get()**
  - For retrieving data from a ticket field.
- **Ticket.set()**
  - For setting data in an already existing ticket field.
- **Ticket.add()**
  - For calculating with a value within a ticket field, i.e. to add a certain time range to a DATE field.
  - For adding a new line in list fields (simple lists and tables).

Another method might be used when a field should be emptied, i.e., when its value should be set to “null”:

- **Ticket.remove()**
  - Sets the value of the field to “null”.

D.8.4.2 Retrieve ticket field values

To retrieve data from a ticket field in a script, you have to reference it by using the technical names of the ticket field group and of the ticket field. The method which has to be used can vary depending on the data type of the ticket field.

Simple data types

The following examples refer to the ticket fields in the figure above. The method which should to be used (because it is the most convenient way) is:

```
ticket.get("<ticketFieldGroupName>.<ticketFieldName>")
```

> Please keep in mind that the **getter** method for a field might return either a value or an object, depending on the type of data field! Please see the following example for an explanation

The following Admin Tool script, e.g., called from a workflow, will display ticket data:

```
// display info from custom types of various data types:
import com.consol.cmas.common.model.ticket.Ticket

Ticket ticket = workflowApi.ticket
log.info 'Display enum from helpdesk CF group ... '

def myfield = ticket.get("helpdesk_standard.priority")
log.info 'Class of helpdesk_standard.priority field is ' + myfield.getClass()
log.info 'Value of helpdesk_standard.priority field is ' + myfield.getName()
log.info 'Retrieve value directly, method 1, using getter method ... '

def my_new_field1 = ticket.get("helpdesk_standard.priority").getName()
log.info 'Result is ' + my_new_field1
log.info 'Retrieve value directly, method 2, using direct access to attribute ... '

def my_new_field2 = ticket.get("helpdesk_standard.priority").name
log.info 'Result is ' + my_new_field2
log.info 'Display value of simple data field ...'

def fb = ticket.get("helpdesk_standard.feedback")
log.info 'Value of feedback boolean field is ' + fb
```

Code example 29: *Admin Tool script used to display ticket fields*

The following output will be displayed in the server.log file:

```
Display enum from helpdesk CF group ...
Class of helpdesk_standard.priority field is class com.consol.cmas.common.model.customfield.enums.EnumValue_$$_jvstb5d_61
Value of helpdesk_standard.priority field is low
Retrieve value directly, method 1, using getter method ...
Result is low
Retrieve value directly, method 2, using direct access to attribute ...
Result is low
 Display value of simple data field ...
Value of feedback boolean field is true
```

Explanation:

If you want to retrieve the value of a ticket field which contains an object (e.g., an EnumValue), you have to use methods like `getName()` to retrieve the actual value, because ticket.get ... only provides the object. The `.name` notation is a simplified (Groovy) version of the getter method.

If you want to retrieve the value of a simple data field, you can do this "directly": ticket.get ... will provide the value.

A precondition script of a workflow activity could look like the following code:

```
boolean vip_info = ticket.get("am_fields.vip")

if(vip_info == true){
   return true
}
else {
   return false
}
```

Code example 30: *Precondition script where boolean value is checked*

Or shorter:

```
return ticket.get("am_fields.vip")
```

Code example 31: *Precondition script where boolean value is checked, short version*

Enum values

An enum (ordered list) field is a field where the value is one of various list values. For example, a list with priorities is the basis for an enum field. To retrieve the value of an enum field, you can use the same syntax as for simple data types. The `get` method provides the enum list value, the `getName()` method provides the STRING attribute with the name of the value.

```
def prio = ticket.get("helpdesk_standard.priority")
log.info 'Priority is now ' + prio.getName()
//or shorter:
log.info 'Priority is now ' + prio.name
```

Code example 32: *Retrieving an enum value for a ticket field*

MLAs

Work with an MLA value

The Java class for MLAs (Multi Level Attributes) is `MultiEnumField`. It is handled like a regular enum field, i.e. to retrieve the current value of an MLA field, you can use the Java/Groovy code like the one shown in the example below.

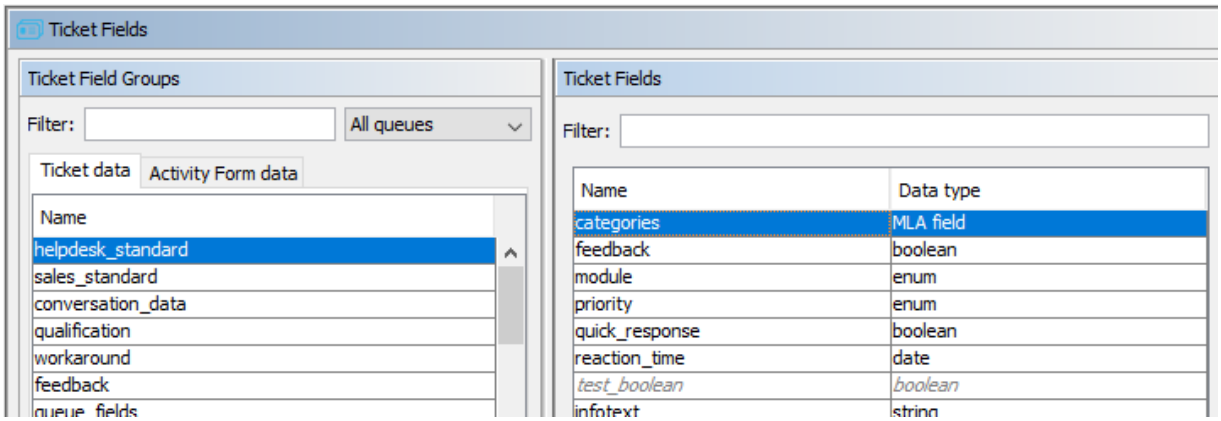


Figure 135: *ConSol CM Admin Tool - Definition of a ticket field of type MLA*

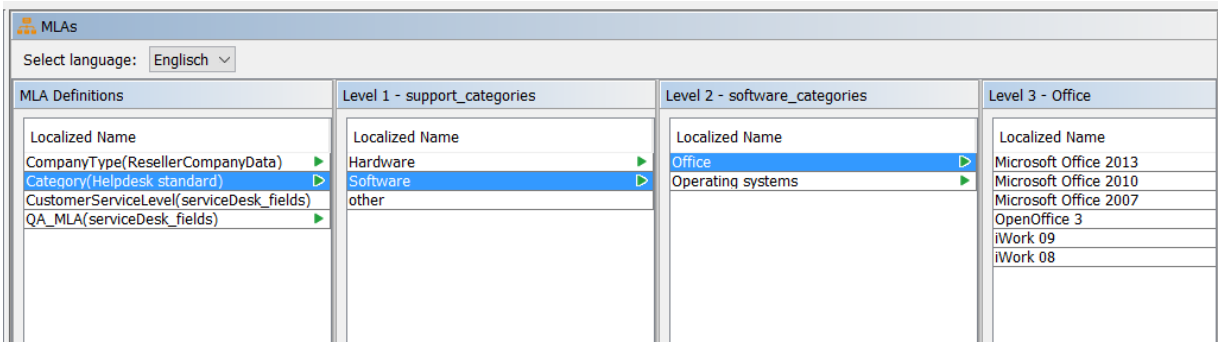


Figure 136: *ConSol CM Admin Tool - MLA definition*

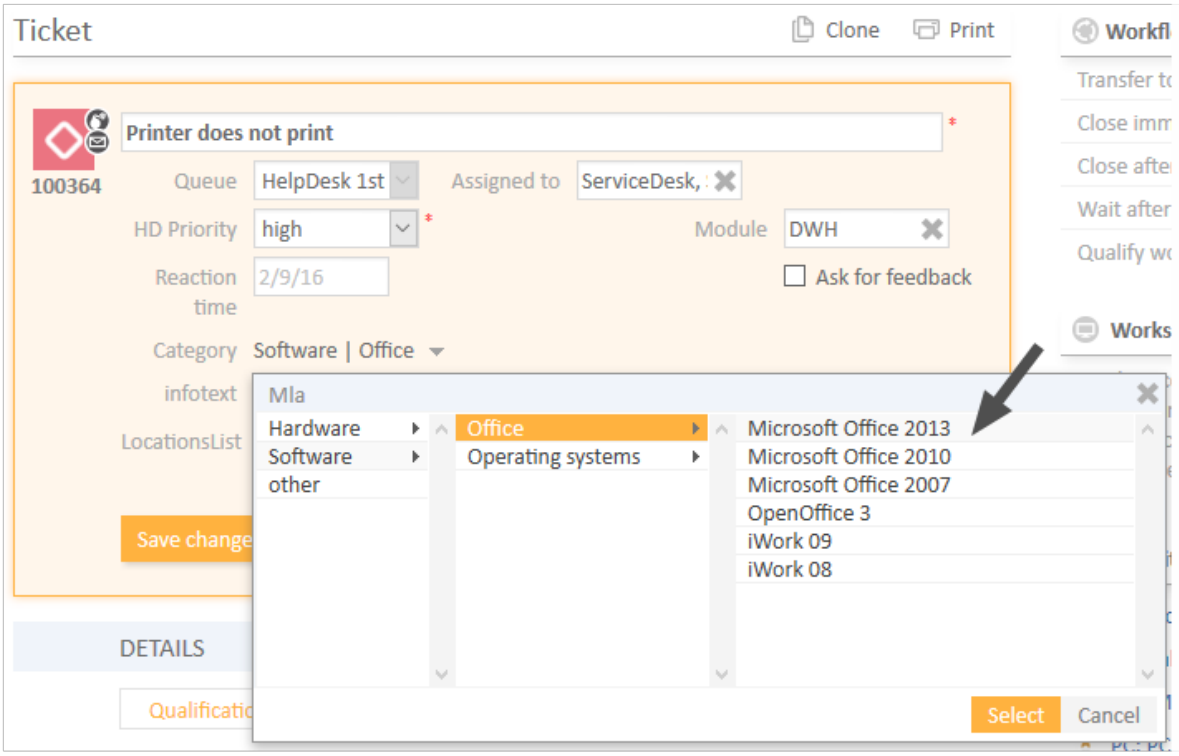


Figure 137: *ConSol CM Web Client - Setting a value (technically: name) of the MLA*

```
// Get
def myEnumValueName = ticket.get("GROUP_NAME.MLA_FIELD_NAME").name
// Set
EnumValue enumValue = enumService.getValueByName("ENUM_NAME","ENUM_VALUE_NAME")
ticket.set("GROUP_NAME.MLA_FIELD_NAME", enumValue)
```

Example with technical name of the field and localized name:

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

log.info 'Displaying MLA value ...'
def my_mla_field = ticket.get("helpdesk_standard.categories")
log.info 'MLA value categories (technical name) is now ' + my_mla_field.name
def my_mla_field_localized = localizationService.getLocalizedProperty
 (EnumValue.class, "name", my_mla_field.id, engineerService.getCurrentLocale())
log.info 'MLA value categories (localized name) is now ' + my_mla_field_localized
```

Code example 33: *Code (Workflow or Admin Tool script) for displaying MLA data*

```
stdout] [Susan-] Displaying MLA value ...
stdout] [Susan-] MLA value categories (technical name) is now ms_2013
stdout] [Susan-] MLA value categories (localized name) is now Microsoft Office 2013
```

Figure 138: *Log output for the example above*

Work with the entire branch of the selected MLA value

If you need to retrieve the entire path to the selected MLA value, you can proceed as follows:

```
log.info 'Displaying MLA branch ...'
List<EnumValue> mlaPathElements = mlaService.getAssignedMla(ticket, "helpdesk_
 standard", "categories")
def mlaPath1=""
mlaPathElements.each() {elem ->
   mlaPath1 = elem.name + ' -- ' + mlaPath1
}
log.info 'mlaPath1 is now ' + mlaPath1

//or shorter:
def mlaPath2 = mlaService.getAssignedMla(ticket, "helpdesk_standard",
 "categories").reverse()*.name.join(" -- ")
log.info 'mlaPath2 is now ' + mlaPath2
```

Code example 34: *Retrieving the entire path to the selected MLA value*

```
stdout] [Susan-] Displaying MLA branch ...
stdout] [Susan-] mlaPath1 is now software -- office -- ms_2013 --
stdout] [Susan-] mlaPath2 is now software -- office -- ms_2013
```

Figure 139: *Log output for the example above*

Lists

Lists of simple data types

A list of simple data types consists of a list (= array) which has a value of a simple data type in each line, a *date* in our example. The ticket field of type *date* has to have the parameter *Belongs to* which points to the list.

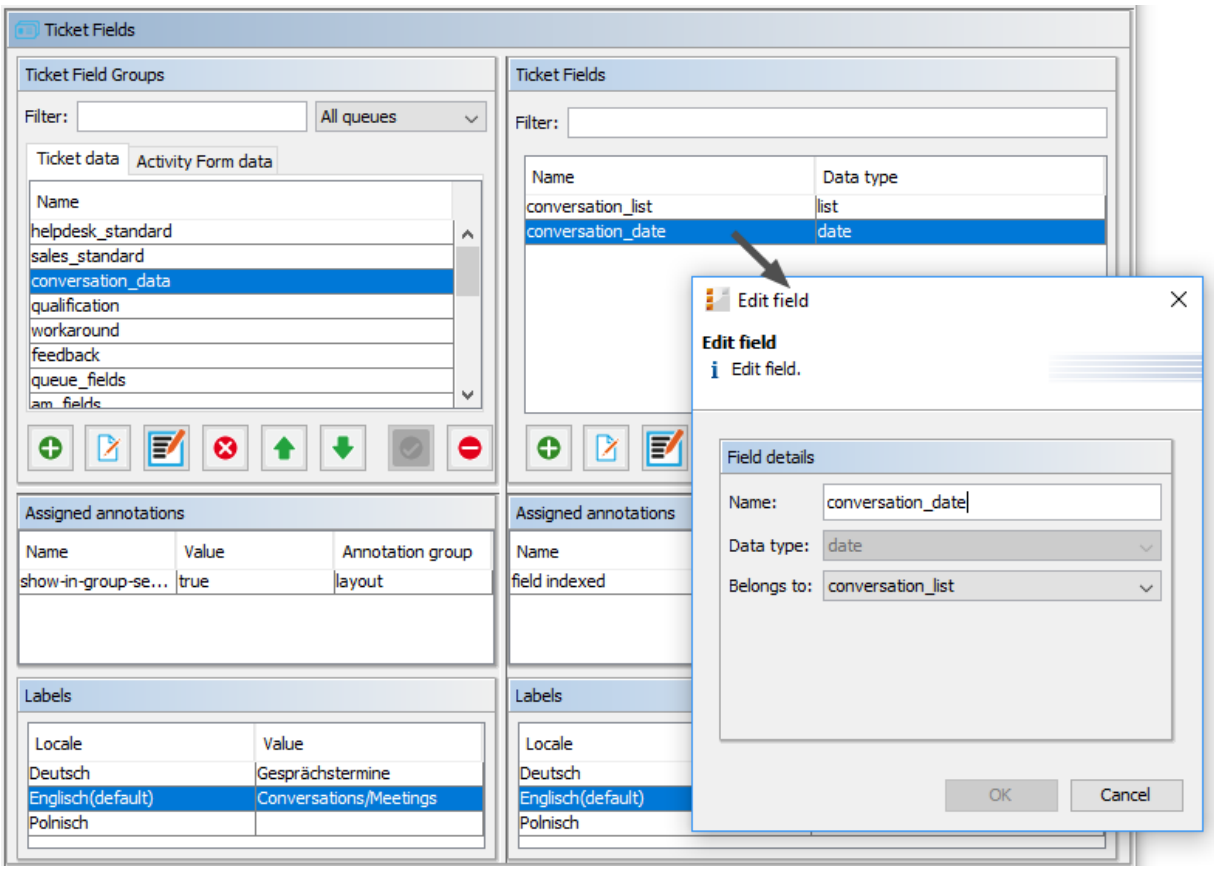


Figure 140: *ConSol CM Admin Tool - Ticket fields for a list of date fields*

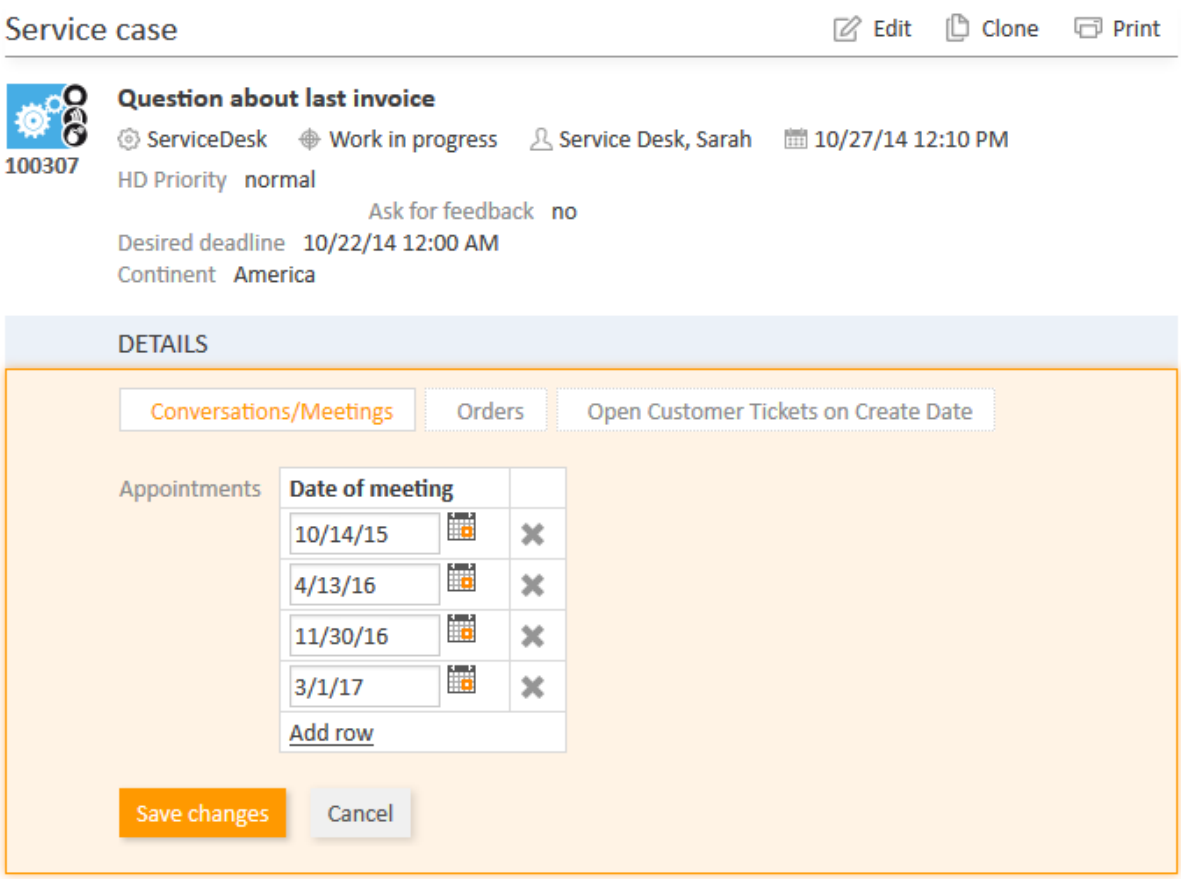


Figure 141: *ConSol CM Web Client - List of date fields in a ticket (edit mode)*

For access to each *date* ticket field within a list use the following lines of code:

```
def convs = ticket.get("conversation_data.conversation_list").each() { conv ->
   log.info "NEXT DATE is :" + conv
   log.info "CLASS of NEXT DATE is " + conv.getClass()
}
```

Code example 35: *Displaying the content of a list of date objects*

```
2014-03-27 12:21:45,274 INFO  [STDOUT                          ] NEXT DATE is :2014-02-03 00:00:00.0
2014-03-27 12:21:45,279 INFO  [STDOUT                          ] CLASS of MEXT DATE is class java.sql.Timestamp
2014-03-27 12:21:45,280 INFO  [STDOUT                          ] NEXT DATE is :2014-03-02 00:00:00.0
2014-03-27 12:21:45,280 INFO  [STDOUT                          ] CLASS of MEXT DATE is class java.sql.Timestamp
2014-03-27 12:21:45,281 INFO  [STDOUT                          ] NEXT DATE is :2014-03-26 00:00:00.0
2014-03-27 12:21:45,281 INFO  [STDOUT                          ] CLASS of MEXT DATE is class java.sql.Timestamp
```

Figure 142: *Log output of the script above*

To access a certain line, you can use the following syntax:

```
def mydate = ticket.get("conversation_data.conversation_list[1]")
```

Code example 36: *Retrieve a certain value from a list of simple data types*

Lists of structs (tables)

The data construct *list of structs* is the technical basis for a multi-column table structure in the Web Client. The list is the parent object which contains lines. Each line is an instance of a struct. Each line (struct) contains as many data fields (table columns) as required. Please see the figures in the introductory part of this section.

To retrieve the data from a list of structs you can work with an iteration over the lines (= structs). In the following example (from an order system, not displayed in the figure above) we work with a table where ...

- the ticket field *orders_list* represents the list.
- the ticket field *orders_list* is located within the ticket field group *order_data.*
- the iterator *str* represents the struct.
- the struct has three fields:
  - *orders_hardware*
    which represents the article that should be ordered (*enum*).
  - *orders_contact*
    which represents the contact person (*string*).
  - *orders_number*
    which represents the number of articles that should be ordered (*integer*).

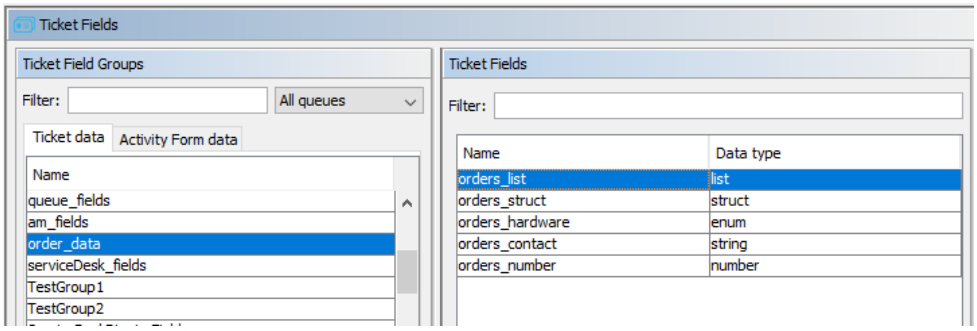


Figure 143: *ConSol CM Admin Tool - ticket fields for a multi-column list (= table)*

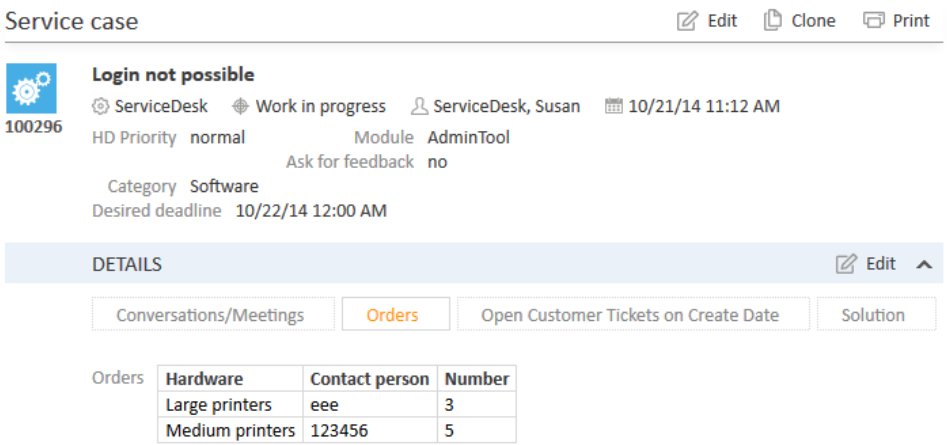


Figure 144: *ConSol CM Web Client - Ticket with filled-in table*

```
def structs = ticket.get("order_data.orders_list").each() { str ->
   log.info("CLASS of LINE is " + str.getClass())
   log.info("FIELD VALUE HARDWARE is " + str.orders_hardware.getName())
   log.info("CLASS of FIELD VALUE HARDWARE is " + str.orders_hardware.getName
    ().getClass())
   log.info("FIELD VALUE CONTACT is " + str.orders_contact)
   log.info("CLASS of FIELD VALUE CONTACT is " + str.orders_contact.getClass())
   log.info("FIELD VALUE NUMBER is " + str.orders_number)
   log.info("CLASS of FIELD VALUE NUMBER is " + str.orders_number.getClass())
}
```

Code example 37: *Retrieve data from a list of structs*

```
2014-03-27 11:39:44,425 INFO  [STDOUT                    ] CLASS of LINE is class com.consol.cmas.common.model.customfield.cfel.Struct
2014-03-27 11:39:44,429 INFO  [STDOUT                    ] FIELD VALUE HARDWARE is large_printers
2014-03-27 11:39:44,429 INFO  [STDOUT                    ] CLASS of FIELD VALUE HARDWARE is class java.lang.String
2014-03-27 11:39:44,430 INFO  [STDOUT                    ] FIELD VALUE CONTACTis Mr. Miller
2014-03-27 11:39:44,431 INFO  [STDOUT                    ] CLASS of FIELD VALUE CONTACT is class java.lang.String
2014-03-27 11:39:44,431 INFO  [STDOUT                    ] FIELD VALUE NUMBER is 2
2014-03-27 11:39:44,454 INFO  [STDOUT                    ] CLASS of FIELD VALUE NUMBER is class java.lang.Long
2014-03-27 11:39:44,455 INFO  [STDOUT                    ] CLASS of LINE is class com.consol.cmas.common.model.customfield.cfel.Struct
2014-03-27 11:39:44,456 INFO  [STDOUT                    ] FIELD VALUE HARDWARE is medium_printers
2014-03-27 11:39:44,456 INFO  [STDOUT                    ] CLASS of FIELD VALUE HARDWARE is class java.lang.String
2014-03-27 11:39:44,456 INFO  [STDOUT                    ] FIELD VALUE CONTACTis Mrs. Summer
2014-03-27 11:39:44,457 INFO  [STDOUT                    ] CLASS of FIELD VALUE CONTACT is class java.lang.String
2014-03-27 11:39:44,458 INFO  [STDOUT                    ] FIELD VALUE NUMBER is 5
2014-03-27 11:39:44,458 INFO  [STDOUT                    ] CLASS of FIELD VALUE NUMBER is class java.lang.Long
```

Figure 145: *Log File - Script output*

D.8.4.3 Setting ticket field values

To set values for ticket fields, you follow the same principle as for getting data: use the ticket field group name and the technical name of the ticket field as a reference. Of course, additionally, the new value is required. And of course it has to be of the correct data type.

```
ticket.set("<ticketFieldGroupName>.<ticketFieldName>", <value>)
```

Setting values for ticket fields with simple data types

```
ticket.set("fields.reaction_time", new Date());
```

Code example 38: *Set a ticket field value for a field of type Date*

When you work with NUMBER or DATE fields, you can even calculate with the ticket field values in a very comfortable way, see following example.

```
//add 24 hours (in millis) to current field value
ticket.add("fields.deadline", 24*60*60*1000)
```

Code example 39: *Calculate with the value of a ticket field of type date*

Setting a value to “null” (i.e. emptying the field) is the same as removing the value:

```
ticket.set("fields.numberOfEmployees", null)
```

Code example 40: *Setting a ticket field value to null*

Or shorter:

```
ticket.remove("fields.numberOfEmployees" )
```

Code example 41: *Setting a ticket field value to null via removing the value*

Setting enum values

To set an ENUM value use the following syntax. Of course, the new value has to be present in the ordered list (enum) which is referenced by the ticket field.

```
ticket.set("ticketFieldGroupName.ticketFieldName",<technical name of value>)
```

```
ticket.set("fields.priority", "URGENT");
```

Code example 42: *Setting an enum value*

Setting list values

Setting values in lists of simple data types

When you want to add a line, you can simply use the `add` method:

```
ticket.add("fields.tags", "my new String")
```

Code example 43: *Adding a new line in a list of strings*

When you want to refer to a certain value to set a new value for it, you have to use the syntax for an array:

```
ticket.set("fields.tags[last]", "consol cm6")
```

Code example 44: *Setting a value in a list of strings*

Setting values in lists of structs

Working with STRUCTS, you always have to work with the key of the value you would like to add or set. When you want to **add** a new line, you have to build a new struct as new line. The `set` method can be used one after another for each new field.

```
ticket.add("order_data.orders_list", new Struct()
.set("tA_Id", id).set("orders_hardware",mynewhardware_model)
.set("orders_contact", thenewcontactname)
.set("orders_number",thenewnumber)
```

Code example 45: *Adding a new line in a list of structs*

D.8.4.4 Fading in and out of ticket field groups

A ticket field group can be faded in (made visible) and faded out (made invisible) using a `workflowApi` method. This works for ticket field groups which are displayed in the main ticket data section as well as for ticket field groups which are displayed in the tabbed (*Details*) section.

A typical use case is a ticket field group which is invisible at first (ticket field group annotation `group-visibility` = “false”) and is faded in when the engineer needs to work with the data in the process. For example, a ticket field group which contains reasons for the dismissal of a request is only displayed (faded in) when the engineer has used the workflow activity *Dismiss ticket ...*. This prevents an information overload of the ticket.

```
workflowApi.setGroupProperty("CF_Group_Dismissal",GroupPropertyType.VISIBLE,"true")
```

Code example 46: *Fade in a ticket field group*

To fade out some ticket field groups, e.g. when the ticket has been qualified and some of the ticket field groups will no longer be required in the process, use code according to the following example:

```
workflowApi.setGroupProperty("CF_Group_
 HardwareInfo",GroupPropertyType.VISIBLE,"false")
```

```
workflowApi.setGroupProperty("CF_Group_
 SoftwareInfo",GroupPropertyType.VISIBLE,"false")
```

Code example 47: *Fade out ticket field groups*

D.8.5 Data fields for customer data: customer fields

The customer data model (FlexCDM) contains customer field groups and customer field. They are both defined in the Admin Tool, navigation group *Customers*, navigation item *Data Models*.

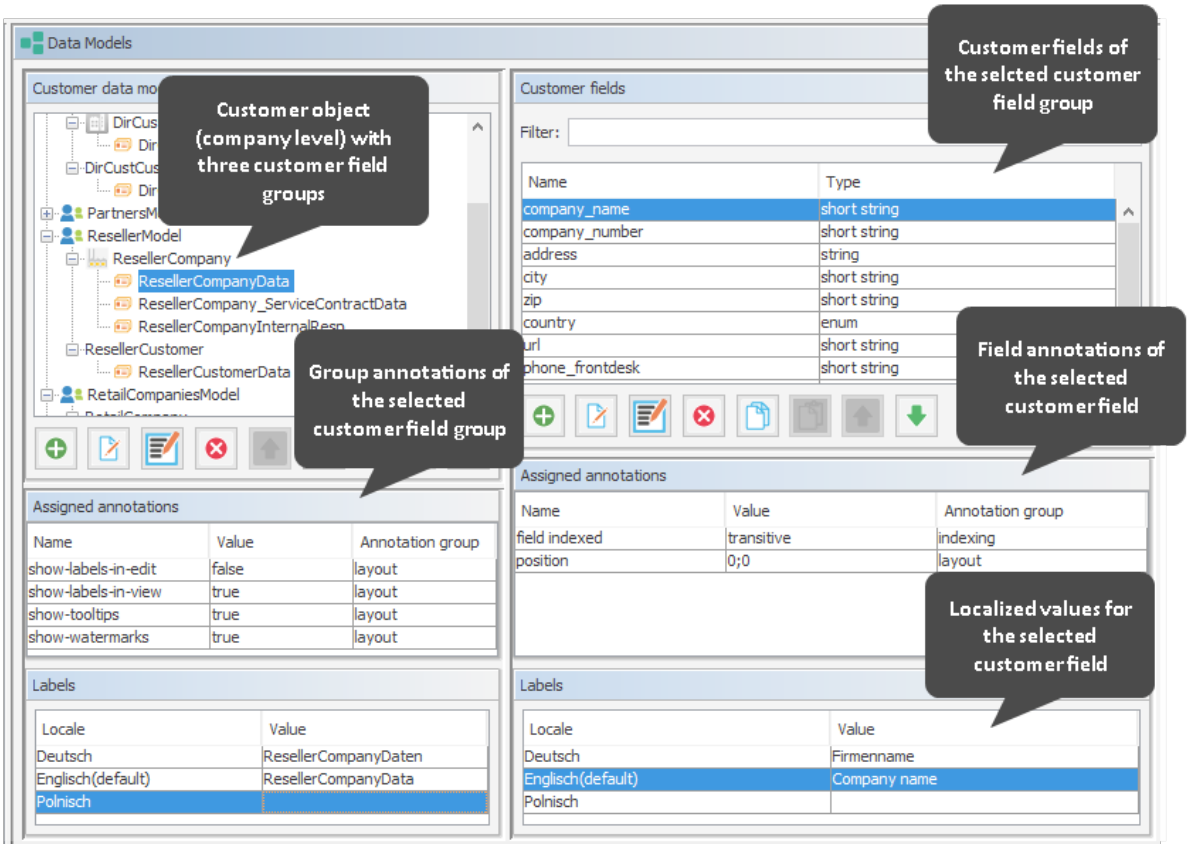


Figure 146: *ConSol CM Admin Tool - Customer field administration for customer data*

D.8.6 Most important methods for accessing customer data

Three methods are of major importance for programming access to customer fields in ConSol CM scripts. They all are methods of the class `Unit`.

For retrieving data from a customer field:

```
Unit.get()
```

For setting data in an already existing customer field:

```
Unit.set()
```

For calculating with a value within a customer field, e.g., to add a certain time range to a date field and for adding a new line in list fields (simple lists and tables).

```
Unit.add()
```

Another method might be used when a field should be emptied, i.e. when its value should be set to null:

```
Unit.remove() // Sets the value of the field to null.
```

D.8.6.1 Retrieving values for customer data

Because the name of a customer field might appear in more than one customer field group, the name of the customer field group has to be provided when accessing the customer data. For example, in the customer data model shown in the figure above, the customer field groups *ResellerCompanyData* and *DirCustCompanyData* could have a customer field named *city*. Therefore, it is important to mention group name and field name.

Please use the following syntax:

```
unit.get("group1:name")
```

For example:

```
def mycity = company.get("ResellerCompanyData:city")
```

Code example 48: *Retrieving a field value for a company*

Methods to access the values of customer fields:

- getField(String pGroupName, String pFieldName)
- getFieldValue(String pGroupName, String pFieldName)
- setFieldValue(String pGroupName, String pFieldName, Object pValue)
- removeField(String pGroupName, String pFieldName)

There are various objects and methods to work with data on different levels of the FlexCDM. Please see the following example where several common objects and methods have been applied. It is an Admin Tool script which is accessed from a workflow activity. The only purpose is to display some data of the ticket's main customer. The following figure shows the Java objects used in the script and the ConSol CM objects in the Admin Tool which are referenced.

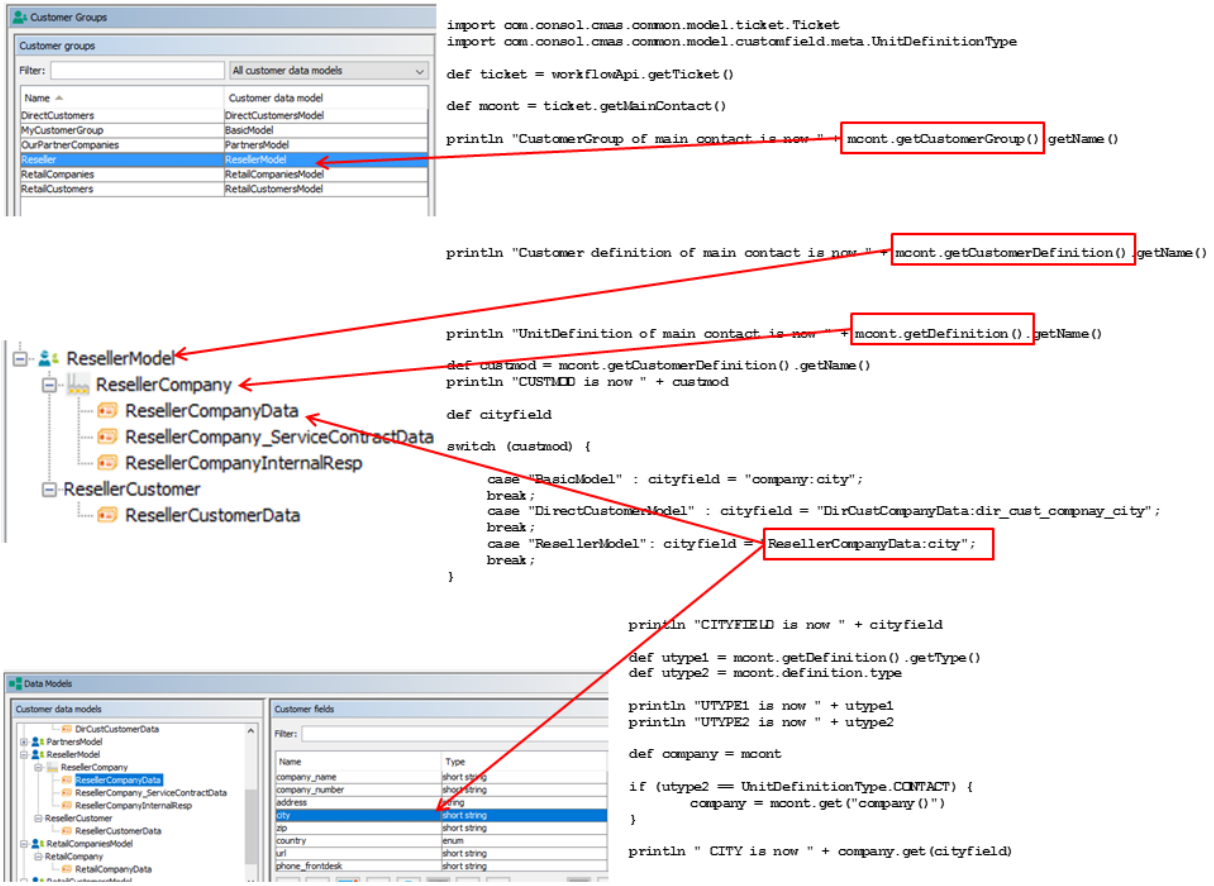


Figure 147: *ConSol CM customer objects in script and Admin Tool*

> Please keep in mind that you might also use the short notation like `unit.definition.type` for getter methods like `unit.getDefinition().getType()`.

```
import com.consol.cmas.common.model.ticket.Ticket
import com.consol.cmas.common.model.customfield.meta.UnitDefinitionType
def ticket = workflowApi.ticket
def mcont = ticket.mainContact
log.info "CustomerGroup of main contact is now " + mcont.customerGroup.name
def custmod = mcont.customerDefinition.name
log.info "Customer definition of main contact is now " + custmod
log.info "UnitDefinition of main contact is now " + mcont.definition.name

def cityfield

switch (custmod) {
   case "BasicModel" : cityfield = "company:city";
   break;
   case "DirectCustomerModel" : cityfield = "DirCustCompanyData:dir_cust_company_
    city";
   break;
   case "ResellerModel": cityfield = "ResellerCompanyData:city";
   break;
   default: cityfield = null;
   break
}

log.info "CITYFIELD is now " + cityfield

def utype1 = mcont.getDefinition().getType()
def utype2 = mcont.definition.type

log.info "UTYPE1 is now " + utype1
log.info "UTYPE2 is now " + utype2

def company = mcont
if (utype2 == UnitDefinitionType.CONTACT) {
   company = mcont.get("company()")
}

def mycity = company.get(cityfield)
log.info " CITY is now " + mycity
```

Code example 49: *Admin Tool script (called from workflow) for displaying customer data*

For the following data set the log file output is shown below. The *Reseller* model of the figure above is used.

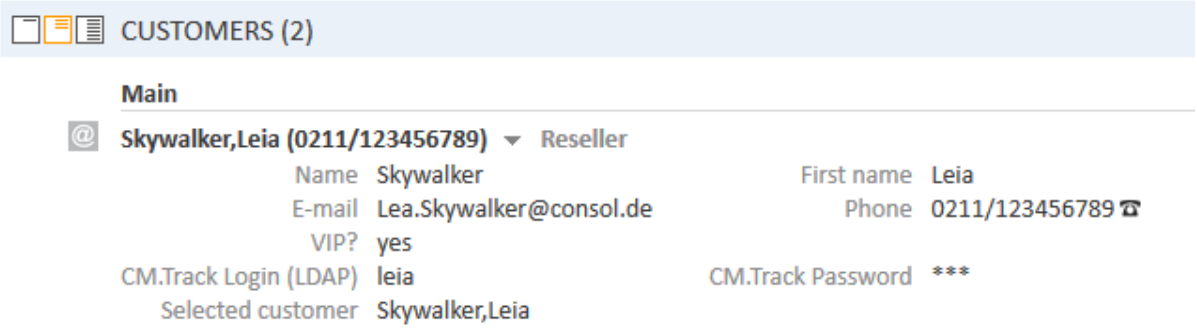


Figure 148: *ConSol CM Web Client - Customer data*

```
2017-03-01 11:00:24,689 INFO  [ase_DisplayCustomerData.groovy] [Susan-] CustomerGroup of main contact is now Reseller
2017-03-01 11:00:24,690 INFO  [ase_DisplayCustomerData.groovy] [Susan-] Customer definition of main contact is now ResellerModel
2017-03-01 11:00:24,692 INFO  [ase_DisplayCustomerData.groovy] [Susan-] UnitDefinition of main contact is now ResellerCustomer
2017-03-01 11:00:24,692 INFO  [ase_DisplayCustomerData.groovy] [Susan-] CITYFIELD is now ResellerCompanyData:city
2017-03-01 11:00:24,695 INFO  [ase_DisplayCustomerData.groovy] [Susan-] UTYPE1 is now CONTACT
2017-03-01 11:00:24,695 INFO  [ase_DisplayCustomerData.groovy] [Susan-] UTYPE2 is now CONTACT
2017-03-01 11:00:24,714 INFO  [ase_DisplayCustomerData.groovy] [Susan-]  CITY is now null
```

Figure 149: *Log file - Script output FlexCDM*

D.8.6.2 Setting values for customer fields

Setting values for customer fields with simple data types

The set and add methods work as described for ticket fields. For example:

```
//set a string field
unit.set("ResellerCompanyData:service_status","ok")
//set number field
company.set("ResellerCompanyData:numberOfEmployees", 1);
//add 1 to field value, afterwards the value of the field is 2
company.add("numberOfEmployees", 1);
```

Code example 50: *Set and add values for a customer field of type integer and set a value for a customer field of type string*

Lists

Setting values in a list of structs for customer data

```
company.set("consultantsFields:responsibleConsultants", [
     new Struct().set("lastName", "Miller").set("email", "miller@consol.com"),
     new Struct().set("lastName", "Smith").set("email", "smith@consol.com"),
     new Struct().set("lastName", "Burger").set("email", "burger@consol.com")
]);
```

Code example 51: *Creating a new list of structs, version 2*

```
company.add("consultantsFields:responsibleConsultants", new Struct().set
 ("lastName", " Nowitzki ").set("email", "dnowitzki@consol.us"));
```

Code example 52: *Adding a new line in a list of structs for company data*

```
company.set("consultantsFields:responsibleConsultants[0].firstName", "John");
```

Code example 53: *Setting a value in a list of structs using index notation*

```
company.set("consultantsFields:responsibleConsultants[last]", null);
```

Code example 54: *Removing a struct (= line) from a list of structs (= table)*

D.8.6.3 Convenience methods for accessing customer data

```
Unit mainContact = ticket.mainContact

// "company" extension returns company for contact
Unit company = mainContact.get("company()");

// it is also possible to set company using "company" extension
mainContact.set("company()", company);

// "contacts" extension returns list of contacts for company
List contacts = company.get("contacts()");

// "tickets" extension returns list of tickets for contact or company
List tickets = company.get("tickets()");
tickets = mainContact.get("tickets()");

// extensions can be chained
Integer count = contact.get("company().contacts()[0].tickets()[count]");

// parentheses can be omitted, but it is not recommended (possible collision with
 name of group or field)
count = contact.get("company.contacts[0].tickets[count]"); // here "company" is not
 extension but name of field
```

Code example 55: *Convenience methods for access to customer data*

D.8.7 Resource data

The optional module ***CM/Resource Pool*** allows it to extend the CM database and store resource objects. This can be various objects like IT assets, contracts, shop items or whatever is required in the respective company. Similar to the customer data model, the resource data model is defined using the Admin Tool. In the navigation group *Resources*, navigation item *Data Models*, the model is defined. In order to be able to work with resource data in workflow scripts, you should first get a profound knowledge of the Resource Pool principle and the setup of the data model, so please read the section about the CM/Resource Pool in the *ConSol CM Administrator Manual* first.

Similar to ticket and customer data, resource data is stored in resource field groups which contain resource fields.

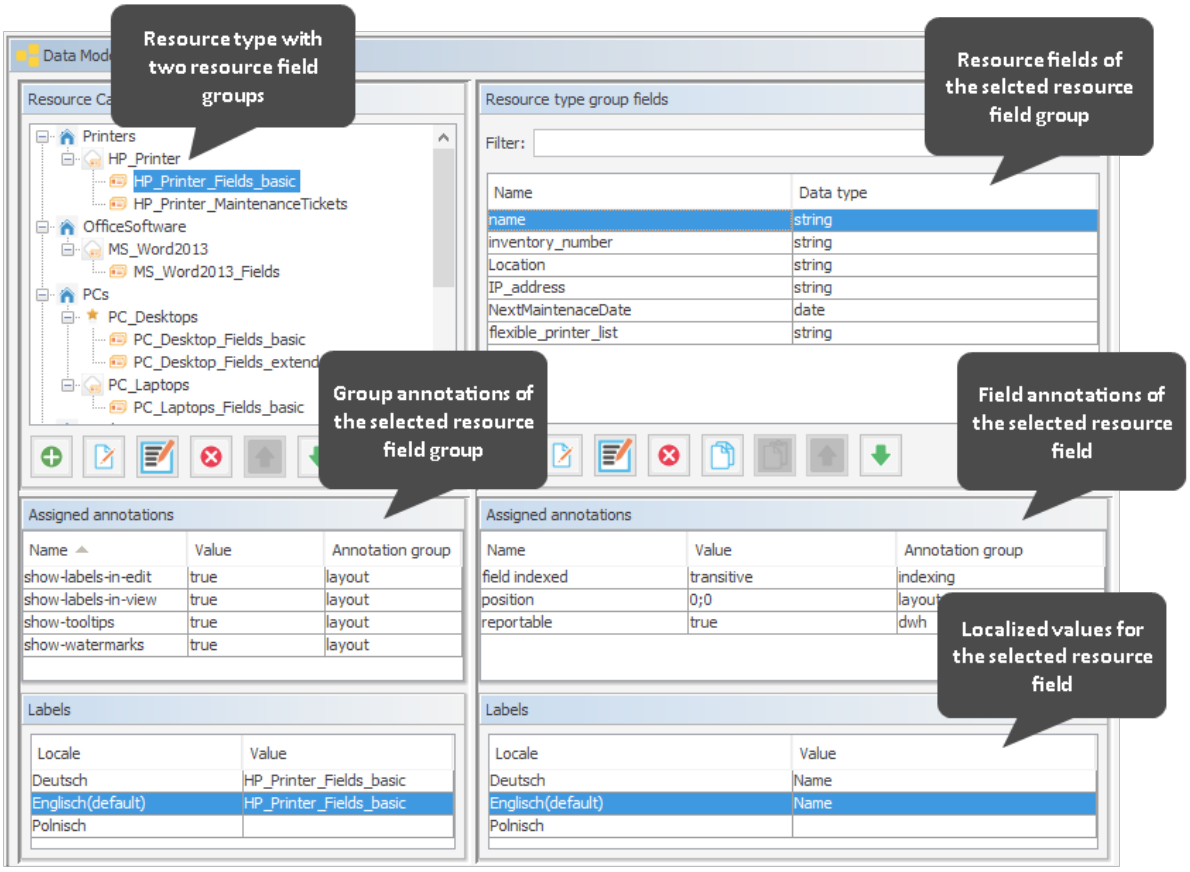


Figure 150: *ConSol CM Admin Tool - Definition of the resource data model*

To retrieve data from resource fields, you can use methods as the ones shown in the following examples.

```
// Display info about HP Printer, printing infos about resource fields into
 server.log

def my_resource_name = resource.get("HP_Printer_Fields_basic.name")
log.info 'Displaying resource information ...'
log.info ' Name is : ' + my_resource_name

def my_resource_inv_number = resource.get("HP_Printer_Fields_basic.inventory_
 number")
log.info ' Inventory number is : ' + my_resource_inv_number
```

Code example 56: *Simple resource action script which displays information about the resource in the log file*

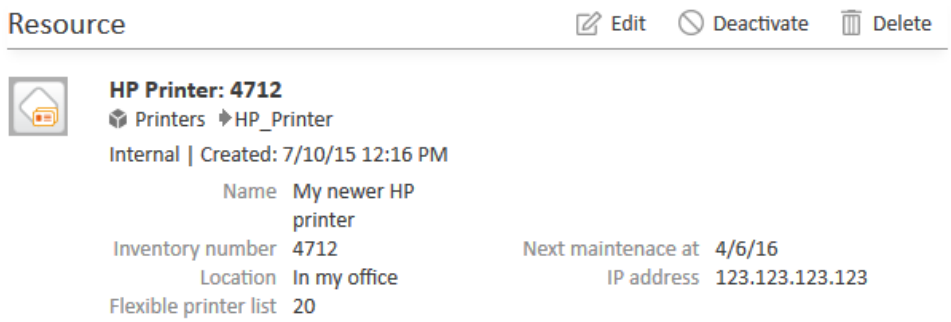


Figure 151: *ConSol CM Web Client - Resource page of a HP printer resource*

```
2017-03-01 12:04:12,532 INFO  [layHpPrinterInfos1488366252515] [Susan-] Displaying resource information ...
2017-03-01 12:04:12,532 INFO  [layHpPrinterInfos1488366252515] [Susan-]  Name is : My newer HP printer
2017-03-01 12:04:12,532 INFO  [layHpPrinterInfos1488366252515] [Susan-]  Inventory number is : 4712
```

Figure 152: *Log output of the script for the printer shown in the GUI example*

In the real world, the circumstances usually are more complex. Often it is required to retrieve all resources of a certain type which are linked by a certain type of relation to a ticket or to a resource. This topic is treated in detail in the section Working with resource relations.

More coding examples, taken from the ConSol CM Action Framework, are provided in the ConSol *CM Administrator Manual*, section *Scripts for the Action Framework*.

D.8.7.1 Fading in and out resource field groups

A resource field group can be faded in (made visible) and faded out (made invisible) using a `workflowApi` method. This works for resource field groups which are displayed in the main resource page data section as well as for resource field groups which are displayed in the tabbed (*Details*) section.

In the following example, the resource *HP_Printers* contains a resource field group with a list of tickets. Each maintenance ticket which is created for the resource using a resource action (see *ConSol CM Administrator Manual* for a detailed explanation of resource actions) should be entered in the list. Initially, the resource field group is configured as invisible (group annotation `group-visibility` = “false”). Only when the action has been performed, the resource field group should be made visible on the resource page.

The code which is shown is an excerpt taken from the resource action script, but the same or similar code can be used in workflow scripts.

```
// set ticket number in list in resource, newtic is the newly created maintenance
 ticket
def newtic_id = newtic.id.toString()
def newtic_name = newtic.name

//resource is the HP printer from which the action script is started
resource.add("HP_Printer_MaintenanceTickets.MaintenanceTicketsList",new Struct
 ().set("MaintenanceTicketID", newtic_id)
          .set("MaintenanceTicketName", newtic_name)
)
if (!executionContext) {
   return actionScriptResultFactory.getPostAction(PostActionType.FAILURE,
    "action.fail.wrong.activity")
}
// if ticktes are in the list in the resource, the field group should be visible
def groupName = "HP_Printer_MaintenanceTickets"
def fieldGroupDefinition = fieldDefinitionService.getGroupByName(groupName)
if (fieldGroupDefinition == null) {
   throw new IllegalArgumentException("There is no group definition with name '" +
    groupName + "'.")
}
resource.getGroupsConfiguration().setProperty(fieldGroupDefinition,
 GroupPropertyType.VISIBLE, "true")
```

Code example 57: *Excerpt from a script which adds a line to a list of tickets on a resource page and makes the resource field group which contains the list visible*

D.8.8 Using data fields for (invisible) variables

Sometimes it is necessary to work with variables which are not used as values for ticket fields, customer fields or resource fields, which are not visible on the GUI, but which are only used as containers for internal programming variables.

The values of these data fields represent **global variables**. You implement global variables by creating regular ticket fields, customer fields or resource fields with the required data type and setting the field to *invisible*. This has to be done by using the annotation `visibility` = “none”. You can even let the variable be visible during the development of the process and control the field's value. Then you can set it to invisible when the system is handed over to QA and users.

D.9 Sending emails

This chapter discusses the following:

D.9.1 Introduction to sending emails 203

D.9.2 Important methods 204

D.9.3 Examples 206

D.9.4 Effects of engineer representation rules for emails sent in scripts 216

D.9.1 Introduction to sending emails

The capability of receiving and sending emails is a core feature of ConSol CM. Please read the detailed introduction in the *ConSol CM Administrator Manual* for information.

In this section we will describe how you can write scripts to send emails from the workflow. This is very useful for use cases like the following:

- You want to send an automatic acknowledgment of receipt to the customer when he/she has opened a ticket.
- You want to inform the engineer and his supervisor when the highest escalation level has been reached.
- You want to inform the customer that a problem has been solved (and how).

Usually, you do not write the text of the email into the script but you work with email templates. So please read the detailed introduction to the *ConSol CM Text Template Manager* in the *ConSol CM Administrator Manual* first.

D.9.2 Important methods

There are different ways of sending emails from ConSol CM workflows. The following list provides an overview of the classes and methods which can be applied and discusses the advantages and disadvantages of each variant. However, please see for yourself, which variant will cover your specific use case the best way.

- Variant 1: Using an object of the class mail
- Variant 2: Working with workflowApi (singleton of class WorkflowContextService)
- Variant 3: Using mailService (singleton of the class MailService)

> Work with the ConSol CM API documentation in order to gain an overview of all methods which are provided by the respective classes!

D.9.2.1 Variant 1: Using an object of the class mail

Here you can define all required parameters for an email (see Example #1 up to Example #3 in the Examples section).

Advantages:

- You can configure the `Mail` object to use the queue-specific email default script ("outgoing mail script"). This is a script which processes the email before it leaves the CM system. This kind of script can be assigned to a queue (*Email script*, see section *Queue Administration* in the *ConSol CM Administrator Manual*). To use such a script can prove helpful, for example when you want to set a `Reply-To` address which is not the standard `Reply-To` address (stored in a system property).

Disadvantages:

- none

Information:

- In case representation rules are in operation: the email is not sent to the representing engineers automatically.
  - Use `Mail setTargetEngineers(List<Engineer> pTargetEngineers)` to send the email to the specified engineers
  - Use `setHandleRepresentatives(boolean pHandleRepresentatives)` to send the email to the engineer and all his representatives.
- You will not find the method `send()` in the API documentation of the class `Mail`! This is because the method is not a method of the `Mail` class, but it is added by a helper class "behind the scenes".

D.9.2.2 Variant 2: Working with workflowApi (singleton of class WorkflowContextService)

The object workflowAPI provides the method `sendEmail()` with several different signatures. All those methods return `void`.

```
sendEmail(Engineer pEngineer, String pSubject, String pText, String pReplyAddr,
List<AttachmentEntry> pTicketAttachments, List<DataSource> pExternalAttachments)
```

Sends an e-mail to the given engineer with the given subject, the given body and the given reply-to address with attachments Email is also send to engineer's representative.

```
sendEmail(List<Engineer> pEngineers, String pSubject, String pText, String pReplyAddr,
List<AttachmentEntry> pTicketAttachments, List<DataSource> pExternalAttachments)
```

Sends an e-mail to the given engineers with the given subject, the given body and the given reply-to address with attachments.

```
sendEmail(Mail pMail)
```

Sends email based on provided mail specification.

```
sendEmail(String pEmailaddr, String pSubject, String pText, String pReplyAddr,
List<AttachmentEntry> pTicketAttachments)
```

Sends an e-mail to the given recipient with the given subject, the given body and the given reply-to address with attachments

```
sendEmail(String pEmailaddr, String pSubject, String pText, String pReplyAddr,
List<AttachmentEntry> pTicketAttachments, List<DataSource> pExternalAttachments)
```

Sends an e-mail to the given recipient with the given subject, the given body and the given reply-to address with attachments

Figure 153: *Methods sendEmail() in workflowApi (WorkflowContextServcice)*

Advantages:

- There are methods where a list of recipients can be provided directly as method parameter

Disadvantages:

- Can only be used in workflow context (i.e. not in every Admin Tool script, e.g. not in TEF scripts)
- Does not apply the queue-specific email default script ("outgoing mail script"), except for the method which takes a `Mail` object as parameter: `sendEmail(Mail pMail)`

Information:

> We recommend to use the rather new `Mail` object instead of working with `workflowApi` for sending emails unless you have a very special reason to use `workflowApi`.

D.9.2.3 Variant 3: Using mailService (singleton of the class MailService)

You can also work with the `mailService` to send emails from a workflow (or an Admin Tool script which is called by a workflow script).

The following methods are available, they all return `void`.

`sendMail(File pMessageContent, boolean pEncrypt, String... pFrom)` Sends message constructed from file that contains an e-mail.
`sendMail(File pMessageContent, String... pFrom)` Sends unencrypted message constructed from file that contains an e-mail.
`sendMail(MimeMessagePreparator pMessagePreparator)` Sends an unencrypted mail.
`sendMail(MimeMessagePreparator pMessagePreparator, boolean pEncrypt)` Sends a mail.
`sendMail(String[] pTo, String pFrom, String pSubject, String pContent)` Creates and sends a simple, unencrypted mail message.
`sendMail(String[] pTo, String pFrom, String pSubject, String pContent, boolean pEncrypt)` Creates and sends a simple mail message.
`sendMail(String pTo, String pFrom, String pSubject, String pContent)` Creates and sends a simple, unencrypted mail message.
`sendMail(String pTo, String pFrom, String pSubject, String pContent, boolean pEncrypt)` Creates and sends a simple mail message.
`sendMailAsynchronous(MailSendHolder pMailMessageHolder)` Send mail asynchronous.

Figure 154: *Methods of the class MailService, they all return void*

Advantages:

- You can send emails in an asynchronous way, this requires to work with a `MailSendHolder` object. See Example 8: Using MailService to send an email in an asynchronous mode.
- Provides a rather high-performance way of sending emails, because the `mailService` is the target for other methods which send emails.
- Can be called from any context within ConSol CM.

Disadvantages:

- Does not offer convenient ways of sending attachments or addressing a list of recipients (you have to implement the loop yourself for the latter case)

D.9.3 Examples

D.9.3.1 Example 1: Sending an automatic acknowledgment of receipt to the customer who created the ticket

This script might be placed in one of the first activities of the workflow.

```
// create new mail object
def mail = new Mail()

// fetch main contact of the ticket
def maincontact = ticket.getMainContact()

// fetch e-mail address of the main contact. The Data Object Group Field has to be
 addressed using Data Object Group name:Data Object Group Field name
def toaddress = maincontact.get("MyCustomerDataObjectGroup:email")

// put the e-mail TO address into the Mail object
mail.setTo(toaddress)

// fetch the REPLY TO address, this is stored in a system property
def replyaddress = configurationService.getValue("cmweb-server-
 adapter","mail.reply.to")

// put the e-mail REPLY TO address into the Mail object
mail.setReplyTo(replyaddress)

// build e-mail text using a template which is stored in the Template Designer
def text = workflowApi.renderTemplate("Acknowledgement_of_receipt")

// put the e-mail text into the Mail object
mail.setText(text)

// create the subject of the e-mail, the ticket number with the correct Regular
 Expression has to be set for correct recognition of incoming e-mails for the
 ticket
def ticketname = ticket.getName()
def subject = "Your case has been registered as Ticket (" + ticketname + ")"

// put the subject into the Mail object
mail.setSubject(subject)

// send out the e-mail
mail.send()
```

Code example 58: *Sending an automatic acknowledgment of receipt to the customer who has opened a ticket, using a Mail object*

D.9.3.2 Example 2: Sending an email to the engineer when an escalation level has been reached

This script might be placed in an automatic activity which is connected to a time trigger. The time trigger measures the escalation interval. When the deadline has been reached, the trigger fires and the ticket enters the automatic activity.

```
// create new mail object
def mail = new Mail()

// fetch current engineer of the ticket and set it as e-mail receiver
if (ticket.engineer){
   mail.setTargetEngineer(ticket.engineer)

   // fetch the REPLY TO address, this is stored in a system property
   def replyaddress = configurationService.getValue("cmweb-server-
    adapter","mail.reply.to")

   // put the e-mail REPLY TO address into the Mail object
   mail.setReplyTo(replyaddress)

   // build e-mail text using a template which is stored in the Template Designer
   def text = workflowApi.renderTemplate("ESCALATION_Mail")

   // put the e-mail text into the Mail object
   mail.setText(text)

   // create the subject of the e-mail, the ticket number with the correct Regular
    Expression has to be set for correct recognition of incoming e-mails for the
    ticket
   def ticketname = ticket.getName()
   def subject = "ESCALATION Level 3 REACHED! Ticket (" + ticket.getId() + ")"

   // put the subject into the Mail object
   mail.setSubject(subject)

   // send out the e-mail
   mail.send()
}
```

Code example 59: *Sending an email to the engineer when a certain escalation level has been reached, using a Mail object*

D.9.3.3 Example 3: Sending an email to a customer integrating the queue-specific email script

This is the same script as shown in the example above, but the queue-specific mail script will be used. For a detailed explanation of this type of script, refer to the *ConSol CM Administrator Manual*, section *Admin Tool Scripts*.

As an effect, the outgoing email will pass through the script before it leaves the CM system. Email parameters, like `Cc`, `Bcc`, or `Reply To` can be changed.

```
// create new mail object
def mail = new Mail()

// fetch main contact of the ticket
def maincontact = ticket.getMainContact()

// fetch e-mail address of the main contact. The Data Object Group Field has
// to be addressed using Data Object Group name:Data Object Group Field name
def toaddress = maincontact.get("MyCustomerDataObjectGroup:email")

// put the e-mail TO address into the Mail object
mail.setTo(toaddress)

// fetch the REPLY TO address, this is stored in a system property
def replyaddress = configurationService.getValue("cmweb-server-
 adapter","mail.reply.to")

// put the e-mail REPLY TO address into the Mail object
mail.setReplyTo(replyaddress)

// build e-mail text using a template which is stored in the Template Designer
def text = workflowApi.renderTemplate("Acknowledgement_of_receipt")

// put the e-mail text into the Mail object
mail.setText(text)

// create the subject of the e-mail, the ticket number with the correct Regular
 Expression
// has to be set for correct recognition of incoming e-mails for the ticket
def ticketname = ticket.getName()
def subject = "Your case has been registered as Ticket (" + ticketname + ")"

// put the subject into the Mail object
mail.setSubject(subject)

// Mail should use the e-mail script which is configured for the queue
mail.useDefaultScript()

// send out the e-mail
mail.send()
```

Code example 60: *Sending an email to a customer integrating the queue-specific mail script, using a Mail object*

D.9.3.4 Example 4: Sending an email to all contacts of the ticket

This will send one email with all customers (that have an email address) as receiver. Please note that this is a simple example which demonstrates the use of a list. The `Reply-To` address is not set, so answers to the email would not be appended to the ticket.

```
def custEmails = workflowApi.getContactList()*.get("email").findAll{it !=
 null}.join(",")
workflowApi.sendEmail(custEmails, "Confirmation", "Good afternoon, we received your
 request!", null, null)
```

Code example 61: *Sending an email to all contacts of the ticket*

D.9.3.5 Example 5: Sending an email to each contact in a list of all contacts of the ticket

This will send one email to every single customer (that has an email address). Please note that this is a simple example which demonstrates the use of a list. The `Reply-To` address is not set, so answers to the email would not be appended to the ticket.

```
workflowApi.getContactList().each {
   def custEmail = it.get("email")
   if (custEmail){
      workflowApi.sendEmail(custEmail, "Confirmation",
      "Good afternoon, we received your request!", null, null)
   }
}
```

D.9.3.6 Example 6: Sending an email and inserting it into the ticket history

The following code shows a script which is called to send a receipt notice to the customer when a tickets for this customer was opened. If the main customer of the ticket is a company, no email is sent, because in the example system, companies do not have email addresses. In the next step, the correct customer field which contains the email address has to be found. The field name can vary between customer groups (with the respective customer model), so this has to be considered. When the email has been sent, this is documented as ticket history entry with the correct entry type.

```
import com.consol.cmas.common.model.mail.Mail
import com.consol.cmas.common.model.content.MailEntryStatus;
import com.consol.cmas.common.model.content.AttachmentEntry;
import com.consol.cmas.common.model.content.ContentFile;
import com.consol.cmas.common.model.content.ContentEntryCategory
import com.consol.cmas.common.model.content.MailEntry;
import com.consol.cmas.common.util.MailHeadersUtil;
import com.consol.cmas.core.server.service.*;

// create new mail object
def mail = new Mail()
def ticket = workflowApi.getTicket()

// fetch main contatc of the ticket
def maincontact = ticket.getMainContact()
def unit_type = maincontact.definition.type.toString()

if(unit_type.equals('COMPANY')) {
   println 'No email address for company; no receipt notice sent.'
   return
```

```
} else if (unit_type.equals('CONTACT')){

   def toaddress_field
   def custgroup = maincontact.customerGroup.name
   println 'Customergroup is now ' + custgroup
   switch(custgroup) {
      case "Reseller": toaddress_field = "email";
      break;
      case "DirectCustomers": toaddress_field = "dir_cust_email"
      break;
      case "MyCustomerGroup": toaddress_field = "email"
      break;
      case "OurPartnerCompanies": toaddress_field = "email"
      break;
      case "RetailCustomers": toaddress_field = "retail_customer_email"
      break;
   }
   def toaddress = maincontact.get(toaddress_field)
   println 'toaddress is now ' + toaddress
   if (!toaddress){
      println 'No email address found for contact, no receipt notice sent.'
   } else {
      // put the e-mail TO address into the Mail object
      mail.setTo(toaddress)

      // fetch the REPLY TO address, theis is stored in a system property
      def replyaddress = configurationService.getValue("cmweb-server-
       adapter","mail.reply.to")
      // put the e-mail REPLY TO address into the Mail object
      mail.setReplyTo(replyaddress)

      // build e-mail text using a template which is stored in the Template Designer
      def text = workflowApi.renderTemplate("Acknowledgement_of_receipt")
      // put the e-mail text into the Mail object
      mail.setText(text)

      // create the subject of the e-mail, the ticket number with the correct
       Regular Expression
      // has to be set for correct recognition of incoming e-mails for the ticket
      def ticketname = ticket.getName()
      def subject = "Your case has been registered as Ticket (" + ticketname + ")"
      // put the subject into the Mail object
      mail.setSubject(subject)
      // Mail should use the e-mail script which is configured for the queue
      mail.useDefaultScript()
      // send out the e-mail and register status
      def attList = new ArrayList<AttachmentEntry>()
      def collection = new HashSet<MailEntry>()
      def mailStatus = true;

      // add attachments to attList .
      try {
         mail.send();
      } catch (Exception e){
```

```
         mailStatus = false;
      }

      MailEntry mailEntry = new MailEntry(subject,text);
      // you can also use an email template -> String text =
       workflowApi.renderTemplate
      mailEntry.setTicket(ticket);
      mailEntry.setCategory(ContentEntryCategory.OUTGOING_MAIL);
      // or other status -> see
       com.consol.cmas.common.model.content.ContentEntryCategory in API
      mailEntry.setCreationDate(new Date());
      mailEntry.setLastModificationDate(new Date());
      mailEntry.setMimeType("text/html"); // maybe "text/plain" shows linebreaks if
       "text/html" doesn't
      mailEntry.setEncoding("UTF-8");
      mailEntry.setAttribute(MailHeadersUtil.FROM_PROPERTY, replyaddress)
      mailEntry.setAttribute(MailHeadersUtil.TO_PROPERTY, toaddress)
      mailEntry.setAttribute(MailHeadersUtil.SUBJECT_PROPERTY,subject);

      if(mailStatus){
         mailEntry.setMailEntryStatus(MailEntryStatus.SUCCESS)
      } else {
         mailEntry.setMailEntryStatus(MailEntryStatus.FAILURE)
      }

      attList?.each { att ->
         mailEntry.addAttachment(att)
      }
      collection.add(mailEntry)
      workflowApi.updateTicket(ticket,collection)
   } // end if (!toaddress){
} // end of else if (unit_type.equals('COMPANY')){
```

Code example 62: *Admin Tool script for sending a notification of receipt and inserting the email as ticket history entry*

D.9.3.7 Example 7: Working with a template for the ticket name in the email subject

In the previous coding example, the following lines are used:

```
def ticketname = ticket.getName()
def subject = "Your case has been registered as Ticket (" + ticketname + ")"
```

This is because the pattern in the email subject guarantees that an incoming email can be assigned to the correct ticket. This means, if you use the coding style shown in the example above, you have to make sure that the pattern you use matches the pattern which has been configured in the Admin Tool, navigation item *Email* (compare following figure).

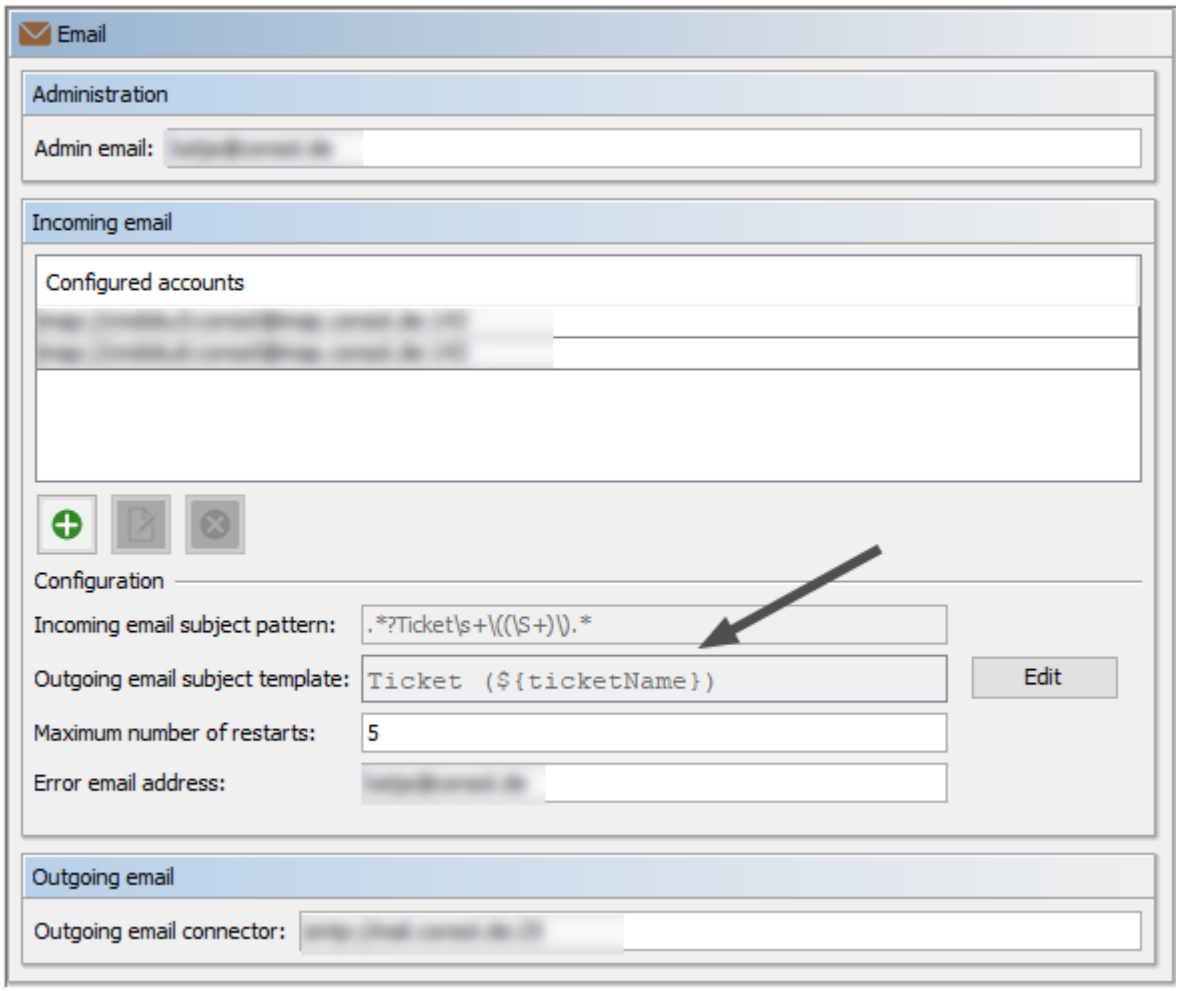


Figure 155: *Admin Tool: Pattern and template for the email subject*

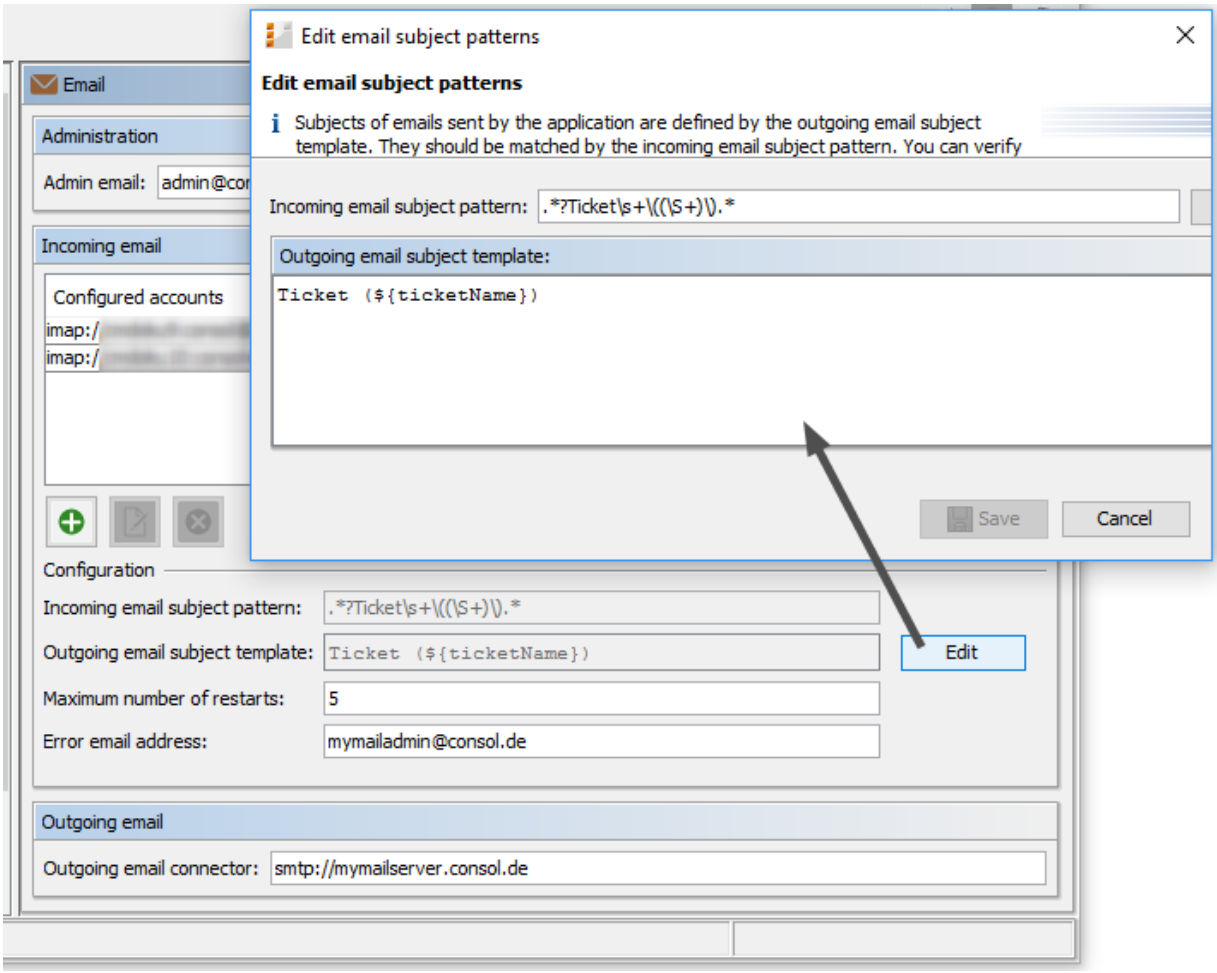
In order to avoid inconsistencies between workflow code and system configuration (Admin Tool), you can also work with the template name which is implicitly available in CM.

Use the following code in the Admin Tool script:

```
import static com.consol.cmas.common.util.TemplateUtil.TICKET_SUBJECT_TEMPLATE_NAME
( ... )
def subject = templateService.merge(TICKET_SUBJECT_TEMPLATE_NAME,
 [ticketName:ticket.name])
```

Note that you have to put a parameter called `ticketName` in the context for the template to work.

Alternative solution: if you modify the template for the outgoing emails to accept either `ticketName` or the `ticket` object (you can then work with `ticket.name`), you can also work with the regular `renderTemplate()` method as shown below.



The outgoing email subject template is then:

```
Ticket (<#if ticketName??>${ticketName}<#elseif ticket??>${ticket.name}</#if>)
```

and the subject is then:

```
def subject = workflowApi.renderTemplate(TICKET_SUBJECT_TEMPLATE_NAME)
```

D.9.3.8 Example 8: Using MailService to send an email in an asynchronous mode

Here, we show an example of sending a single email. However, the code might be used to improve performance if a great number of emails should be sent in a bulk operation. Then you can really benefit from the asynchronous mechanism. The sending of the emails will not block other processes which are triggered in the same script, for example.

```
import com.consol.cmas.common.model.mail.MailSendHolder

def text = workflowApi.renderTemplate("Acknowledgement_of_receipt")
def holder = MailSendHolder.createSelfSendHolder(text, true, null, null)

def ticket = workflowApi.ticket
def ticketname = ticket.name
```

```
def maincontact = ticket.mainContact
def toaddress_field
def custgroup = maincontact.customerGroup.name


switch(custgroup) {
   case "Reseller":toaddress_group = "ResellerCustomerData";
   toaddress_field = "email";
   break;
   case "DirectCustomers": toaddress_group = "DirCustCustomerData" ;
   toaddress_field = "dir_cust_email"
   break;
   case "MyCustomerGroup": toaddress_group = "customer";
   toaddress_field = "email"
   break;
   case "OurPartnerCompanies": toaddress_group = "PartnersContactData";
   toaddress_field = "email"
   break;
   case "RetailCustomers": toaddress_group = "RetailCustomersContactData";
   toaddress_field = "retail_customer_email"
   break;
}

def toaddressComplete = toaddress_group + ":" + toaddress_field

def toaddress = maincontact.get("${toaddressComplete}")

if (!toaddress){
   log.info 'No email address found for contact, no receipt notice sent.'
} else {
   holder.setTo(toaddress)
   def subject = "Your case has been registered as Ticket (" + ticketname + ")"
   holder.setSubject(subject)

   // fetch the REPLY TO address, this is stored in a system property
   def replyaddress = configurationService.getValue("cmweb-server-
    adapter","mail.reply.to")
   holder.setFrom(replyaddress)
   mailService.sendMailAsynchronous(holder)
   log.info 'Acknowledgement of receipt has been sent!'
 }
```

Code example 63: *Script for sending an email in an asynchronous mode, please see also alternative code in the following example*

```
log.info 'Customergroup is now ' + custgroup
def sourceGroupMap = [
   Reseller: [toaddress_group: "ResellerCustomerData",toaddress_field : "email"],
   DirectCustomers: [toaddress_group: "DirCustCustomerData",toaddress_field : "dir_
    cust_email"],
   OurPartnerCompanies: [toaddress_group: "PartnersContactData",toaddress_field :
    "email"],
   RetailCustomers: [toaddress_group: "RetailCustomersContactData",toaddress_field :
    "retail_customer_email"],
]
def toaddressComplete = sourceGroupMap[custgroup].toaddress_
 group+":"+sourceGroupMap[custgroup].toaddress_field
def toaddress = maincontact.get("${toaddressComplete}")
```

Code example 64: *Alternative code to switch statement: use a multi-dimensional hashmap for more efficient coding*

D.9.4 Effects of engineer representation rules for emails sent in scripts

When you write scripts where emails are sent out to engineers and/or customers, you have to take the engineer representation rules into consideration. For detailed explanations of the engineer representation feature, please refer to the ConSol CM Administrator Manual (section *Role Administration*) and the ConSol CM User Manual (section *User Profile*).

Important information about representation configurations

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

1. An engineer writes an email using the Ticket-Email Editor: the representation rule is applied and the representing engineer receives a copy of the email. This means all emails which are sent manually using CM are sent to the original recipient and their current representative. The CM system checks if a representation rule is active for a certain email address! Please keep this in mind when you configure the representation permissions in the Admin Tool and inform your CM users (engineers) about this behavior! It might lead to unwanted effects, especially when persons are registered as engineers and as contacts in the ConSol CM system (e.g. for an internal help desk).
2. An email is sent automatically from the CM system: it depends on the specific configuration of the CM system which engineers receive a copy of the email, the email is (!) not sent to the representing engineers automatically! It might be implemented that the representing engineer gets a copy, but this is not mandatory. The original email might be sent from a workflow script or from an Admin Tool script (which might also be called from a workflow). It depends on the implementation in this script who receives a copy of the email. Please read the introductory section of Important methods and the following section for details!

For emails sent out using scripts, the CM system behavior concerning representation rules depends on the method which is used to send the email.

All of the three methods listed here belong to the java Class `Mail`.

D.9.4.1 setTo(String pTo)

```
Mail.setTo(<email of originalReceivingEngineer>)
```

This method receives a String object as parameter. This is the email address of the recipient. This method will not (!) send a copy of the original email to representing engineers.

D.9.4.2 setTargetEngineer(Engineer pTargetEngineer)

```
Mail.setTargetEngineer(<current engineer of the ticket>)
```

This method receives an Engineer object as parameter and will send the original email to the engineer's email address (if set) and to the representing engineer who is registered as representation for the original email address. Be careful here - please read the info box above!

D.9.4.3 setTargetEngineers(List<Engineer> pTargetEngineers)

This method receives a list of Engineer objects as parameter and will send the original email to the email address of each engineer in the list (if email address is set) and to the representing engineers for all of the listed engineers' email addresses. Be careful here - please read the info box above!

All three methods are used in the same way, i.e. you have to use only one of them to set the recipient's address in a script which should send an email. Please see the following example which uses the `Mail.setTargetEngineer()` method.

```
import com.consol.cmas.common.model.mail.Mail
import com.consol.cmas.core.server.service.*
import static com.consol.cmas.common.util.TemplateUtil.TICKET_SUBJECT_TEMPLATE_NAME
// create new mail object
def mail = new Mail()
def ticket = workflowApi.getTicket()
def eng = ticket.engineer
def toaddress = eng?.email
println 'toaddress is now ' + toaddress
if (!toaddress){
   log.info 'No email address found for engineer, no info notice sent.'
} else {
   // put the e-mail TO address into the Mail object
   // mail.setTo(toaddress)
   mail.setTargetEngineer(eng)
   // fetch the REPLY TO address, theis is stored in a system property
   def replyaddress = configurationService.getValue("cmweb-server-
    adapter","mail.reply.to")
   // put the e-mail REPLY TO address into the Mail object
   mail.setReplyTo(replyaddress)

   // build e-mail text using a template which is stored in the Template Designer
   def text = "Hello I am your info ticket"
   // put the e-mail text into the Mail object
   mail.setText(text)

   def subject = templateService.merge(TICKET_SUBJECT_TEMPLATE_NAME,
    [ticketName:ticket.name])
   // put the subject into the Mail object
   mail.setSubject(subject)
   // Mail should use the e-mail script which is configured for the queue
   mail.useDefaultScript()
   // send out the e-mail and register status

   try {
      mail.send();
   } catch (Exception e){
 mailStatus = false;
   }
}
```

Code example 65: *Workflow or Admin Tool script used to send an email to an engineer, using the representation feature by using the setTargetEngineer() method*

D.10 Working with path information

This chapter discusses the following:

D.10.1 Introduction 219

D.10.2 Retrieve path information for a workflow element 220

D.10.3 Example for the use of path information 220

D.10.1 Introduction

Like a file in a file system on a computer, every element of a workflow can be addressed using the path of this element. This might be required when you want to work with the element within a workflow script. A path represents the hierarchical structure of the workflow.

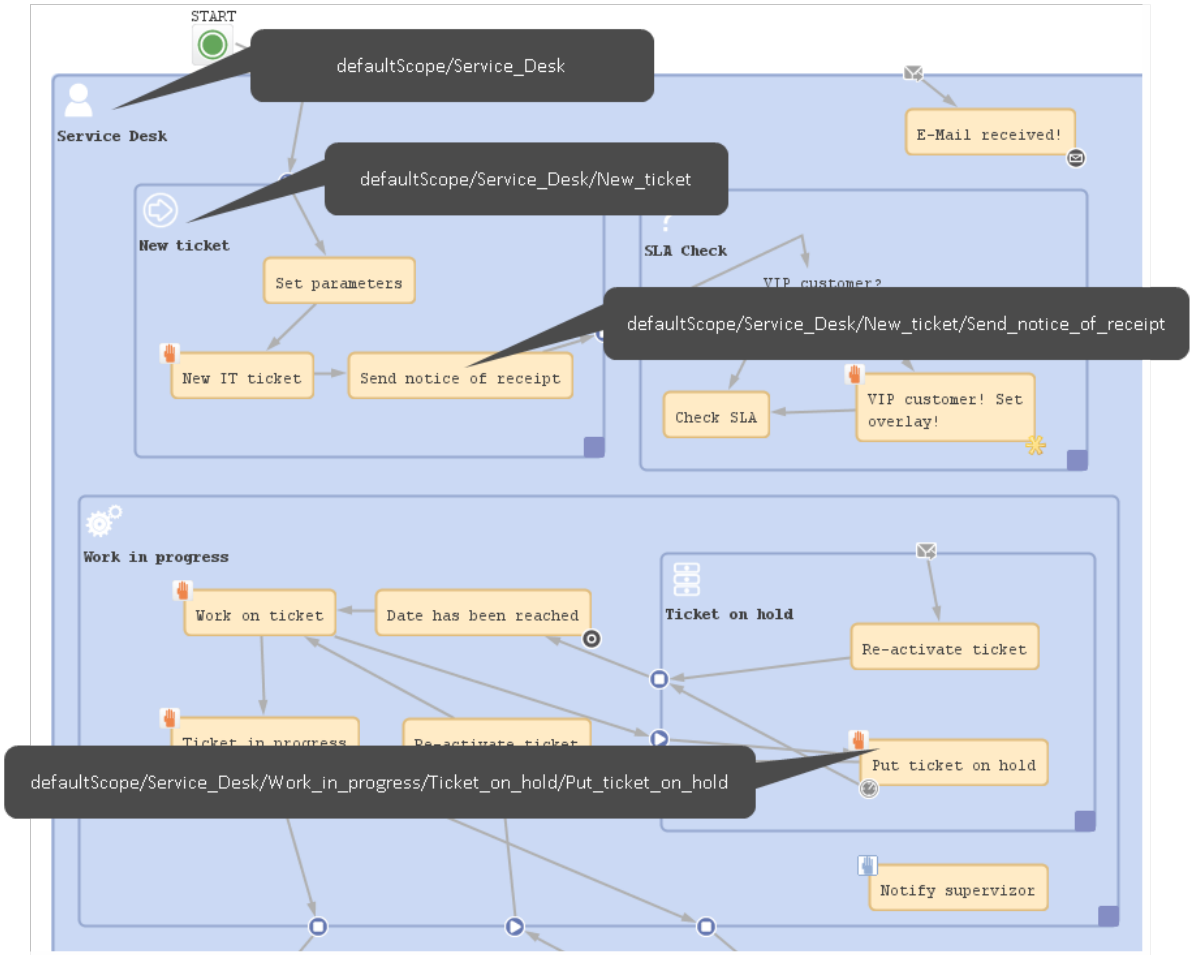


Figure 156: *ConSol CM Process Designer - Path information (example: activities and scopes)*

D.10.2 Retrieve path information for a workflow element

You can copy the path of an element by clicking on the element (an adornment in the example) with the right mouse tab and selecting *Copy adornment's path to clipboard*. For activities and scopes the entry will be *Copy node's path to clipboard*.

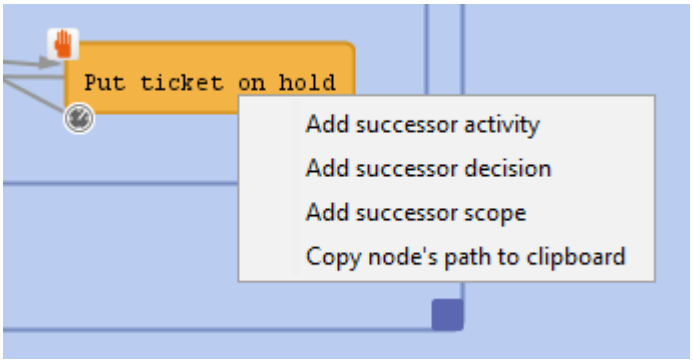


Figure 157: *ConSol CM Process Designer - Copying the path of a workflow element*

D.10.3 Example for the use of path information

D.10.3.1 Example 1: Deactivate and/or re-initialize a time trigger

A typical case for the use of path information is the re-initialization of a time trigger, e.g. if you want to measure the time after an email has been received and make sure that the email is taken care of within a period of 10 minutes maximum. That means you have to use a time trigger over and over again and re-initialize it after each email which has been received by the ticket.

When the ticket is created, the time trigger has to be deactivated. The following code would be used:

```
workflowApi.deactivateTimer("defaultScope/Service_Desk/TimeTrigger1")
```

Code example 66: *Deactivate a time trigger*

When an email has been received, the trigger has to be re-initialized. The following code would be used:

```
workflowApi.reinitializeTrigger("defaultScope/Service_Desk/TimeTrigger1")
```

Code example 67: *Re-initialize time trigger*

D.11 Working with calendars and times

This chapter discusses the following:

D.11.1 Introduction 221

D.11.2 Calculating with dates and times without a business calendar 222

D.11.3 Calculating with dates and times using a business calendar 222

D.11.1 Introduction

Calculating dates and times plays an important role in ConSol CM workflow programming. For a time trigger (see section Time triggers), the exact point in time when it is supposed to fire can be set via script. This adds various possibilities in controlling escalation times, reminders for engineers, and other **active** components of a ConSol CM process. Examples for potential calculations with dates and/or times are:

- escalation dates with time triggers
- DATE fields, like a desired (or required) deadline

When you calculate a date and/or time, you have to decide if a business calendar should be used or not. A business calendar defines working hours for a process. It is defined using the Admin Tool and assigned to one or more queues.

For example, the service desk team might have working hours from 8 to 6 for 6 days a week, whereas the administration team works on a 9-to-5 basis, 5 days a week. Using a CM business calendar makes sure that an escalation will not be set during spare time and that non-working hours are not included into the calculation of the elapsed escalation time. Please refer to the *ConSol CM Administrator Manual* for a detailed introduction to business calendars.

On the other hand, there are examples, when a business calendar is not required but the **pure** time, based on the regular calendar, should be used. For example, when it is required to get back to a customer three weeks after the initial contact. The following paragraphs will show you examples for both use cases.

> 1 day means 24 hrs of absolute time, it has nothing to do with the use of a calendar. The calendar only plays a role when the time trigger is activated, then the 24 hrs, i.e. 86400000 milliseconds, will be taken as business calendar input (if the calendar is enabled).
>
> **Example:**
> When we have as trigger time 1 day = 24 hrs without calendar, the 24 hrs are calculated like regular time, so the escalation will fire one day later at the same time.
> In contrast: When we use a calendar (with, for example, 7 work hrs per work day), the 24 hrs will be split-up according to the calendar, resulting in the firing event more than 3 days later (24 hrs = 3 x 7 hrs + 3 hrs).

D.11.2 Calculating with dates and times without a business calendar

D.11.2.1 Example: Setting a dynamic time range for a time trigger

Depending on the priority, the time trigger for an escalation is configured:

```
// prio is 'medium'
def escalationTime = configurationService.getValue("custom-mycompany-
 properties","escalation.time.medium")
def escalationTimeMillisecs = escalationTime * 60 * 1000L
trigger.setDueTime( escalationTimeMillisecs )
```

Code example 68: *Setting time for a time trigger*

D.11.3 Calculating with dates and times using a business calendar

D.11.3.1 Example: Using a time trigger with a business calendar to calculate the escalation time

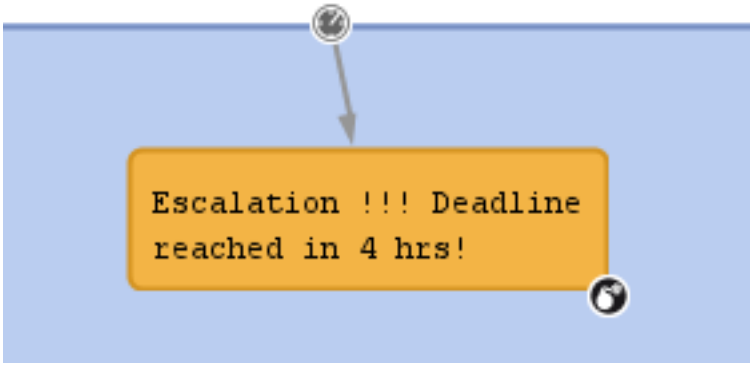


Figure 158: *ConSol CM Process Designer - Time trigger for escalation 4 hours before deadline*

```
def deadl = ticket.get("serviceDesk_fields.desiredDeadline")
if (deadl == null){
} else {
   // log.info """DEADL: ${deadl}: ${deadl.time}"""
   // 4hrs before deadline the escalation should be set
   // business calendar should be used
   // ServiceDeskCalendar is assigned to queue ServiceDesk, this is transparent here
   def now = new Date()
   // log.info """NOW: ${now}: ${now.time}"""
   // time required in millisecds
   def four_hours = -4*60*60*1000L

   // calculate escalation date
   def escalDate = BusinessCalendarUtil.getEscalationTime(deadl, four_hours,
    ticket.queue.calendar)
   // calculate and set due time
   // log.info """escalDate: ${escalDate}: ${escalDate.time}"""
   def dueTime = escalDate.time - now.time
   trigger.setDueTime(dueTime)
}
```

Code example 69: *Script for time trigger for escalation 4 hours before deadline (method BusinessCalendarUtil.getEscalationTime for CM versions 6.10.6 or 6.11.0.4 and up)*

> Please remember that the method `setDueTime()` sets the trigger time (delta) in milliseconds starting from the point in time when the ticket has entered the scope where the trigger is located!

The following methods are available with the class `BusinessCalendarUtil`. They are helpful for calculating times when a business calendar is in use.

- **getBusinessTime(Date pCurrentDate, Date pEndTime, BusinessCalendar pCalendar):**
  Calculates the interval (milliseconds) between two timestamps taking a business calendar into respect.
- **getEscalationTime(Date pCurrentDate, long pDurationInMillis, BusinessCalendar pCalendar):**
  Calculates a timestamp by adding an interval (milliseconds) to another timestamp taking a business calendar into respect. (Replaces the deprecated method). Returns a `Date` object.
- **getBusinessTime(Date pCurrentDate, long pDurationInMillis, BusinessCalendar pCalendar) DEPRECATED in CM 6.10.6.0 and 6.11.0.4!:**
  Calculates a timestamp by adding an interval (milliseconds) to another timestamp taking a business calendar into respect.

D.12 Working with object relations

In ConSol CM, you can work with three types of relations:

Relation type	Explanation
Ticket Relations	Hierarchical or one-level relations between two tickets, see section Working with ticket relations.
Customer Relations	Relations between customer objects, i.e. contacts and companies, see section Working with customer relations.
Resource Relations	Relations between a resource object and another object. The latter can be of type ticket, resource or customer. See section Working with resource relations.

D.12.1 Working with ticket relations

This chapter discusses the following:

- Introduction
- Simple ticket relation without a hierarchy
- Master-slave relations
- Parent-child relations
- Important methods for working with ticket relations

D.12.1.1 Introduction

Relations between tickets can help to model your business processes in a very efficient way.

ConSol CM offers three types of relations:

- **Simple ticket relations**
  Non-hierarchical, simple reference. Each ticket can have any number of references.
  A simple ticket relation can be built by an engineer using the Web Client or by a programmer using the ConSol CM programming interface.
  In both cases, a reference can only be established between two existing tickets.
- **Master-slave relations**
  Hierarchical. A master ticket can have several slave tickets. A slave ticket always has exactly one master ticket.
  This construct can be built by an engineer using the Web Client or by a programmer using the ConSol CM programming interface.
  A master-slave relation can only be established between two existing tickets, i.e. the tickets both have to exist first, then a master-slave relation can be built to connect them.
- **Parent-child relations**
  Hierarchical. A parent ticket can have several child tickets. A child ticket always has exactly one parent ticket.
  This construct can only be built and manipulated using the ConSol CM programming interface.
  A parent-child relation can be built between existing tickets. Also a new child ticket can be created during the process.

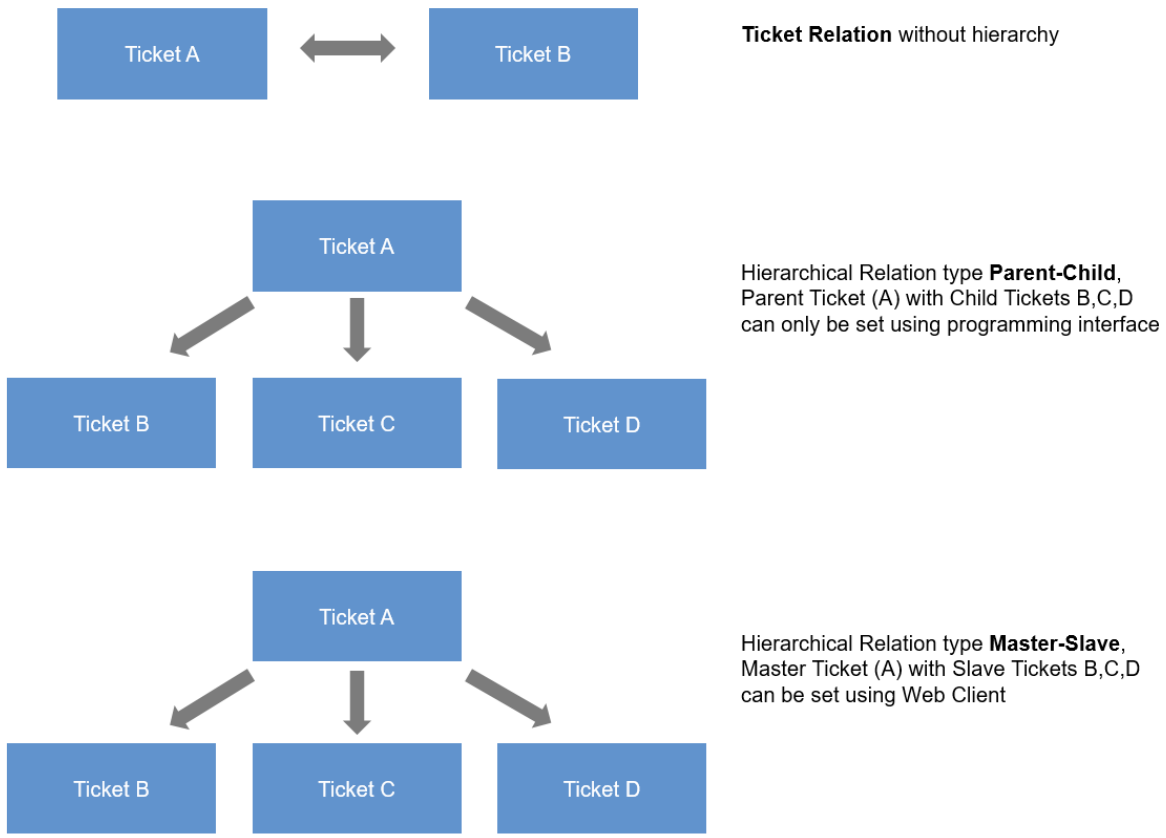


Figure 159: *ConSol CM relation types*

In this section, we will not explain how to set-up ticket relations using the Web Client, this is explained in detail in the *ConSol CM User Manual*. In the current section, we will show you how to establish relations using the programming interface, namely workflow scripts.

In the ConSol CM Workflow API, the reference type is represented by the class (enum) `com.consol.cmas.common.model.ticket.TicketRelationType`. This offers three values:

- REFERENCE
- MASTER_SLAVE
- PARENT_CHILD

D.12.1.2 Simple ticket relation without a hierarchy

This relation type can be helpful when you want to create references which help to find the tickets related to one ticket easier than using the search function.

Example use cases are:

- When a new ticket is created you want to see if there are any other open tickets from the same customer. If yes, you create a relation between the tickets. In this way, an engineer can easily jump from one open ticket of the customer to the next.
- When a new ticket is created for a certain hardware category, you want to establish references to all other tickets with the same hardware type.

This relation type can be built and manipulated using either the Web Client or the programming interface. Thus, a relation of type `REFERENCE` can be built within a workflow script and can then be manipulated by an engineer using the Web Client, provided he/she has the required access rights.

Example: Creating a simple relation between two tickets

```
workflowApi.addRelation(TicketRelationType.REFERENCE, "This is a very important
 relation", pSourceTicketId, pTargetTicketId)
```

Code example 70: *Creating a ticket relation of type REFERENCE using workflowAPI*

Syntax: Find all referenced tickets

```
List<Ticket> mytickets = workflowApi.getReferencedTickets()
```

D.12.1.3 Master-slave relations

This relation type can be helpful when you want to create a hierarchy between a certain number of existing tickets. Remember that this relation type can be established using either the Web Client or using the programming interface. However, here, only the programming approach will be explained.

Example use cases are:

- In a company, there are several projects, each represented by a ticket. When the decision has been made to integrate one of the projects in an overall program (also represented by a ticket), the project manager uses the workflow activity *Integrate into Program*. There, the correct program has to be selected (e.g. using an ACF). In the script of the workflow activity *Integrate into Program*, the program ticket is set as *Master* ticket of the current project ticket.
- In a service team, tickets for several different products are managed. For each product, there is one product ticket. When a new service ticket has been opened, the engineer uses the activity *Set product* where he can select the related product from a drop-down menu. In the workflow script of the activity *Set product*, the service ticket is automatically set as *Slave* of the product ticket.

> ⚠ A master-slave relation can be built and manipulated using either the Web Client or the programming interface. Thus, a relation of type *MASTER_SLAVE* can be built within a workflow script and can then be manipulated by an engineer using the Web Client, provided the engineer has the required access rights. Use the *parent-child* construct if you want to make sure that no engineer can manipulate the ticket hierarchy.

Example: Creating a master-slave relation between two tickets

```
//in this script the project ticket (= current ticket) is set as slave ticket to
// the program ticket which becomes the master

// fetch the program ticket ID. The ID of the program ticket is already stored
// in a CF in the project (=current) ticket
def progTicketId = ticket.get("ReferencesFields.ProgramTicketId")

// fetch ID of current ticket (which will become the slave)
def mySlaveProjectId = ticket.id

workflowApi.addRelation(TicketRelationType.MASTER_SLAVE, "Slave Ticket: This
 project is part of the program indicated in the master ticket", progTicketId,
 mySlaveProjectId)
```

Code example 71: *Creating a ticket relation of type MASTER_SLAVE using workflowAPI*

Syntax: Finding all slave tickets

```
List<Ticket> mytickets = workflowApi.getSlaveTickets()
```

Code example 72: *Finding all slave tickets of the current ticket*

Syntax: Finding the master ticket

```
def mticket = workflowApi.getMasterTicket()
```

Code example 73: *Finding the master ticket of the current ticket*

D.12.1.4 Parent-child relations

This relation type can be helpful when you want to create a hierarchy between a certain number of tickets which should not be manipulated manually.

Example use cases are:

- A project should be managed by the project management ticket which becomes the parent. All tasks within the project are represented as child tickets. This structure is automatically created by a workflow script during set-up of the project ticket.
- A system migration is planned using one parent ticket. For each single component which has to be migrated a child ticket is built. This structure is automatically created by a workflow script during set-up of the project ticket.

The relation type *PARENT_CHILD* can only be built and manipulated using the programming interface. Thus, a relation of this type can be built within a workflow script and can then only be manipulated by other scripts.

Example 1: Creating a new child ticket as a child of the current ticket

```
// this script creates a ticket for a task which will be child ticket
// of a project ticket (which will be the parent)

// create a new ticket, which will become the task (=child) ticket
Ticket newTask = new Ticket()

// fetch the subject of the parent-to-be ticket, i.e. of the current ticket
def subj = ticket.subject
// or longer: def subj = ticket.getSubject()

// set the subject of the new task (= child) ticket
newTask.setSubject("New Task for project " + subj)

// put the task (= child) ticket into the tasks queue
def tasksQueue = queueService.getByName("Tasks")
newTask.setQueue(tasksQueue)

// Initially, the new task ticket will not have an engineer
newTask.setEngineer(null)

// define the ticket text, i.e. the first comment in the new task ticket
def taskTicketText = "Please work on this task asap"

// the contact for the new task ticket should be the same as the one for the
 project ticket:
def taskContact = workflowApi.getPrimaryContact()

//create PARENT_CHILD relation between project (parent) and task (child)
workflowApi.createChildTicket(newTask, taskTicketText, taskContact)
```

Code example 74: *Creating a child ticket*

Example 2: Finding the parent ticket of a ticket

```
def my_parent = workflowAPI.getParentTicket()
```

Code example 75: *Finding the parent ticket of a ticket*

Example 3: Finding all child tickets of a ticket

```
// only works for current ticket:
List<Ticket> my_childtickets = workflowApi.getChildTickets()
```

Code example 76: *Finding all child tickets of a ticket*

Example 4: Finding all brother tickets (other child tickets) of the same parent ticket

```
// only works for current ticket:
List<Ticket> my_brothers = workflowApi.getBrotherTickets()
```

Code example 77: *Finding all brother tickets of a (child) ticket*

D.12.1.5 Important methods for working with ticket relations

Note the following rules for the work with ticket relations:

- In MASTER_SLAVE relations, the master is always the source.
- In PARENT_CHILD relations, the parent is always the source.
- In simple REFERENCE relations the source is the ticket from which the relation has been created.

WorkflowApi methods

The following methods are methods of the class `WorkflowContextService` which is implicitly available as `workflowApi` object in workflow scripts.

Method of workflowApi (workflowContextService)	Explanation
Ticket createChildTicket(Ticket pTicket, String pTicketText, Unit pCustomer)	Creates a new child ticket. Queue and ticket fields have to be set correctly.
List<Ticket> getChildTickets()	IntSet containing the ticket objects of the child tickets of the current ticket.
List<Ticket> getBrotherTickets()	IntSet containing the ticket objects of the brother tickets of the current ticket.
Ticket getParentTicket()	Ticket object of the parent ticket or “null” if the current ticket does not have a parent ticket.
List<Ticket> getTargetTickets(TicketRelationType pType)	Get list of ticket objects that current ticket has relations of certain type to. For those relations, the current ticket is the source ticket.
List<Ticket> getTargetTickets(long pTicketId, TicketRelationType pType)	Get list of ticket objects that current ticket has relations of certain type to. For those relations, the ticket given with `pTicketId` is the source ticket.
List<Ticket> getSourceTickets(TicketRelationType pType)	Get list of ticket objects that current ticket has relations of certain type from. For those relations, the current ticket is the destination ticket.
List<Ticket> getSourceTickets(long pTicketId, TicketRelationType pType)	Get list of ticket objects that current ticket has relations of certain type from. For those relations, the given ticket is the destination ticket.

Method of workflowApi (workflowContextService)	Explanation
boolean hasTargetTickets(TicketRelationType pType)	Check if ticket has target tickets. Check if relations exist that have this ticket as source ticket.
boolean hasTargetTickets(long pTicketId, TicketRelationType pType)	Check if given ticket has target tickets. Check if relations exist that have this ticket as source ticket.
boolean hasSourceTickets(TicketRelationType pType)	Check if ticket has source tickets. Check if relations exist that have this ticket as target ticket.
boolean hasSourceTickets(long pTicketId, TicketRelationType pType)	Check if given ticket has source tickets. Check if relations exist that have this ticket as target ticket.
void changeSourceTickets(TicketRelationType pType, long pTargetTicketId, List<Long> pSourceTicketIds)	For the target ticket (e.g. a child ticket) the relations of a given type (e.g. PARENT_CHILD) are removed. For the same relation type a new relation is created with the provided source tickets.
void changeTargetTickets(TicketRelationType pType, long pSourceTicketId, List<Long> pTargetTicketsIds)	For the given source ticket all relations of the given type are removed. For the list of provided target tickets new relations of the given type are created.
void removeRelation(TicketRelationType pType, long pSourceTicketId, long pTargetTicketId)	Remove ticket relation between two tickets with specified type.
void addRelation(TicketRelationType pType, String pComment, long pSourceTicketId, long pTargetTicketId)	Add relation of the specified type between ticket `sourceTicketId` and `targetTicketId`.
Ticket getMasterTicket()	Return the master ticket of the current ticket.
List<Ticket> getSlaveTickets()	Returns a list of all slave tickets of the current ticket
List<Ticket> getReferencedTickets()	Returns a list of all tickets which have a simple reference to the current ticket.

Table 1: *Important methods of workflowApi for the work with ticket relations*

TicketRelationService methods

If you work with scripts in the Admin Tool (which are then called from a workflow script), the workflowApi is not available. You can use methods of the class `TicketRelationService` which is available as singleton `ticketRelationService`.

Method of ticketRelationService	Explanation
List<TicketRelation> getByTicket(Ticket pTicket, TicketRelationDirection pTicketRelationEnd, TicketRelationType... pRelationType)	Get a list of all ticket relations for a given ticket constrained by the relation direction (ANY|SOURCE|TARGET) and the relation type (MASTER_SLAVE|PARENT_CHILD|REFERENCE)
Set<TicketRelation> getByTickets (Set<Ticket> pTickets, TicketRelationDirection pTicketRelationEnd, TicketRelationType... pRelationType)	Get set of all tickets relations for a given set of tickets constrained by the relation direction (ANY|SOURCE|TARGET) and the relation type (MASTER_SLAVE|PARENT_CHILD|REFERENCE)

Table 2: *Important methods of TicketRelationService for the work with ticket relations*

Example with workflowApi and ticketRelationService methods

```
// use wflApi:
println 'Displaying slave tickets from wfl script ...'
List<Ticket> slave_tics = workflowApi.getSlaveTickets()
slave_tics?.each(){ st ->
   log.info ' Slave Ticket is now ' + st.getId() + ' -- ' + st.getSubject()
}
```

Code example 78: *Example Script, display IDs and names of slave tickets workflow version*

```
// DisplaySlaveTickets.groovy
// use in AT:
import com.consol.cmas.common.service.*
import com.consol.cmas.common.model.ticket.TicketRelation
import com.consol.cmas.common.model.ticket.TicketRelationDirection
import com.consol.cmas.common.model.ticket.TicketRelationType
println 'Displaying slave tickets from AT script ...'
def ticket = workflowApi.getTicket()

List<TicketRelation> t_rel = ticketRelationService.getByTicket(ticket,
 TicketRelationDirection.ANY, TicketRelationType.MASTER_SLAVE)
t_rel?.each(){ tr ->
   log.info 'Source ticket is now ' + tr.sourceTicket.id + ' -- ' +
    tr.sourceTicket.subject
   log.info 'Target ticket is now ' + tr.targetTicket.id + ' -- ' +
    tr.targetTicket.subject
}
```

Code example 79: *Example Script, display IDs and names of slave tickets Admin Tool script version*

```
scriptExecutionService.execute("DisplaySlaveTickets.groovy")
```

Code example 80: *Calling previous AT script from workflow activity*

D.12.2 Working with customer relations

This chapter discusses the following:

- Introduction
- Creating customer relations using the programming interface
- Important Java classes for the work with customer relations

D.12.2.1 Introduction

You need in a good knowledge of the customer data model used in ConSol CM to be able to work with customer relations. Please refer to the *ConSol CM Administrator Manual* for a detailed introduction. Since a customer is represented by the class `Unit` in CM programming, the terms *customer relation* and *unit relation* are synonyms.

Three objects are essential:

Object	Java class	Admin Tool description	Explanation
Customer	Unit	<none>	The general description or the general object which represents a customer, i.e. some person (contact) or company or another object on company or contact level which is registered in the CM database
Company	Unit	Customer object of type *company*	An object on company level (i.e. the highest level in the customer model). This can be a real company or this can be a machine or another object which represents the level. An object on the **company level** can be the **parent level** for an object on the **contact level**. From a logical point of view, a company can have several contacts.
Contact	Unit	Customer object of type *contact*	An object on contact level (i.e. the lowest level in the customer model). This can be a real person or another object which represents the level. An object on the **contact level** can be a stand-alone object (in a one-level customer model) or can belong to a **company level** object. From a logical point of view, a contact can belong to none or exactly one company.

> ⚠ Keep in mind that the main customer of a ticket can be a contact or a company! The method used is `ticket.getMainContact()`. This returns an object of class `Unit`. The object can be a contact or a company!

Customer relations represent relations between customers, i.e. companies and contacts.

They can be:

- **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.

In the programming interface, a customer object (i.e. a contact or a company) is represented by an object of the class `Unit`.

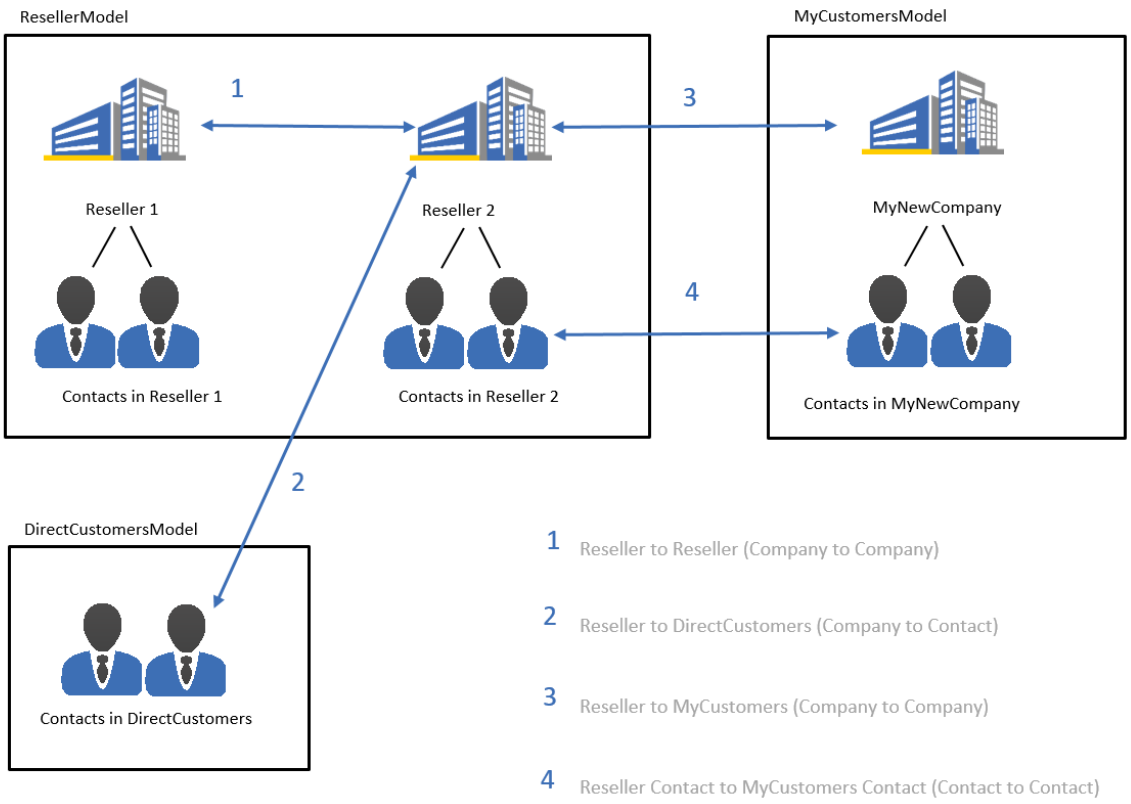


Figure 160: *Customer relations in ConSol CM*

> ⚠ To work with customer relations in workflow scripts, make sure you have established and configured all required relations using the Admin Tool before you start programming.

D.12.2.2 Creating customer relations using the programming interface

> ⚠ In this book we sometimes use the terms *customer* and *customer definition* which are part of the customer model of ConSol CM (FlexCDM). However, the names of the corresponding Java classes are `Unit` and `UnitDefinition`. All other Java classes which deal with customer objects are also 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* documentation for details.

Example 1: Add a reseller - end customer relation

In the following example, a relation has been defined in the Admin Tool to reflect a *reseller - end customer* relation. A company of the customer group *Reseller* sells products to a customer (a person, a contact) of the customer group *DirectCustomers*.

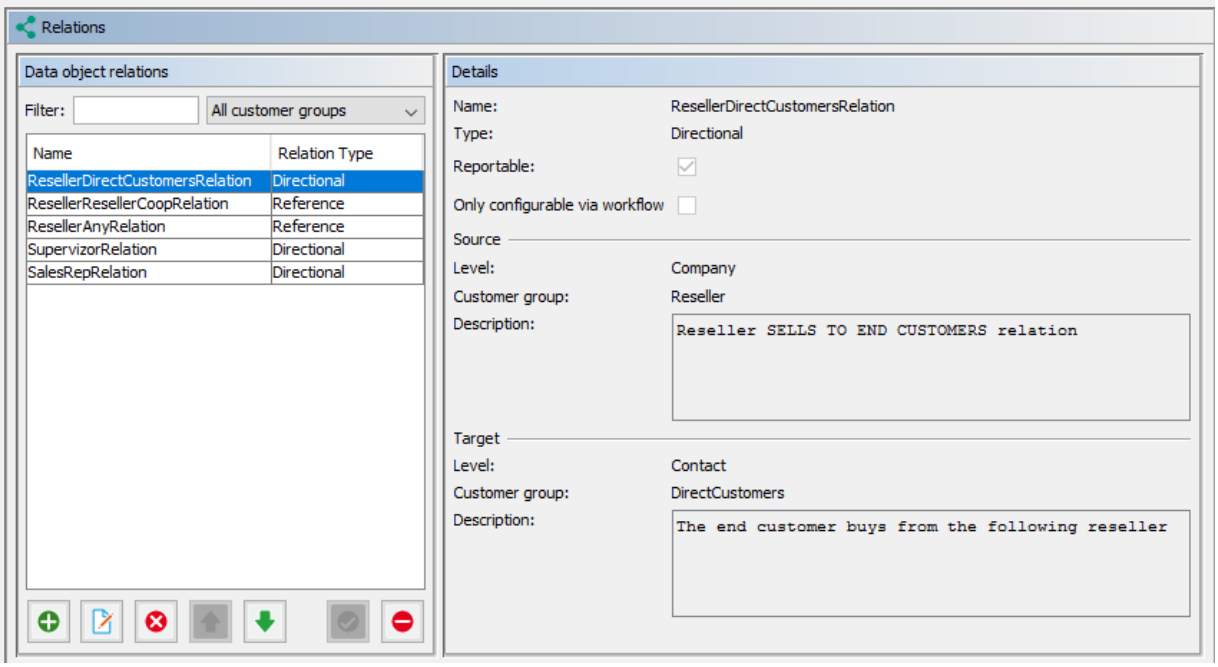


Figure 161: *ConSol CM Admin Tool - Definition of reseller - end customer relation*

A ticket is created with a main customer. This customer is an employee of a reseller company. The end customer to whom the reseller company sells products is added as additional customer in the role *end customer* to the ticket. The engineer who works on the ticket should be able to create a relation between the reseller company (source) and the end customer person (target) using a workflow activity.

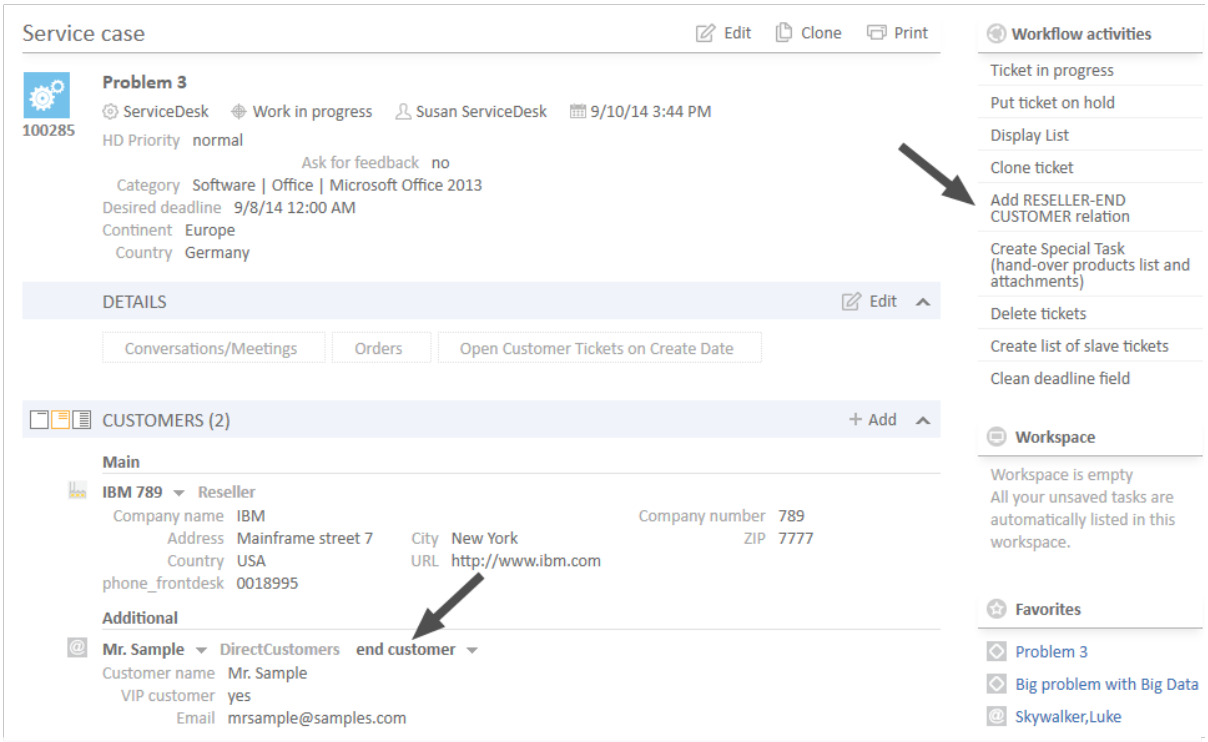


Figure 162: *ConSol CM Web Client - Example ticket with main customer and one additional customer*

In the *Service Desk* workflow, there is a workflow activity *Add RESELLER-END CUSTOMER relation* (see next figure).

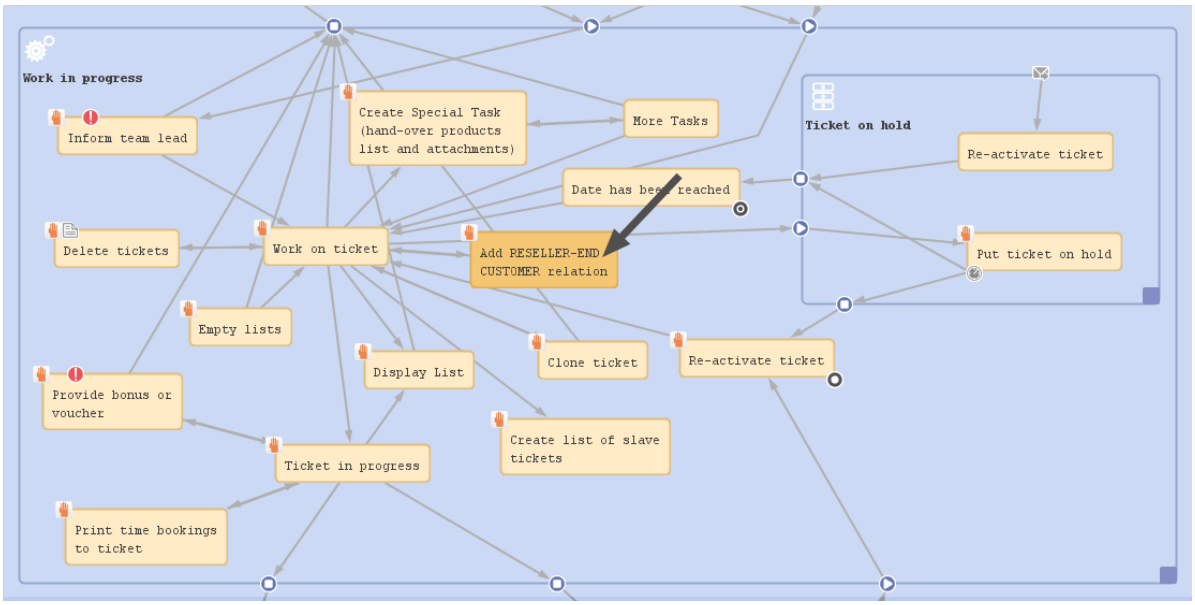


Figure 163: *ConSol CM Process Designer - Workflow activity for adding a unit relation*

The following script is used in the workflow activity *Add RESELLER-END CUSTOMER relation*:

```
// get Company of the main customer of the ticket, this is the RESELLER company
def maincust = ticket.mainContact
def unit_type = maincust.definition.type
log.info 'TYPE is now : ' + unit_type
def comp

if(unit_type == UnitDefinitionType.CONTACT) {
   comp = maincust.get("company()")
} else if (unit_type == UnitDefinitionType.COMPANY){
   comp = maincust
}

// works only for Resellers!
def custGroup = comp.customerGroup.name

if (custGroup.equals("Reseller")){
   def end_custs = ticket.getContacts("end customer").each() { e_cust ->
      log.info "E_CUST is now " + e_cust.get("DirCustCustomerData.dir_cust_customer_
       name")
      //build all components for new unit relation
      // -- get the unit definitoin
      def unitrel_def = unitRelationDefinitionService.getByName
       ("ResellerDirectCustomersRelation")
      def new_rel = new UnitRelation(unitrel_def, comp, e_cust, "This Reseller sells
       to the end customer")
      def new_rel2= unitRelationService.create(new_rel)
   }
} else {
   workflowApi.addValidationError("INFO", "Please set Reseller Company or contcat as
    main customer first.")
}
```

Code example 81: *Adding a data object relation using a workflow script*

When the engineer has executed the workflow activity, the relation from the *reseller* company to the *end user* has been established.

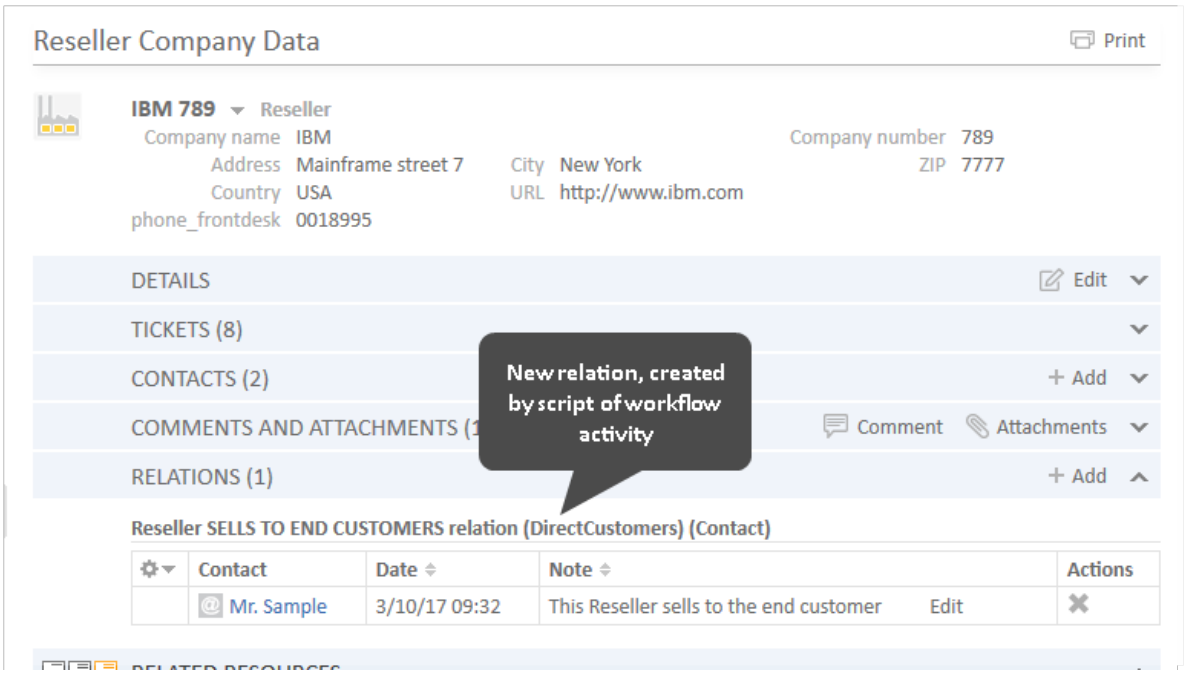


Figure 164: *ConSol CM Web Client - New unit relation (created by workflow script)*

D.12.2.3 Important Java classes for the work with customer relations

Java class	Explanation
`Unit`	A customer object (unit): a contact or a company.
`UnitRelation`	A relation between two customer objects (units). Visible in the Web Client on the contact or company page under *Relations*.
`UnitRelationDefinition`	The definition of a unit relation as configured in the Admin Tool under *Customers - Relations*. A `UnitRelation` always has a certain `UnitRelationDefinition`.
`UnitRelationDefinitionService`	Singleton. Available as object `unitRelationDefinitionService`. Service which provides helpful methods for the work with customer (unit) relations. See the *ConSol CM Java API* documentation for details.
`UnitRelationService`	Singleton. Available as object `unitRelationService`. Service which provides helpful methods for the work with customer (unit) relations. See the *ConSol CM Java API* documentation for details.

D.12.3 Working with resource relations

This chapter discusses the following:

- Introduction
- Example: Calculating the reaction time of a ticket

D.12.3.1 Introduction

You can establish relations between resource objects and other objects using the module CM/Resource Pool. A resource relation can connect a resource to

- a ticket
- a customer (contact or company, i.e. a unit)
- another resource

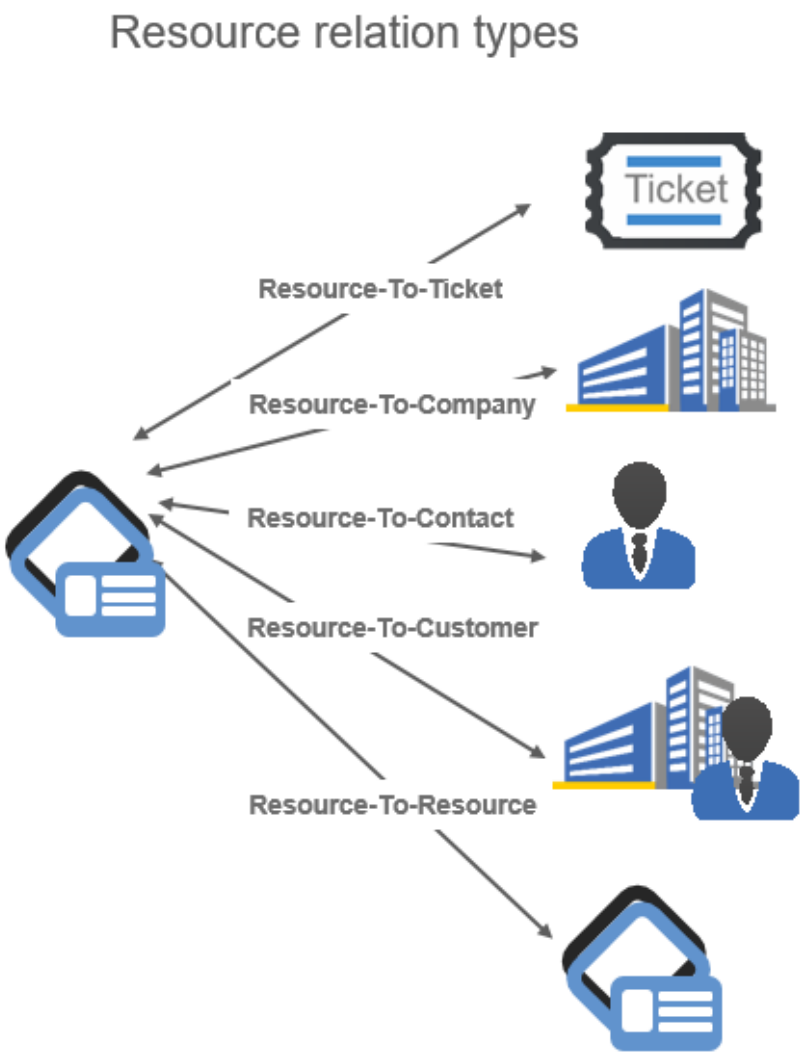


Figure 165: *Resource relation types*

To be able to work with resource relations, you have to have a profound knowledge of the resource data model. Please refer to the *ConSol CM Administrator Manual*, section *CM/Resource Pool* for a detailed introduction.

The following Java classes are essential for working with resources, but of course they represent only a small subset of Java classes which form the Resource Pool. Please refer to the ConSol CM Java API Doc for a comprehensive description of all classes and methods.

Object	Java class	Admin Tool descrip-tion	Explanation
resource	Resource	Resource	An object of the defined Resource Type.
resourceRela-tionService	Interface ResourceRelationService	-	the service which offers a lot of helpful meth-ods for dealing with resources
ResourceRelation xy	ResourceRelation	Resource (type) relation	Relation between a resource and another object
	ResourceRela-tionWithTargetResourceCriteria ResourceRela-tionWithTargetTicketCriteria ResourceRela-tionWithTargetUnitCriteria	-	Stores the criteria which are the basis for a search, performed by the `resourceRelationSer vice`

D.12.3.2 Example: Calculating the reaction time of a ticket

A company can have a relation to a resource of type *SLA*. In this resource, SLA data are stored, e.g., the reaction time. The desired deadline for a service ticket is calculated by extracting the value of the reaction time from the resource object. With this value, the deadline for the service ticket is calculated and the result is entered in the *Desired deadline* data field of the ticket.

```
import com.consol.cmas.common.service.resource.*
import com.consol.cmas.common.model.resource.*

// if it is a clone - fill the respective custom field:

//Take SLA (resource) from company of main customer and calculate deadline from it
log.info('starting script in Set Parameters in ServiceDesk2 ...')

def maincust = ticket.mainContact
def unit_type = maincust.definition.type
log.info 'TYPE is now : ' + unit_type
def comp

if(unit_type == UnitDefinitionType.CONTACT) {
   comp = maincust.get("company()")
} else if (unit_type == UnitDefinitionType.COMPANY){
   comp = maincust
}

// find SLA relation from SLA resource to the main customer (if company) or the
 company of the main customer (if contact)
def crit = new ResourceRelationWithTargetUnitCriteria()
crit.setUnit(comp)
crit.setResourceTypeName("SLAs")
List<ResourceRelationWithTargetUnit> myrelations =
 resourceRelationService.getByCriteria(crit)
log.info 'myrelations size is now ' + myrelations.size()


if (myrelations.size() > 0) {
   // one unit can have only one SLA as relation, see AT definition
   def my_sla = myrelations[0].getSourceResource()
   def sla_name = my_sla.get("SLA_Fields_basic.SLA_Name")
   println 'SLA name is now ' + sla_name
   def react_days = my_sla.get("SLA_Fields_basic.ReactionTime")
   react_days = Integer.valueOf(react_days.intValue())
   // calculate reaction time
   def now = new Date()
   def deadline = now + react_days
   ticket.set("serviceDesk_fields.desiredDeadline",deadline)
}
```

Code example 82: *Calculate ticket deadline from SLA. SLA as resource which is linked to the ticket.*

D.13 Working with text classes

This chapter discusses the following:

D.13.1 Introduction 242

D.13.2 Examples 244

D.13.1 Introduction

A **class of text** is a classification that you assign to a ticket entry. This entry can be:

- a comment
- an email that was sent from the ticket
- an email that was received in the ticket
- an attachment

Assigning a class of text can serve one or more of the following purposes:

- Highlighting the text in the ticket with a special color to make it easier to find (e.g., an important note, as shown in the following figure). An icon can also be used for each class of text.
- Marking a ticket entry to make it visible in CM/Track, i.e., to make it available for customers who log in to the ConSol CM customer portal.
- Marking the entry to control the process flow, e.g., a ticket can only be finished when exactly one entry has been marked as *solution*.
- Marking the entry for hand-off to another process, e.g., the entries marked *question* and *answer* are automatically used for an FAQ ticket.

Thus, with classes of text you can organize ticket information within the ticket and can also control the process flow and the availability of information.

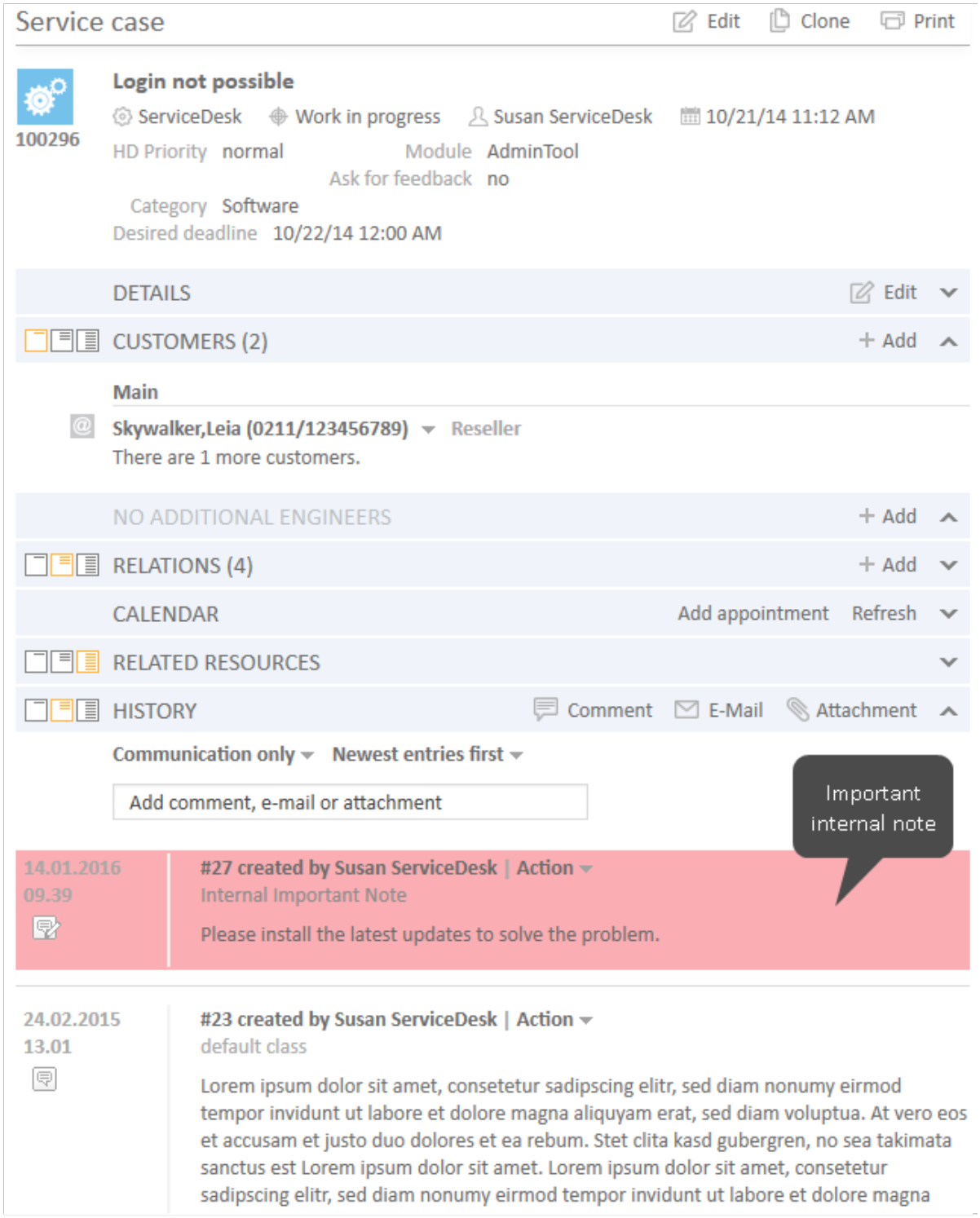


Figure 166: *ConSol CM Web Client - Using a class of text for an internal important note*

A detailed explanation about how to configure and manage classes of text is provided in the *ConSol CM Administrator Manual*. The work with classes of text in the Web Client is explained in the *ConSol CM User Manual*.

The following code sections show examples from use cases which occurred rather often in our every-day consulting life.

D.13.2 Examples

D.13.2.1 Example 1: Checking if a solution exists before the ticket can be closed

In the following example, all text entries of a ticket (i.e. comments, emails) are checked. If the class of text *Solution* is set at least once (this could be checked even more exactly to see if there is exactly one Solution entry), it is possible to close the ticket, i.e. the script will continue.

If the Solution class has not yet been set, an error message is displayed in the Web Client. We work with self-defined labels here, see the *ConSol CM Administrator Manual*, section *Labels* for a detailed explanation. In this way, it is not required to define the messages for different locales here in the script, but this happens automatically, because the labels have been localized.

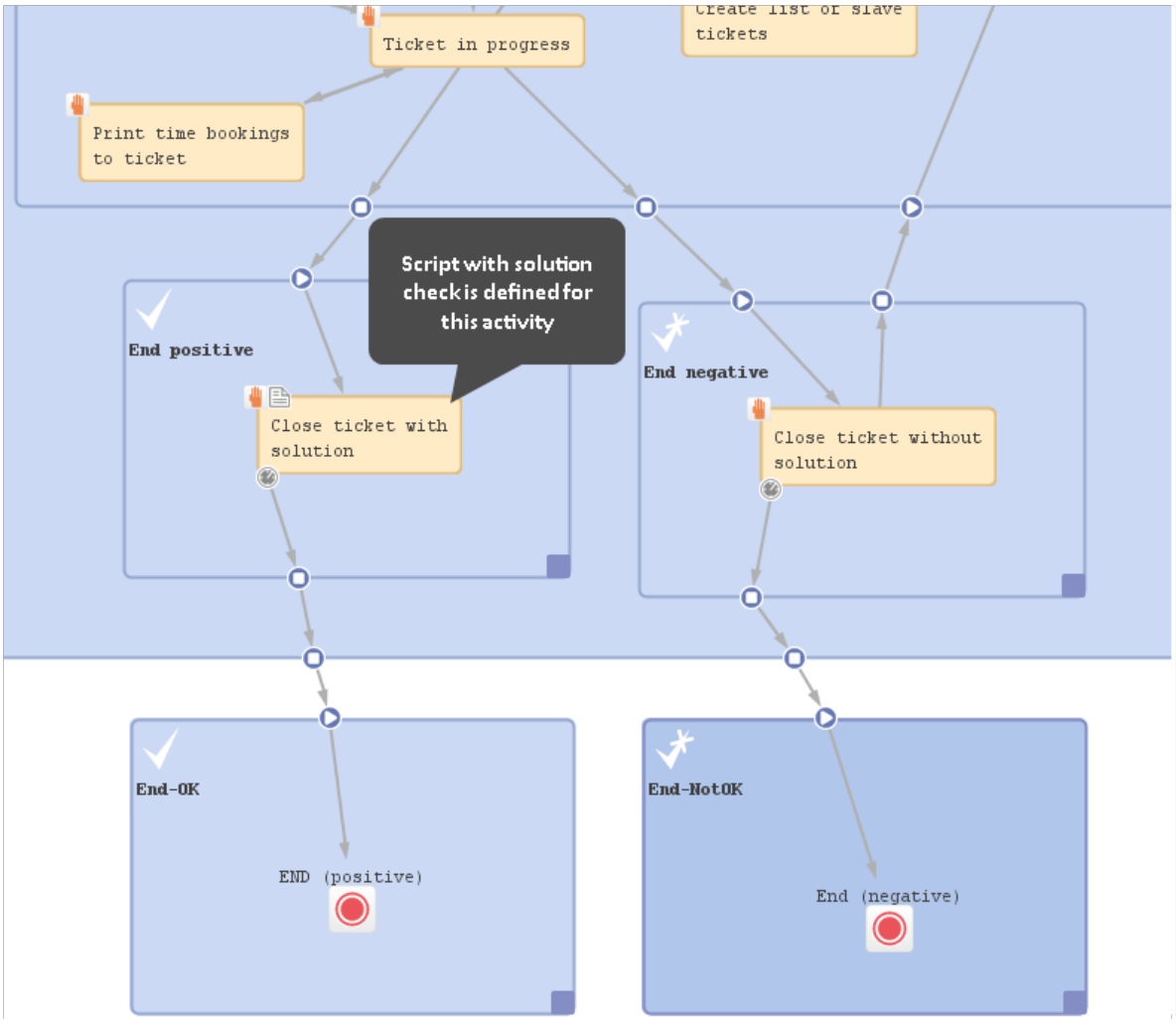


Figure 167: *Workflow activity where the script is used*

```
// ticket can only be positively closed when solution - see class of text - is
 provided
def sol_ok = false
List<TextEntry> te_list = workflowApi.ticketText

te_list.each() { te ->
   def cont_class = te.contentEntryClass?.name
   if(cont_class.equals('Solution')){
      sol_ok = true
   }
}

if (!sol_ok) {
   def mylocale = engineerService.getCurrentLocale()
   text = messageProviderService.getMessage("error.solution", mylocale)
   workflowApi.addValidationError("INFO", text)
}
```

Code example 83: *Workflow script to check if a solution has been defined in the ticket*

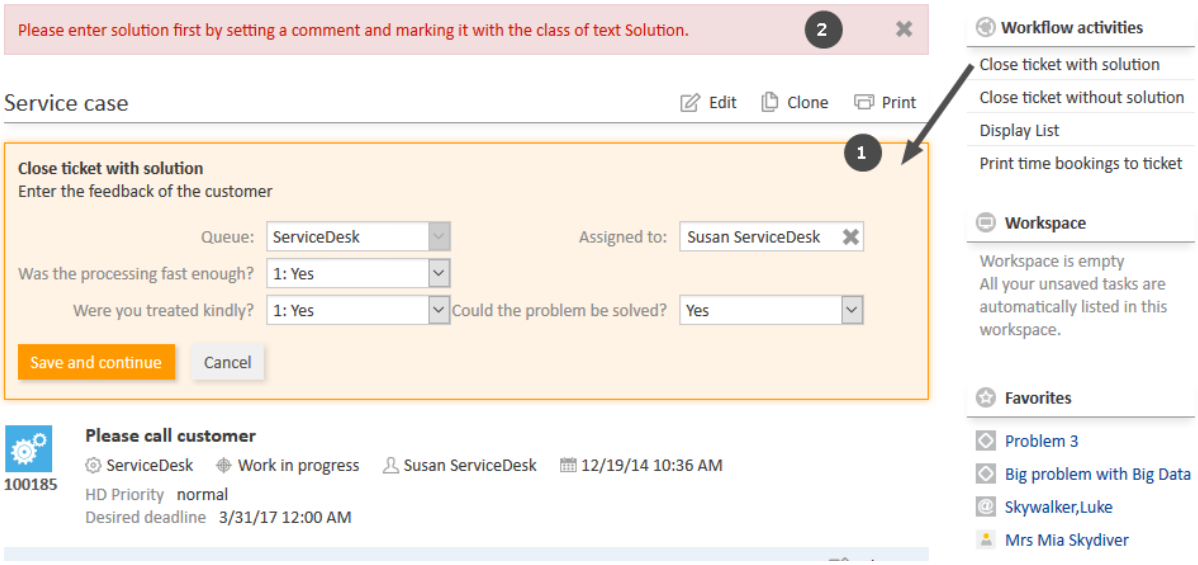


Figure 168: *ConSol CM Web Client - Error message if no solution has been defined in the ticket*

D.13.2.2 Example 2: Adding a text as ticket comment and setting a class of text

The following example shows a postActivityExecutionScript. When the activity *Close with solution* in the ServiceDesk workflow is called, an FAQ ticket is created automatically. The text which has been marked as *Solution* in the ServiceDesk ticket (compare previous example) is written into the new FAQ ticket as comment and is marked by the class of text Solution again. In this way, a new FAQ suggestion is created with a solution which has proved to work.

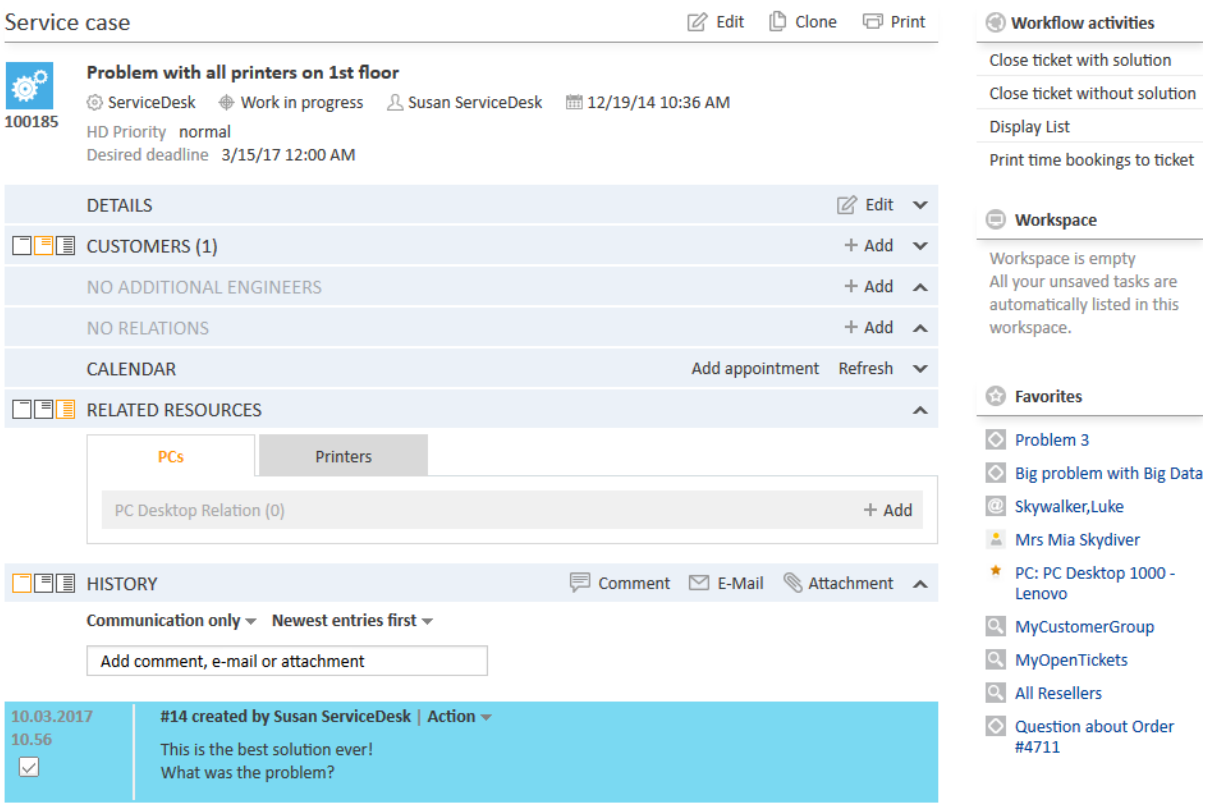


Figure 169: *ConSol CM Web Client - Service Desk ticket with solution*

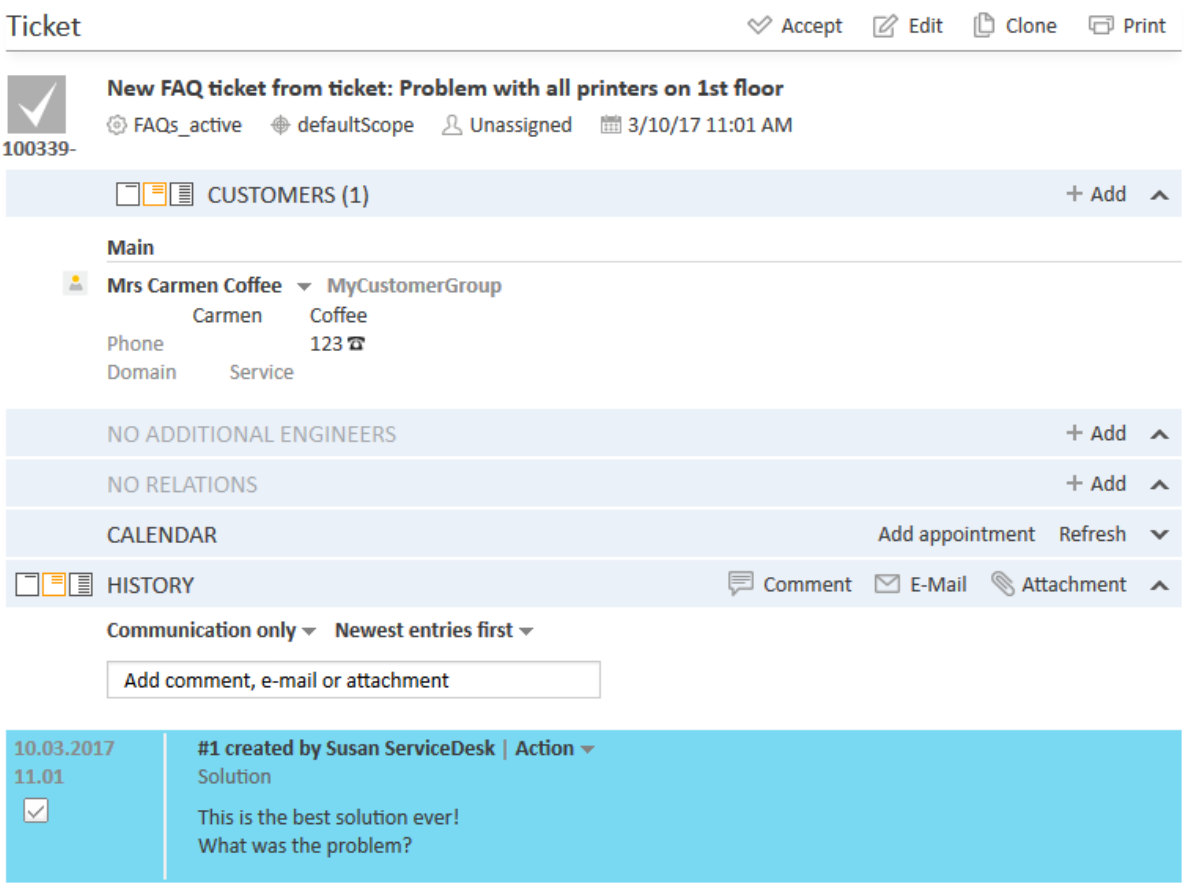


Figure 170: *ConSol CM Web Client - New FAQ ticket with solution text*

```
import com.consol.cmas.common.model.ticket.Ticket
import com.consol.cmas.common.model.customfield.Unit
import com.consol.cmas.core.server.service.action.PostActionType
import com.consol.cmas.common.model.content.TextEntry
import com.consol.cmas.common.model.content.ContentEntry
switch(activity.name){
   // other cases ...

   case 'defaultScope/Service_Desk/End_positive/Close_ticket_with_solution': M:{
      Ticket newtic = new Ticket()
      def faq_queue = queueService.getByName("FAQs_active")
      newtic.setQueue(faq_queue)
      Unit mycont = ticket.getMainContact()
      newtic.setSubject("New FAQ ticket from ticket: " + ticket.getSubject())
      ticketService.createWithUnit(newtic,mycont)
      // add solution text from parent ticket as ticket comment
      List<ContentEntry> ce_list = ticketContentService.getContentEntries
       (ticket,TextEntry.class)

      ce_list.each() { ce ->
         def cont_class = ce.contentEntryClass?.name
         if(cont_class.equals('Solution')){
            def mytext = ce.text
            def new_te = new TextEntry("Solution from SD Ticket", mytext)
            def ce_class = ce.contentEntryClass
            new_te.setContentEntryClass(ce_class)
            ticketContentService.createContentEntry(newtic, new_te)
         }
      }
   }
}
```

Code example 84: *Excerpt from the postActivityExecutionScript*

Please note that, in this example, for the engineer it is not obvious that "behind the scenes" an FAQ ticket is created. This might be required in special cases. If you do not want to have such an action "behind the scenes", please put the entire code into the workflow activities. If you write the script code directly into the workflow activity script, you can then use methods like `workflowApi.getTicketText()` and `workflowApi.addTicketText()`.

If you put the code into an Admin Tool script and call this script from the workflow script, you have to use the methods of the `TicketContentService` as shown in the example above.

D.14 Working with attachments

This chapter discusses the following:

D.14.1 Introduction 249

D.14.2 Example: Attaching all attachments of a ticket to the child ticket 249

D.14.1 Introduction

In ConSol CM, you can add attachments to tickets. These attachments can be of various file types and can be opened directly from the ticket history, provided the client machine has the correct application installed. For a detailed explanation about how to work with attachments, please refer to the *ConSol CM User Manual*.

Attachments can be integrated into CM in different process steps:

- An email with an attachment is sent to ConSol CM.
  - The email is appended to an existing ticket. The attachment is attached to the existing ticket.
  - A new ticket is created. The attachment is attached to the new ticket.
- An engineer attaches a file to the ticket using the Ticket Editor.
- An engineer uses CM/Doc and an attachment (MS Word or OpenOffice) is automatically created and attached to the ticket.

In the end, all of these attachments are ticket attachments of various file types. The Groovy class which is used for the respective objects is `com.consol.cmas.common.model.content.AttachmentEntry`.

> Please note that ...
>
> ... `workflowApi.workflowApi.getAttachmentList()` returns only the attachments which are directly attached to the ticket, i.e. which have been manually or automatically attached to the ticket. Email attachments are not included!
>
> ... `ticketContentService.getAttachmentEntries(ticket, ContentEntryCategory.values())` returns all ticket attachments.
>
> ... `ticketContentService.getAttachmentEntries(ticket, ContentEntryCategory.INCOMING_MAIL)` returns all ticket attachments which originated as email attachments in incoming emails. Other possible values are OUTGOING_MAIL or DEFAULT.

D.14.2 Example: Attaching all attachments of a ticket to the child ticket

The following code is taken from an Admin Tool script which is called from the workflow activity *Create Special Task (hand-over products list and attachments)*. From the Service Desk ticket, a new ticket in the Special Tasks queue is created. All attachments (or only the important ones, see alternative solution) are transferred to the new child ticket. The product list is also transferred.

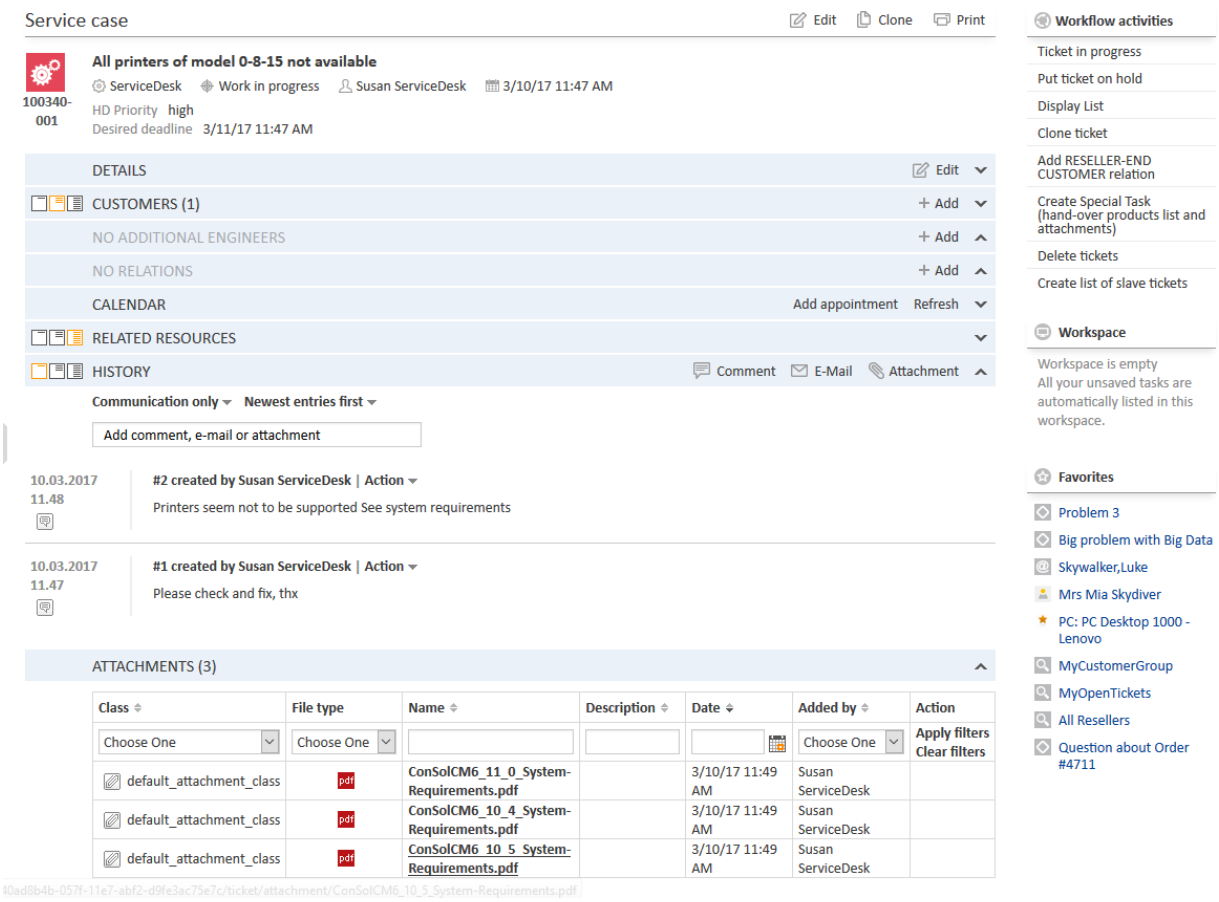


Figure 171: *ConSol CM Web Client - ServiceDesk ticket with three attachments*

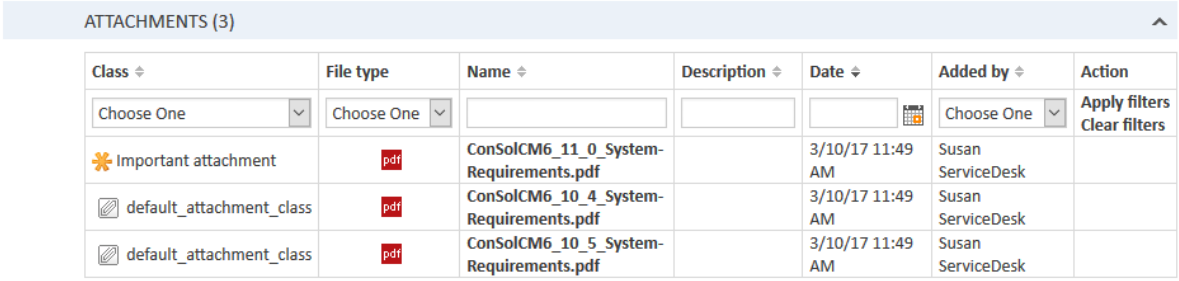


Figure 172: *ConSol CM Web Client - ServiceDesk ticket (only attachment section displayed) with three attachments, one marked as Important attachment using the respective class of text*

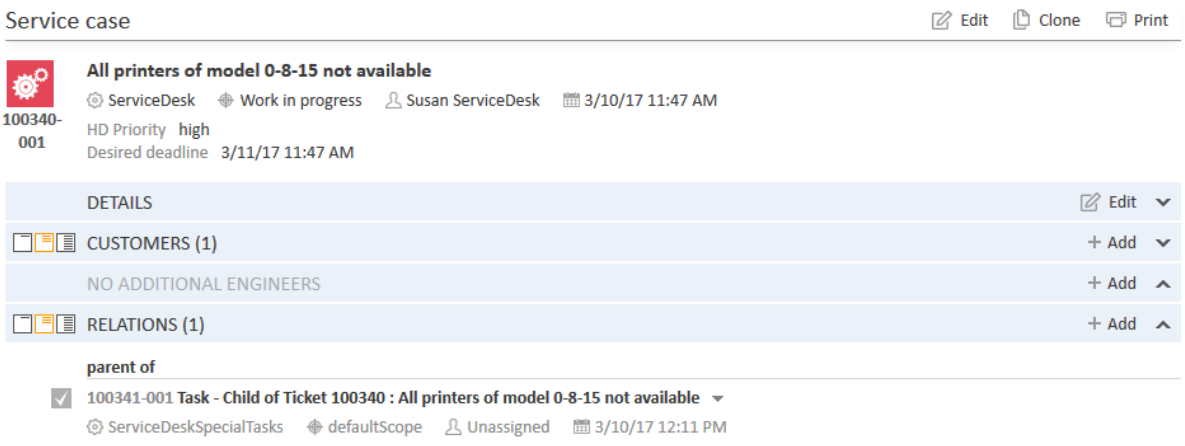


Figure 173: *ConSol CM Web Client - ServiceDesk ticket with newly created relation to new Special Tasks ticket*

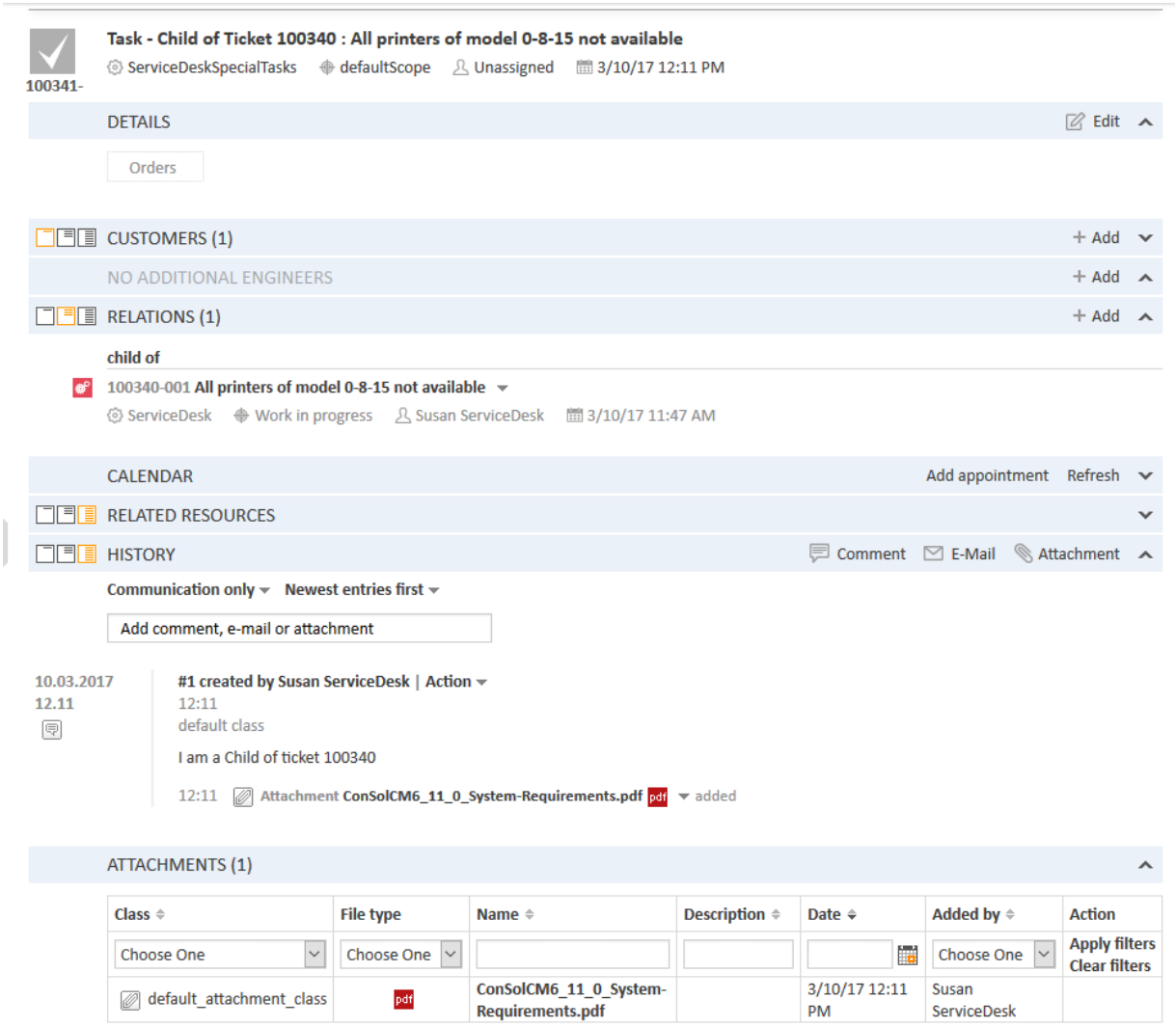


Figure 174: *ConSol CM Web Client - Child ticket with one attachment (which was marked as important in parent ticket)*

```
import com.consol.cmas.common.model.ticket.Ticket
import com.consol.cmas.common.service.*
import com.consol.cmas.common.model.content.AttachmentEntry
import com.consol.cmas.common.model.customfield.cfel.Struct
import com.consol.cmas.common.model.content.ContentEntryCategory
import com.consol.cmas.common.model.content.ContentFile
Ticket ticket = workflowApi.ticket
Ticket nt = new Ticket()
def qu = queueService.getByName("SpecialTasks")
nt.setQueue(qu)
def subj = "Task - Child of Ticket " + ticket.getId() + " : " + ticket.getSubject()
nt.setSubject(subj)
nt.setEngineer(null)
// main contact:
// def cont = ticket.getMainContact()
def cont = workflowApi.getPrimaryContact()
def text = "I am a Child of ticket " + ticket.getId()
def orders = ticket.get("order_data.orders_list")?.each() { ord ->
   nt.add("order_data.orders_list", new Struct().set("orders_hardware", ord.orders_
    hardware.getName())
   .set("orders_contact", ord.orders_contact)
   .set("orders_number", ord.orders_number)
   )
}

// put copy of each attachment to each child ticket
workflowApi.createChildTicket(nt, text,cont)

List<AttachmentEntry> attachmnts = ticketContentService.getAttachmentEntries
 (ticket, ContentEntryCategory.values())

attachmnts.each(){ at ->
   if ( at.file ) { // ignores deleted attachments
      def new_at = new AttachmentEntry()
      new_at.mimeType = at.file.mimeType ?: at.mimeType
      new_at.file = new ContentFile(at.file.name,new_
       at.mimeType,at.file.inputStream,at.file.size)
      new_at.description = at.description
      workflowApi.addAttachment(nt,new_at)
   }
}
```

Code example 85: *Admin Tool script called from a workflow activity: creating a child ticket and handing over the products list and all ticket attachments*

An attachment can have a class of text. If you want to transfer only the attachments of the class of text *Important attachment*, use the following code in the each loop:

```
attachmnts.each(){ at ->
   if ( at.file && at.contentEntryClass?.name?.equals("Important attachment")) { //
    ignores deleted attachments
      def new_at = new AttachmentEntry()
      new_at.mimeType = at.file.mimeType ?: at.mimeType
      new_at.file = new ContentFile(at.file.name,new_
       at.mimeType,at.file.inputStream,at.file.size)
      new_at.description = at.description
      workflowApi.addAttachment(nt,new_at)
   }
}
```

Code example 86: *Alternative (additional) solution: handing over only the important attachments*

D.15 Working with overlays

Overlays are one way of informing the engineer that some event has occurred concerning a ticket. As described in section Properties of an activity, subsection *overlay*, an overlay can be set using the Process Designer GUI, namely the properties editor of an activity. In numerous cases, it is sufficient to configure the removal of the overlay by using the overlay range. However, in some cases it might be required to set or remove an overlay using a workflow script. The most common example of this is the removal of the mail overlay when an email has been read. A detailed example of working with overlays for this use case is provided in section Example 2: Activity to acknowledge the email.

CM/ is a registered trademark of ConSol Consulting & Solutions Software GmbH

D.16 Searching for tickets, customers and resources

This chapter discusses the following:

D.16.1 Introduction 255

D.16.2 Searching for tickets 256

D.16.3 Searching for customers (contacts and companies) 262

D.16.4 Searching for resources 264

D.16.1 Introduction

Using the ConSol CM API you can search the database for tickets or for units (contacts and companies). If your CM system contains the module CM/Resource Pool, you can also search the database for resources. All search modes are based on the same principle:

1. A `criteria` object is created where all parameters for the target objects are stored.
    a. `TicketCriteria` for tickets
    b. `UnitCriteria` for contacts and companies
    c. `ResourceCriteria` for resources
2. This criteria object is handed over to a service which then returns a list with the result objects.
    a. `TicketService` for tickets
    b. `UnitService` for units
    c. `ResourceService` for resources

The fields which are set as parameters for the criteria objects have to be indexed, i.e. the annotation `field-indexed` has to be set.

D.16.2 Searching for tickets

To search for tickets you have to create the `TicketCriteria` object. For example, the following fields can be set.

- Date of ticket creation or closure
- Engineer
- System-specific ticket fields
- Ticket history criteria
- Ticket IDs
- Modification date
- Ticket name
- Pattern for the ticket subject
- Queue IDs
- IDs for current workflow scopes
- Current status (closed/open)
- Additional engineers

> Please see the ConSol CM API documentation for a complete list of available criteria.

The `TicketCriteria` object has to be handed over to the `TicketService` which is implicitly available as singleton `ticketService` in each script. Please see the following examples and refer to the *ConSol CM Workflow API Java* documentation for details about classes and methods.

D.16.2.1 Example 1: General example to search for tickets

```
def ticketCrit = new TicketCriteria()
ticketCrit.subject = "TICKET_SUBJECT"
ticketCrit.setQueueIds([new Long(workflowApi.getQueueByName("QUEUE_NAME").id)] as
 Set)
ticketCrit.setFields([new StringField(new FieldKey("FIELD_GROUP", "FIELD_NAME"),
 "SEARCH_VALUE")] as Set)
List<Ticket> foundTickets = ticketService getByCriteria(ticketCrit)
def firstTicket = foundTickets?.first()
```

Code example 87: *Search for tickets (pseudocode)*

D.16.2.2 Example 2: Find all tickets with the same module as the current ticket

The following example is taken from a workflow of a Service Desk environment. When the ticket has been created and the module has been set from a list, the workflow should check automatically if there are other open tickets with the same module. An ENUM is used for the module.

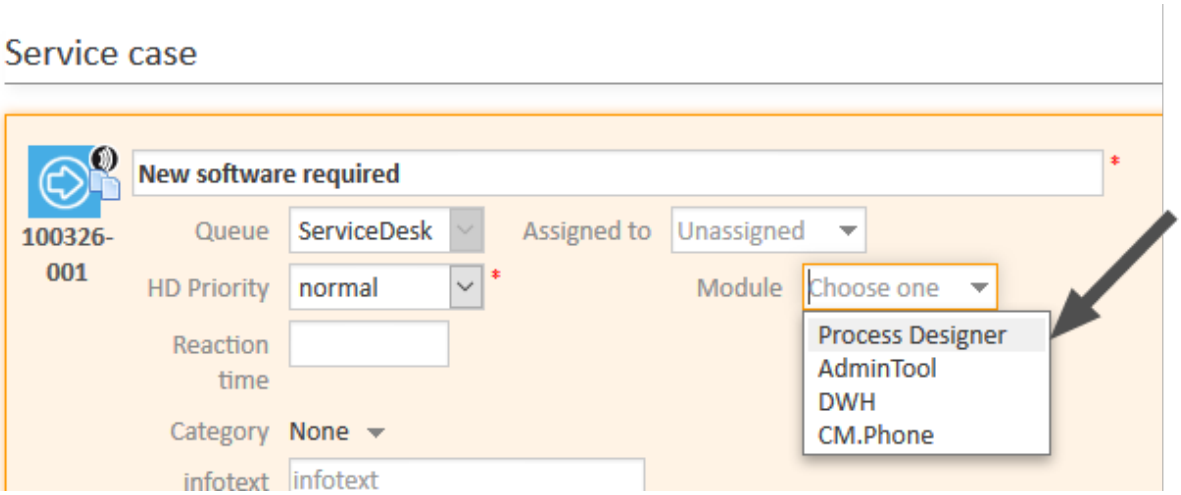


Figure 175: *ConSol CM Web Client - Selecting the module for a Service Desk ticket*

```
def mod = ticket.getField("helpdesk_standard", "module")
if (mod) {
   def crit = new TicketCriteria()

   //Find only tickets with same module:
   Set<AbstractField> myfields = [mod]
   crit.setFields(myfields)

   // Find only open tickets:
   crit.setStatus(TicketCriteria.Status.OPEN)

   //Find only tickets in same queue:
   def qu_id = ticket.queue.id
   crit.setQueueIds([qu_id] as Set)

   // Start search:
   List<Ticket> tics = ticketService.getByCriteria(crit)

   tics.each() { tic ->
      log.info 'Next Ticket subject is now ' + tic.subject
   }
} else {
   workflowApi.addValidationError(ticket.getField("helpdesk_standard", "module"),
    "Please provide a module, otherwise this activity cannot be performed.")
}
```

Code example 88: *Find tickets with the same module as the current ticket and display result in server.log file*

D.16.2.3 Example 3: Search for tickets by customer

In this example, we look for the *Account Management* ticket for a certain company.

```
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 Memberfield Key
def contactSearchFieldKey = new FieldKey("queue_fields","contacts_member")

// Create Unit Memberfield 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 prepared fields into TicketCriteria
crit.setFields([contactsListField] as Set)

// Search ... and Result
def foundTickets = ticketService.getByCriteria(crit)
println "Found tickets: ${foundTickets}"
if ( foundTickets ) {
   def AM_tic = foundTickets.first()
   def AM_tic_id = AM_tic.id
}
```

Code example 89: *Search for tickets by unit*

D.16.2.4 Example 4: Search for tickets by engineer to manage workload

In this example, an engineer can only call the workflow activity *New IT ticket (Accept ticket)* if he does not have too many tickets already. The maximum number of tickets which is allowed is stored in the custom-specific system property `custom-servicedesk, engineer.max.open.tickets`. In this way, the number can be changed by a CM administrator without a workflow developer being involved.

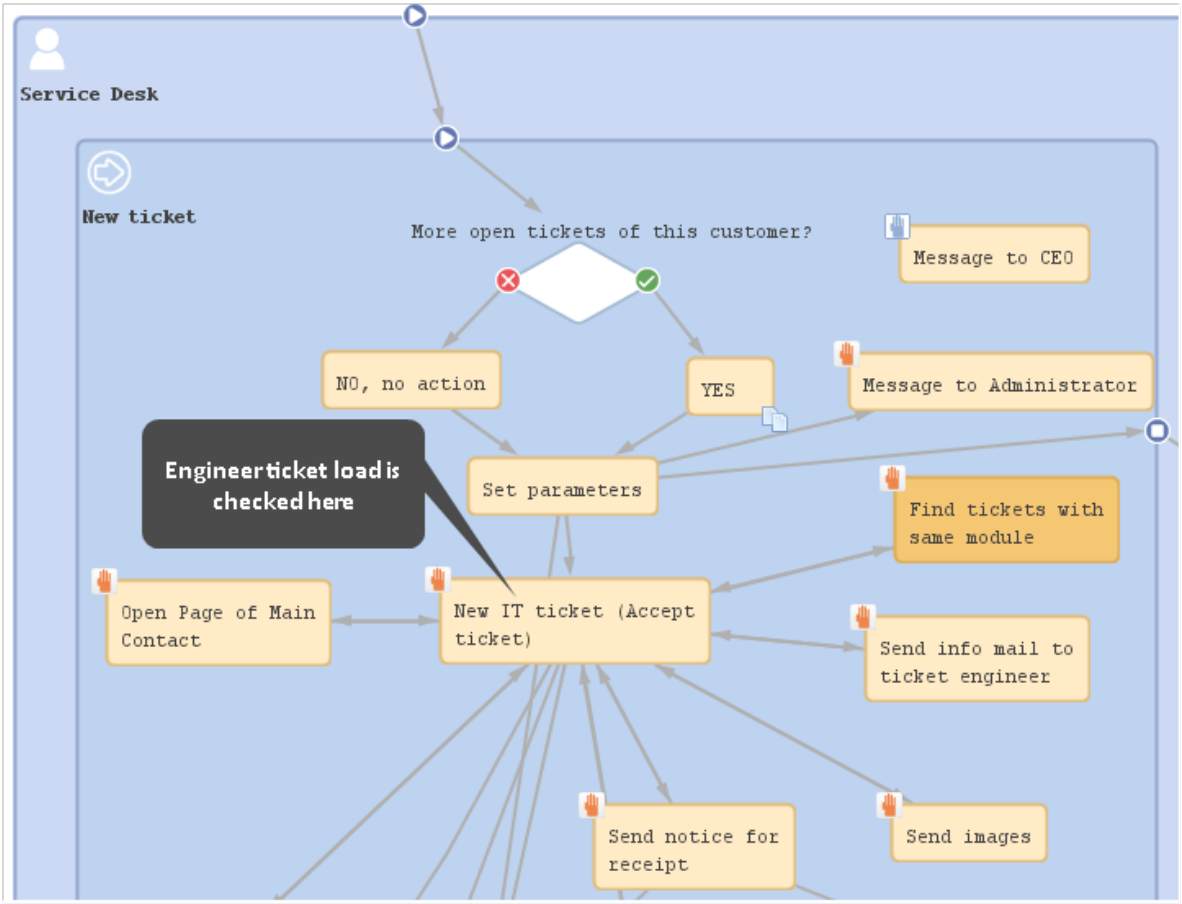


Figure 176: *Workflow activity which contains the ticket number control script*

```
// Engineer can only accept ticket if he does not have too many tickets already
def curr_eng = workflowApi.currentEngineer
def max_tics = configurationService.getValue("custom-
 servicedesk","engineer.max.open.tickets")

// look for open tickets of current engineer
def engs = []
engs.add(curr_eng.id)

TicketCriteria tic_crit = new TicketCriteria()
tic_crit.engineerCriteria = TicketCriteria.EngineerCriteria.assigned(engs as Set)
tic_crit.status = TicketCriteria.Status.OPEN

List<Ticket> open_eng_tics = ticketService.getByCriteria(tic_crit)

def tic_number = open_eng_tics.size
def loc = engineerService.currentLocale

if (tic_number > max_tics) {
   log.info 'Too many tickets for engineer ' + engineerService.current + '. Current
    number is ' + tic_number
   // get text from labels defined in AT:

   def infoText1 = messageProviderService.getMessage("info.wfl.toManyTicketsError1",
    loc)
   def infoText2 = messageProviderService.getMessage("info.wfl.toManyTicketsError2",
    loc) + max_tics
   // alternative solution: workflowApi.addValidationError("INFO","You have too many
    tickets (" + tic_number + ") already, so you cannot accept another ticket.
    Maximum allowed number is " + max_tics)
   workflowApi.addValidationError("INFO",infoText1 + " " + infoText2)

} else {
   ticket.setEngineer(curr_eng)
}
workflowApi.reinitializeTrigger("defaultScope/Service_
 Desk/TimeTriggerDesiredDeadline")
```

Code example 90: *Script of activity "New IT ticket (Accept ticket)", engineer workload is checked*

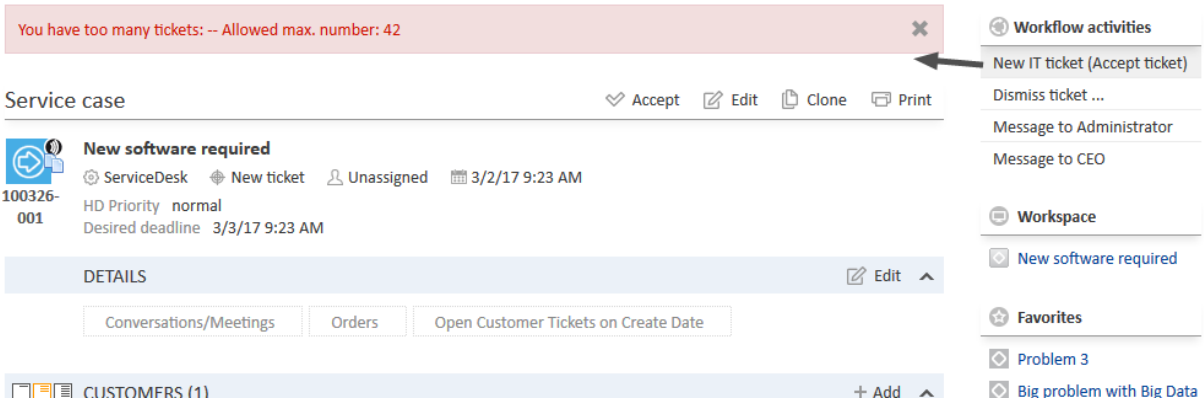


Figure 177: *ConSol CM Web Client - Control of the number of tickets which can be assigned to one engineer*

D.16.3 Searching for customers (contacts and companies)

To search for customers (i.e. for contacts and/or companies) you have to create the `UnitCriteria` object. For example, the following fields can be set:

- Customer group
- System-specific customer fields
- Unit IDs
- Patterns for units
- Phone number (used for CM/Phone)
- TicketCriteria
- UnitDefinition names
- Boolean UseInCriterion

Then you use the `unitService` to get the search result.

> Please see the ConSol CM API documentation for a complete list of available criteria.

D.16.3.1 General syntax for customer search by enum value

```
import com.consol.cmas.common.model.customfield.UnitCriteria
import com.consol.cmas.common.model.customfield.EnumSearchField
import com.consol.cmas.common.model.customfield.meta.FieldKey

def unitCrit = new UnitCriteria()
def companyEnumField = new EnumSearchField(new FieldKey("customer", "company"),
 [enumService.getValueByName("ENUM_GROUP_NAME",ENUM_VALUE_NAME)] as Set)
unitCrit.setFields([companyEnumField] as Set)
unitService.getByCriteria(unitCrit).each { foundContact ->
   println "Processing found contact: "+foundContact.get("name")
}
```

Code example 91: *Search for customers by enum value (general syntax)*

D.16.3.2 Example 1: Search for contacts by first name and last name

```
def unitCrit = new UnitCriteria()
unitCrit.setFields([new StringField(new FieldKey("UNIT_GROUP_NAME", "firstname"),
 "Max"),
new StringField(new FieldKey("UNIT_GROUP_NAME", "lastname"), "Mustermann")] as Set)
def foundContacts = unitService.getByCriteria(unitCrit)
def firstContact = foundContacts?.first()
```

Code example 92: *Search for contacts by first name and last name*

D.16.3.3 Example 2: Search for customers by enum value

```
def unitCrit = new UnitCriteria()

//all other UnitCriteria init operations skipped

// this is the requested value inside the list:
def secLvl = ticket.get("transportEntryData.securityLevel")

//ShipperData/securityLevel is the path of the EnumField inside the list
def secLvlEnumFieldKey = new FieldKey("ShipperData","securityLevel")

//create the template field with FieldKey and our value to search for
def secLvlTemplateField = new EnumField(secLvlEnumFieldKey, secLvl)

//ShipperData/securityLevels is the path of the list itself
def secLvlListTemplateFieldKey = new FieldKey("ShipperData","securityLevels")

//init the template list with the value to be searched for
def secLvlListTemplateField = new ListField(secLvlListTemplateFieldKey,
 [secLvlTemplateField])

// put the template list into the UnitCriteria object
def unitCrit.setFields([secLvlListTemplateField] as Set)

// Search ... and Result
def shippers = unitService.getByCriteria(unitCrit)
```

Code example 93: *Search for customers by enum value (example)*

D.16.4 Searching for resources

To search for resources, you have to create the `ResourceCriteria` object. For example, the following fields can be set:

- Date range
- Excluded resource IDs
- Resource category IDs of the desired resources

Then you use the `resourceService` to get the search result.

> Please see the ConSol CM API documentation for a complete list of available criteria.

D.16.4.1 Example 1: Writing an IT inventory list as a comment into a ticket

In the following example, a list of all IT assets, which are represented as resources of certain resource types, is written into the current ticket.

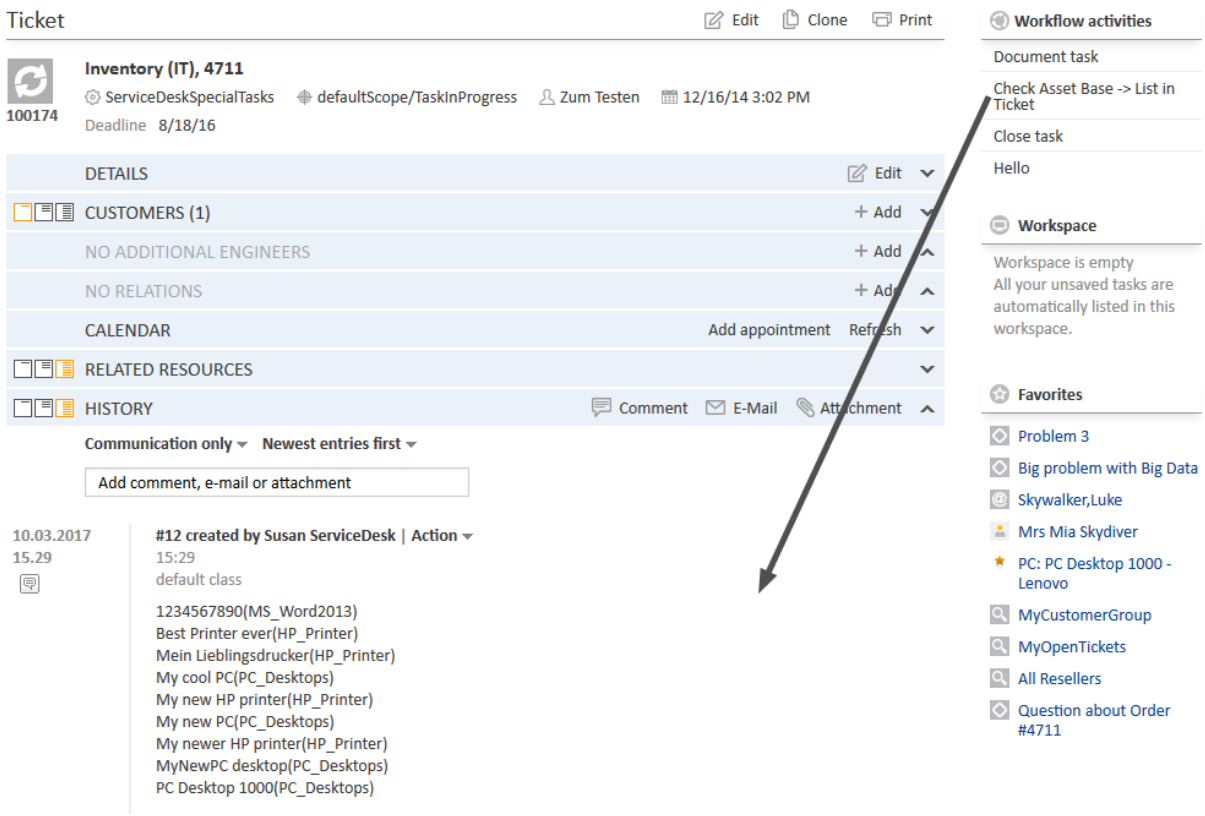


Figure 178: *ConSol CM Web Client - Calling a workflow activity which writes the IT asset list as comment into the ticket (script: see following code example)*

```
// Make an inventory of the asset base (CM/Resource Pool) and write the info into
 the current ticket
// Asset base contains Hardware and Software resources

import com.consol.cmas.common.model.resource.*
import java.util.Arrays
import com.consol.cmas.common.model.ticket.Ticket
import com.consol.cmas.common.model.content.TextEntry

ticket = workflowApi.getTicket()
def crit = new ResourceCriteria()

// only three resource groups are required:
Set<String> res_groups = ["Printers","OfficeSoftware","PCs"]
crit.setResourceGroupsTechnicalNames(res_groups)

List<Resource> res_list = resourceService.getByCriteria(crit)
def desc_name def printout_list = []

res_list.each(){res ->
   def res_tname = res.getResourceType().getName()
   // log.info 'Resource type is ' + res_tname
   // find the field which is used for description of a single resource
   // this depends on the resource fields of the resource field group
   switch (res_tname){
      case "HP_Printer": desc_name = "HP_Printer_Fields_basic.name"
      break;
      case "MS_Word2013": desc_name = "MS_Word2013_Fields.OrderNumber"
      break;
      case "PC_Desktops": desc_name = "PC_Desktop_Fields_basic.name"
      break;
      case "PC_Laptops": desc_name = "PC_Laptop_Fields_basic.laptopname"
      break;
   }

   def res_final = res.get(desc_name) + '(' + res_tname + ')<br>'
   printout_list += res_final
}
printout_list = printout_list.sort{it}

TextEntry new_te = new TextEntry("Asset Base contains the following
 assets",printout_list.toString()
.replace('[','')
.replace(']','')
.replace(',',''))

ticketContentService.createContentEntry(ticket,new_te)
```

Code example 94: *Print an IT asset (resource) list into the ticket*

D.17 Displaying messages and notifications

Sometimes it might be required to display messages or notifications to the engineer who is currently working with ConSol CM. You can use messages for information concerning the process or the changes which the engineer makes while editing an object. To inform the engineer about events which are not related to the current object, you can create a system notification.

In the following sections, you will learn how to implement messages and notifications for the Web Client.

D.17.1 Messages

The most important method for displaying messages in the Web Client is `addValidationError()`, in the object `workflowApi`. It can be used in workflow scripts or in scripts of type *Workflow* which are called from the workflow but stored in the Admin Tool.

> Alternatively to the methods described in this manual, error messages can also be implemented using the Action Framework. Please see the *ConSol CM Administrator Manual* for further information.

The place where the messages are displayed depends on the parameters passed to the method. There are two options:

- Use `workflowApi.addValidationError("INFO","message text")` to display the message in a red box above the ticket.
- Use `workflowApi.addValidationError(ticket.getField("group name", "field name")"message text")` to display a red message next to the affected field in an ACF.

D.17.1.1 Displaying a simple error message

In the following example, a simple message is displayed when the engineer clicks a workflow activity to close a ticket which still has open child tickets. Obviously, the text is not localized and therefore cannot be used in multi-language environments.

```
// only when all child tickets are closed, the parent ticket (this ticket) can be
 closed
import com.consol.cmas.common.model.ticket.Ticket

Ticket ticket = workflowApi.getTicket()
List<Ticket> ch_tickets = workflowApi.getChildTickets()

Boolean all_closed = true

ch_tickets.each() { ch_tic ->
   Boolean closed = ch_tic.getScopeInfo().isClosedOrDeleted()
   log.info 'Child Ticket {$ch_tic.id} is ' + closed
   if (closed == false) {
      all_closed = false
   }
}

def text = "Please close all child tickets first. Then you can close the ticket";

if (!all_closed) {
   workflowApi.addValidationError("INFO",text)
} else {
   log.info "Ticket ${ticket.id} was closed."
}
```

Code example 95: *Checking if all child tickets are closed and displaying a message for the engineer, non-localized version*

D.17.1.2 Displaying a localized message using labels

In order to localize the text which is displayed in the Web Client, you can work with labels. Labels are defined in the Admin Tool, navigation group *Global Configuration*, navigation item *Labels*. A detailed introduction to labels is provided in the *ConSol CM Administrator Manual*.

In the following example, an error message is displayed when the engineer clicks a workflow activity to accept the ticket although he is assigned to too many open tickets already. The text which is displayed on the GUI is composed of two components:

- text which is retrieved from a label, here using the `MessageProviderService`
- a variable which is filled dynamically in the context of the script, here `max_tics` which is actually retrieved from a CM system property

As a comment, the alternative solution with a simple text message is shown.

```
// Engineer can only accept ticket if he does not have too many tickets already
def curr_eng = workflowApi.currentEngineer
def max_tics = configurationService.getValue("custom-
 servicedesk","engineer.max.open.tickets")

// look for open tickets of current engineer
def engs = []
engs.add(curr_eng.id)

TicketCriteria tic_crit = new TicketCriteria()
tic_crit.engineerCriteria = TicketCriteria.EngineerCriteria.assigned(engs as Set)
tic_crit.status = TicketCriteria.Status.OPEN

List<Ticket> open_eng_tics = ticketService.getByCriteria(tic_crit)

def tic_number = open_eng_tics.size
def loc = engineerService.currentLocale

if (tic_number > max_tics) {
   log.info 'Too many tickets for engineer ' + engineerService.current + '. Current
    number is ' + tic_number
   // get text from labels defined in AT:

   def infoText1 = messageProviderService.getMessage("info.wfl.toManyTicketsError1",
    loc)
   def infoText2 = messageProviderService.getMessage("info.wfl.toManyTicketsError2",
    loc) + max_tics
   // alternative solution: workflowApi.addValidationError("INFO","You have too many
    tickets (" + tic_number + ") already, so you cannot accept another ticket.
    Maximum allowed number is " + max_tics)
   workflowApi.addValidationError("INFO",infoText1 + " " + infoText2)

} else {
   ticket.setEngineer(curr_eng)
}
workflowApi.reinitializeTrigger("defaultScope/Service_
 Desk/TimeTriggerDesiredDeadline")
```

Code example 96: *Script of activity "New IT ticket (Accept ticket)", engineer workload is checked*

D.17.1.3 Displaying a localized message without using labels

You can also work with localized values by simply checking the locale as shown in the following example. The major difference between this and the previous example is that the labels can easily be modified using the Admin Tool, thereby being accessible by CM administrators. In contrast, in the solution demonstrated in the current section, the text which is displayed can only be changed by deploying a new version of the workflow. You can decide for yourself which solution might be the best for your requirements.

```
// only when all child tickets are closed, the parent ticket (this ticket) can be
 closed
import com.consol.cmas.common.model.ticket.Ticket

Ticket ticket = workflowApi.getTicket()
List<Ticket> ch_tickets = workflowApi.getChildTickets()

Boolean all_closed = true

ch_tickets.each() { ch_tic ->
   Boolean closed = ch_tic.getScopeInfo().isClosedOrDeleted()
   log.info 'Child Ticket {$ch_tic.id} is ' + closed
   if (closed == false) {
      all_closed = false
   }
}

def text
def mylocale = engineerService.getCurrentLocale().toString()
log.info "LOCALE IS NOW $mylocale"

if (mylocale.startsWith('en')) {
   text = "Please close all child tickets first. Then you can close the ticket";
} else if (mylocale.startsWith('de')) {
   text= "Bitte erst alle Child Tickets abschließen. Dann können Sie das Ticket
    abschließen"
} else { //default
   text = "Please close child tickets."
}
if (!all_closed) {
   workflowApi.addValidationError("INFO",text)
} else {
   println "Ticket ${ticket.id} was closed."
}
```

Code example 97: *Checking if all child tickets are closed and displaying a message for the engineer*

D.17.1.4 Displaying a field-specific message in an ACF

It is also possible to display messages which refer to single fields when filling in ACFs. In this way, you can check if a data field has been filled in correctly and display a message if some value is missing or wrong.

The following example shows the general syntax used for this kind of validation messages. If a ticket field is provided, the message is displayed within the ACF next to this field.

```
workflowApi.addValidationError(ticket.getField("GROUP_NAME", "FIELD_NAME"), "Value
 is not valid! Please enter a new one.")
```

D.17.2 Notifications

Notifications are displayed when clicking the bell icon in the header of the Web Client, next to the link to the engineer profile. If the user allows notifications in the browser, they can also appear in the Windows system tray.

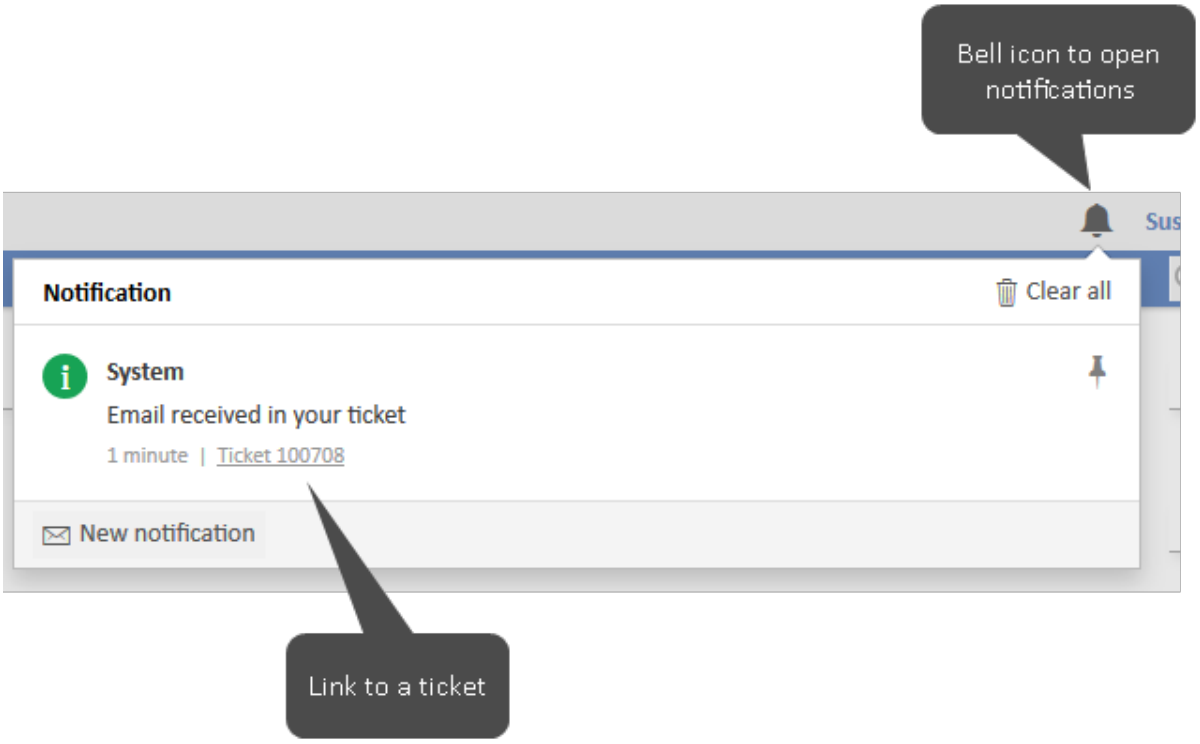


Figure 179: *ConSol CM Web Client - Notification with a ticket link*

D.17.2.1 Example: Sending a notification when an email is received

When an email is received for a ticket, you can add an overlay to the ticket icon and send a notification to the assigned engineer. For this purpose, you only need to add an email trigger connected to an automatic activity, where the overlay is added and the notification is sent. The notification can include a link to the ticket.

The script for this activity could look as follows:

```
def myEngineer = workflowApi.engineer.name
def myTicket = workflowApi.ticket.name
if (myEngineer) {
   broadcasterService.notificationBuilder().
   withSenderName("System").
   withMessage("Email received in your ticket").
   withNature("INFO").
   withOrigin(myTicket).
   withOriginContext("TICKET").
   addRecipientName(myEngineer).
   build().
   send();
}
```

Code example 98: *Workflow script to send a notification*

D.18 Debug information

This chapter discusses the following:

D.18.1 Introduction 271

D.18.2 Log file configuration 271

D.18.3 Using statements for debug output 272

D.18.1 Introduction

D.18.1.1 Relevant log files

A detailed description of all ConSol CM log files is provided in the *ConSol CM Administrator Manual* and the *ConSol CM Operations Manual*. Please refer to one of those manuals if you want to know more about CM log files. In the current section, only a short introduction is provided to help you work with the log files which are relevant for the Process Designer.

Log file path

In ConSol CM, the default configuration for the log file path places all log files in the following directory:

- **In JBoss 7 (single instance):**
  <JBOSS_HOME>/standalone/server.log
- **In Oracle WebLogic:**
  <DOMAIN_HOME>/cm-logs and
  <DOMAIN_HOME>/cmrf-logs/server.log

server.log

In ConSol CM, the standard output usually is written to `server.log`. You can write information into this file using, e.g., the `log.info()` method in your scripts, see section Debug output to the server.log file

audit.log

Operations which are performed using the Admin Tool or using the Process Designer are logged in the `audit.log` file. For example, login/logout to the Process Designer and deployment of workflows is logged.

D.18.2 Log file configuration

The logging configuration can be changed by editing the respective configuration file:

- In JBoss:
  <JBOSS_HOME>/standalone/configuration/cm6.xml or cm6-cmrf.xml
- In Weblogic:
  <WLS_HOME>user_projects\domains\consolcm6_domain\log4j.xml.

A comprehensive documentation of logging and log files in ConSol CM is provided in the *ConSol CM Setup Manual*.

As an alternative, you can write information into the ticket as text.

D.18.3 Using statements for debug output

D.18.3.1 Debug output to the server.log file

The following statements can be used to write log information to the `server.log` file. This works in workflow scripts as well as in Admin Tool scripts.

```
println 'This is my debug message.'
```

```
println("This is my debug message.")
```

```
log.info("This is my debug message.")
```

```
log.info "This is my debug message."
```

In a WebLogic system, usually the `log.info` statement has to be used. The `println` might not work.

Please note that we recommend to work with the `log` methods instead of `println`, because

- the log level can be defined. Besides `log.info`, you might want to use `log.warn`, `log.error` or other levels.
- the name of the script which produces the entry is written into the log file. This does not happen if you work with `println`.

D.18.3.2 Debug output as a text entry in a ticket

If you would like to display the information to the ticket (e.g. because you do not have access to the file system where the log files are stored) you can write the text into the ticket as regular comment:

```
workflowApi.addTicketText('This is my debug message', 'This is the subject of my
 debug message', false)
```

D.18.3.3 Debugging ConSol CM standard scripts

In ConSol CM standard scripts, e.g. `createTicket.groovy`, you will find statements similar to the following:

```
if (log.isDebugEnabled()) {
   log.debug("Extracted email from from-field is $email")
}
```

Code example 99: *Debug entry in ConSol CM standard email script*

To activate the debug output, i.e. to have CM write the debug information into the log file, you have to set the log level of the respective module (here: email) to `DEBUG`. This is done in the file `jboss-log4j.xml`.

We will not elaborate on this topic here. If you would like to learn more about CM logging, please refer to the *ConSol CM Operations Manual*.

E - Workflow deployment

This chapter discusses the following:

E.1 Introduction and workflow life cycle 275

E.2 Engineer permissions required for workflow deployment 276

E.3 Actions during workflow deployment 277

E.1 Introduction and workflow life cycle

During the development of a workflow you use the following functions which reflect the workflow life cycle:

- Click the *Load ...* button to load the workflow or create a new workflow, e.g., version 1.2.
- Edit the workflow.
- Click the *Save as new version* button to save the workflow as a new version. A new version number will be used, e.g., 2.0.
- Continue editing the workflow.
- Click the *Save ...* button to save the workflow in the current version, e.g., version 2.0.
- Continue editing the workflow.
- Click the *Deploy* button to deploy the workflow. This will save and deploy the workflow, e.g., version 3.0.
  A deployed workflow always has an increased major number compared to the last saved version.
  The workflow which was active/deployed before is now no longer active, but the new version of the workflow is in operation at once. The ConSol CM system does not have to be stopped.
  The new version is marked in bold characters and with status *currently deployed* in the workflow list which is opened for the *Load* and *Delete* operations.
  After this step, the next saved version will be saved as *new version*.

> ⚠ Make sure you are aware of the number of tickets which have to be transferred when a new workflow is deployed! The deploy operation might take some time in large environments!
> See section Actions during workflow deployment.

E.2 Engineer permissions required for workflow deployment

An engineer who is supposed to deploy workflows must have at least one role with one of the following access rights:

- **Global Permissions:**
  Administrate system (full access) or *Administrate system (configuration only)*
- **Workflow Permissions:**
  Deploy workflow

E.3 Actions during workflow deployment

When a workflow is deployed, it will be active at once. Thus, consider well what will happen to open tickets which are in a queue where the new workflow will be applied. They will be transferred to the new workflow.

In case you have performed one or more of the following steps:

- removed one or more activities
- added one or more automatic activities
- added one or more triggers

the following actions will be initiated after you have pressed the *Deploy* button.

You will be prompted for a decision concerning the open tickets in the respective queues which cannot stay at their previous position within the process because the workflow architecture was changed:

- Stay as close as possible to the previous position (default).
- Let all those tickets start the process from the beginning.

In case you choose the first option (*keep position*), the following actions will be performed:

1. The transfer of tickets starts.
2. The name of the ticket's last executed activity is compared to the names in the current workflow definition. If the ticket's activity is no longer in the workflow definition, a new target activity for the ticket must be found.
3. The history for the ticket is loaded. The transfer engine iterates over all activities executed from the beginning of the process instance and tries to find one which would be suitable, i.e., which
   a. is still present in the workflow definition,
   b. is not a trigger target element,
   c. is not a dead end activity.

Each ticket which cannot keep its position will be moved to the suitable position according to those criteria. In any case the tickets will be moved backwards, never forwards, within the workflow.

For a summary of all ticket transfers click on *View* in the main menu and select *Show ticket transfer history*:

- **Workflow name**
  Name of the workflow.
- **Version**
  Version of the old workflow.
- **Start time**
  Start of the transfer. Will be the start time of the *Deploy* operation.
- **End time**
  End of the transfer. After this time the new workflow will be in full operation.
- **Transferred tickets**
  Number of tickets which have been transferred, i.e., which had to be touched by the system during workflow deployment. Should be identical to the sum of open tickets in all queues which use the workflow.
- **Details**
  Additional information concerning the deployment with ticket transfer.

In the bottom right corner of the Process Designer GUI, the overall status of the ticket transfer is displayed.

CM/ is a registered trademark of ConSol Consulting & Solutions Software GmbH

F - Best practices

This chapter discusses the following:

F.1 How to use scopes for organizing the workflow 280
F.1.1 Variant A: Using a global scope 281
F.1.2 Variant B: Using three or more main scopes 283
F.2 Where to position the start node 285
F.3 Where to save scripts 286
F.3.1 When to use Admin Tool workflow scripts 287
F.3.2 How to use Admin Tool workflow scripts 288
F.4 How to optimize trigger combinations 289
F.5 How to avoid unneeded ticket update events 292
F.6 How to use the disable auto update parameter 293
F.7 How to avoid self-triggering business event triggers 295
F.8 Which components to use with time triggers 296

CM/ is a registered trademark of ConSol Consulting & Solutions Software GmbH

F.1 How to use scopes for organizing the workflow

One of the first things you have to consider, when you start making a concept for a workflow, is the number and organization of scopes. If you would like to refresh your knowledge about scopes, please refer to the introduction to scopes.

> Of course you can always modify the workflow in later steps, but this might have implications for existing tickets, views, and reports. This is particularly significant if the workflow is used in a production environment.

Consider the following points when setting up the basic structure of a workflow:

- Which trigger should be active for the ticket in which states of the process?
  For example, should a time trigger, which monitors the new tickets, also be active for tickets which are already in progress? Or, should a mail trigger be active when the ticket has been finished by the engineer?
- Which views are required?
  Views are based on the position of tickets in scopes, see *ConSol CM Administrator Manual* section *View Administration* for details.

F.1.1 Variant A: Using a global scope

A global scope is a scope which contains all other scopes of the workflow. You might want to use such a global scope because some processes require reactions to events during the entire process. Those events are implemented using triggers which are attached to the global scope. For example, if you want to supervise for the entire process, if an email has been received, you attach a mail trigger (see section Mail triggers) to the global scope. All sub-scopes of the global scope inherit the sensitivity to this trigger. If the email should only be monitored for a sub-scope, you can attach the mail trigger to this sub-scope.

The same applies to all kinds of triggers, i.e., business event triggers (see section Business event triggers) and time triggers (see section Time triggers).

The START node always has to be positioned outside the global scope!

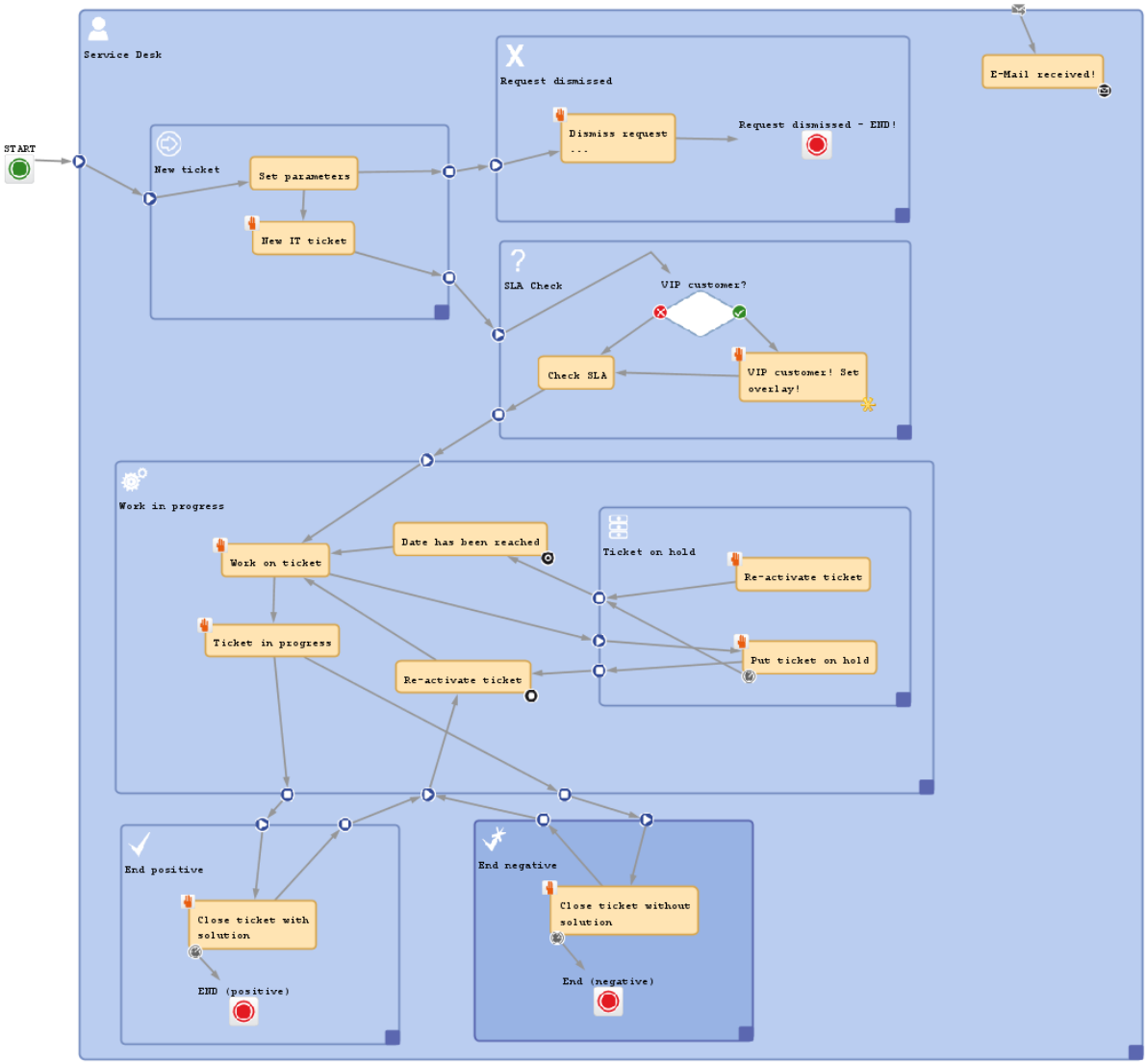


Figure 180: *ConSol CM Process Designer - Workflow with global scope*

Please keep in mind that you can always use triggers in inner scopes which will then consume the event (see section Firing order of business event triggers in hierarchical scopes as an example for busi-

ness event triggers). For example, if you would like to use a mail trigger in the entire process in the global scope but you need a certain reaction of the ticket in the *Finished* scope, you can use a mail trigger which is attached to the *Finished* scope.

CM/ is a registered trademark of ConSol Consulting & Solutions Software GmbH

F.1.2 Variant B: Using three or more main scopes

An alternative way to construct a workflow is to use three or more main scopes:

- New tickets
- In progress (only here, a mail trigger is applied)
- Closed tickets (in one or more separate scopes)

The following picture shows an example for a workflow which has been built according to this principle.

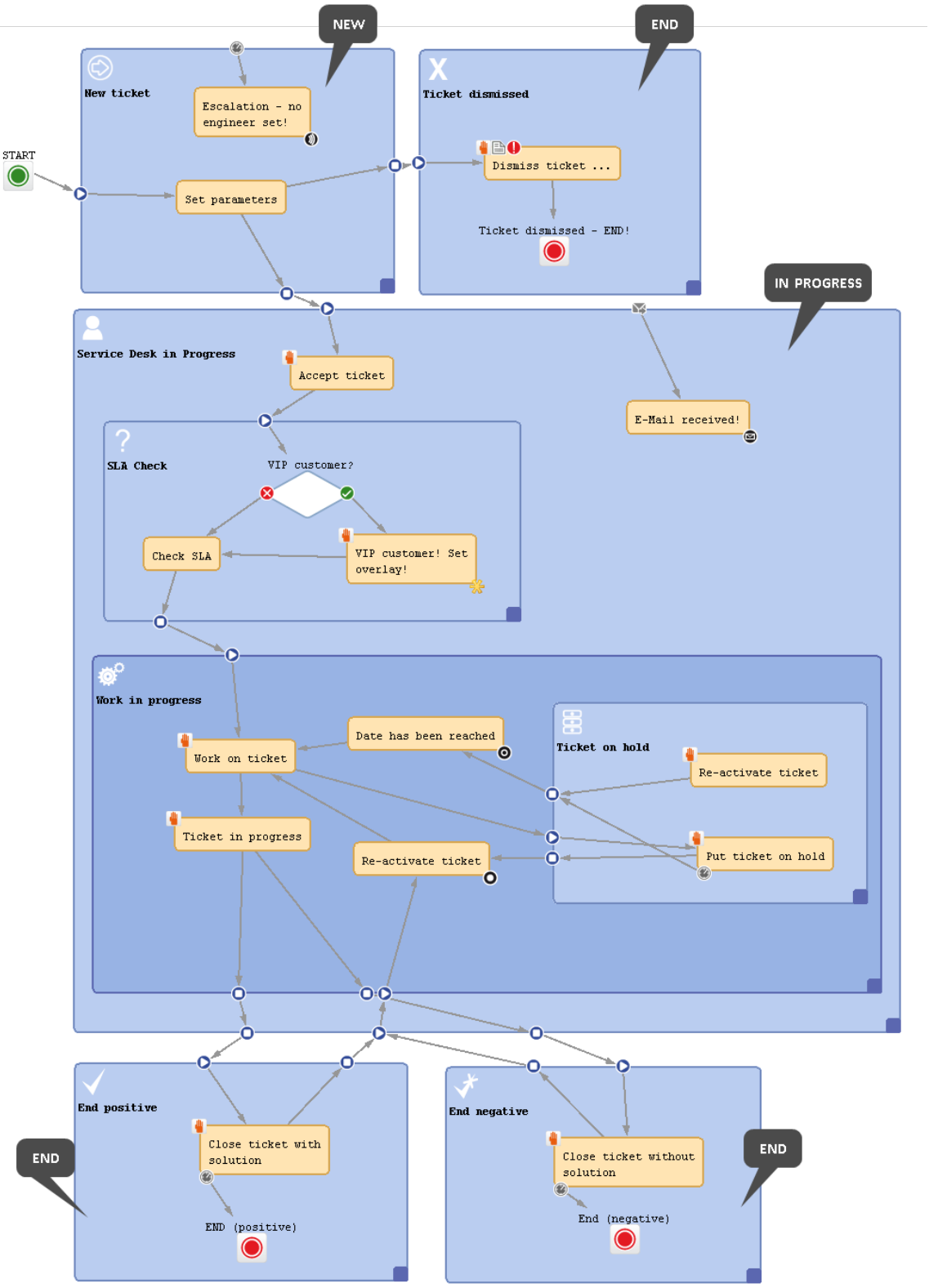


Figure 181: *ConSol CM Process Designer - Workflow with three types of main scopes*

F.2 Where to position the start node

The best position of the start node (see section Start nodes) depends on the use of triggers in the following scope. If time triggers are used in the first scope, where tickets are forwarded after the start node, the start node should be placed outside the scope. In case the start node is placed inside the first scope, the time trigger might not be initialized correctly. So place the start node in the default scope.

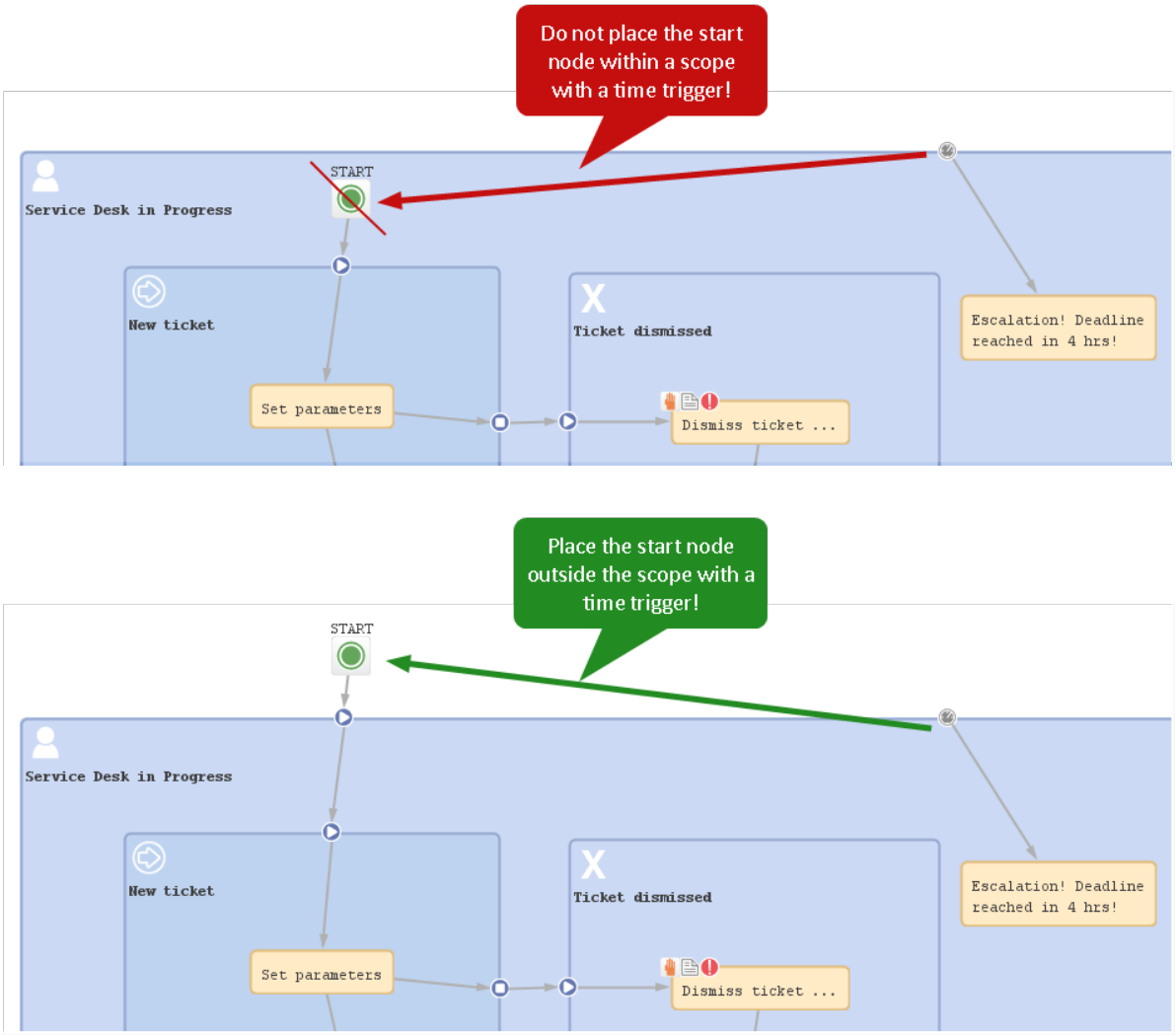


Figure 182: *ConSol CM Process Designer - Position of start node*

F.3 Where to save scripts

For scripts, which are used over and over again in workflow activity and/or precondition scripts, it might be better to store them (in our example: `myscript.groovy`) in the *Script* section of the Admin Tool and call them from the workflow script.

Use the following syntax in the workflow script (see also the example in the following section):

```
scriptExecutionService.execute("myscript.groovy")
```

F.3.1 When to use Admin Tool workflow scripts

We would neither recommend to always use this method nor would we advise against it. We will illustrate the advantages and disadvantages of this approach and you can then decide for yourself where in your system you want to apply it.

The **advantages** of storing workflow scripts in the Admin Tool are the following:

- The script is stored only once and has to be maintained / changed at only one place.
- Changes of the scripts are executed in the system just in-time, no deployment (as for workflows) is required.

The **disadvantages** of storing workflow scripts in the Admin Tool are the following:

- The process logic is stored at two separate places, i.e., you always have to work with the Process Designer as well as with the Admin Tool to see the entire process.
- Most objects have to be imported into Admin Tool scripts, because they are not present implicitly.
- A workflow export alone is not sufficient to move the workflow, because scripts in the Admin Tool are not included in the export.

F.3.2 How to use Admin Tool workflow scripts

Admin Tool scripts which are used in the workflow have to be of type *Workflow*. An Admin Tool script usually is called from the workflow using the `ScriptExecutionService`.

```
scriptExecutionService.execute("myscript.groovy")
```

Code example 100: *Calling an Admin Tool script from the workflow*

```
// Define a HashMap with the key-value pairs which you would like to pass to the
 Admin Tool
def params = [ "templateName": "newCustomer" ]

// Execute the script. The passed parameters are available in the Admin Tool
 script. In the
// example, the variable templateName does not have to be defined in the Admin Tool
 script
// but it is present based on the definition in the passed HashMap.
// The variable r will contain the return value of the script or Null if there is
 no return
// value
def r = scriptExecutionService.execute("myscript.groovy", params)
```

Code example 101: *Calling an Admin Tool script from the workflow with use of parameters*

> Since the object `workflowApi` (see section workflowAPI) is not available in the Admin Tool scripts, you will have to find other classes with methods which you can use instead of the methods of `workflowApi`.

F.4 How to optimize trigger combinations

> ⚠ Beware of unnecessary trigger executions! They will consume resources and slow down application performance.

Example 1:
This example shows many business event triggers in one big **global** scope.

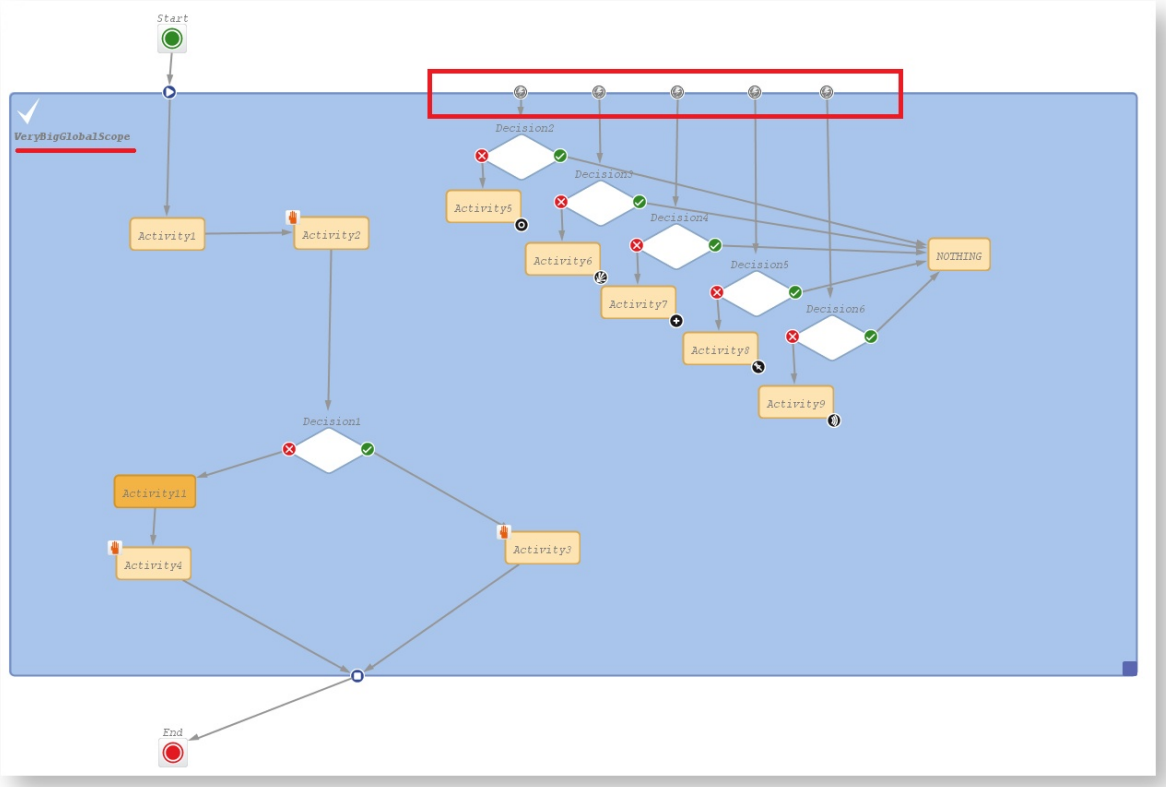


Figure 183: *ConSol CM Process Designer - Scope with triggers*

Example 2:
If it is possible, please use triggers in the smallest scope possible (in this example, the trigger with *Decision6* was moved to a smaller scope).

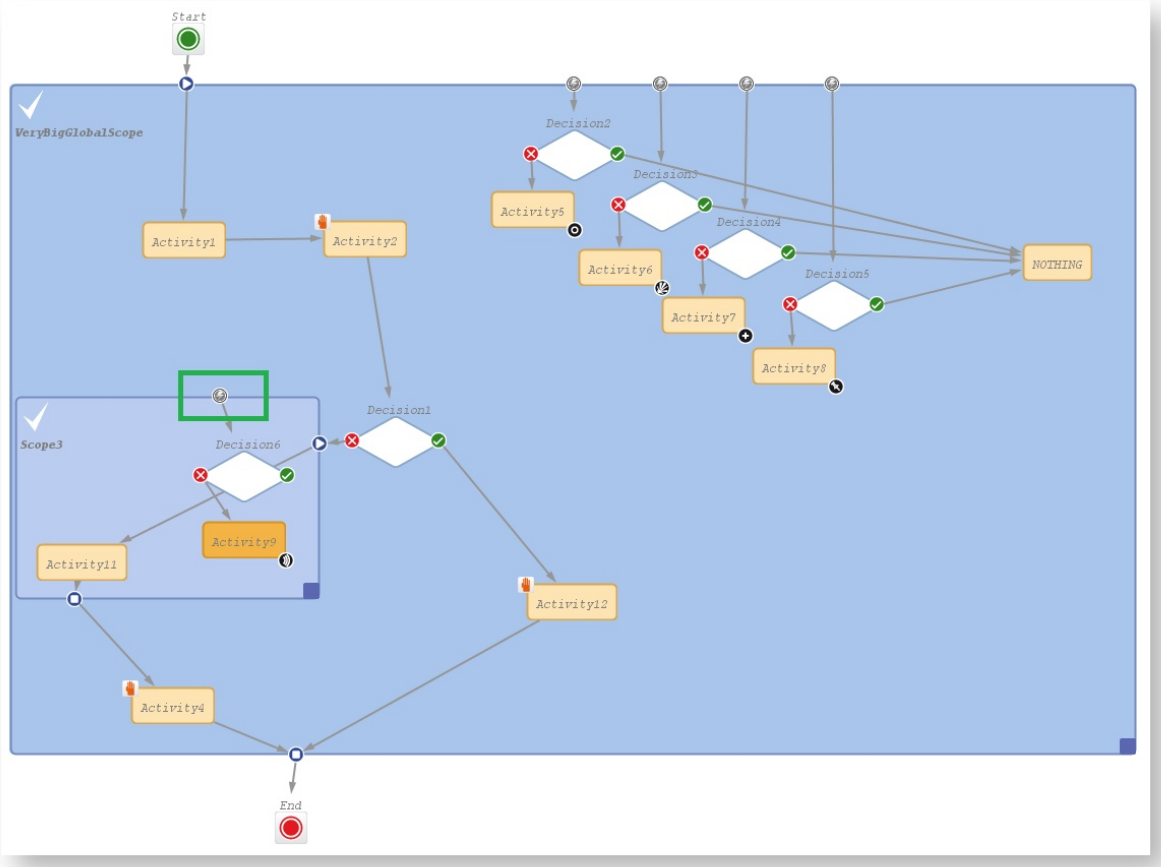


Figure 184: *ConSol CM Process Designer - Move trigger to smaller scope*

Example 3:
If it is **not** possible to move triggers to smaller scopes and you do not want to call all of the triggers while executing some activity, move this activity to an outside scope without any triggers.

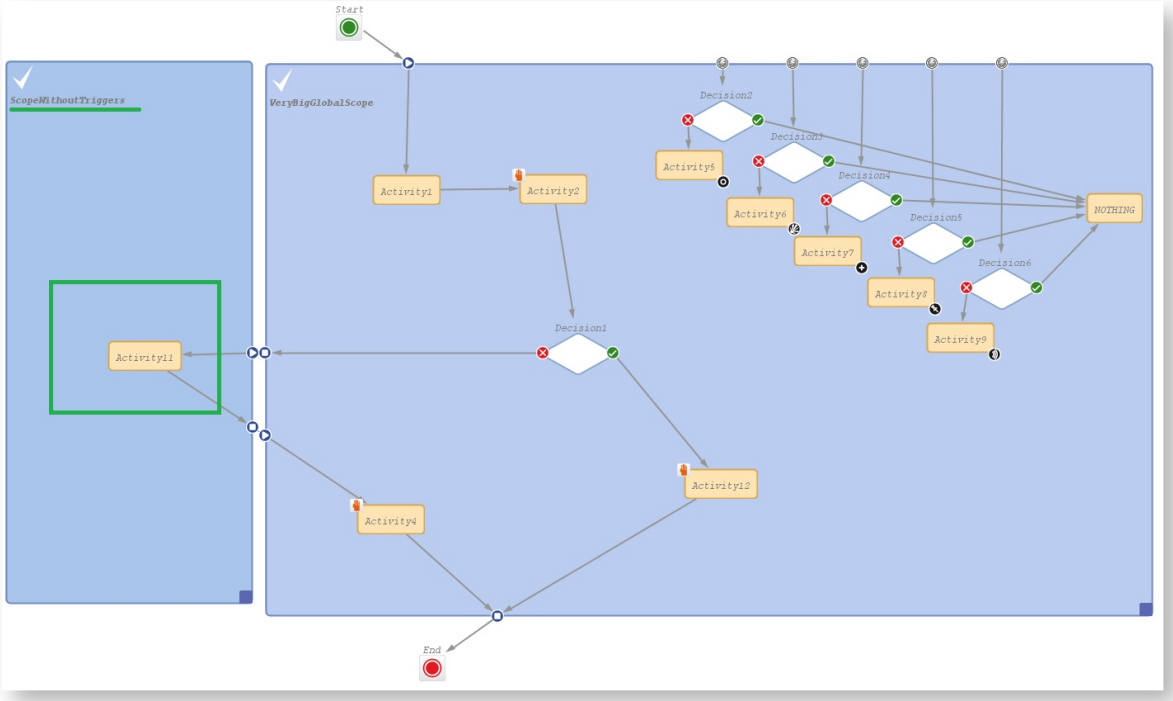


Figure 185: *ConSol CM Process Designer - Separate scopes with and without triggers*

In this example, the position of *Activity11* is optimized. It triggered many decision calls and all of them went to **NOTHING**. Executing *Activity11* outside of the global scope keeps a good quality of workflow performance!

F.5 How to avoid unneeded ticket update events

⚠ Beware of unnecessary ticket update events (Java class `TicketUpdateEvent`)!

For example, assigning the current engineer (the engineer who is logged in and working with the Web Client) to a ticket can be done in two ways. In one solution a ticket update event is fired, in the other this does not happen. If it is not necessary for a business case to throw a `TicketUpdateEvent`, avoid it, because an unnecessary call of `TicketUpdateEvent` causes a decrease in performance.

```
//this method throws a TicketUpdateEvent after assigning the current engineer to
 the ticket
workflowApi.assignEngineer(workflowApi.currentEngineer)
```

Code example 102: *Code which triggers TicketUpdateEvent*

```
//this method does NOT throw a TicketUpdateEvent!
ticket.setEngineer(workflowApi.currentEngineer)
```

Code example 103: *Code which does not trigger TicketUpdateEvent*

F.6 How to use the *disable auto update* parameter

⚠ Use the *disable auto update* flag for workflow components with care!

Please remember that a ticket update event is by default fired after every activity execution. A ticket update event is an operation that has a great impact and must be used with care!

To avoid performance problems, you can use the *disable auto update* flag. It depends on the business logic, if it makes sense to use this flag or not.

For example, when we have a series of automatic activities, a good practice is:

- The **1st** automatic activity has the *disable auto update* flag **on**.
  (It will **not** call the ticket update service method after activity execution.)
- The **2nd** automatic activity has the *disable auto update* flag **on**.
  (It will **not** call the ticket update service method after activity execution.)
- The **3rd** automatic activity has the *disable auto update* flag **on**.
  (It will **not** call the ticket update service method after activity execution.)
  ...
- The **last** automatic activity has the *disable auto update* flag **off**.
  (It **will** call `TicketUpdateEvent` **once**, at the **end** of the pipeline!)

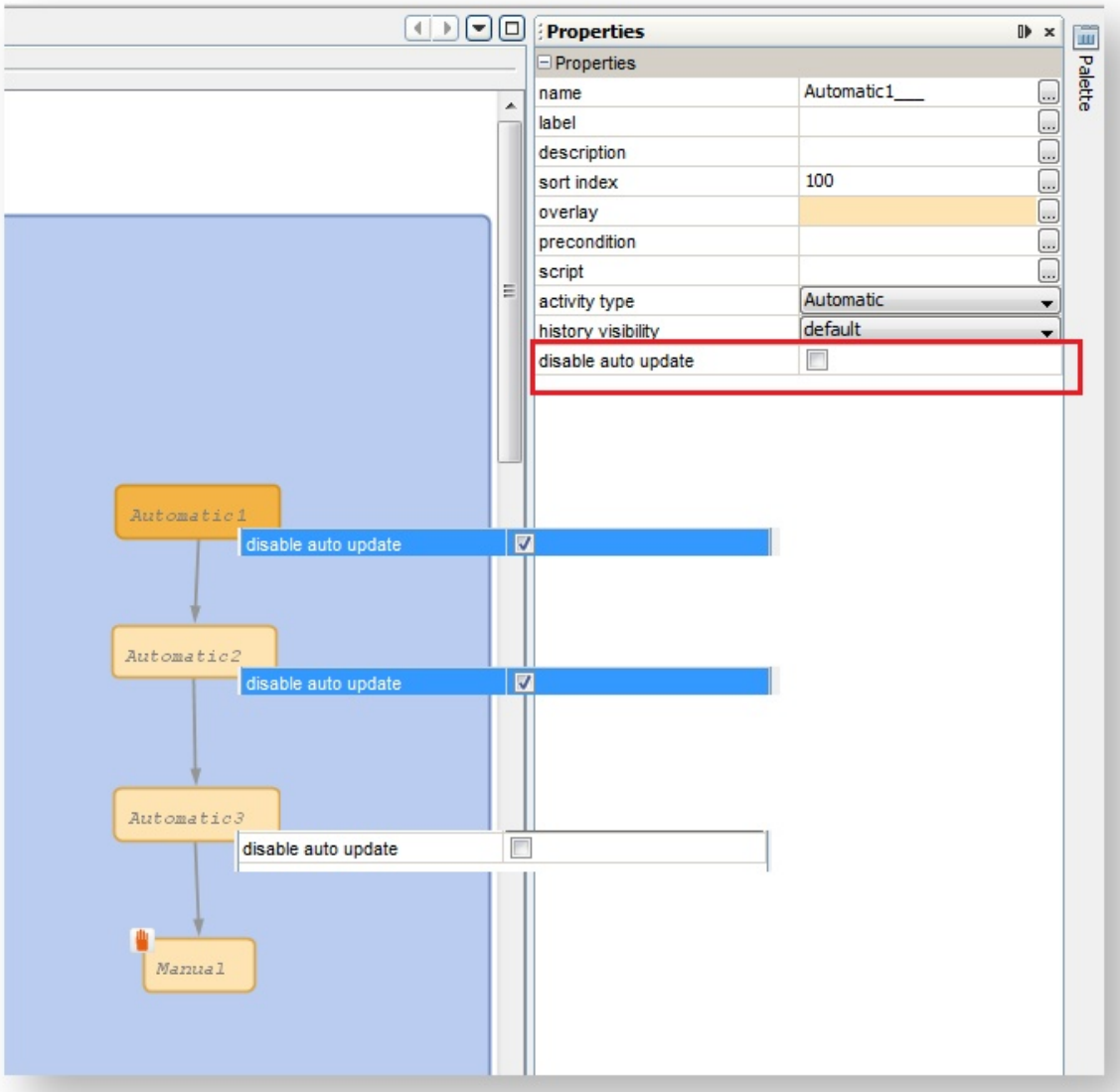


Figure 186: *ConSol CM Process Designer - Activities with "disable auto update" option*

F.7 How to avoid self-triggering business event triggers

When you use a business event trigger which is followed by an automatic activity, be careful that in this automatic activity the fields or objects, which trigger the business event trigger, are **not** changed again (which would fire the trigger again)!

If the use case requires that the fields, which caused the firing of the trigger, have to be changed again, then the logic, where the fields are changed, has to be placed in an activity outside the scope which hosts the trigger.

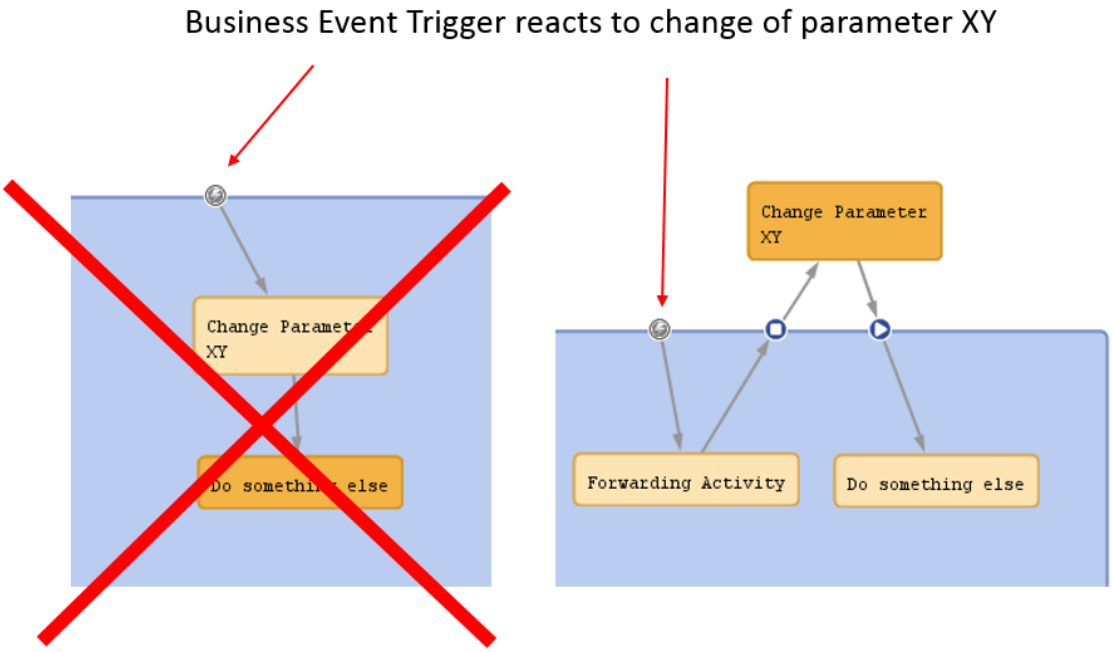


Figure 187: *ConSol CM Process Designer - Avoiding self-triggering business event triggers*

F.8 Which components to use with time triggers

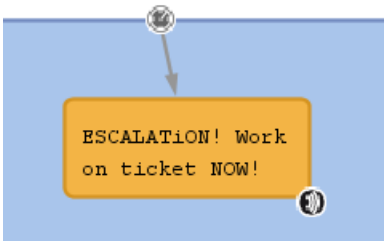


Figure 188: *A time trigger with an automatic activity*

There are two ways of implementing an event which should happen, when a time trigger has fired:

- Use the script after timer
- Use an automatic activity

The ConSol CM best practice for this use case is the following:

- Use the *script after timer* to control if the following automatic activity should be executed or not.
  This is slightly better than using a decision node, because a decision node is too complex at this location of the process and it would require two return values. Of course, it is - in principle - possible to use a decision node and in the end, it is up to you what you prefer.
- Use an automatic activity for everything else which should happen when the time trigger has fired.
  This can be seen in the graphical representation of the process logic and thus becomes instantly clear when the workflow is opened. You can easily add an overlay.

G - Appendix

This section contains several appendices:

- Annotations
- System Properties
- Default Java Imports
- List of Code Examples
- Trademarks
- Glossary

G.1 Annotations

There are two types of annotations: field annotations and group annotations. Field annotations are applied to a single ticket, customer or resource field. Group annotations are applied to a ticket, customer or resource field group. Please see:

- List of Field Annotations
- List of Group Annotations

G.1.1 List of Field Annotations

This chapter describes the following field annotations grouped by annotation type:

personal-data 301
groupable 301
sortable 301
autocomplete-script 301
leave-trailing-zeros 302
readonly 302
string-content-type 302
visibility 302
visualization 303
visualize-when-empty 303
boolean-type 303
enum-in-search-type 303
enum-type 303
list-type 304
text-type 304
ldapid 305
password 305
username 305
dwh-no-history-field 305
reportable 305
field indexed 306
phonetic 306
colspan 306
field-group 306
fieldsize 307
label-group 307
label-in-view 307
order-in-result 307
position 308
rowspan 308
show-label-in-edit 308
show-label-in-view 308
show-tooltip 308
show-watermark 308

ticket-list-colspan 309
ticket-list-position 309
ticket-list-rowspan 309
no-history-field 309
dialable 309
resource-color 310
customer exposure 310
contact search result column 310
contains contacts 311
enum field with ticket color 311
accuracy 311
email 311
format 311
matches 312
maxLength 312
maxValue 312
minLength 312
minValue 312
required 313
visibility configuration 313

G.1.1.1 anonymize (type)

personal-data

- **Type**: anonymize
- **Description**: Can be assigned to ticket and contact fields. Determines that the field holds personal data. Contact fields with this annotation will be deleted when the contact is anonymized. Ticket fields with this annotation will be deleted when the main customer of the ticket is anonymized.
- **Values**:
  - *true / false*: Set “true” if the field contains personal data.

G.1.1.2 cmweb-common (type)

groupable

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

sortable

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

G.1.1.3 common (type)

autocomplete-script

- **Type**: common
- **Description**: Specifies the script (type *Text autocomplete*) used for the scripted autocomplete field.
- **Values**: <name of the script>

> **Note:**
>
> This annotation is part of the way of configuring autocomplete fields which should only be used in ConSol CM versions up to 6.11.1.0. Starting with version 6.11.1.1, a new way of implementing scripted autocomplete lists is available. See section *Scripted Autocomplete Lists* in the *ConSol CM Administrator Manual*.
>
> Existing fields which use the old method will remain functional as of version 6.11.1.1. These fields will not be modified during an update. The previously used method `onEditDisplayEntered` still works for existing fields. Nevertheless, it is not needed anymore and will be removed in a future version of ConSol CM.

leave-trailing-zeros

- **Type**: common
- **Description**: Used for the display of fixed point numbers.
- **Values**:
  - *true / false*: Trailing zeros in the fractional part are not cut off when value is “true”.

readonly

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

string-content-type

- **Type**: common
- **Description**: Specifies the type of display for a rich text field in view mode.
- **Values**:
  - **html-inline**: The content of the rich text field is displayed directly on the GUI.
  - **html-link**: The content of the rich text field is displayed in a pop-up window which the user can access by clicking the *Show content* link.

visibility

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

visualization

- **Type**: common
- **Description**: Specifies the script (type *Field visualization*) used to render the content of the data field. For a detailed explanation, please refer to the section *Scripts of Type Field Visualization* in the *ConSol CM Administrator Manual*.
- **Values**: <name of the script>

visualize-when-empty

- **Type**: common
- **Description**: Determines if the visualization script of a data field with the annotation `visualization` should be executed even if the field is empty. By default, the visualization script is only executed if the field contains a value.
- **Values**:
  - *true / false*: Set “true” if the field visualization script should be executed even if the field is empty.

> This annotation is applied for scripted field visualization both in the Web Client and CM/Track.

G.1.1.4 component type (type)

boolean-type

- **Type**: component-type
- **Description**: Definition of the layout of a boolean field.
- **Values**:
  - *checkbox* (default): Field that can be checked (set to “false ”by default).
  - *radio*: 2 radio buttons (yes/no) for selection (only one can be active).
  - *select*: Drop-down field with 2 values (yes/no).

enum-in-search-type

- **Type**: component-type
- **Description**: Defines whether an ENUM field used in a search accepts searching over multiple values.
- **Values**:
  - *single* (default) / *multiple*: Accepts searching over multiple values if value “multiple” is set.

enum-type

- **Type**: component-type
- **Description**: Layout definition of list display

- **Values**:
  - *select* (default): Drop-down list for selection.
  - *radio*: List of radio buttons to select (only one option can be active).
  - *autocomplete*: Drop-down list for selections where the field is an input field used to filter the list.

list-type

- **Type**: component-type
- **Description**: Disables the add and/or delete options for data fields of type LIST or STRUCT.
- **Values**:
  - *fixed-size*: It is not possible to add or delete fields/rows.
  - *non-shrinkable*: It is not possible to delete fields/rows.
  - *non-growable*: It is not possible to add fields/rows.

text-type

- **Type**: component-type
- **Description**: Defines the possible types of a STRING field.
- **Values**:
  - *text* (default): Single-line input field
  - *textarea*: Multi-line input field
  - *password*: Input field for passwords.
    Password will be displayed as ******* in view mode.
  - *label*: Input will be displayed as a label, i.e., the field is displayed only, no input is possible.
  - *autocomplete*: The field will be used as autocomplete list. Please see the detailed explanationin the *ConSol CM Administrator Manual*, section *Admin Tool Scripts*, *Scripted Autocomplete Lists*.
  - *url*: The input will be displayed as a hyperlink in view mode. If no protocol is provided, *http://* is added automatically in front of the entered string. If a protocol, e.g. “http”, “https”, “mailto”, “file”, or “ftp” is used, the URL is rendered as is. The display text for the URL can be entered after a whitespace.
    Example: “http://consol.de ConSol”
  - *file-url*: Input will be displayed as a link to a file on the file system. The web browser has to allow/support those links! See section [Details about string fields: use annotations to fine-tune strings] on how to achieve this. The link will also be displayed as tooltip.

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

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

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

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

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

Example URLs:

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

G.1.1.5 contact authentication (type)

ldapid

- **Type**: contact authentication
- **Description**: Used in a customer field group of type *contact*, for the customer field which contains the LDAP ID for CM/Track authentication.
- **Values**: Indicates that this field will be used as an LDAP ID in the authentication process. Data type string is required.
  Since the definition is made at the customer group level, the LDAP authentication can be run in mixed mode. I.e., use LDAP for some customer groups and regular authentication for other customer groups.

password

- **Type**: contact authentication
- **Description**: Indicates that this field will be used as a password in the authentication process.
- **Values**:
  - **<string>**: Used for CM/Track.

username

- **Type**: contact authentification
- **Description**: Indicates that this field will be used as a login name in the authentication process.
- **Values**:
  - *true / false*: Used for CM/Track.

G.1.1.6 dwh (type)

dwh-no-history-field

- **Type**: dwh
- **Description**: Annotation used to indicate that field will not be historized in DWH
- **Values**:
  - *true / false*: Since version 6.10.2.0.

reportable

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

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

G.1.1.7 indexing (type)

field indexed

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

phonetic

- **Type**: indexing
- **Description**: activates the phonetic search for this field. Can only be used for data fields of type String (also long or short string).
- **Values**: *true/false*. Will automatically set to “true” when the annotation is added.

G.1.1.8 layout (type)

colspan

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

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

field-group

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

fieldsize

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

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

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

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

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

label-group

- **Type**: layout
- **Description**: Indicates a group of fields along with its descriptive label in view mode. Annotation is ignored in edit mode.
- **Values**:
  - **<string>**: Indicates a group of data fields along with its descriptive label. The annotation is used in view mode, ignored in edit mode. The group can have exactly one label (a data field of type STRING with assigned additional annotation `text-type` with value “label”). The label is shown when at least one data field from its group has a value set. All fields with the same label value are grouped and displayed under this label.
    The annotation `label-group` has to be assigned to the label, too.

label-in-view

- **Type**: layout
- **Description**: Shows data field value as a label in view mode. Annotation is ignored in edit mode.
- **Values**:
  - *true*: Remove the annotation if the label should not be visible in view mode.

order-in-result

- **Type**: layout
- **Description**: Shows field as a column at given position in the search result list.
- **Values**:
  - **<number>**: The columns are sorted in ascending order.
    Since CM version 6.0.1. Please see detailed explanation in info box in the *ConSol CM Administrator Manual*, section *Search Configuration*.

position

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

rowspan

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

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

show-label-in-edit

- **Type**: layout
- **Description**: Whether the data field should be displayed in edit mode with label.
- **Values**:
  - *true / false*: Since version 6.9.4

show-label-in-view

- **Type**: layout
- **Description**: Whether the data field should be displayed in view mode with label.
- **Values**:
  - *true / false*: Since version 6.9.4

show-tooltip

- **Type**: layout
- **Description**: Whether the data field should be displayed with tooltip.
- **Values**:
  - *true / false*: Since version 6.9.4

show-watermark

- **Type**: layout
- **Description**: Whether the data field should be displayed with watermark.
- **Values**:
  - *true / false*: Since version 6.9.4

ticket-list-colspan

- **Type**: layout
- **Description**: Defines how many columns are occupied by the field in the ticket list box.
- **Values**:
  - **<number>**: Number of columns.

> ⚠ This annotation only works if the annotation `ticket-list-position` is also set for the field.

ticket-list-position

- **Type**: layout
- **Description**: Defines the position of the field in the ticket list box.
- **Values**:
  - **<number>;<number>**: Values define row and column (row;column), numbering starts at 0;0.

ticket-list-rowspan

- **Type**: layout
- **Description**: Defines how many rows are occupied by the field in the ticket list box.
- **Values**:
  - **<number>**: Number of rows.

> ⚠ This annotation only works if the annotation `ticket-list-position` is also set for the field.

G.1.1.9 performance (type)

no-history-field

- **Type**: performance
- **Description**: Indicates that a single data field should not be historicized. Overwrites the group annotation `no-history`.
- **Values**:
  - *true / false*: Annotation is active if value is set to “true”. For fields that should be stored but not be visible in history use annotation `visibility configuration`.
    In CM versions up to 6.10.2, the DWH transfer of a field history is also controlled by this annotation.
    Starting with CM version 6.10.2, use the annotation `dwh-no-history-field` for this.

G.1.1.10 phone commander (type)

dialable

- **Type**: phone commander (CM/Phone)
- **Description**: Defines a field with a phone number.

- **Values**:
  - *true*: Used with CM/Phone only. Marks a phone number as automatically dialable for outgoing calls for the CTI system.

G.1.1.11 resource (type)

resource-color

- **Type**: resource
- **Description**:
- **Values**:
  - *true / false*: Should be assigned to a color ENUM. Color of selected enum value should be applied to a resource icon's background.

G.1.1.12 restapi (type)

customer exposure

- **Type**: restapi
- **Description**: Indicates whether a data field should be available to customers using the REST API, e.g. in CM/Track.
- **Values**:
  - *full* (default): The data field is available for reading and writing.
  - *read*: The data field is available for reading only.
  - *none*: The data field is not available. This value can be helpful, for example, when an entire ticket field group has been configured with `customer exposure group` = “full |read” and dedicated single fields should not be available in CM/Track (or over the REST API in general).

G.1.1.13 search result (type)

contact search result column

- **Type**: search result
- **Description**: Identifies whether the field should be presented in the search result by default. **Deprecated! Do not use!** Removed in ConSol CM version 6.11.
- **Values**:
  - *true*:
    Remove the annotation if the field should not be visible by default.
    Since CM version 6.1.3.
    (Replaced by `order-in-result`! `contact search result column` is obsolete!)

G.1.1.14 ticket contact relation type (type)

contains contacts

- **Type**: ticket contact relation type
- **Description**: Only one instance of this annotation is allowed in one data model! Used only for list field definition, indicates that it can hold unit references to units annotated as contacts. Starting with ConSol CM version 6.11, the field annotated with contains contact will not produce history entries. This decreases the loading time for tickets which contain a great number of customers.
- **Values**:
  - *true/false*: Value type is boolean. Specifies whether the list is shown with the contact ("true") or with the ticket ("false").

G.1.1.15 ticket display (type)

enum field with ticket color

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

G.1.1.16 validation (type)

accuracy

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

email

- **Type**: validation
- **Description**: Used for email addresses to validate that the format is correct, i.e., that it matches <name>@<domain>.
- **Values**:
  - *true*: May be used with STRING data fields. Remove the annotation if the format should not be validated.

format

- **Type**: validation
- **Description**: Used for validating the format of date fields.

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

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

matches

- **Type**: validation
- **Description**: Checks if input of STRING data fields matches the given RegEx.
- **Values**:
  - **<string>**: May be used with STRING data fields.

maxLength

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

maxValue

- **Type**: validation
- **Description**: Defines the maximum value for number data fields.
- **Values**:
  - **<number>**: May be used with NUMBER data fields, i.e., NUMBER and FIXED-POINT NUMBER.

minLength

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

minValue

- **Type**: validation
- **Description**: Defines the minimum value for NUMBER data fields.
- **Values**:
  - **<number>**: May be used with NUMBER data fields, i.e., NUMBER and FIXED-POINT NUMBER.

required

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

G.1.1.17 visibility (type)

visibility configuration

- **Type**: visibility
- **Description**: Indicates the visibility of this field in history.
- **Values**:
  - *on every level*: Field is shown on every level of history.
  - *2nd level and 3rd level*: Field is shown only on the 2nd and the 3rd level of history.
  - *only 3rd level*: Field is shown only on the 3rd level of history.

G.1.2 List of Group Annotations

This chapter describes the following group annotations grouped by annotation type:

group-visibility 315
dwh-no-history 315
reportable group 315
auto-open-group 315
show-contact-in-ticket-list 315
show-in-group-section 316
show-labels-in-edit 316
show-labels-in-view 316
show-tooltips 316
show-watermarks 316
no-history 316
resource-fields-group-mode 317
resource-custom-fields-group-mode 317
customer group exposure 317
unit is a contact 317

G.1.2.1 common (type)

group-visibility

- **Type**: common
- **Description**: Defines the default visibility of a data field group.
- **Values**:
  - *true / false*: The annotation can be overwritten at the field level.

G.1.2.2 dwh (type)

dwh-no-history

- **Type**: dwh
- **Description**: Indicates that all fields in the group will not be historicized in DWH
- **Values**:
  - *true / false*: Since version 6.10.2.0

reportable group

- **Type**: dwh
- **Description**: Indicates that all data fields belonging to this group are reportable and should be transferred to CMRF.
- **Values**:
  - *true / false*: A value has to be set. Annotation is active if value is set to “true”.

G.1.2.3 layout (type)

auto-open-group

- **Type**: layout
- **Description**: The group will be opened initially. More than one value can be entered as a comma- or semicolon-separated list (can be used for the customer annotation).
- **Values**:
  - *ticket:create*: Group is opened initially when a new ticket is created.
  - *ticket:view*: Group is opened initially when a ticket page is opened.
  - *customer:create*: Group is opened initially when a new customer is created.
  - *customer:view*: Group is opened when the customer (contact or company) page is opened.

show-contact-in-ticket-list

- **Type**: layout
- **Description**: Obsolete! Use page customization! accordionTicketList.mainCustomerDescriptionVisible={true, false}
- **Values**: obsolete

show-in-group-section

- **Type**: layout
- **Description**: Defines that a data field group is displayed in the Details section (as tab).
- **Values**:
  - *true / false*: Without this annotation the group is shown in the non-tabbed ticket, customer or resource section.

show-labels-in-edit

- **Type**: layout
- **Description**: Whether the data fields in this group should be displayed in edit mode with labels.
- **Values**:
  - *true / false*: Since version 6.9.4

show-labels-in-view

- **Type**: layout
- **Description**: Whether the data fields in this group should be displayed in view mode with labels.
- **Values**:
  - *true / false*: Since version 6.9.4

show-tooltips

- **Type**: layout
- **Description**: Whether the data fields in this group should be displayed with tooltips.
- **Values**:
  - *true / false*: Since version 6.9.4

show-watermarks

- **Type**: layout
- **Description**: Whether the data fields in this group should be displayed with watermarks.
- **Values**:
  - *true / false*: Since version 6.9.4

G.1.2.4 performance (type)

no-history

- **Type**: performance
- **Description**: Indicates that all data fields belonging to this group will not be historicized.
- **Values**:
  - *true / false*: Indicates that all data fields that belong to this group should not be historicized. Possible values are “true” if this annotation should be active or “false”, which is the same as removing the annotation. Use this annotation if you want to prevent storing history for all/many fields in a group. If you only want to prevent historization for a single/some field(s), use the annotation `no-history-field` at the field level.

In CM versions up to 6.10.2, the DWH transfer of a field history is also controlled by this annotation.
Starting with CM version 6.10.2, use the annotation `dwh-no-history` for this.

G.1.2.5 resource (type)

resource-fields-group-mode

- **Type**: resource
- **Description**: Controls the mode of a resource field group concerning editing via Web Client.
- **Values**:
  - *internal / external*: Possible values: internal, external. A resource field is not editable in the Web Client if value is “external”.
    Since 6.10.4.0
    Removed 6.10.5.0

resource-custom-fields-group-mode

- **Type**: resource
- **Description**: Controls the mode of a resource field group concerning editing via Web Client.
- **Values**:
  - *internal / external*: Possible values: internal, external. A resource field is not editable in the Web Client if value is “external”.
    Since 6.10.5.0

G.1.2.6 restapi (type)

customer group exposure

- **Type**: restapi
- **Description**: Indicates whether a data field group should be available to customers using the REST API, e.g. in CM/Track.
- **Values**:
  - *full* (default): The data field group is available for reading and writing.
  - *read*: The data field group is available for reading only.

G.1.2.7 ticket contact relation (type)

unit is a contact

- **Type**: ticket contact relation
- **Description**: deprecated
- **Values**:
  - *true / false*: Removed in version 6.9.0.

G.2 System Properties

The following chapter provides detailed information about the system properties used in ConSol CM.

- Alphabetical List of System Properties
- List of System Properties by Module
- List of System Properties by Area

CM/ is a registered trademark of ConSol Consulting & Solutions Software GmbH

G.2.1 Alphabetical List of System Properties

This chapter describes the following properties:

access.token.signing.key 330
admin.email 330
admin.login 330
admin.tool.consumed.licences.check.interval 330
admin.tool.consumed.licences.pool.name 331
admin.tool.session.check.interval 331
archive.uri 331
attachment.allowed.types 331
attachment.max.size 332
attachment.type.hints 332
attachment.upload.timeout 332
authentication.method 333
autocommit.cf.changes 333
autocomplete.enabled 333
automatic.booking.enabled 334
batch-commit-interval 334
big.task.minimum.size 334
cache-cluster-name 335
calendar.csv.dateFormat 335
calendar.csv.separator 335
checkUserOnlineIntervalInSeconds 335
client.archive.access.token.validity.seconds 336
client.archive.enabled 336
client.archive.refresh.token.validity.seconds 336
client.archive.secret 337
client.was.access.token.validity.seconds 337
client.was.refresh.token.validity.seconds 337
client.was.secret 337
cluster.mode 338
cluster.unicast 338
cmas.dropSchemaBeforeSetup 338
cmoffice.enabled 339
cmoffice.oo.path.NUMBER 339
cmoffice.strict.versioning.enabled 339

cmoffice.websocket.port 339
comment.authors.disabled 340
commentRequiredForTicketCreation 340
communication.channel 340
config.data.version 341
config.import.global.transaction.enabled 341
connection.release.mode 341
contact.authentication.method 341
contact.inherit.permissions.only.to.own.customer.group 342
csrf.domain.allow.none 342
csrf.domain.white.list 342
csrf.domain.white.list 343
csrf.request.filter.enabled 343
csrf.request.filter.enabled 343
customfield.content.file.max.size 343
customizationVersion 344
dao.log.threshold.milliseconds 344
dao.log.username 344
data.directory 345
data.optimization 345
database.notification.enabled 345
database.notification.redelivery.delay.seconds 346
database.notification.redelivery.max.attempts 346
defaultAttachmentEntryClassName 346
defaultCommentClassName 346
defaultContentEntryClassName 347
defaultIncommingMailClassName 347
defaultNumberOfCustomFieldsColumns 347
defaultOutgoingMailClassName 347
delete.ticket.enabled 348
diffTrackingEnabled 348
diffTrackingEnabledForUnitAndResource 348
diff.tracking.disabled 349
disable.admin.task.auto.commit 349
domain.map.for.client.config.<DOMAIN_NAME> 349
dwh.administration.refresh.interval.seconds 350

dwh.mode 350
engineer.description.cache.enabled 350
engineer.description.mode 350
engineer.description.template.name 351
eviction.event.queue.size 351
eviction.lifeSpan 351
eviction.max.nodes 352
eviction.strategy 352
eviction.wakeup.interval 352
expert.mode 352
external.line.access.prefix 353
favoritesSizeLimit 353
fetchLock.interval 353
fetchSize.strategy 353
fetchSize.strategy.FetchSizeFixedStrategy.value 354
fetchSize.strategy.FetchSizePageBasedStrategy.limit 354
fetchSize.strategy.FetchSizeThresholdStrategy.value 354
filesystem.polling.threads.number 355
filesystem.polling.threads.shutdown.timeout.seconds 355
filesystem.polling.threads.watchdog.interval.seconds 355
filesystem.task.enabled 355
filesystem.task.interval.seconds 356
filesystem.task.polling.folder 356
filesystem.task.timeout.seconds 356
filesystem.task.transaction.timeout.seconds 356
forward.mails.to.representatives 357
globalSearchResultSizeLimit 357
heartbeat 357
helpFilePath 358
hibernate.dialect 358
hideTicketSubject 358
ignore-queues 359
index.attachment 359
index.history 359
index.status 359
index.task.worker.threads 360

index.version.current 360
index.version.newest 360
indexed.assets.per.thread.in.memory 361
indexed.engineers.per.thread.in.memory 361
indexed.resources.per.thread.in.memory 361
indexed.tickets.per.thread.in.memory 361
indexed.units.per.thread.in.memory 362
initialized 362
internal.line.access.prefix 362
is.cmrf.alive 363
java.naming.factory.initial 363
java.naming.factory.url.pkgs 363
java.naming.provider.url 363
jobExecutor.adminMail 364
jobExecutor.idleInterval 364
jobExecutor.idleInterval.seconds 364
jobExecutor.jobExecuteRetryNumber 365
jobExecutor.jobMaxRetries 365
jobExecutor.jobMaxRetriesReachedSubject 365
jobExecutor.lockingLimit 365
jobExecutor.lockTimeout.seconds 366
jobExecutor.mailFrom 366
jobExecutor.maxInactivityInterval.minutes 366
jobExecutor.threads 367
jobExecutor.timerRetryInterval 367
jobExecutor.timerRetryInterval.seconds 367
jobExecutor.txTimeout.seconds 367
kerberos.v5.enabled 368
kerberos.v5.username.regex 368
last.config.change 368
last.config.change.templates 368
last.ping.timestamp 369
ldap.authentication 369
ldap.basedn 369
ldap.certificate.basedn 370
ldap.certificate.content.attribute 370

ldap.certificate.password 370
ldap.certificate.providerurl 370
ldap.certificate.searchattr 371
ldap.certificate.userdn 371
ldap.contact.name.basedn 371
ldap.contact.name.password 371
ldap.contact.name.providerurl 372
ldap.contact.name.searchattr 372
ldap.contact.name.userdn 372
ldap.initialcontextfactory 372
ldap.password 373
ldap.providerurl 373
ldap.searchattr 373
ldap.userdn 373
live.start 374
local.country.prefix 374
mail.attachments.validation.info.sender 374
mail.attachments.validation.info.subject 375
mail.db.archive 375
mail.encryption 375
mail.error.from.address 375
mail.error.to.address 376
mail.from 376
mail.notification.engineerChange 376
mail.notification.sender 376
mail.on.error 377
mail.redelivery.retry.count 377
mail.reply.to 377
mail.sender.address 378
mail.smtp.email 378
mail.smtp.email.password 378
mail.smtp.email.user 378
mail.smtp.envelopesender 379
mail.smtp.tls.enabled 379
mail.ticketname.pattern 379
mailbox.<NUMBER>.name 380

mailbox.1.connection.host 380
mailbox.1.connection.password 380
mailbox.1.connection.port 380
mailbox.1.connection.protocol 380
mailbox.1.connection.username 380
mailbox.2.connection.host 380
mailbox.2.connection.password 381
mailbox.2.connection.port 381
mailbox.2.connection.protocol 381
mailbox.2.connection.username 381
mailbox.default.connection.host 381
mailbox.default.connection.password 381
mailbox.default.connection.port 382
mailbox.default.connection.protocol 382
mailbox.default.connection.username 382
mailbox.default.session.mail.debug 382
mailbox.default.session.mail.mime.address.strict 383
mailbox.default.session.mail.<PROTOCOL>.connectiontimeout 383
mailbox.default.session.mail.<PROTOCOL>.fetchsize 383
mailbox.default.session.mail.<PROTOCOL>.partialfetch 384
mailbox.default.session.mail.<PROTOCOL>.timeout 384
mailbox.default.task.delete.read.messages 384
mailbox.default.task.enabled 385
mailbox.default.task.interval.seconds 385
mailbox.default.task.max.message.size 385
mailbox.default.task.max.messages.per.run 386
mailbox.default.task.timeout.seconds 386
mailbox.default.task.transaction.timeout.seconds 386
mailbox.polling.threads.mail.log.enabled 386
mailbox.polling.threads.number 387
mailSender.executionInterval.seconds 387
mailSender.lockingLimit 387
mailSender.lockTimeout.seconds 387
mailSender.maxAttempts 388
mailSender.nodeId 388
mailSender.retryInterval.seconds 388

mailSender.threads.number 389
mailSender.txTimeout.seconds 389
mailTemplateAboveQuotedText 389
max.licences.perUser 389
maxSizePerPagemapInMegaBytes 390
monitoring.engineer.login 390
monitoring.unit.login 390
nimh.enabled 391
notification.error.description 391
notification.error.from 391
notification.error.subject 391
notification.error.to 392
notification.finished_successfully.description 392
notification.finished_successfully.from 392
notification.finished_successfully.subject 392
notification.finished_successfully.to 393
notification.finished_unsuccessfully.description 393
notification.finished_unsuccessfully.from 393
notification.finished_unsuccessfully.subject 393
notification.finished_unsuccessfully.to 394
notification.host 394
notification.password 394
notification.port 394
notification.protocol 395
notification.tls.enabled 395
notification.username 395
notifications.enabled 396
number.of.tasks 396
outdated.lock.age 396
pagemapLockDurationInSeconds 396
password.reset.mail.from 397
policy.password.age 397
policy.password.pattern 397
policy.rotation.ratio 398
policy.track.username.case.sensitive 398
policy.username.case.sensitive 398

postActivityExecutionScriptName 398
queue.polling.threads.number 399
queue.polling.threads.shutdown.timeout.seconds 399
queue.polling.threads.watchdog.interval.seconds 399
queue.task.error.pause.seconds 399
queue.task.interval.seconds 400
queue.task.max.retries 400
queue.task.timeout.seconds 400
queue.task.transaction.timeout.seconds 401
queuesExcludedFromGS 401
recent.items.cleanup.cluster.node.id 401
recent.items.cleanup.interval.minutes 401
recent.items.max.per.engineer 402
recent.items.persistence.enabled 402
recoverable.exceptions 402
refreshTimeInCaseOfConcurrentRememberMeRequests 403
rememberMeLifetimeInMinutes 403
request.scope.transaction 403
resetCode.expirationPeriod 403
resource.replace.batchSize 404
resource.replace.timeout 404
scene 404
script.evict.unused.after.hours 405
script.logging.threshold.seconds 405
script.validation.interval.seconds 405
searchPageSize 405
searchPageSizeOptions 406
security.fields.customer.exposure.check.enabled 406
security.restrict.unit.access.to.own.data 406
serial.mods.tracking.enabled 407
server.instance.task.crash.period.seconds 407
server.instance.task.period.seconds 407
server.session.archive.reaper.interval 408
server.session.archive.timeout 408
server.session.reaper.interval 408
server.session.timeout 408

serverPoolingInterval 409
skip-ticket 409
skip-ticket-history 410
skip-unit 410
skip-unit-history 410
skip.wfl.transfer.cleanup 410
skip.wfl.transfer.translations.cleanup 411
split.history 411
start.groovy.task.enabled 411
statistics.calendar 412
statistics.client.group 412
statistics.contact.role 412
statistics.content.entry 413
statistics.content.entry.class 413
statistics.content.entry.history 413
statistics.customer.definition 414
statistics.engineer 414
statistics.enum.group 414
statistics.field.definition 415
statistics.group.definition 415
statistics.locale 415
statistics.localized.property 416
statistics.mla 416
statistics.project 416
statistics.queue 417
statistics.resource 417
statistics.resource.group 417
statistics.resource.history 418
statistics.resource.relation.definition 418
statistics.resource.type 418
statistics.ticket 419
statistics.ticket.function 419
statistics.ticket.history 419
statistics.time.booking 420
statistics.timestamp 420
statistics.unit 420

statistics.unit.history 421
statistics.unit.relation.definition 421
statistics.workflow 421
strict.utf.bmp.enabled 421
supportEmail 422
synchronize.archive.timeout.minutes 422
synchronize.master.address 423
synchronize.master.security.token 423
synchronize.master.security.user 423
synchronize.master.timeout.minutes 424
synchronize.megabits.per.second 424
synchronize.sleep.millis 424
task.execution.interval.seconds 424
task.execution.node.id 425
task.panel.refresh.interval.seconds 425
themeOverlay 425
ticket.delete.timeout 426
ticket.from.incoming.message.accepted.links 426
ticketListRefreshIntervalInSeconds 426
ticketListSizeLimit 427
tickets.delete.size 427
time.buffer 427
transaction.timeout.minutes 427
ttl.days 428
tx.read.only.mode.enabled 428
unit.description.mode 428
unit.replace.batchSize 429
unit.replace.timeout 429
unit.transfer.order 429
unitIndexSearchResultSizeLimit 429
unused.content.remover.cluster.node.id 430
unused.content.remover.enabled 430
unused.content.remover.polling.minutes 430
unused.content.remover.ttl.minutes 431
update.6.11.0.0.sleep 431
update.6.11.0.0.timezone 431

urlLogoutPath 432
voCacheEnabled 432
warmup.executor.enabled 433
webSessionTimeoutInMinutes 433
wfl.sticky.transfer.disabled 434
wicketAjaxRequestHeaderFilterEnabled 434
workflow.deploy.cache.eviction.disabled 434
X-Frame-Options 434

access.token.signing.key

- **Module**: cmas-auth-server
- **Description**: Determines the secret shared between the authorization server and all client applications using OAuth2, e.g. CM/Archive. The value of this property needs to match the value of *archive.oauth2.access.token.signing.key* in the configuration file of CM/Archive.
- **Type**: password
- **Restart required**: yes
- **System**: no
- **Optional**: no
- **Example value**: my_password
- **Since**: 6.11.2.0

admin.email

- **Module**: cmas-core-security
- **Description**: The email address of the ConSol CM administrator. The value which you entered during system set-up is used initially.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: myuser@consol.de
- **Since**: 6.0

admin.login

- **Module**: cmas-core-security
- **Description**: The name of the ConSol CM administrator. The value which you entered during system set-up is used initially.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: admin
- **Since**: 6.0

admin.tool.consumed.licences.check.interval

- **Module**: cmas-app-admin-tool
- **Description**: Sets the interval (in seconds) to monitor the number of consumed licenses.
- **Type**: integer
- **Restart required**: no
- **System**: no

- **Optional**: yes
- **Example value**: 30 (default value)
- **Since**: 6.11.0.0

admin.tool.consumed.licences.pool.name

- **Module**: cmas-app-admin-tool
- **Description**: Sets the license pool name to monitor the number of consumed licenses.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: CONCURRENT_USERS (default value)
- **Since**: 6.11.0.0

admin.tool.session.check.interval

- **Module**: cmas-app-admin-tool
- **Description**: Configures the time interval (in seconds) in which the system checks for inactive (ended) Admin Tool sessions.
- **Type**: integer
- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: 30
- **Since**: 6.7.5

archive.uri

- **Module**: cmas-archive-core-server
- **Description**: Sets the URL from which the CM/Archive application can be accessed
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: <server_url>:8090
- **Since**: 6.11.2.0

attachment.allowed.types

- **Module**: cmas-core-server
- **Description**: Comma-separated list of allowed file name extensions (if no value is defined, all file extensions are allowed).
- **Type**: string

- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: txt,zip,doc
- **Since**: 6.5.0

attachment.max.size

- **Module**: cmas-core-server
- **Description**: Maximum attachment size, in MB. This is a validation property of the CM API. It controls the size of attachments at tickets, units, and resources. It also controls the size of incoming (not outgoing!) email attachments. The value of this property needs to be aligned with the respective setting in the application server configuration.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 100 (default value)
- **Since**: 6.4.0

attachment.type.hints

- **Module**: cmas-core-server
- **Description**: Allows you to assign MIME types to not yet officially supported file extensions, so these file extensions can be detected correctly.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 7z:application/x-7z-compressed, zip:application/zip, wav:audio/x-wav
- **Since**: 6.11.2.0

attachment.upload.timeout

- **Module**: cmweb-server-adapter
- **Description**: Defines the transaction timeout in minutes for adding attachments to a ticket, resource or customer. Counts the time for the upload of all attachments of one transaction. When the timeout occurs, all files which have been temporarily stored on the server are deleted. No file is uploaded.
- **Type**: Integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes

- **Example value**: 3
- **Since**: 6.10.5.3

authentication.method

- **Module**: cmas-core-security
- **Description**: User authentication method (internal CM database or LDAP authentication). Allowed values are LDAP or DATABASE.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: DATABASE
- **Since**: 6.0

autocommit.cf.changes

- **Module**: cmas-dwh-server
- **Description**: Defines whether DWH tasks which result from configurational changes on ticket fields are executed automatically without manual interaction in the Admin Tool. Can be also set in the Admin Tool in the navigation item *DWH*. The default and recommended value is “false”.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false (default value)
- **Since**: 6.7.0

autocomplete.enabled

- **Module**: cmas-app-admin-tool
- **Description**: If the flag is missing or its value is “false”, then the *Autocomplete address* navigation item is hidden in Admin Tool.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: true
- **Since**: 6.9.2.0

automatic.booking.enabled

- **Module**: cmweb-server-adapter
- **Description**: If enabled, time spend on creating comment/email will be measured and automatic time booking will be added.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: true
- **Since**: 6.9.4.2

batch-commit-interval

- **Module**: cmas-dwh-server
- **Description**: Number of objects in a DWH message. Larger values mean better transfer performance at the cost of higher memory usage.
  Starting with ConSol CM version 6.11, this property is only used if the package size of a DWH operation is not set. This can only happen when the command is directly addressed to the Java MBean `consol.cmas.global.dwh.synchronizationService`, e.g. using the `update()` method. When a DWH operation is started using the Admin Tool, there is always a value for the package size. If not explicitly set, the default value of 1000 is used as value for `batch.commit.interval`.
- **Default value**: 1000
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 100
- **Since**: 6.0.0

big.task.minimum.size

- **Module**: cmas-core-index-common
- **Description**: Indicates the minimum size of index task (in parts, each part has 100 entities) to qualify this task as a big one. Big tasks have a lower priority than normal tasks.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 15 (default value)
- **Since**: 6.8.3

cache-cluster-name

- **Module**: cmas-core-cache
- **Description**: JBoss cache cluster name.
- **Type**: string
- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: 635a6de1-629a-4129-8299-2d98633310f0
- **Since**: 6.4.0

calendar.csv.dateFormat

- **Module**: cmas-core-server
- **Description**: Format of the date given in the csv file containing the list of holidays.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: dd/MM/yyyy
- **Since**: 6.9.3.2

calendar.csv.separator

- **Module**: cmas-core-server
- **Description**: Separator used in the csv file containing the list of holidays.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: ,
- **Since**: 6.9.3.2

checkUserOnlineIntervalInSeconds

- **Module**: cmweb-server-adapter
- **Description**: The interval in seconds to check which users are online (default 180sec = 3min).
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 180

- **Since**: 6.0
- **Removed in**: 6.5 / 6.11.0.1

client.archive.access.token.validity.seconds

- **Module**: cmas-auth-server
- **Description**: Determines the validity period in seconds of the access token required for using CM/Archive.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: 43200 (default value)
- **Since**: 6.11.2.0

client.archive.enabled

- **Module**: cmas-auth-server
- **Description**: Determines if CM/Archive is enabled (“true”) or disabled (“false”).
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: false
- **Since**: 6.11.2.0
- **Removed in**: 6.11.2.2

client.archive.refresh.token.validity.seconds

- **Module**: cmas-auth-server
- **Description**: Determines the validity period in seconds of the refresh token required for obtaining new access tokens for CM/Archive.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: 2592000 (default value)
- **Since**: 6.11.2.0

client.archive.secret

- **Module**: cmas-auth-server
- **Description**: Determines the secret shared between the authorization server and CM/Archive. The value of the property needs to match the value of *archive.oauth2.client.secret* in the configuration file of CM/Archive.
- **Type**: password
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: my_password
- **Since**: 6.11.2.0

client.was.access.token.validity.seconds

- **Module**: cmas-auth-server
- **Description**: Determines the validity period in seconds of the access token required for using the Web Admin Suite.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: 43200 (default value)
- **Since**: 6.12.0.0

client.was.refresh.token.validity.seconds

- **Module**: cmas-auth-server
- **Description**: Determines the validity period in seconds of the refresh token required for obtaining new access tokens for the Web Admin Suite.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: 2592000 (default value)
- **Since**: 6.12.0.0

client.was.secret

- **Module**: cmas-auth-server
- **Description**: Determines the secret shared between the authorization server and the Web Admin Suite.
- **Type**: password
- **Restart required**: no

- **System**: no
- **Optional**: no
- **Example value**: my_password
- **Since**: 6.12.0.0

cluster.mode

- **Module**: cmas-core-shared
- **Description**: Specifies whether ConSol CM is running in a cluster.
- **Type**: boolean
- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.1.0

cluster.unicast

- **Module**: cmas-core-shared
- **Description**: Flag to activate jgroups unicast mode for ConSol CM clusters (as opposed to the default multicast mode causing problems in some data center environments). If set to “true”, remember to set the JVM start parameters: `jgroups.bind.port`, `jgroups.bind.address` and `jgroups.initial_hosts`.
- **Type**: boolean
- **Restart required**: yes
- **System**: yes
- **Optional**: yes
- **Example value**: false (default value)
- **Since**: 6.11.0.0
- **Removed in**: 6.12.0.0

cmas.dropSchemaBeforeSetup

- **Module**: cmas-setup-hibernate
- **Description**: Flag if schema is to be (was) dropped during setup
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true
- **Since**: 6.0

cmoffice.enabled

- **Module**: cmweb-server-adapter
- **Description**: Flag if CM/Doc (former CM/Office) is enabled.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.4.0

cmoffice.oo.path.NUMBER

- **Module**: cmweb-server-adapter
- **Description**: Possible location of the OpenOffice installation. The properties are numbered starting with 0.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: c:\Program Files (x86)\LibreOffice 3.6\program
- **Since**: 6.10.1.0
- **Removed in**: 6.12.0.0

cmoffice.strict.versioning.enabled

- **Module**: cmweb-server-adapter
- **Description**: Controls if the SAVE operation in Microsoft Word / OpenOffice documents creates a new attachment ("true") or overwrites the existing attachment ("false"). This concerns the behavior within one session using the text editing program. If the program is stopped, the overwrite mechanism will not work anymore.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true
- **Since**: 6.10.5.4

cmoffice.websocket.port

- **Module**: cmweb-server-adapter
- **Description**: Determines the port on which the CM/Doc application should be started and connected to.
- **Type**: integer

- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 30333
- **Since**: 6.12.0.0

comment.authors.disabled

- **Module**: cmas-restapi-core
- **Description**: Disables the display of the content's author via REST API.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: false (default value)
- **Since**: 6.11.0

commentRequiredForTicketCreation

- **Module**: cmweb-server-adapter
- **Description**: Flag if the comment is a required field for ticket creation.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true (default value)
- **Since**: 6.2.0

communication.channel

- **Module**: cmas-dwh-server
- **Description**: Communication channel. Only possible value since CM version 6.11.0.0: DIRECT
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: DIRECT
- **Since**: 6.8.5.0
- **Removed in**: 6.11.0.0 (DIRECT mode is the only available mode and is set automatically)

config.data.version

- **Module**: cmas-core-server
- **Description**: The internal version number of the current system configuration. This property is maintained internally, please do not change it unless advised by ConSol.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 11
- **Since**: 6.0

config.import.global.transaction.enabled

- **Module**: cmas-core-server
- **Description**: Flag deciding whether configuration (without localizations) should be imported within a single transaction.
- **Type**: Boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true
- **Since**: 6.11.1.0

connection.release.mode

- **Module**: cmas-setup-hibernate
- **Description**: Describes the JEE connection handling strategy for transactions. If set to “AFTER_TRANSACTION”, the connection will be cached during the transaction and released at the end. If set to “AFTER_STATEMENT”, the connection will be released to the pool after each statement execution. Please do not change the default here unless advised by ConSol.
- **Type**: string
- **Restart required**: yes
- **System**: no
- **Optional**: yes
- **Example value**: AFTER_STATEMENT (default for JEE environment)
- **Since**: 6.0

contact.authentication.method

- **Module**: cmas-core-security
- **Description**: Indicates contact authentication method, where possible values are DATABASE or LDAP or LDAP,DATABASE or DATABASE,LDAP.
- **Type**: string

- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: LDAP
- **Since**: 6.9.3.0

contact.inherit.permissions.only.to.own.customer.group

- **Module**: cmas-core-security
- **Description**: Indicates whether an authenticated contact inherits all customer group permissions from the CM/Track user profile ("false") or only has permissions to his own customer group ("true").
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.9.2.3

csrf.domain.allow.none

- **Module**: cmas-restapi-core
- **Description**: Determines whether empty `Origin/Referer` headers are accepted. By default, the property is set to "false", so that existing REST requests without `Origin/Referer` headers work.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false (default value)
- **Since**: 6.11.2.2

csrf.domain.white.list

- **Module**: cmas-restapi-core
- **Description**: The list of domains (separated with "|") which are allowed in `Origin/Referer` headers and will not be blocked by the CSRF (cross-site request forgery) filter. By default, the property is empty so that cross-site requests are blocked.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: https://example.com:80 | http://www.consol.de:8080
- **Since**: 6.11.2.2

csrf.domain.white.list

- **Module**: cmweb-server-adapter
- **Description**: The list of domains (separated with "|") which are allowed and will not be checked by CSRF (cross-site request forgery) filter
- **Type**: String
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: example.com | consol.de
- **Since**: 6.10.7.0

csrf.request.filter.enabled

- **Module**: cmas-restapi-core
- **Description**: It allows to disable the CSRF (cross-site request forgery) request filter for the REST API.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true (default value)
- **Since**: 6.11.2.2

csrf.request.filter.enabled

- **Module**: cmweb-server-adapter
- **Description**: It allows to disable CSRF (Cross-site request forgery) request filter
- **Type**: Boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true
- **Since**: 6.10.7.0

customfield.content.file.max.size

- **Module**: cmas-core-server
- **Description**: Determines the maximum size in MB of the images which can be added as content of rich text fields.
- **Type**: integer
- **Restart required**: no
- **System**: yes

- **Optional**: yes
- **Example value**: 10
- **Since**: 6.11.2.2

customizationVersion

- **Module**: cmweb-server-adapter
- **Description**: UID representing the latest web customization version. Used only internally, please do not change the value.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: cd58453e-f3cc-4538-8030-d15e8796a4a7
- **Since**: 6.5.0

dao.log.threshold.milliseconds

- **Module**: cmas-core-server
- **Description**: Used to configure database operation times logging. DAO methods whose execution takes longer than the time set in this property (in milliseconds) are logged.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 500 (default value)
- **Since**: 6.11.1.0

dao.log.username

- **Module**: cmas-core-server
- **Description**: Used to configure database operation times logging. The execution of DAO methods which are related to the user name stated in this property is logged. Only one user name can be provided. The value is empty by default.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: webadmin
- **Since**: 6.11.1.0

data.directory

- **Module**: cmas-core-shared
- **Description**: Directory for ConSol CM data (e.g., index)
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: C:\Users\user\cmas
- **Since**: 6.0

data.optimization

- **Module**: cmweb-server-adapter
- **Description**: Defines optimization to be applied on response data. So far, the following values are supported (for setting more than one value, separate values by '|'): MINIFICATION and COMPRESSION. MINIFICATION minifies HTML data by e.g. stripping whitespaces and comments. COMPRESSION applies gzip compression to HTTP response. (Note: If you are running in cluster mode and want to test different configurations in parallel, you can set different values for each cluster node by specifying property `data.optimization.nodeId` to override default property.)
- **Type**: string
- **Restart required**: COMPRESSION can be switched on/off without restart, MINIFICATION requires restart.
- **System**: yes
- **Optional**: yes
- **Example value**: MINIFICATION|COMPRESSION

database.notification.enabled

- **Module**: cmas-core-index-common
- **Description**: Indicates whether the notification channel “database” should be used for index updates instead of JMS. The default value is “true.”
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true
- **Since**: 6.8.4.7
- **Removed in**: 6.12.0.0

database.notification.redelivery.delay.seconds

- **Module**: cmas-core-index-common
- **Description**: If the notification channel “database” is used for index updates, this property indicates the delay for notification redelivery when an exception occurs.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 60
- **Since**: 6.8.4.7

database.notification.redelivery.max.attempts

- **Module**: cmas-core-index-common
- **Description**: In case of index update database notification channel, indicates maximum redelivery attempts when an exception occurs.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 60
- **Since**: 6.8.4.7

defaultAttachmentEntryClassName

- **Module**: cmweb-server-adapter
- **Description**: The default content entry class used to classify an attachment if no other class was set explicitly.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: DefaultTextElement
- **Since**: 6.9.2.0

defaultCommentClassName

- **Module**: cmas-core-server
- **Description**: Name of the default text class for comments.
- **Type**: string
- **Restart required**: no
- **System**: no

- **Optional**: yes
- **Example value**: default_class
- **Since**: 6.3.0

defaultContentEntryClassName

- **Module**: cmweb-server-adapter
- **Description**: Default text class for new ACIMs.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: default_class
- **Since**: 6.3.0

defaultIncommingMailClassName

- **Module**: cmas-core-server
- **Description**: Name of the default text class for incoming emails.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: default_class
- **Since**: 6.3.0

defaultNumberOfCustomFieldsColumns

- **Module**: cmweb-server-adapter
- **Description**: Default number of columns for ticket fields.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 3
- **Since**: 6.2.0

defaultOutgoingMailClassName

- **Module**: cmas-core-server
- **Description**: Name of the default text class for outgoing emails.
- **Type**: string
- **Restart required**: no

- **System**: no
- **Optional**: yes
- **Example value**: default_class
- **Since**: 6.3.0

delete.ticket.enabled

- **Module**: cmas-app-admin-tool
- **Description**: Controls if the menu entry *Delete* is displayed in the context menu in the Admin Tool for the ticket list in the ticket administration.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true
- **Since**: 6.9.4.0

diffTrackingEnabled

- **Module**: cmweb-server-adapter
- **Description**: Removed in ConSol CM version 6.11.
  Defines if parallel editing of a ticket by different engineers should be possible.
  “false”: Previous way of handling changes when editing a ticket. If the ticket has been changed in the meantime, the current engineer will not be able to submit his changes without being forced to reload the page before submitting.
  “true”: New changes handling mode. If the ticket has been changed, this will not block the submission of other changes anymore. If the part of the ticket that was changed was exactly the part that is changed by the submitting engineer, then an information message will be displayed, but the ticket change will be persisted/stored anyway.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true (default)
- **Since**: 6.10.1
- **Removed in**: 6.11.0

diffTrackingEnabledForUnitAndResource

- **Module**: cmweb-server-adapter
- **Description**: Enables the prevention of concurrent modifications on units / resources.
- **Type**: boolean
- **Restart required**: no
- **System**: no

- **Optional**: yes
- **Example value**: 3
- **Since**: 6.11.0.0

diff.tracking.disabled

- **Module**: cmas-restapi-core
- **Description**: Fallback property for disabling diff tracking for CM/Track, which is history-based so it can be heavy.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: false (default value)
- **Since**: 6.10.5.6

disable.admin.task.auto.commit

- **Module**: cmas-core-index-common
- **Description**: All tasks created for index update will be automatically executed right after creation.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.6.1

domain.map.for.client.config.<DOMAIN_NAME>

- **Module**: cmas-restapi-core
- **Description**: Enables mapping multiple instances of CM/Track to a specific domain. DOMAIN_NAME is the name of the client configuration in the Admin Tool (if your CM/Track configuration is called “trackV2customized”, the name of the property is `domain.map.for.client.config.trackV2customized`). You can provide several URLs separated by a comma.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: https://www.consol.de, https://www.consol.com
- **Since**: 6.10.7.0

dwh.administration.refresh.interval.seconds

- **Module**: cmas-app-admin-tool
- **Description**: Internal DWH property, not to be changed manually. This property is used to set the interval for refreshing the list of actions in the Admin Tool, section *Data Warehouse -> Administration -> Actions*.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 10
- **Since**: 6.11.0.1

dwh.mode

- **Module**: cmas-dwh-server
- **Description**: Current mode for DWH data transfer. Possible values are OFF, ADMIN, LIVE
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: OFF
- **Since**: 6.0.1

engineer.description.cache.enabled

- **Module**: cmas-core-server
- **Description**: Defines whether user descriptions are cached. The default value is “true”, please do not change it unless advised by ConSol.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true (default value)
- **Since**: 6.11.0

engineer.description.mode

- **Module**: cmas-core-server
- **Description**: Defines whether user names in the ticket history are taken from the database or dynamically rendered using templates. The default value “DYNAMIC” is a bit more costly from the performance perspective, while “PROTOCOL” is faster but returns historical names which might be outdated. Use “PROTOCOL” if you have lots of history entries from many different users.

- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: PROTOCOL
- **Since**: 6.11.0

engineer.description.template.name

- **Module**: cmas-core-server
- **Description**: Defines the name of the template which is used to render engineer names for display in the Web Client. The template has to be stored in the *Templates* section of the Admin Tool.
- **Type**: String
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: engineer description template name (default value)
- **Since**: 6.11.0

eviction.event.queue.size

- **Module**: cmas-core-cache
- **Description**: The size of the queue holding cache events. The default value is 200000. It is recommended to increase the value slightly (up to 400000) on systems with high traffic or load.
- **Type**: integer
- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: 200000 (default value)
- **Since**: 6.4.0

eviction.lifeSpan

- **Module**: cmas-core-cache
- **Description**: Sets the interval (in milliseconds) for cache entry validity. When this time is elapsed, the entry is removed from the cache.
- **Type**: integer
- **Restart required**: yes
- **System**: no
- **Optional**: yes
- **Example value**: 86400000 (default value)
- **Since**: 6.11.1.0

eviction.max.nodes

- **Module**: cmas-core-cache
- **Description**: Sets the maximum size of internal caches. The default value is 100000. Increasing it will lead to higher memory consumption and is not recommended unless explicitly advised by ConSol.
- **Type**: integer
- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: 100000 (default value)
- **Since**: 6.4.0

eviction.strategy

- **Module**: cmas-core-cache
- **Description**: Determines the cache eviction strategy. Please see the Infinispan documentation for details.
- **Type**: string
- **Restart required**: yes
- **System**: no
- **Optional**: yes
- **Example value**: LRU, LIRS (default value)
- **Since**: 6.11.1.0

eviction.wakeup.interval

- **Module**: cmas-core-cache
- **Description**: Sets the interval (in milliseconds) between two cache queue event processing cycles. The default value is 3000. It is recommended to decrease it (minimum is 1500) on systems with high traffic or load.
- **Type**: integer
- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: 3000 (default value)
- **Since**: 6.4.0

expert.mode

- **Module**: cmas-core-shared
- **Description**: Switches expert mode on/off thereby unblocking/blocking expert features. For example, the CM system property `initialized` is only available in the expert mode.
- **Type**: boolean

- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.0

external.line.access.prefix

- **Module**: cmas-core-server
- **Description**: General prefix to dial before an area code. Set for each customer group separately.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 411
- **Since**: 6.9.3.0

favoritesSizeLimit

- **Module**: cmweb-server-adapter
- **Description**: Maximum number of items in Favorites list.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 10
- **Since**: 6.0

fetchLock.interval

- **Module**: cmas-workflow-jbpm
- **Description**:
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 5000
- **Removed in**: 6.8.0

fetchSize.strategy

- **Module**: cmas-core-server
- **Description**: Strategy for selecting the fetch size on JDBC result sets.

- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: FetchSizePageBasedStrategy, FetchSizeThresholdStrategy, FetchSizeFixedStrategy
- **Since**: 6.8.4.1

fetchSize.strategy.FetchSizeFixedStrategy.value

- **Module**: cmas-core-server
- **Description**: Sets fetch size value if the selected strategy to set the fetch size is `FetchSizeFixedStrategy`.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 150
- **Since**: 6.8.4.1

fetchSize.strategy.FetchSizePageBasedStrategy.limit

- **Module**: cmas-core-server
- **Description**: Sets maximum fetch size value if the selected strategy to set the fetch size is `FetchSizePageBasedStrategy`.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 10000
- **Since**: 6.8.4.1

fetchSize.strategy.FetchSizeThresholdStrategy.value

- **Module**: cmas-core-server
- **Description**: Sets fetch size threshold border values if the selected strategy to set the fetch size is `FetchSizeThresholdStrategy`.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 150,300,600,1000
- **Since**: 6.8.4.1

filesystem.polling.threads.number

- **Module**: cmas-nimh
- **Description**: Number of threads started for database emails' queue polling. The default value is 1
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 10
- **Since**: 6.4.0

filesystem.polling.threads.shutdown.timeout.seconds

- **Module**: cmas-nimh
- **Description**: Waiting time after the shutdown signal. When the timeout is reached, the thread will be terminated.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60 (default value)
- **Since**: 6.4.0

filesystem.polling.threads.watchdog.interval.seconds

- **Module**: cmas-nimh
- **Description**: Determines the interval in seconds to execute the watchdog which checks the activity of the threads of the file system poller.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 30 (default value)
- **Since**: 6.4.0

filesystem.task.enabled

- **Module**: cmas-nimh
- **Description**: With this property the service thread related to a given poller can be disabled.
- **Type**: boolean
- **Restart required**: no
- **System**: no

- **Optional**: yes
- **Example value**: true (default value)
- **Since**: 6.4.0

filesystem.task.interval.seconds

- **Module**: cmas-nimh
- **Description**: Default interval for polling mailboxes in seconds.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60 (default value)
- **Since**: 6.4.0

filesystem.task.polling.folder

- **Module**: cmas-nimh
- **Description**: Polling folder location which will be scanned for emails in the format of eml files. The default value is the `mail` subdirectory of the ConSol CM data directory
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: c://cmas//mail
- **Since**: 6.4.0

filesystem.task.timeout.seconds

- **Module**: cmas-nimh
- **Description**: After this time (of inactivity) the service thread is considered damaged and automatically restarted. Default: 120 seconds
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.4.0

filesystem.task.transaction.timeout.seconds

- **Module**: cmas-nimh
- **Description**: Default transaction timeout (in seconds) for email fetching transactions. Should be correlated with number of messages fetched at once.

- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60 (default value)
- **Since**: 6.4.0

forward.mails.to.representatives

- **Module**: cmweb-server-adapter
- **Description**: Determines if emails which are manually sent from the Web Client are also sent to representing engineers. The default value of the property is “false”, meaning that this kind of emails are not forwarded to the representing engineer. Set the property to “true” if you want to restore the previous behavior, i.e., all emails which are sent to the represented engineer are automatically forwarded to the representing engineer. Please take into account that this might not be desired if the same person is an engineer and a customer in the CM system.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: false (default value)
- **Since**: 6.11.1.7

> This property only configures the handling of manually sent emails. The handling of automatically sent emails depends on the used Java method.

globalSearchResultSizeLimit

- **Module**: cmweb-server-adapter
- **Description**: Maximum number of items in Quick Search result.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 10
- **Since**: 6.0

heartbeat

- **Module**: cmas-core-server
- **Description**: Timestamp that indicates if an instance of the application is connected to the database schema.
- **Type**: integer

- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 1234567899
- **Since**: 6.10.5.3

helpFilePath

- **Module**: cmweb-server-adapter
- **Description**: URL for online help. If not empty, the *Help* link is displayed in the Web Client.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: http://www.consol.de
- **Since**: 6.2.1

hibernate.dialect

- **Module**: cmas-setup-hibernate
- **Description**: The dialect used by hibernate. Usually set during initial set-up (depending on the database system).
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: org.hibernate.dialect.MySQL5InnoDBDialect
- **Since**: 6.0

hideTicketSubject

- **Module**: cmweb-server-adapter
- **Description**: If set to “true”, the ticket subject is hidden.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.2.1

ignore-queues

- **Module**: cmas-dwh-server
- **Description**: A comma-separated list of queue names which are not not transferred to the DWH.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: QueueName1,QueueName2,QueueName3
- **Since**: 6.6.19
- **Removed in**: 6.8.1

index.attachment

- **Module**: cmas-core-index-common
- **Description**: Specifies whether the content of attachments is indexed.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true
- **Since**: 6.4.3

index.history

- **Module**: cmas-core-index-common
- **Description**: Specifies whether unit and ticket history are indexed.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.1.0
- **Removed in**: 6.11.0

index.status

- **Module**: cmas-core-index-common
- **Description**: Status of the Indexer, possible values RED, YELLOW, GREEN, will be displayed in the Admin Tool.
- **Type**: string
- **Restart required**: no

- **System**: yes
- **Optional**: no
- **Example value**: GREEN
- **Since**: 6.6.1

index.task.worker.threads

- **Module**: cmas-core-index-common
- **Description**: How many threads will be used to execute index tasks (synchronization, administrative and repair tasks). We recommend to use a value not larger than 2.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 1 (default value)
- **Since**: 6.6.14, 6.7.3. Since 6.8.0 and exclusively in 6.6.21 also normal (live) index updates are affected by this property.

index.version.current

- **Module**: cmas-core-index-common
- **Description**: Holds information about the current (possibly old) index version.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 1 (default value)
- **Since**: 6.7.0

index.version.newest

- **Module**: cmas-core-index-common
- **Description**: Holds information about which index version is considered newest.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 1 (default value)
- **Since**: 6.7.0

indexed.assets.per.thread.in.memory

- **Module**: cmas-core-index-common
- **Description**: Determines how many assets should be loaded into memory at once, per thread, during indexing.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 200 (default value)
- **Since**: 6.8.0

indexed.engineers.per.thread.in.memory

- **Module**: cmas-core-index-common
- **Description**: Determines how many engineers should be loaded into memory at once, per thread, during indexing.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 300 (default value)
- **Since**: 6.6.14, 6.7.3

indexed.resources.per.thread.in.memory

- **Module**: cmas-core-index-common
- **Description**: Determines how many resources should be loaded into memory at once, per thread, during indexing.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 200 (default value)
- **Since**: 6.10.0.0

indexed.tickets.per.thread.in.memory

- **Module**: cmas-core-index-common
- **Description**: Determines how many tickets should be loaded into memory at once, per thread, during indexing.
- **Type**: integer
- **Restart required**: no
- **System**: yes

- **Optional**: no
- **Example value**: 100 (default value)
- **Since**: 6.6.14, 6.7.3

indexed.units.per.thread.in.memory

- **Module**: cmas-core-index-common
- **Description**: Determines how many units should be loaded into memory at once, per thread, during indexing.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 200 (default value)
- **Since**: 6.6.14, 6.7.3

initialized

- **Module**: cmas-setup-manager
- **Description**: Flag if ConSol CM is initialized. If this value is missing or not “true”, the setup will be performed. Starting with ConSol CM version 6.11, this property is only available in expert mode.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true
- **Since**: 6.0

> ⚠ Be careful with using this property! When you set the value to “false”, the ConSol CM server will perform the system setup at the next start, i.e. all data of the existing system is lost, including system properties!

internal.line.access.prefix

- **Module**: cmas-core-server
- **Description**: Prefix that the company’s telephony system requires for outside lines. Set for each customer group separately.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes

- **Example value**: 199
- **Since**: 6.9.3.0

is.cmrf.alive

- **Module**: cmas-dwh-server
- **Description**: As a starting point, the time the last message was sent to CMRF should be used. If a response from CMRF is not received after the set value (in seconds), it should create a DWH operation status with an error message indicating that CMRF is down.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 1200
- **Since**: 6.7.0

java.naming.factory.initial

- **Module**: cmas-dwh-server
- **Description**: Factory class for the DWH context factory.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: org.jnp.interfaces.NamingContextFactory
- **Since**: 6.0.1
- **Removed in**: 6.11.0.0

java.naming.factory.url.pkgs

- **Module**: cmas-dwh-server
- **Description**:
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: org.jboss.naming:org.jnp.interfaces
- **Since**: 6.0.1
- **Removed in**: 6.11.0.0

java.naming.provider.url

- **Module**: cmas-dwh-server
- **Description**: URL of naming provider.

- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: localhost
- **Since**: 6.0.1
- **Removed in**: 6.11.0.0

jobExecutor.adminMail

- **Module**: cmas-workflow-engine
- **Description**: Email address which will get notified about job execution problems (when retry counter is exceeded).
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: admin@consol.de
- **Since**: 6.8.0

jobExecutor.idleInterval

- **Module**: cmas-workflow-jbpm
- **Description**:
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 45000
- **Removed in**: 6.8.0
- **Replaced by**: jobExecutor.idleInterval.seconds

jobExecutor.idleInterval.seconds

- **Module**: cmas-workflow-engine
- **Description**: Determines how often the job executor thread will look for new jobs to execute.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 45 (default value up to CM version 6.10.5.2. The default value for CM versions

6.10.5.3 and up is 5)

- **Since**: 6.8.0

jobExecutor.jobExecuteRetryNumber

- **Module**: cmas-workflow-jbpm
- **Description**:
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 5
- **Removed in**: 6.8.0
- **Replaced by**: jobExecutor.jobMaxRetries

jobExecutor.jobMaxRetries

- **Module**: cmas-workflow-engine
- **Description**: Controls the number of retry attempts the job executor will do before declaring a job as failed.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 5 (default value)
- **Since**: 6.8.0

jobExecutor.jobMaxRetriesReachedSubject

- **Module**: cmas-workflow-engine
- **Description**: The subject used in the notification mail which administrators receive about failed job executors.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: Job maximum retries reached. Job was removed!!! (default value)
- **Since**: 6.8.0

jobExecutor.lockingLimit

- **Module**: cmas-workflow-engine
- **Description**: Number of jobs locked at once (marked for execution) by the job executor thread.
- **Type**: integer

- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 5 (default value since CM version 6.10.5.3)
- **Since**: 6.8.0

jobExecutor.lockTimeout.seconds

- **Module**: cmas-workflow-engine
- **Description**: Determines how long the job can be locked (marked for execution) by the job executor.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 360 (default value)
- **Since**: 6.8.0

jobExecutor.mailFrom

- **Module**: cmas-workflow-engine
- **Description**: Email address which will be set as From header for notifications to the administrator.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: jobexecutor@consol.de
- **Since**: 6.8.0

jobExecutor.maxInactivityInterval.minutes

- **Module**: cmas-workflow-engine
- **Description**: Number of minutes of allowed job executor inactivity (e.g. when it is blocked by long timer execution). After this time executors threads are restarted.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 30 (default value)
- **Since**: 6.9.2.0

jobExecutor.threads

- **Module**: cmas-workflow-engine
- **Description**: Number of job execution threads.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 1 (default value)
- **Since**: 6.8.0

jobExecutor.timerRetryInterval

- **Module**: cmas-workflow-jbpm
- **Description**:
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 10000
- **Removed in**: 6.8.0
- **Replaced by**: jobExecutor.timerRetryInterval.seconds

jobExecutor.timerRetryInterval.seconds

- **Module**: cmas-workflow-engine
- **Description**: Determines how long the job executor thread will wait after job execution error.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 10 (default up to CM version 6.10.5.2. The default value for CM versions 6.10.5.3 and up is 30)
- **Since**: 6.8.0

jobExecutor.txTimeout.seconds

- **Module**: cmas-workflow-engine
- **Description**: Transaction timeout used for job execution.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes

- **Example value**: 60 (default value)
- **Since**: 6.8.0

kerberos.v5.enabled

- **Module**: cmas-core-security
- **Description**: Indicates whether SSO via Kerberos is enabled.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false (default value if Kerberos was not enabled during system setup)
- **Since**: 6.2.0

kerberos.v5.username.regex

- **Module**: cmas-core-security
- **Description**: Regular expression used for mapping Kerberos principals to CM user login names.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: (.*)@.*
- **Since**: 6.2.0

last.config.change

- **Module**: cmas-core-server
- **Description**: Random UUID created during the last configuration change. This is a value maintained internally, please do not change it unless advised by ConSol.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 2573c7b7-2bf5-47ff-b5a2-bad31951a266
- **Since**: 6.1.0, 6.2.1

last.config.change.templates

- **Module**: cmas-core-server
- **Description**: Random UUID created during the last change in templates. This is a value maintained internally, please do not change it unless advised by ConSol.
- **Type**: string
- **Restart required**: no

- **System**: yes
- **Optional**: no
- **Example value**: 2573c7c7-2af5-4eff-b9c2-bad31951a266
- **Since**: 6.10.5.0

last.ping.timestamp

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property shows the status of CMRF and is filled automatically by the CMRF. It contains the date of the last response on the ping from the ConSol CM server to the CMRF.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 32323214
- **Since**: 6.11.0.1

ldap.authentication

- **Module**: cmas-core-security
- **Description**: Authentication method used when using LDAP authentication. Possible values are “anonymous” and “simple”.
- **Type**: string
- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: simple (default value)
- **Since**: 6.0

ldap.basedn

- **Module**: cmas-core-security
- **Description**: Base DN used for looking up LDAP user accounts when using LDAP authentication.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: ou=accounts,dc=consol,dc=de
- **Since**: 6.0

ldap.certificate.basedn

- **Module**: cmas-core-server
- **Description**: Base DN for certificates location in the LDAP tree. If not provided, `cmas-core-security, ldap.basedn` is used.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: ou=accounts,dc=consol,dc=de
- **Since**: 6.8.4

ldap.certificate.content.attribute

- **Module**: cmas-core-server
- **Description**: LDAP attribute name used where certificate data is stored in the LDAP tree.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: usercertificate (default value)
- **Since**: 6.8.4

ldap.certificate.password

- **Module**: cmas-core-server
- **Description**: LDAP Certificates manager password. If not set, `cmas-core-security, ldap.password` is used.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.8.4

ldap.certificate.providerurl

- **Module**: cmas-core-server
- **Description**: LDAP Certificates provider URL. If not set, `cmas-core-security, ldap.providerurl` is used.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes

- **Example value**: ldap://ldap.consol.de:389
- **Since**: 6.8.4

ldap.certificate.searchattr

- **Module**: cmas-core-server
- **Description**: LDAP attribute name used to search for certificate in the LDAP tree.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: mail (default value)
- **Since**: 6.8.4

ldap.certificate.userdn

- **Module**: cmas-core-server
- **Description**: LDAP Certificates manager DN. If not set, `cmas-core-security, ldap.userdn` is used.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.8.4

ldap.contact.name.basedn

- **Module**: cmas-core-security
- **Description**: Base path to search for contact DN by LDAP ID.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: ou=accounts,dc=consol,dc=de
- **Since**: 6.9.3.0

ldap.contact.name.password

- **Module**: cmas-core-security
- **Description**: Password to look up contact DN by LDAP ID. If not set, the anonymous account is used.
- **Type**: string
- **Restart required**: no
- **System**: no

- **Optional**: yes
- **Since**: 6.9.3.0

ldap.contact.name.providerurl

- **Module**: cmas-core-security
- **Description**: Address of the LDAP server (ldap[s]://host:port).
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: ldap://ldap.consol.de:389
- **Since**: 6.9.3.0

ldap.contact.name.searchattr

- **Module**: cmas-core-security
- **Description**: Attribute to search for contact DN by LDAP ID.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: uid
- **Since**: 6.9.3.0

ldap.contact.name.userdn

- **Module**: cmas-core-security
- **Description**: User DN to look up contact DN by LDAP ID. If not set, the anonymous account is used.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Since**: 6.9.3.0

ldap.initialcontextfactory

- **Module**: cmas-core-security
- **Description**: Class name for the initial context factory of the LDAP implementation when using LDAP authentication. If it is not set, `com.sun.jndi.ldap.LdapCtxFactory` is used.
- **Type**: string
- **Restart required**: yes
- **System**: yes

- **Optional**: no
- **Example value**: com.sun.jndi.ldap.LdapCtxFactory
- **Since**: 6.0

ldap.password

- **Module**: cmas-core-security
- **Description**: Password for connecting to LDAP to look up users when using LDAP authentication. Only needed if look-up cannot be performed anonymously.
- **Type**: password
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.1.2

ldap.providerurl

- **Module**: cmas-core-security
- **Description**: LDAP provider when using LDAP authentication.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: ldap://myserver.consol.de:389
- **Since**: 6.0

ldap.searchattr

- **Module**: cmas-core-security
- **Description**: Search attribute for looking up LDAP entry associated with a CM login.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: uid
- **Since**: 6.0

ldap.userdn

- **Module**: cmas-core-security
- **Description**: LDAP user for connecting to LDAP to look up users when using LDAP authentication. Only needed if look-up cannot be performed anonymously.
- **Type**: string
- **Restart required**: no

- **System**: yes
- **Optional**: yes
- **Since**: 6.1.2

live.start

- **Module**: cmas-dwh-server
- **Description**: When the DWH synchronization mode is set to LIVE using the Admin Tool (navigation group *Data Warehouse*, navigation item *Administration*, *Configuration* button), this property is created and set to the current date.
  If LIVE mode is not enabled and there is no data in `cmas_dwh_ser_sync_object`, the property `live.start` is deleted.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes (automatically added in DWH "LIVE" mode)
- **Example value**: 15028802377645
- **Since**: 6.7.0

local.country.prefix

- **Module**: cmas-core-server
- **Description**: Prefix of the local country code. Set for each customer group separately.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 48
- **Since**: 6.9.3.0

mail.attachments.validation.info.sender

- **Module**: cmas-nimh-extension
- **Description**: Sets the From header for error notification emails which are sent when the validation of the attachment type fails.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: admin@mail.com
- **Since**: 6.7.5

mail.attachments.validation.info.subject

- **Module**: cmas-nimh-extension
- **Description**: Sets the subject for error notification emails which are sent when the validation of the attachment type fails.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: Email was not processed because its attachments were rejected!
- **Since**: 6.7.5

mail.db.archive

- **Module**: cmas-nimh-extension
- **Description**: If property is set to “true”, incoming emails are archived in the database.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: false (default value)
- **Since**: 6.8.5.5

mail.encryption

- **Module**: cmas-core-server
- **Description**: If the property is set to “true”, the encrypt checkbox in the Ticket Email Editor is checked by default. The default value of the property is “false”.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true
- **Since**: 6.8.4.0

mail.error.from.address

- **Module**: cmas-nimh-extension
- **Description**: From address for error emails from NIMH
- **Type**: email
- **Restart required**: no
- **System**: yes
- **Optional**: no

- **Example value**: myuser@consol.de
- **Since**: 6.4.0

mail.error.to.address

- **Module**: cmas-nimh-extension
- **Description**: To address for error emails from NIMH. As a default the email address of the administrator which you have entered during system setup is used.
- **Type**: email
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: myuser@consol.de
- **Since**: 6.4.0

mail.from

- **Module**: cmweb-server-adapter
- **Description**: This email address is used instead of the engineer's email address during email conversations.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: consolcm@example.com
- **Since**: 6.1.2

mail.notification.engineerChange

- **Module**: cmas-core-server
- **Description**: Determines whether notification emails should be sent when the engineer of a ticket is changed.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true
- **Since**: 6.1.0

mail.notification.sender

- **Module**: cmas-core-server
- **Description**: From address for notification emails when the engineer of a ticket is changed. If not set, `cmas-core-security, admin.email` is used instead.

- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: cm6notification@cm6installation
- **Since**: 6.6.3

mail.on.error

- **Module**: cmas-nimh-extension
- **Description**: If set to “true” an error email is sent to the above configured address in case the email message could not be processed.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true (default)
- **Since**: 6.4.0

mail.redelivery.retry.count

- **Module**: cmas-core-server
- **Description**: Number of redelivery attempts of an outgoing email.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 3
- **Since**: 6.1.0

mail.reply.to

- **Module**: cmweb-server-adapter
- **Description**: When set, the Ticket Email Editor in the Web Client displays a Reply-To field pre-filled with this value.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: consolcm@example.com
- **Since**: 6.0.1

> ⚠ Please read the detailed information about ConSol CM Reply-To addresses in section *Scripts of Type Email* in the *ConSol CM Administrator Manual*.

mail.sender.address

- **Module**: cmas-workflow-jbpm
- **Description**: From address for emails from the workflow engine.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: myuser@consol.de
- **Removed in**: 6.8.0
- **Replaced by**: jobExecutor.mailFrom

mail.smtp.email

- **Module**: cmas-core-server
- **Description**: SMTP email URL for outgoing emails
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: smtp://mail.mydomain.com:25
- **Since**: 6.0

mail.smtp.email.password

- **Module**: cmas-core-server
- **Description**: SMTP password for outgoing email communication. If this property is set, it takes precedence over the value from the SMTP URL.
- **Type**: password
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: password
- **Since**: 6.11.2.2

mail.smtp.email.user

- **Module**: cmas-core-server
- **Description**: SMTP user name for outgoing email communication. If this property is set, it takes precedence over the value from the SMTP URL.

- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: myuser
- **Since**: 6.11.2.2

mail.smtp.envelopesender

- **Module**: cmas-core-server
- **Description**: Email address used as sender in SMTP envelope. If not set, the From address of the email is used.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: mysender@mydomain.com
- **Since**: 6.5.7

mail.smtp.tls.enabled

- **Module**: cmas-core-server
- **Description**: Activates SMTP via SSL/TLS (SMTPS) for sending emails from the Web Client and scripts. The default value is “false”. If it is set to “true”, SMTPS is activated for sending emails.
- **Type**: boolean
- **Restart required**: yes
- **System**: no
- **Optional**: yes
- **Example value**: true
- **Since**: 6.11.1.6

mail.ticketname.pattern

- **Module**: cmas-nimh-extension
- **Description**: Regular expression pattern used to identify the ticket name in the subject of incoming mails.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: .*?Ticket\s+\((\S+)\).*
- **Since**: 6.4.0

mailbox.<NUMBER>.name

- **Module**: cmas-nimh
- **Description**: Identifier (name) of the mailbox.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: myEmailAccount
- **Since**: 6.11.2.0

mailbox.1.connection.host

- **Module**: cmas-nimh
- **Description**: Host (server) for first configured mailbox. Will overwrite the default parameter `mailbox.default.connection.host`.

mailbox.1.connection.password

- **Module**: cmas-nimh
- **Description**: Password for first configured mailbox. Will overwrite the default parameter `mailbox.default.connection.password`.

mailbox.1.connection.port

- **Module**: cmas-nimh
- **Description**: Port for first configured mailbox. Will overwrite the default parameter `mailbox.default.connection.port`.

mailbox.1.connection.protocol

- **Module**: cmas-nimh
- **Description**: Protocol (e.g., IMAP or POP3) for first configured mailbox. Will overwrite the default parameter `mailbox.default.connection.protocol`.

mailbox.1.connection.username

- **Module**: cmas-nimh
- **Description**: User name for first configured mailbox. Will overwrite the default parameter `mailbox.default.connection.username`.

mailbox.2.connection.host

- **Module**: cmas-nimh
- **Description**: Host (server) for second configured mailbox. Will overwrite the default parameter `mailbox.default.connection.host`.

mailbox.2.connection.password

- **Module**: cmas-nimh
- **Description**: Password for second configured mailbox. Will overwrite the default parameter `mailbox.default.connection.password`.

mailbox.2.connection.port

- **Module**: cmas-nimh
- **Description**: Port for second configured mailbox. Will overwrite the default parameter `mailbox.default.connection.port`.

mailbox.2.connection.protocol

- **Module**: cmas-nimh
- **Description**: Protocol (e.g., IMAP or POP3) for second configured mailbox. Will overwrite the default parameter `mailbox.default.connection.protocol`.

mailbox.2.connection.username

- **Module**: cmas-nimh
- **Description**: User name for second configured mailbox. Will overwrite the default parameter `mailbox.default.connection.username`.

> For all NIMH-related mailbox properties, the following principle is used: a default property is defined (e.g. `mailbox.default.connection.port`). If no mailbox-specific value is configured, this default value will be used.

mailbox.default.connection.host

- **Module**: cmas-nimh
- **Description**: Host (server name) of a given mailbox from which the poller reads emails.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 10.10.1.157
- **Since**: 6.4.0

mailbox.default.connection.password

- **Module**: cmas-nimh
- **Description**: Password for given mailbox from which the poller reads emails.
- **Type**: password
- **Restart required**: no
- **System**: no
- **Optional**: yes

- **Example value**: consol
- **Since**: 6.4.0

mailbox.default.connection.port

- **Module**: cmas-nimh
- **Description**: Port for a given mailbox from which the poller reads emails.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 143
- **Since**: 6.4.0

mailbox.default.connection.protocol

- **Module**: cmas-nimh
- **Description**: Poller's protocol e.g., IMAP or POP3. No default value
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: imap
- **Since**: 6.4.0

mailbox.default.connection.username

- **Module**: cmas-nimh
- **Description**: User name for a given mailbox from which the poller reads emails.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: username
- **Since**: 6.4.0

mailbox.default.session.mail.debug

- **Module**: cmas-nimh
- **Description**: JavaMail property, see JavaMail API documentation for further information. Allows for more detailed JavaMail session debugging.
- **Type**: boolean
- **Restart required**: no

- **System**: no
- **Optional**: yes
- **Example value**: true
- **Since**: 6.4.0

mailbox.default.session.mail.mime.address.strict

- **Module**: cmas-nimh
- **Description**: JavaMail property, see JavaMail API documentation for further information. Counterpart of the old `mule mail.mime.strict`, allows to set not so strict email header parsing.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true
- **Since**: 6.4.0

mailbox.default.session.mail.<PROTOCOL>.connectiontimeout

- **Module**: cmas-nimh
- **Description**: JavaMail property, see JavaMail API documentation for further information. Determines the connection timeout in milliseconds.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 3000
- **Since**: 6.4.0 (IMAP and POP3) / 6.11.2.0 (IMAPS and POP3S)

mailbox.default.session.mail.<PROTOCOL>.fetchsize

- **Module**: cmas-nimh
- **Description**: JavaMail property, see JavaMail API documentation for further information. Determines the size of partial fetch in bytes for the indicated protocol.
  For IMAP systems: in CM versions 6.10.7.0 and up, the value of `mailbox.default.session.mail.imap.fetchsize` is set to 1048576 (equals 1 MB) during the initial setup of a ConSol CM system. During an update of an existing ConSol CM system, the value of the property is left unchanged, if the property is already present. In case the property is not yet present, it is added with the default value.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes

- **Example value**: 1048576
- **Since**: 6.9.4.0 (IMAP and POP3) / 6.11.2.0 (IMAPS and POP3S)

mailbox.default.session.mail.<PROTOCOL>.partialfetch

- **Module**: cmas-nimh
- **Description**: JavaMail property, see JavaMail API documentation for further information. Determines whether the protocol's partial fetch capability should be used.
  For IMAP systems: in CM versions 6.10.7.0 and up, the value of `mailbox.default.session.mail.imap.partialfetch` is set to "false" during the initial setup of a ConSol CM system. During an update of an existing ConSol CM system, the value of the property is left unchanged, if the property is already present. In case the property is not yet present, it is added with the default value.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true
- **Since**: 6.9.4.0 (IMAP and POP3) / 6.11.2.0 (IMAPS and POP3S)

mailbox.default.session.mail.<PROTOCOL>.timeout

- **Module**: cmas-nimh
- **Description**: JavaMail property, see JavaMail API documentation for further information. Determines the I/O timeout in milliseconds.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 120 for IMAP(S) and 3000 for POP3(S)
- **Since**: 6.4.0 (IMAP and POP3), 6.11.2.0 (IMAPS and POP3S)

mailbox.default.task.delete.read.messages

- **Module**: cmas-nimh
- **Description**: This defines whether messages should be removed from the mailbox after processing. For IMAP protocol messages are marked as SEEN by default. For POP3 protocol, when flag is set to "true" the message is removed, otherwise remains on server and will result in infinite reads.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes

- **Example value**: false (default value)
- **Since**: 6.4.0

mailbox.default.task.enabled

- **Module**: cmas-nimh
- **Description**: With this property, the service thread related to the given poller can be disabled.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true (default value)
- **Since**: 6.4.0

mailbox.default.task.interval.seconds

- **Module**: cmas-nimh
- **Description**: Default interval for polling mailboxes in seconds.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60 (default value)
- **Since**: 6.4.0

mailbox.default.task.max.message.size

- **Module**: cmas-nimh
- **Description**: Maximum size of email messages (i.e., email plus attachment). Emails exceeding the size limit will not be automatically processed by NIMH but will be stored in the database (table `cmas_nimh_archived_mail`) and will therefore appear in the email backups in the Admin Tool (see section *Email Backups* in the *ConSol CM Administrator Manual*). From there they can be resent, downloaded to the file system, or deleted. For those operations the message size is not relevant.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 10485760 (default value, 10 MB)
- **Since**: 6.4.0

mailbox.default.task.max.messages.per.run

- **Module**: cmas-nimh
- **Description**: Number of messages fetched at once from mailbox. Must be correlated with transaction timeout. The default value is 20
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.4.0

mailbox.default.task.timeout.seconds

- **Module**: cmas-nimh
- **Description**: After this time (of inactivity) the service thread is considered damaged and automatically restarted. The default value is 120 seconds.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.4.0

mailbox.default.task.transaction.timeout.seconds

- **Module**: cmas-nimh
- **Description**: Default transaction timeout (in seconds) for email fetching transactions. Should be correlated with number of messages fetched at once.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60 (default value)
- **Since**: 6.4.0

mailbox.polling.threads.mail.log.enabled

- **Module**: cmas-nimh
- **Description**: Enables email logging which is especially crucial in cluster environment (used as semaphore there)
- **Type**: boolean
- **Restart required**: no
- **System**: no

- **Optional**: yes
- **Example value**: true (default value)
- **Since**: 6.9.4.1

mailbox.polling.threads.number

- **Module**: cmas-nimh
- **Description**: Number of threads for accessing mailboxes.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 1 (default value)
- **Since**: 6.4.0

mailSender.executionInterval.seconds

- **Module**: cmas-core-server
- **Description**: Determines the period in seconds before the system checks if there are new emails to be sent.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 10
- **Since**: 6.12.0.0

mailSender.lockingLimit

- **Module**: cmas-core-server
- **Description**: Determines the number of emails fetched from the database for sending in a single transaction.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 1
- **Since**: 6.12.0.0

mailSender.lockTimeout.seconds

- **Module**: cmas-core-server
- **Description**: Determines the period in seconds before a locked email is released for another lock.

- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 199
- **Since**: 6.12.0.0

mailSender.maxAttempts

- **Module**: cmas-core-server
- **Description**: Determines the maximum number of attempts, including the first attempt, before an email is removed.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 3
- **Since**: 6.12.0.0

mailSender.nodeId

- **Module**: cmas-core-server
- **Description**: Determines the ID of the node on which the thread for sending emails is started (only in clustered environments). If the value is “all” or no value is provided, the threads are started on every node.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: all
- **Since**: 6.12.0.0

mailSender.retryInterval.seconds

- **Module**: cmas-core-server
- **Description**: Determines the period in seconds before another attempt to send an email is made, if the previous attempt failed.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.12.0.0

mailSender.threads.number

- **Module**: cmas-core-server
- **Description**: Determines the number of threads for sending emails.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 1
- **Since**: 6.12.0.0

mailSender.txTimeout.seconds

- **Module**: cmas-core-server
- **Description**: Determines the transaction timeout for sending emails in seconds.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.12.0.0

mailTemplateAboveQuotedText

- **Module**: cmweb-server-adapter
- **Description**: Determines the behavior of the email template in the Ticket Email Editor when another email is quoted, i.e. forwarded or replied to. Often used to place the signature correctly.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.2.4

max.licences.perUser

- **Module**: cmas-core-server
- **Description**: Sets the maximum number of licenses which a single user can use (e.g., logging in from different browsers). By default this value is not restricted.
- **Type**: integer
- **Restart required**: no
- **System**: yes

- **Optional**: yes
- **Example value**: 10
- **Since**: 6.8.4.5

maxSizePerPagemapInMegaBytes

- **Module**: cmweb-server-adapter
- **Description**: The parameter defines the size (in MB) of the file which is created by the Wicket framework per user session. i.e. for each engineer which is currently logged in. The file is used to save pages during the running session. When the defined size limit has been reached and new entries are added, the oldest entries are removed. In the Web Client, due to this behavior, an engineer who works with an "old" page will be redirected to the *Overview*/Start page (usually the dashboard page) when the "old" page is removed from the file.
  So in case engineers who work with a great number of open tabs in ConSol CM and complain about being redirected to the *Overview* page, it might be useful to increase this parameter. In large systems, you could use e.g. a value of 45 or 50. Since this is the size of the file which is saved on disk, the maximum value depends on the available disk space, however, a value which is too large is not recommended either.
- **Type**: integer
- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: 15
- **Since**: 6.3.5

monitoring.engineer.login

- **Module**: cmas-core-server
- **Description**: Login of monitoring engineer.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: nagios
- **Since**: 6.9.3.0

monitoring.unit.login

- **Module**: cmas-core-server
- **Description**: Login of monitoring unit.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes

- **Example value**: nagios
- **Since**: 6.9.3.0

nimh.enabled

- **Module**: cmas-core-server
- **Description**: Enables the NIMH service. Must be suffixed with the cluster node ID, e.g., `nimh.enabled.NODEID` = “true”.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: false
- **Since**: 6.9.4.0

notification.error.description

- **Module**: cmas-dwh-server
- **Description**: Text for error emails from the DWH.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: Error occurred
- **Since**: 6.0.1

notification.error.from

- **Module**: cmas-dwh-server
- **Description**: From address for error emails from the DWH
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: myuser@consol.de
- **Since**: 6.0.1

notification.error.subject

- **Module**: cmas-dwh-server
- **Description**: Subject for error emails from the DWH
- **Type**: string
- **Restart required**: no

- **System**: yes
- **Optional**: no
- **Example value**: Error occurred
- **Since**: 6.0.1

notification.error.to

- **Module**: cmas-dwh-server
- **Description**: To address for error emails from the DWH
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: myuser@consol.de
- **Since**: 6.0.1

notification.finished_successfully.description

- **Module**: cmas-dwh-server
- **Description**: Text for emails from the DWH when a transfer finishes successfully.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: Transfer finished successfully
- **Since**: 6.0.1

notification.finished_successfully.from

- **Module**: cmas-dwh-server
- **Description**: From address for emails from the DWH when a transfer finishes successfully.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: myuser@consol.de
- **Since**: 6.0.1

notification.finished_successfully.subject

- **Module**: cmas-dwh-server
- **Description**: Subject for emails from the DWH when a transfer finishes successfully.
- **Type**: string

- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: Transfer finished successfully
- **Since**: 6.0.1

notification.finished_successfully.to

- **Module**: cmas-dwh-server
- **Description**: To address for emails from the DWH when a transfer finishes successfully.
- **Type**: string
- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: myuser@consol.de
- **Since**: 6.0.1

notification.finished_unsuccessfully.description

- **Module**: cmas-dwh-server
- **Description**: Text for emails from the DWH when a transfer finishes unsuccessfully.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: Transfer finished unsuccessfully
- **Since**: 6.0.1

notification.finished_unsuccessfully.from

- **Module**: cmas-dwh-server
- **Description**: From address for emails from the DWH when a transfer finishes unsuccessfully.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: myuser@consol.de
- **Since**: 6.0.1

notification.finished_unsuccessfully.subject

- **Module**: cmas-dwh-server
- **Description**: Subject for emails from the DWH when a transfer finishes unsuccessfully.

- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: Transfer finished unsuccessfully
- **Since**: 6.0.1

notification.finished_unsuccessfully.to

- **Module**: cmas-dwh-server
- **Description**: To address for emails from the DWH when a transfer finishes unsuccessfully.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: myuser@consol.de
- **Since**: 6.0.1

notification.host

- **Module**: cmas-dwh-server
- **Description**: Email (SMTP) server hostname for sending DWH emails.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: myserver.consol.de
- **Since**: 6.0.1

notification.password

- **Module**: cmas-dwh-server
- **Description**: Password for sending DWH emails (optional).
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.0.1

notification.port

- **Module**: cmas-dwh-server
- **Description**: SMTP port for sending DWH emails.

- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 25
- **Since**: 6.0.1

notification.protocol

- **Module**: cmas-dwh-server
- **Description**: The protocol used for sending emails from the DWH.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: pop3

notification.tls.enabled

- **Module**: cmas-dwh-server
- **Description**: Activates SMTP via SSL/TLS (SMTPS) for sending notification emails from the DWH. The default value is “false”. If it is set to “true”, SMTPS is activated for sending notifications from the DWH.
- **Type**: string
- **Restart required**: yes
- **System**: no
- **Optional**: yes
- **Example value**: false (default value)
- **Since**: 6.11.1.6

notification.username

- **Module**: cmas-dwh-server
- **Description**: (SMTP) user name for sending DWH emails.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: myuser
- **Since**: 6.0.1

notifications.enabled

- **Module**: cmas-web-notifications
- **Description**: Determines whether the notifications feature is enabled ("true") or disabled ("false"). The default value is "true" for ConSol CM systems running on JBoss application servers, and "false" for ConSol CM systems running on WebLogic application servers.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: true
- **Since**: 6.12.0.0

number.of.tasks

- **Module**: cmas-core-server
- **Description**: Number of threads to use by the Task Execution Framework (TEF).
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 1
- **Since**: 6.9.4.0

outdated.lock.age

- **Module**: cmas-workflow-jbpm
- **Description**:
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 60000
- **Removed in**: 6.8.0
- **Replaced by**: cmas-workflow-engine, jobExecutor.lockTimeout.seconds

pagemapLockDurationInSeconds

- **Module**: cmweb-server-adapter
- **Description**: Number of seconds to pass before pagemap is considered to be locked for too long.
- **Type**: integer
- **Restart required**: yes

- **System**: yes
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.7.3

password.reset.mail.from

- **Module**: cmas-core-security
- **Description**: The `From` address for the email which is sent to a customer who requests a new password (using the *Forgot your password?* link) in CM/Track and to an engineer who requests a new password (using the *Forgot your password?* link) in the Web Client.
- **Type**: String
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: mypwreset@consol.de
- **Since**: 6.11.0.1

policy.password.age

- **Module**: cmas-core-security
- **Description**: Maximum validity period, in number of days, example “183” (6 months), default value: “5500” (= 15 years, i.e., no password change enforced).
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 5500 (15 years, default value)
- **Since**: 6.10.1.0

policy.password.pattern

- **Module**: cmas-core-security
- **Description**: RegEx pattern for the password, default value: “^.3,$” (at least 3 characters).
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: "^.3,$" (default value)
- **Since**: 6.10.1.0

policy.rotation.ratio

- **Module**: cmas-core-security
- **Description**: Defines how often passwords may repeat. For example, setting the value to X means that the new password cannot be present among the user's X previous passwords.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 1 (default value)
- **Since**: 6.10.1.0

policy.track.username.case.sensitive

- **Module**: cmas-core-security
- **Description**: Defines whether customer (user) names in CM/Track are treated case-sensitive on login.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true (default value)
- **Since**: 6.11.0.0

policy.username.case.sensitive

- **Module**: cmas-core-security
- **Description**: Defines whether user names are case-sensitive.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true (default value)
- **Since**: 6.10.1.0

postActivityExecutionScriptName

- **Module**: cmweb-server-adapter
- **Description**: Defines the name for the script which should be executed after every workflow activity, see section *PostActivityExecutionScript* in the *ConSol CM Administrator Manual*. If no script should be executed, leave the value empty.
- **Type**: string
- **Restart required**: no
- **System**: yes

- **Optional**: no
- **Example value**: postActivityExecutionHandler
- **Since**: 6.2.0

queue.polling.threads.number

- **Module**: cmas-nimh
- **Description**: Number of threads started for polling the email queue in the database.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 1 (default value)
- **Since**: 6.4.0

queue.polling.threads.shutdown.timeout.seconds

- **Module**: cmas-nimh
- **Description**: Waiting time after the shutdown signal. When the timeout is reached, the thread will be terminated.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60 (default value)
- **Since**: 6.4.0

queue.polling.threads.watchdog.interval.seconds

- **Module**: cmas-nimh
- **Description**: Determines the interval in seconds to execute the watchdog which checks the activity of the threads of the email queue poller.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 30 (default value)
- **Since**: 6.4.0

queue.task.error.pause.seconds

- **Module**: cmas-nimh
- **Description**: Maximum number of seconds, the queue poller waits after infrastructure (e.g. database) error.

- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 180 (default value)
- **Since**: 6.4.0

queue.task.interval.seconds

- **Module**: cmas-nimh
- **Description**: Main emails' queue polling thread interval.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 15 (default value)
- **Since**: 6.4.0

queue.task.max.retries

- **Module**: cmas-nimh
- **Description**: Maximum number of email processing retries after an exception. When reached, the email is moved to the email archive. This email can be rescheduled again using NIMH API (or the Admin Tool).
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 10
- **Since**: 6.4.0

queue.task.timeout.seconds

- **Module**: cmas-nimh
- **Description**: After this time of inactivity (in seconds), the service thread is considered damaged and automatically restarted.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 600 (default value)
- **Since**: 6.4.0

queue.task.transaction.timeout.seconds

- **Module**: cmas-nimh
- **Description**: Transaction timeout for email processing in the pipe.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60 (default value)
- **Since**: 6.4.0

queuesExcludedFromGS

- **Module**: cmweb-server-adapter
- **Description**: Comma-separated list of queue names which are excluded from Quick Search.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: Helpdesk,FAQ
- **Since**: 6.0

recent.items.cleanup.cluster.node.id

- **Module**: cmas-core-server
- **Description**: Value of a `-Dcmas.clusternode.id` designating the node which will clean up recent items.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 1 (assuming the cluster node started with `-Dcmas.clusternode.id=1` parameter)
- **Since**: 6.11.0.1

recent.items.cleanup.interval.minutes

- **Module**: cmas-core-server
- **Description**: Controls the time interval (in minutes) in which recent items should be checked for removal.
- **Type**: integer
- **Restart required**: no
- **System**: yes

- **Optional**: no
- **Example value**: 60
- **Since**: 6.11.0.1

recent.items.max.per.engineer

- **Module**: cmas-core-server
- **Description**: Maximum number of preserved recent items per engineer while cleaning up (older recent items will be deleted).
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 500
- **Since**: 6.11.0.1

recent.items.persistence.enabled

- **Module**: cmas-core-server
- **Description**: Enables persistence of recent items. If the property is set to “false”, it prevents storing new recent items. If a recently visited widget (`recentlyVisitedWidget`) is used on a dashboard, this property needs to be set to “true”. Otherwise, you can set it to “false” to save system resources.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true
- **Since**: 6.11.1.0

recoverable.exceptions

- **Module**: cmas-dwh-server
- **Description**: Comma-separated list of exception definitions: CLASS[+][:REGEX]. The exceptions included in the list do not stop CM from sending to the CMRF process, but force it to try again. If optional '+' after CLASS is present, classes which extend CLASS are matched.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: java.sql.SQLRecoverableException,java.lang.RuntimeException+:.*T.1\,2T.*
- **Since**: 6.8.4.6

refreshTimeInCaseOfConcurrentRememberMeRequests

- **Module**: cmas-workflow-jbpm
- **Description**: It sets the refresh time (in seconds) after which the page will be reloaded in case of concurrent remember me requests. This feature prevents one user from occupying many licenses. Please increase that time if sessions are still occupying.
- **Type**: integer
- **Restart required**: yes
- **System**: yes
- **Optional**: yes
- **Example value**: 5
- **Since**: 6.8.2

rememberMeLifetimeInMinutes

- **Module**: cmweb-server-adapter
- **Description**: Lifetime for *remember me* in minutes.
- **Type**: integer
- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: 1440
- **Since**: 6.0

request.scope.transaction

- **Module**: cmweb-server-adapter
- **Description**: It allows to disable request scope transaction. By default one transaction is used per request. Setting this property to “false” there will cause one transaction per service method invocation.
- **Type**: boolean
- **Restart required**: yes
- **System**: yes
- **Optional**: yes
- **Example value**: true
- **Since**: 6.8.1

resetCode.expirationPeriod

- **Module**: cmas-core-security
- **Description**: Defines the expiration period for the link when resetting the password in CM/Track.
- **Type**: Integer
- **Restart required**: no

- **System**: no
- **Optional**: yes
- **Example value**: 86400000 (default value, 24 hours)
- **Since**: 6.10.1

resource.replace.batchSize

- **Module**: cmas-core-server
- **Description**: Defines the number of objects to be processed in a resource replace action.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 5
- **Since**: 6.10.0.0

resource.replace.timeout

- **Module**: cmas-core-server
- **Description**: Transaction timeout (in seconds) of a resource replacement action step.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 120
- **Since**: 6.10.0.0

scene

- **Module**: cmas-setup-scene
- **Description**: Scene file which was imported during setup (can be empty).
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: vfszip:/P:/dist/target/jboss/server/cmas/deploy/cm-dist-6.5.1-SNAPSHOT.ear/APP-INF/lib/dist-scene-6.5.1-SNAPSHOT.jar/META-INF/cmas/scenes/helpdesk-sales_scene.jar/
- **Since**: 6.0

script.evict.unused.after.hours

- **Module**: cmas-core-server
- **Description**: Determines the number of hours for which unused scripts remain in the cache. After this time, the compiled class of the script is removed. The ConSol CM server checks for scripts to evict every hour.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 24 (default)
- **Since**: 6.11.1.14

script.logging.threshold.seconds

- **Module**: cmas-core-server
- **Description**: When this time, in seconds, is exceeded during script execution, a warning is emitted in the logs.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 10 (default)
- **Since**: 6.10.1.0

script.validation.interval.seconds

- **Module**: cmas-app-admin-tool
- **Description**: Interval in seconds between two code checks in the Admin Tool or the Process Designer code editor
- **Type**: Integer
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: 1 (default value)
- **Since**: 6.11.0.1

searchPageSize

- **Module**: cmweb-server-adapter
- **Description**: Default page size for search results.
- **Type**: integer
- **Restart required**: no
- **System**: yes

- **Optional**: no
- **Example value**: 20
- **Since**: 6.0

searchPageSizeOptions

- **Module**: cmweb-server-adapter
- **Description**: Options for the page size for search results.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 10|20|30|40|50|75|100
- **Since**: 6.0

security.fields.customer.exposure.check.enabled

- **Module**: cmas-restapi-core
- **Description**: Enables customer exposure annotation checks for ticket fields.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true (default value)
- **Since**: 6.10.5.4

security.restrict.unit.access.to.own.data

- **Module**: cmas-restapi-core
- **Description**: If set to “true”, an additional check is performed when a user logs in as a customer using the REST API, e.g. CM/Track. When requesting customer data, only the company of the user or other contacts of the user’s company are returned. If set to “false”, no additional security check is performed and the former security rules apply.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true (default value)
- **Since**: 6.9.2.14

serial.mods.tracking.enabled

- **Module**: cmas-core-server
- **Description**: Low level technical flag deciding whether serial diff tracking for entities is enabled. If enabled, there will be no `StackOverflow` error in case a dependency between two entities (for example engineer and ticket) causes an infinite loop first and then as a result, the StackOverflow. The property must be added to the configuration manually. It will not be added to a system configuration during setup or update.

> ⚠ Please enable the restricted ticket change behavior described in this section only when advised by a ConSol representative! It is a low level technical flag with intricate consequences for system behavior and thus should not be used without thorough scrutiny.

- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: false (default value)
- **Since**: 6.10.7.0, 6.11.0.5

server.instance.task.crash.period.seconds

- **Module**: cmas-core-server
- **Description**: Determines the period (in seconds) for detecting live and crashed ConSol CM server instances.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 15 (default value)
- **Since**: 6.11.1.6

server.instance.task.period.seconds

- **Module**: cmas-core-server
- **Description**: Determines the interval (in seconds) between repeated task executions. The task will be executed repeatedly with the given number of seconds between each execution.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 5 (default value)
- **Since**: 6.11.1.6

server.session.archive.reaper.interval

- **Module**: cmas-core-server
- **Description**: Determines the interval (in seconds) when the reaper for archived server sessions is executed, refers to the database table `cmas_user_session`.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.7.1

server.session.archive.timeout

- **Module**: cmas-core-server
- **Description**: Server sessions archive validity timeout (in days). After this time session info is removed from the database (refers to the database table `cmas_user_session`).
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 3 or 7
- **Since**: 6.7.1

server.session.reaper.interval

- **Module**: cmas-core-server
- **Description**: Determines the interval (in seconds) when the reaper for inactive (ended) server sessions is executed, refers to the database table `cmas_user_session`.
- **Type**: integer
- **Restart required**: only Session Service
- **System**: yes
- **Optional**: no
- **Example value**: 10800 (3 hours)
- **Since**: 6.6.1, 6.7.1

server.session.timeout

- **Module**: cmas-core-server
- **Description**: Server session timeout (in seconds) for connected clients (database table `cmas_user_session`). Each client can overwrite this timeout with custom value using its ID (ADMIN_TOOL, WEB_CLIENT, WORKFLOW_EDITOR, TRACK (before 6.8, please use PORTER), ETL, REST)

appended to property name, e.g., `server.session.timeout.ADMIN_TOOL`.
Please see also the Page Customization attributes *updateTimeServerSessionActivityEnabled* and *updateTimeServerSessionActivity*, both of type *cmApplicationCustomization*.

- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 1800
- **Since**: 6.6.1, 6.7.1

Detailed explanation for the Admin Tool:

- server.session.timeout.ADMIN_TOOL
  Defines the time interval how long the server considers a session valid while there is no activity from the Admin Tool holding the session. The Admin Tool is not aware of this value, it only suffers having an invalid session, if the last activity has been longer in the past.
- admin.tool.session.check.interval
  Defines the time between two checks done by the Admin Tool, if the server still considers its session valid.

For example, if admin.tool.session.check.interval = 60, the Admin Tool queries the server every minute if its session is still active/valid. In case server.session.timeout.ADMIN_TOOL = 600 the Admin Tool will get the response that the session is now invalid after ten minutes of inactivity.

serverPoolingInterval

- **Module**: cmweb-server-adapter
- **Description**: Defines the time in seconds for pooling server to invalidate caches on the web layer.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 5
- **Since**: 6.1.0

skip-ticket

- **Module**: cmas-dwh-server
- **Description**: Tickets are not transferred during transfer/update.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false

- **Since**: 6.6.19
- **Removed in**: 6.8.1

skip-ticket-history

- **Module**: cmas-dwh-server
- **Description**: History of ticket is not transferred during transfer/update.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.6.19
- **Removed in**: 6.8.1

skip-unit

- **Module**: cmas-dwh-server
- **Description**: Units are not transferred during transfer/update.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.6.19
- **Removed in**: 6.8.1

skip-unit-history

- **Module**: cmas-dwh-server
- **Description**: History of unit is not transferred during transfer/update.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.6.19
- **Removed in**: 6.8.1

skip.wfl.transfer.cleanup

- **Module**: cmas-core-server
- **Description**: If set to “true”, skips workflow cleanup after transfer.

- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: false (default value)
- **Since**: 6.9.4.1

skip.wfl.transfer.translations.cleanup

- **Module**: cmas-core-server
- **Description**: Enables skipping the cleanup of localized properties of removed workflow elements.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: false (default value)
- **Since**: 6.10.5.5

split.history

- **Module**: cmas-dwh-server
- **Description**: Changes the SQL that fetches the history for the tickets during DWH transfer so that the history is not fetched for all tickets at once but only for one ticket per SQL.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: false
- **Since**: 6.8.0

start.groovy.task.enabled

- **Module**: cmas-app-admin-tool
- **Description**: For being able to run Admin Tool scripts of type *Task* in the Admin Tool (navigation group *Services*, navigation item *Task Execution*). It is required to enable the *Start task* button, which is hidden by default. This is done by setting this system property to “true”.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true
- **Since**: 6.9.4.0

statistics.calendar

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.client.group

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.contact.role

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.content.entry

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.content.entry.class

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.content.entry.history

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.customer.definition

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.engineer

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.enum.group

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.field.definition

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.group.definition

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.locale

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.localized.property

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.mla

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.project

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.queue

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.resource

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.resource.group

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.resource.history

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.resource.relation.definition

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.resource.type

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.ticket

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.ticket.function

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.ticket.history

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.time.booking

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.timestamp

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.unit

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.unit.history

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.unit.relation.definition

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.workflow

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

strict.utf.bmp.enabled

- **Module**: cmas-core-server
- **Description**: In ConSol CM versions lower than 6.10.6, incoming emails with a subject line containing four-byte UTF8 characters could not be handled by some installations using the MySQL database engine. The reason is the encoding/collation configuration of the database using a

two-byte BMP (Basic Multilingual Plane) 0 plane which cannot be changed in some installations for technical reasons. Other database engines were unaffected. Emails with this encoding could not be imported into the system at all in CM versions lower than 6.10.6. In order to accommodate this issue this system property for configuration is available.
Setting it to “true” will filter out all four-byte UTF8 characters before any database interaction, so the problems mentioned above will not occur.
The property value is “true” by default for MySQL databases, and “false” for any other database where it should not be necessary at all. Change it for a MySQL database only, if the settings positively will support four-byte characters.

- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.10.6.0

supportEmail

- **Module**: cmweb-server-adapter
- **Description**:
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.0
- **Removed in**: 6.11.0.1

synchronize.archive.timeout.minutes

- **Module**: cmas-core-index-common
- **Description**: Determines the time period in minutes during which index updates are archived. The default value of this property is “60” for clustered environments and “0” for non-clustered environments. If the master node is switched, the new master can recover missing data in case it had not been fully synchronized with the previous master indexing server at the moment of switch.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 60
- **Since**: 6.12.0.0

synchronize.master.address

- **Module**: cmas-core-index-common
- **Description**: Value of `-Dcmas.http.host.port` specifying how to connect to the indexing master server. The default value is null. Since 6.6.17 this value is configurable in set-up to designate the initial indexing master server. Since 6.12.0.0 it is possible to switch the node acting as master node by entering the address of the new master node. The index updates are archived for the time configured in the property `cmas-core-index-common`, `synchronize.archive.timeout.minutes`, so the new master can recover missing data in case it had not been fully synchronized with the previous master indexing server at the moment of switch.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 127.0.0.1:80
- **Since**: 6.6.0

synchronize.master.security.token

- **Module**: cmas-core-index-common
- **Description**: The password for accessing the index snapshot via URL, e.g., for index synchronization or for backups.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: token
- **Since**: 6.6.0

synchronize.master.security.user

- **Module**: cmas-core-index-common
- **Description**: The user name for accessing the index snapshot via URL, e.g., for index synchronization or for backups.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: user
- **Since**: 6.6.0

synchronize.master.timeout.minutes

- **Module**: cmas-core-index-common
- **Description**: Determines how long the master server may continually fail until a new master gets elected. Since 6.6.17, this value is configurable in setup, where 0 means that master server will never change (failover is disabled).
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 5 (default value)
- **Since**: 6.6.0

synchronize.megabits.per.second

- **Module**: cmas-core-index-common
- **Description**: Determines how much bandwidth the master server may consume when transferring index changes to all slave servers. Do not use all available bandwidth to transfer index changes between hosts, as doing so will most probably partition the cluster due to some subsystems being unable to communicate.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 85 (default value)
- **Since**: 6.6.0

synchronize.sleep.millis

- **Module**: cmas-core-index-common
- **Description**: Determines how often each slave server polls the master server for index changes.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 1000 (default value)
- **Since**: 6.6.0

task.execution.interval.seconds

- **Module**: cmas-core-server
- **Description**: Time in seconds between the end of an accomplished task in the TEF (Task Execution Framework) and the start of the next task.
- **Type**: Integer

- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: 5
- **Since**: 6.9.4.0

task.execution.node.id

- **Module**: cmas-core-server
- **Description**: Only relevant in clustered environments. The ID of the node where scripts of the TEF (Task Execution Framework) will be executed. This applies to both scripts called from the workflow and scripts called manually using the Admin Tool. The Admin Tool can be started from any node.
- **Type**: Integer
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: 2
- **Since**: 6.11.0.1

task.panel.refresh.interval.seconds

- **Module**: cmas-app-admin-tool
- **Description**: Time in seconds after which the task list (in the Admin Tool) of the Task Execution Framework is refreshed.
- **Type**: Integer
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: 10
- **Since**: 6.10.5.3 (not added automatically during update from versions prior to 6.10.5.3!)

themeOverlay

- **Module**: cmweb-server-adapter
- **Description**: Name of the folder containing the skin used to customize the layout of the Web Client.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: consolINT
- **Since**: 6.0, functionality changed in 6.11.2.0

ticket.delete.timeout

- **Module**: cmas-core-server
- **Description**: Transaction timeout (in seconds) for deleting tickets.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 60
- **Since**: 6.1.3

ticket.from.incoming.message.accepted.links

- **Module**: cmas-core-server
- **Description**: List of domains to which links in incoming emails and links in comments added via REST API are clickable in the ticket history. Regular expressions can be used to specify the allowed URLs. It is possible to add several URLs by using a whitespace as delimiter. The URL must start with one of the allowed protocols (http, https, ftp, ftps, file, mailto). All other links are removed, i.e., the link is displayed in the ticket history as text but it cannot be clicked. If the property is left empty, all links are removed. The regular expression `.+` can be used to allow all domains.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: https://.*\.consol\.de (allows links to “https://<any>.consol.de”)
- **Since**: 6.11.1.7

> ⚠ Please note that whitelisting domains might make ConSol CM vulnerable to cross-site scripting and other attacks. Choose the domains you whitelist carefully!

ticketListRefreshIntervalInSeconds

- **Module**: cmweb-server-adapter
- **Description**: Refresh interval for the ticket list (in seconds).
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 180
- **Since**: 6.0

ticketListSizeLimit

- **Module**: cmweb-server-adapter
- **Description**: Maximum number of tickets in ticket list.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 100
- **Since**: 6.0

tickets.delete.size

- **Module**: cmas-core-server
- **Description**: Defines the number of tickets deleted per transaction.
- **Type**: integer
- **Restart required**: only Session Service
- **System**: yes
- **Optional**: no
- **Example value**: 10 (default value)
- **Since**: 6.8.1

time.buffer

- **Module**: cmas-dwh-server
- **Description**: Number of minutes to extend date of start live mode.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 5
- **Since**: 6.8.1.11

transaction.timeout.minutes

- **Module**: cmas-core-server
- **Description**: Sets the transaction timeout for the task execution service, i.e., one run of a task must finish before this timeout is reached. The changes are visible only for new tasks, the execution of which started after the configuration change.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes

- **Example value**: 10*3600 (10 hours - default value)
- **Since**: 6.10

ttl.days

- **Module**: cmas-web-notifications
- **Description**: Determines the time (in days) after which unpinned notifications are removed.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: 15
- **Since**: 6.12.0.0

tx.read.only.mode.enabled

- **Module**: cmweb-server-adapter
- **Description**: Enables read-only transactions for faster page loading. This transactional behavior was introduced in 6.11.0, and this property acts as a safety guard to restore the old behaviors. Do not change this value unless facing tx problems and advised by ConSol.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true (default value)
- **Since**: 6.11

unit.description.mode

- **Module**: cmas-core-server
- **Description**: Defines whether unit (contact) descriptions in the ticket history are taken from the database or dynamically rendered using templates. The value, “DYNAMIC”, is a bit more costly from the performance perspective, while “PROTOCOL” is faster but returns historical names which might be outdated. Use “PROTOCOL” if you have lots of history entries from many different units. This is also the default value in CM versions 6.11.1.1 and up. In CM versions up to 6.11.1.0, “DYNAMIC” is the default.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: PROTOCOL
- **Since**: 6.11.0

unit.replace.batchSize

- **Module**: cmas-core-server
- **Description**: Defines the number of objects to be processed in a unit replace action.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 5
- **Since**: 6.8.2

unit.replace.timeout

- **Module**: cmas-core-server
- **Description**: Transaction timeout (in seconds) of a unit replacement action step.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 120
- **Since**: 6.8.2

unit.transfer.order

- **Module**: cmas-dwh-server
- **Description**: Defines in which order customer field groups should be transferred to the DWH.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: company;customer
- **Since**: 6.6.19
- **Removed in**: 6.8.1

unitIndexSearchResultSizeLimit

- **Module**: cmweb-server-adapter
- **Description**: Maximum number of units in unit search result (e.g. when searching for contact).
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no

- **Example value**: 5
- **Since**: 6.0

unused.content.remover.cluster.node.id

- **Module**: cmas-core-server
- **Description**: Value of a `cmas.clusternode.id` designating which node will remove unused ticket attachments and unit content entries.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 1 (assuming cluster node started with the parameter `-Dcmas.clusternode.id=1`)
- **Since**: 6.9.0.0

unused.content.remover.enabled

- **Module**: cmas-core-server
- **Description**: Specifies whether a removal of unused ticket attachments and unit content entries should take place.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true
- **Since**: 6.9.0.0

unused.content.remover.polling.minutes

- **Module**: cmas-core-server
- **Description**: Determines how often unused ticket attachments and unit content entries should be checked for removal.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 15
- **Since**: 6.9.0.0

unused.content.remover.ttl.minutes

- **Module**: cmas-core-server
- **Description**: Minimum interval, in minutes, after which unused ticket attachments and unit content entries can be removed.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 1440
- **Since**: 6.9.0.0

update.6.11.0.0.sleep

- **Module**: cmas-setup-hibernate
- **Description**: Helper property for the update preparation scripts introduced in context of CM database refactoring in version 6.11. This is an optional setting allowing a delay (in milliseconds) after each loop iteration of the preparation scripts. Setting the delay should lower the database load, for example during working hours. This property may be removed after the update preparation tasks finish.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.0, for use in 6.10.5.x

update.6.11.0.0.timezone

- **Module**: cmas-setup-hibernate
- **Description**: Helper property for the ticket history migration (the new way of counting history groups). Since 6.11.0.0 the groups are constant (2h time span), but before 6.11.0.0 groups were not constant and depended on the customer's time zone. Migration scripts use an old algorithm to calculate groups and therefore need information about the time zone. The property should be set to the timezone which is most commonly used by the customers. If the property is not set, the default server time zone is used (`TimeZone.getDefault()`). The property should be set before updating to 6.11.0.0 and will be removed automatically after migration. The list of accepted timezones can be found for example here: http://joda-time.sourceforge.net/timezones.html.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: Europe/Berlin
- **Since**: 6.11.0.0, for use before updating to this version

urlLogoutPath

- **Module**: cmweb-server-adapter
- **Description**: URL which is opened when the user logs out. There are three configuration possibilities:
  - Empty value: The user is redirected to the login page.
  - <CM URL>/cm-client/logout: The user is redirected to the logout page, which includes a link to log in again.
  - Any URL: The user is redirected to the given URL outside ConSol CM.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: https://mycmserver/cm-client/logout
- **Since**: 6.3.1, behavior changed in 6.11.2.0

voCacheEnabled

- **Module**: cmweb-server-adapter
- **Description**: This property enables additional caching for the Web Client, voCaching, of complete objects, thus improving performance.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: true
- **Since**: 6.11.1.0

Notes:

- Since ConSol CM version 6.11.1.1, the default value is “true” for non-clustered environments. The value is set to “true” automatically during the setup or update of ConSol CM 6.11.1.1.
- When voCaching is enabled and lazy loading is used for folding ticket history entries, once the engineer unfolded an entry, he cannot fold it again by reloading the page or opening the ticket from the workspace.
- When using the dynamic mode for displaying engineer and customer names in the ticket history (as configured in the system properties cmas-core-server, engineer.description.mode and cmas-core-server, unit.description.mode), the new version of the engineer and/or customer name is only displayed after the ticket has been changed.
- This system property is ignored for clustered environments (environments with cmas-core-shared, cluster.mode set to “true”). In clustered environments, voCaching is always disabled to avoid problems that changes made to objects on one node are not visible on the other nodes.

warmup.executor.enabled

- **Module**: cmas-core-server
- **Description**: Specifies whether the server should asynchronously warm up during startup (e.g., fill some of the internal caches).
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true
- **Since**: 6.9.4.2

webSessionTimeoutInMinutes

- **Module**: cmweb-server-adapter
- **Description**: Session timeout in minutes.
- **Type**: integer
- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: 180
- **Removed in**: 6.7.1
- **Replaced by**: cmas-core-server, server.session.timeout

wfl.sticky.transfer.disabled

- **Module**: cmas-core-server
- **Description**: Enables using preserved original names of workflow elements.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: false
- **Since**: 6.10.1.0

wicketAjaxRequestHeaderFilterEnabled

- **Module**: cmweb-server-adapter
- **Description**: This enables filter for Wicket AJAX requests, coming from stale pages with Wicket 1.4 scripting (CM pre-6.8.0), after update to CM6 post-6.8.0.
- **Type**: boolean
- **Restart required**: yes
- **System**: yes
- **Optional**: yes
- **Example value**: false
- **Since**: 6.8.1

workflow.deploy.cache.eviction.disabled

- **Module**: cmas-core-server
- **Description**: Determines if the infinispan cache should be cleared after deployment (“true”) or not (“false”). If the property is set to “true”, all mappings are removed from the cache.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: false (default value)
- **Since**: 6.11.1.4

X-Frame-Options

- **Module**: cmweb-server-http-headers
- **Description**: Example property to illustrate the configuration of HTTP headers. In this case the delivered HTTP header contains the field *X-Frame-Options* with the value “SAMEORIGIN”.

  Each property in the module `cmweb-server-http-headers` represents one header field. The property name/key identifies the response header field and the value of the property is the field value sent in this header.

> ⚠ Please be aware that additional HTTP response headers must be correctly defined with the exact spelling as officially specified! Please note also that the correct interpretation and application of these headers is fully in the realm and responsibility of the client browser which requested the page!

- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: SAMEORIGIN
- **Since**: 6.10.8

G.2.2 List of System Properties by Module

This chapter lists the system properties included in the following modules.

- cmas-app-admin-tool (module)
- cmas-archive-core-server (module)
- cmas-auth-server (module)
- cmas-core-cache (module)
- cmas-core-index-common (module)
- cmas-core-security (module)
- cmas-core-server (module)
- cmas-core-shared (module)
- cmas-dwh-server (module)
- cmas-nimh (module)
- cmas-nimh-extension (module)
- cmas-restapi-core (module)
- cmas-restapi-http-headers (module)
- cmas-setup-hibernate (module)
- cmas-setup-manager (module)
- cmas-setup-scene (module)
- cmas-web-notifications (module)
- cmas-workflow-engine (module)
- cmas-workflow-jbpm (module)
- cmweb-server-http-headers (module)
- cmweb-server-adapter (module)

G.2.2.1 cmas-app-admin-tool (module)

admin.tool.consumed.licences.check.interval

- **Module**: cmas-app-admin-tool
- **Description**: Sets the interval (in seconds) to monitor the number of consumed licenses.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 30 (default value)
- **Since**: 6.11.0.0

admin.tool.consumed.licences.pool.name

- **Module**: cmas-app-admin-tool
- **Description**: Sets the license pool name to monitor the number of consumed licenses.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: CONCURRENT_USERS (default value)
- **Since**: 6.11.0.0

admin.tool.session.check.interval

- **Module**: cmas-app-admin-tool
- **Description**: Configures the time interval (in seconds) in which the system checks for inactive (ended) Admin Tool sessions.
- **Type**: integer
- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: 30
- **Since**: 6.7.5

autocomplete.enabled

- **Module**: cmas-app-admin-tool
- **Description**: If the flag is missing or its value is “false”, then the *Autocomplete address* navigation item is hidden in Admin Tool.
- **Type**: boolean
- **Restart required**: no
- **System**: yes

- **Optional**: yes
- **Example value**: true
- **Since**: 6.9.2.0

delete.ticket.enabled

- **Module**: cmas-app-admin-tool
- **Description**: Controls if the menu entry *Delete* is displayed in the context menu in the Admin Tool for the ticket list in the ticket administration.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true
- **Since**: 6.9.4.0

dwh.administration.refresh.interval.seconds

- **Module**: cmas-app-admin-tool
- **Description**: Internal DWH property, not to be changed manually. This property is used to set the interval for refreshing the list of actions in the Admin Tool, section *Data Warehouse -> Administration -> Actions*.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 10
- **Since**: 6.11.0.1

script.validation.interval.seconds

- **Module**: cmas-app-admin-tool
- **Description**: Interval in seconds between two code checks in the Admin Tool or the Process Designer code editor
- **Type**: Integer
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: 1 (default value)
- **Since**: 6.11.0.1

start.groovy.task.enabled

- **Module**: cmas-app-admin-tool
- **Description**: For being able to run Admin Tool scripts of type *Task* in the Admin Tool (navigation group *Services*, navigation item *Task Execution*). It is required to enable the *Start task* button, which is hidden by default. This is done by setting this system property to “true”.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true
- **Since**: 6.9.4.0

task.panel.refresh.interval.seconds

- **Module**: cmas-app-admin-tool
- **Description**: Time in seconds after which the task list (in the Admin Tool) of the Task Execution Framework is refreshed.
- **Type**: Integer
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: 10
- **Since**: 6.10.5.3 (not added automatically during update from versions prior to 6.10.5.3!)

G.2.2.2 cmas-archive-core-server (module)

archive.uri

- **Module**: cmas-archive-core-server
- **Description**: Sets the URL from which the CM/Archive application can be accessed
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: <server_url>:8090
- **Since**: 6.11.2.0

G.2.2.3 cmas-auth-server (module)

access.token.signing.key

- **Module**: cmas-auth-server
- **Description**: Determines the secret shared between the authorization server and all client applications using OAuth2, e.g. CM/Archive. The value of this property needs to match the value of *archive.oauth2.access.token.signing.key* in the configuration file of CM/Archive.
- **Type**: password
- **Restart required**: yes
- **System**: no
- **Optional**: no
- **Example value**: my_password
- **Since**: 6.11.2.0

client.archive.access.token.validity.seconds

- **Module**: cmas-auth-server
- **Description**: Determines the validity period in seconds of the access token required for using CM/Archive.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: 43200 (default value)
- **Since**: 6.11.2.0

client.archive.enabled

- **Module**: cmas-auth-server
- **Description**: Determines if CM/Archive is enabled (“true”) or disabled (“false”).
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: false
- **Since**: 6.11.2.0
- **Removed in**: 6.11.2.2

client.archive.refresh.token.validity.seconds

- **Module**: cmas-auth-server
- **Description**: Determines the validity period in seconds of the refresh token required for obtaining new access tokens for CM/Archive.
- **Type**: integer

- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: 2592000 (default value)
- **Since**: 6.11.2.0

client.archive.secret

- **Module**: cmas-auth-server
- **Description**: Determines the secret shared between the authorization server and CM/Archive. The value of the property needs to match the value of *archive.oauth2.client.secret* in the configuration file of CM/Archive.
- **Type**: password
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: my_password
- **Since**: 6.11.2.0

client.was.access.token.validity.seconds

- **Module**: cmas-auth-server
- **Description**: Determines the validity period in seconds of the access token required for using the Web Admin Suite.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: 43200 (default value)
- **Since**: 6.12.0.0

client.was.refresh.token.validity.seconds

- **Module**: cmas-auth-server
- **Description**: Determines the validity period in seconds of the refresh token required for obtaining new access tokens for the Web Admin Suite.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: 2592000 (default value)
- **Since**: 6.12.0.0

client.was.secret

- **Module**: cmas-auth-server
- **Description**: Determines the secret shared between the authorization server and the Web Admin Suite.
- **Type**: password
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: my_password
- **Since**: 6.12.0.0

G.2.2.4 cmas-core-cache (module)

cache-cluster-name

- **Module**: cmas-core-cache
- **Description**: JBoss cache cluster name.
- **Type**: string
- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: 635a6de1-629a-4129-8299-2d98633310f0
- **Since**: 6.4.0

eviction.event.queue.size

- **Module**: cmas-core-cache
- **Description**: The size of the queue holding cache events. The default value is 200000. It is recommended to increase the value slightly (up to 400000) on systems with high traffic or load.
- **Type**: integer
- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: 200000 (default value)
- **Since**: 6.4.0

eviction.lifeSpan

- **Module**: cmas-core-cache
- **Description**: Sets the interval (in milliseconds) for cache entry validity. When this time is elapsed, the entry is removed from the cache.
- **Type**: integer
- **Restart required**: yes
- **System**: no

- **Optional**: yes
- **Example value**: 86400000 (default value)
- **Since**: 6.11.1.0

eviction.max.nodes

- **Module**: cmas-core-cache
- **Description**: Sets the maximum size of internal caches. The default value is 100000. Increasing it will lead to higher memory consumption and is not recommended unless explicitly advised by ConSol.
- **Type**: integer
- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: 100000 (default value)
- **Since**: 6.4.0

eviction.strategy

- **Module**: cmas-core-cache
- **Description**: Determines the cache eviction strategy. Please see the Infinispan documentation for details.
- **Type**: string
- **Restart required**: yes
- **System**: no
- **Optional**: yes
- **Example value**: LRU, LIRS (default value)
- **Since**: 6.11.1.0

eviction.wakeup.interval

- **Module**: cmas-core-cache
- **Description**: Sets the interval (in milliseconds) between two cache queue event processing cycles. The default value is 3000. It is recommended to decrease it (minimum is 1500) on systems with high traffic or load.
- **Type**: integer
- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: 3000 (default value)
- **Since**: 6.4.0

G.2.2.5 cmas-core-index-common (module)

big.task.minimum.size

- **Module**: cmas-core-index-common
- **Description**: Indicates the minimum size of index task (in parts, each part has 100 entities) to qualify this task as a big one. Big tasks have a lower priority than normal tasks.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 15 (default value)
- **Since**: 6.8.3

database.notification.enabled

- **Module**: cmas-core-index-common
- **Description**: Indicates whether the notification channel “database” should be used for index updates instead of JMS. The default value is “true.”
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true
- **Since**: 6.8.4.7
- **Removed in**: 6.12.0.0

database.notification.redelivery.delay.seconds

- **Module**: cmas-core-index-common
- **Description**: If the notification channel “database” is used for index updates, this property indicates the delay for notification redelivery when an exception occurs.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 60
- **Since**: 6.8.4.7

database.notification.redelivery.max.attempts

- **Module**: cmas-core-index-common
- **Description**: In case of index update database notification channel, indicates maximum redelivery attempts when an exception occurs.
- **Type**: integer

- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 60
- **Since**: 6.8.4.7

disable.admin.task.auto.commit

- **Module**: cmas-core-index-common
- **Description**: All tasks created for index update will be automatically executed right after creation.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.6.1

index.attachment

- **Module**: cmas-core-index-common
- **Description**: Specifies whether the content of attachments is indexed.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true
- **Since**: 6.4.3

index.history

- **Module**: cmas-core-index-common
- **Description**: Specifies whether unit and ticket history are indexed.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.1.0
- **Removed in**: 6.11.0

index.status

- **Module**: cmas-core-index-common
- **Description**: Status of the Indexer, possible values RED, YELLOW, GREEN, will be displayed in the Admin Tool.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: GREEN
- **Since**: 6.6.1

index.task.worker.threads

- **Module**: cmas-core-index-common
- **Description**: How many threads will be used to execute index tasks (synchronization, administrative and repair tasks). We recommend to use a value not larger than 2.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 1 (default value)
- **Since**: 6.6.14, 6.7.3. Since 6.8.0 and exclusively in 6.6.21 also normal (live) index updates are affected by this property.

index.version.current

- **Module**: cmas-core-index-common
- **Description**: Holds information about the current (possibly old) index version.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 1 (default value)
- **Since**: 6.7.0

index.version.newest

- **Module**: cmas-core-index-common
- **Description**: Holds information about which index version is considered newest.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no

- **Example value**: 1 (default value)
- **Since**: 6.7.0

indexed.assets.per.thread.in.memory

- **Module**: cmas-core-index-common
- **Description**: Determines how many assets should be loaded into memory at once, per thread, during indexing.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 200 (default value)
- **Since**: 6.8.0

indexed.engineers.per.thread.in.memory

- **Module**: cmas-core-index-common
- **Description**: Determines how many engineers should be loaded into memory at once, per thread, during indexing.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 300 (default value)
- **Since**: 6.6.14, 6.7.3

indexed.resources.per.thread.in.memory

- **Module**: cmas-core-index-common
- **Description**: Determines how many resources should be loaded into memory at once, per thread, during indexing.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 200 (default value)
- **Since**: 6.10.0.0

indexed.tickets.per.thread.in.memory

- **Module**: cmas-core-index-common
- **Description**: Determines how many tickets should be loaded into memory at once, per thread, during indexing.
- **Type**: integer

- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 100 (default value)
- **Since**: 6.6.14, 6.7.3

indexed.units.per.thread.in.memory

- **Module**: cmas-core-index-common
- **Description**: Determines how many units should be loaded into memory at once, per thread, during indexing.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 200 (default value)
- **Since**: 6.6.14, 6.7.3

synchronize.archive.timeout.minutes

- **Module**: cmas-core-index-common
- **Description**: Determines the time period in minutes during which index updates are archived. The default value of this property is “60” for clustered environments and “0” for non-clustered environments. If the master node is switched, the new master can recover missing data in case it had not been fully synchronized with the previous master indexing server at the moment of switch.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 60
- **Since**: 6.12.0.0

synchronize.master.address

- **Module**: cmas-core-index-common
- **Description**: Value of `-Dcmas.http.host.port` specifying how to connect to the indexing master server. The default value is null. Since 6.6.17 this value is configurable in set-up to designate the initial indexing master server. Since 6.12.0.0 it is possible to switch the node acting as master node by entering the address of the new master node. The index updates are archived for the time configured in the property `cmas-core-index-common`, `synchronize.archive.timeout.minutes`, so the new master can recover missing data in case it had not been fully synchronized with the previous master indexing server at the moment of switch.
- **Type**: integer

- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 127.0.0.1:80
- **Since**: 6.6.0

synchronize.master.security.token

- **Module**: cmas-core-index-common
- **Description**: The password for accessing the index snapshot via URL, e.g., for index synchronization or for backups.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: token
- **Since**: 6.6.0

synchronize.master.security.user

- **Module**: cmas-core-index-common
- **Description**: The user name for accessing the index snapshot via URL, e.g., for index synchronization or for backups.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: user
- **Since**: 6.6.0

synchronize.master.timeout.minutes

- **Module**: cmas-core-index-common
- **Description**: Determines how long the master server may continually fail until a new master gets elected. Since 6.6.17, this value is configurable in setup, where 0 means that master server will never change (failover is disabled).
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 5 (default value)
- **Since**: 6.6.0

synchronize.megabits.per.second

- **Module**: cmas-core-index-common
- **Description**: Determines how much bandwidth the master server may consume when transferring index changes to all slave servers. Do not use all available bandwidth to transfer index changes between hosts, as doing so will most probably partition the cluster due to some subsystems being unable to communicate.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 85 (default value)
- **Since**: 6.6.0

synchronize.sleep.millis

- **Module**: cmas-core-index-common
- **Description**: Determines how often each slave server polls the master server for index changes.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 1000 (default value)
- **Since**: 6.6.0

G.2.2.6 cmas-core-security (module)

admin.email

- **Module**: cmas-core-security
- **Description**: The email address of the ConSol CM administrator. The value which you entered during system set-up is used initially.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: myuser@consol.de
- **Since**: 6.0

admin.login

- **Module**: cmas-core-security
- **Description**: The name of the ConSol CM administrator. The value which you entered during system set-up is used initially.
- **Type**: string

- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: admin
- **Since**: 6.0

authentication.method

- **Module**: cmas-core-security
- **Description**: User authentication method (internal CM database or LDAP authentication). Allowed values are LDAP or DATABASE.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: DATABASE
- **Since**: 6.0

contact.authentication.method

- **Module**: cmas-core-security
- **Description**: Indicates contact authentication method, where possible values are DATABASE or LDAP or LDAP,DATABASE or DATABASE,LDAP.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: LDAP
- **Since**: 6.9.3.0

contact.inherit.permissions.only.to.own.customer.group

- **Module**: cmas-core-security
- **Description**: Indicates whether an authenticated contact inherits all customer group permissions from the CM/Track user profile (“false”) or only has permissions to his own customer group (“true”).
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.9.2.3

kerberos.v5.enabled

- **Module**: cmas-core-security
- **Description**: Indicates whether SSO via Kerberos is enabled.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false (default value if Kerberos was not enabled during system setup)
- **Since**: 6.2.0

kerberos.v5.username.regex

- **Module**: cmas-core-security
- **Description**: Regular expression used for mapping Kerberos principals to CM user login names.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: (.*)@.*
- **Since**: 6.2.0

ldap.authentication

- **Module**: cmas-core-security
- **Description**: Authentication method used when using LDAP authentication. Possible values are “anonymous” and “simple”.
- **Type**: string
- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: simple (default value)
- **Since**: 6.0

ldap.basedn

- **Module**: cmas-core-security
- **Description**: Base DN used for looking up LDAP user accounts when using LDAP authentication.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no

- **Example value**: ou=accounts,dc=consol,dc=de
- **Since**: 6.0

ldap.contact.name.basedn

- **Module**: cmas-core-security
- **Description**: Base path to search for contact DN by LDAP ID.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: ou=accounts,dc=consol,dc=de
- **Since**: 6.9.3.0

ldap.contact.name.password

- **Module**: cmas-core-security
- **Description**: Password to look up contact DN by LDAP ID. If not set, the anonymous account is used.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Since**: 6.9.3.0

ldap.contact.name.providerurl

- **Module**: cmas-core-security
- **Description**: Address of the LDAP server (ldap[s]://host:port).
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: ldap://ldap.consol.de:389
- **Since**: 6.9.3.0

ldap.contact.name.searchattr

- **Module**: cmas-core-security
- **Description**: Attribute to search for contact DN by LDAP ID.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes

- **Example value**: uid
- **Since**: 6.9.3.0

ldap.contact.name.userdn

- **Module**: cmas-core-security
- **Description**: User DN to look up contact DN by LDAP ID. If not set, the anonymous account is used.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Since**: 6.9.3.0

ldap.initialcontextfactory

- **Module**: cmas-core-security
- **Description**: Class name for the initial context factory of the LDAP implementation when using LDAP authentication. If it is not set, `com.sun.jndi.ldap.LdapCtxFactory` is used.
- **Type**: string
- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: com.sun.jndi.ldap.LdapCtxFactory
- **Since**: 6.0

ldap.password

- **Module**: cmas-core-security
- **Description**: Password for connecting to LDAP to look up users when using LDAP authentication. Only needed if look-up cannot be performed anonymously.
- **Type**: password
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.1.2

ldap.providerurl

- **Module**: cmas-core-security
- **Description**: LDAP provider when using LDAP authentication.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no

- **Example value**: ldap://myserver.consol.de:389
- **Since**: 6.0

ldap.searchattr

- **Module**: cmas-core-security
- **Description**: Search attribute for looking up LDAP entry associated with a CM login.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: uid
- **Since**: 6.0

ldap.userdn

- **Module**: cmas-core-security
- **Description**: LDAP user for connecting to LDAP to look up users when using LDAP authentication. Only needed if look-up cannot be performed anonymously.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.1.2

password.reset.mail.from

- **Module**: cmas-core-security
- **Description**: The `From` address for the email which is sent to a customer who requests a new password (using the *Forgot your password?* link) in CM/Track and to an engineer who requests a new password (using the *Forgot your password?* link) in the Web Client.
- **Type**: String
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: mypwreset@consol.de
- **Since**: 6.11.0.1

policy.password.age

- **Module**: cmas-core-security
- **Description**: Maximum validity period, in number of days, example “183” (6 months), default value: “5500” (= 15 years, i.e., no password change enforced).
- **Type**: integer
- **Restart required**: no

- **System**: no
- **Optional**: yes
- **Example value**: 5500 (15 years, default value)
- **Since**: 6.10.1.0

policy.password.pattern

- **Module**: cmas-core-security
- **Description**: RegEx pattern for the password, default value: “^.3,$” (at least 3 characters).
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: "^.3,$" (default value)
- **Since**: 6.10.1.0

policy.rotation.ratio

- **Module**: cmas-core-security
- **Description**: Defines how often passwords may repeat. For example, setting the value to X means that the new password cannot be present among the user's X previous passwords.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 1 (default value)
- **Since**: 6.10.1.0

policy.username.case.sensitive

- **Module**: cmas-core-security
- **Description**: Defines whether user names are case-sensitive.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true (default value)
- **Since**: 6.10.1.0

policy.track.username.case.sensitive

- **Module**: cmas-core-security
- **Description**: Defines whether customer (user) names in CM/Track are treated case-sensitive on login.

- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true (default value)
- **Since**: 6.11.0.0

resetCode.expirationPeriod

- **Module**: cmas-core-security
- **Description**: Defines the expiration period for the link when resetting the password in CM/Track.
- **Type**: Integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 86400000 (default value, 24 hours)
- **Since**: 6.10.1

G.2.2.7 cmas-core-server (module)

attachment.allowed.types

- **Module**: cmas-core-server
- **Description**: Comma-separated list of allowed file name extensions (if no value is defined, all file extensions are allowed).
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: txt,zip,doc
- **Since**: 6.5.0

attachment.max.size

- **Module**: cmas-core-server
- **Description**: Maximum attachment size, in MB. This is a validation property of the CM API. It controls the size of attachments at tickets, units, and resources. It also controls the size of incoming (not outgoing!) email attachments. The value of this property needs to be aligned with the respective setting in the application server configuration.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no

- **Example value**: 100 (default value)
- **Since**: 6.4.0

attachment.type.hints

- **Module**: cmas-core-server
- **Description**: Allows you to assign MIME types to not yet officially supported file extensions, so these file extensions can be detected correctly.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 7z:application/x-7z-compressed, zip:application/zip, wav:audio/x-wav
- **Since**: 6.11.2.0

calendar.csv.dateFormat

- **Module**: cmas-core-server
- **Description**: Format of the date given in the csv file containing the list of holidays.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: dd/MM/yyyy
- **Since**: 6.9.3.2

calendar.csv.separator

- **Module**: cmas-core-server
- **Description**: Separator used in the csv file containing the list of holidays.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: ,
- **Since**: 6.9.3.2

config.data.version

- **Module**: cmas-core-server
- **Description**: The internal version number of the current system configuration. This property is maintained internally, please do not change it unless advised by ConSol.
- **Type**: integer
- **Restart required**: no

- **System**: yes
- **Optional**: no
- **Example value**: 11
- **Since**: 6.0

config.import.global.transaction.enabled

- **Module**: cmas-core-server
- **Description**: Flag deciding whether configuration (without localizations) should be imported within a single transaction.
- **Type**: Boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true
- **Since**: 6.11.1.0

customfield.content.file.max.size

- **Module**: cmas-core-server
- **Description**: Determines the maximum size in MB of the images which can be added as content of rich text fields.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 10
- **Since**: 6.11.2.2

dao.log.threshold.milliseconds

- **Module**: cmas-core-server
- **Description**: Used to configure database operation times logging. DAO methods whose execution takes longer than the time set in this property (in milliseconds) are logged.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 500 (default value)
- **Since**: 6.11.1.0

dao.log.username

- **Module**: cmas-core-server
- **Description**: Used to configure database operation times logging. The execution of DAO methods which are related to the user name stated in this property is logged. Only one user name can be provided. The value is empty by default.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: webadmin
- **Since**: 6.11.1.0

defaultCommentClassName

- **Module**: cmas-core-server
- **Description**: Name of the default text class for comments.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: default_class
- **Since**: 6.3.0

defaultIncommingMailClassName

- **Module**: cmas-core-server
- **Description**: Name of the default text class for incoming emails.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: default_class
- **Since**: 6.3.0

defaultOutgoingMailClassName

- **Module**: cmas-core-server
- **Description**: Name of the default text class for outgoing emails.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes

- **Example value**: default_class
- **Since**: 6.3.0

engineer.description.cache.enabled

- **Module**: cmas-core-server
- **Description**: Defines whether user descriptions are cached. The default value is “true”, please do not change it unless advised by ConSol.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true (default value)
- **Since**: 6.11.0

engineer.description.mode

- **Module**: cmas-core-server
- **Description**: Defines whether user names in the ticket history are taken from the database or dynamically rendered using templates. The default value “DYNAMIC” is a bit more costly from the performance perspective, while “PROTOCOL” is faster but returns historical names which might be outdated. Use “PROTOCOL” if you have lots of history entries from many different users.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: PROTOCOL
- **Since**: 6.11.0

engineer.description.template.name

- **Module**: cmas-core-server
- **Description**: Defines the name of the template which is used to render engineer names for display in the Web Client. The template has to be stored in the *Templates* section of the Admin Tool.
- **Type**: String
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: engineer description template name (default value)
- **Since**: 6.11.0

external.line.access.prefix

- **Module**: cmas-core-server
- **Description**: General prefix to dial before an area code. Set for each customer group separately.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 411
- **Since**: 6.9.3.0

fetchSize.strategy

- **Module**: cmas-core-server
- **Description**: Strategy for selecting the fetch size on JDBC result sets.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: FetchSizePageBasedStrategy, FetchSizeThresholdStrategy, FetchS-izeFixedStrategy
- **Since**: 6.8.4.1

fetchSize.strategy.FetchSizeFixedStrategy.value

- **Module**: cmas-core-server
- **Description**: Sets fetch size value if the selected strategy to set the fetch size is `FetchSizeFixedStrategy`.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 150
- **Since**: 6.8.4.1

fetchSize.strategy.FetchSizePageBasedStrategy.limit

- **Module**: cmas-core-server
- **Description**: Sets maximum fetch size value if the selected strategy to set the fetch size is `FetchSizePageBasedStrategy`.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes

- **Example value**: 10000
- **Since**: 6.8.4.1

fetchSize.strategy.FetchSizeThresholdStrategy.value

- **Module**: cmas-core-server
- **Description**: Sets fetch size threshold border values if the selected strategy to set the fetch size is `FetchSizeThresholdStrategy`.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 150,300,600,1000
- **Since**: 6.8.4.1

heartbeat

- **Module**: cmas-core-server
- **Description**: Timestamp that indicates if an instance of the application is connected to the database schema.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 1234567899
- **Since**: 6.10.5.3

internal.line.access.prefix

- **Module**: cmas-core-server
- **Description**: Prefix that the company's telephony system requires for outside lines. Set for each customer group separately.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 199
- **Since**: 6.9.3.0

last.config.change

- **Module**: cmas-core-server
- **Description**: Random UUID created during the last configuration change. This is a value maintained internally, please do not change it unless advised by ConSol.
- **Type**: string

- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 2573c7b7-2bf5-47ff-b5a2-bad31951a266
- **Since**: 6.1.0, 6.2.1

last.config.change.templates

- **Module**: cmas-core-server
- **Description**: Random UUID created during the last change in templates. This is a value maintained internally, please do not change it unless advised by ConSol.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 2573c7c7-2af5-4eff-b9c2-bad31951a266
- **Since**: 6.10.5.0

ldap.certificate.basedn

- **Module**: cmas-core-server
- **Description**: Base DN for certificates location in the LDAP tree. If not provided, `cmas-core-security, ldap.basedn` is used.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: ou=accounts,dc=consol,dc=de
- **Since**: 6.8.4

ldap.certificate.content.attribute

- **Module**: cmas-core-server
- **Description**: LDAP attribute name used where certificate data is stored in the LDAP tree.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: usercertificate (default value)
- **Since**: 6.8.4

ldap.certificate.password

- **Module**: cmas-core-server
- **Description**: LDAP Certificates manager password. If not set, `cmas-core-security, ldap.password` is used.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.8.4

ldap.certificate.providerurl

- **Module**: cmas-core-server
- **Description**: LDAP Certificates provider URL. If not set, `cmas-core-security, ldap.providerurl` is used.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: ldap://ldap.consol.de:389
- **Since**: 6.8.4

ldap.certificate.searchattr

- **Module**: cmas-core-server
- **Description**: LDAP attribute name used to search for certificate in the LDAP tree.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: mail (default value)
- **Since**: 6.8.4

ldap.certificate.userdn

- **Module**: cmas-core-server
- **Description**: LDAP Certificates manager DN. If not set, `cmas-core-security, ldap.userdn` is used.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.8.4

local.country.prefix

- **Module**: cmas-core-server
- **Description**: Prefix of the local country code. Set for each customer group separately.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 48
- **Since**: 6.9.3.0

mail.encryption

- **Module**: cmas-core-server
- **Description**: If the property is set to “true”, the encrypt checkbox in the Ticket Email Editor is checked by default. The default value of the property is “false”.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true
- **Since**: 6.8.4.0

mail.notification.engineerChange

- **Module**: cmas-core-server
- **Description**: Determines whether notification emails should be sent when the engineer of a ticket is changed.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true
- **Since**: 6.1.0

mail.notification.sender

- **Module**: cmas-core-server
- **Description**: From address for notification emails when the engineer of a ticket is changed. If not set, `cmas-core-security, admin.email` is used instead.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes

- **Example value**: cm6notification@cm6installation
- **Since**: 6.6.3

mail.redelivery.retry.count

- **Module**: cmas-core-server
- **Description**: Number of redelivery attempts of an outgoing email.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 3
- **Since**: 6.1.0

mail.smtp.email

- **Module**: cmas-core-server
- **Description**: SMTP email URL for outgoing emails
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: smtp://mail.mydomain.com:25
- **Since**: 6.0

mail.smtp.email.password

- **Module**: cmas-core-server
- **Description**: SMTP password for outgoing email communication. If this property is set, it takes precedence over the value from the SMTP URL.
- **Type**: password
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: password
- **Since**: 6.11.2.2

mail.smtp.email.user

- **Module**: cmas-core-server
- **Description**: SMTP user name for outgoing email communication. If this property is set, it takes precedence over the value from the SMTP URL.
- **Type**: string
- **Restart required**: no

- **System**: yes
- **Optional**: yes
- **Example value**: myuser
- **Since**: 6.11.2.2

mail.smtp.envelopesender

- **Module**: cmas-core-server
- **Description**: Email address used as sender in SMTP envelope. If not set, the From address of the email is used.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: mysender@mydomain.com
- **Since**: 6.5.7

mail.smtp.tls.enabled

- **Module**: cmas-core-server
- **Description**: Activates SMTP via SSL/TLS (SMTPS) for sending emails from the Web Client and scripts. The default value is “false”. If it is set to “true”, SMTPS is activated for sending emails.
- **Type**: boolean
- **Restart required**: yes
- **System**: no
- **Optional**: yes
- **Example value**: true
- **Since**: 6.11.1.6

mailSender.executionInterval.seconds

- **Module**: cmas-core-server
- **Description**: Determines the period in seconds before the system checks if there are new emails to be sent.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 10
- **Since**: 6.12.0.0

mailSender.lockingLimit

- **Module**: cmas-core-server
- **Description**: Determines the number of emails fetched from the database for sending in a single transaction.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 1
- **Since**: 6.12.0.0

mailSender.lockTimeout.seconds

- **Module**: cmas-core-server
- **Description**: Determines the period in seconds before a locked email is released for another lock.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 199
- **Since**: 6.12.0.0

mailSender.maxAttempts

- **Module**: cmas-core-server
- **Description**: Determines the maximum number of attempts, including the first attempt, before an email is removed.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 3
- **Since**: 6.12.0.0

mailSender.nodeId

- **Module**: cmas-core-server
- **Description**: Determines the ID of the node on which the thread for sending emails is started (only in clustered environments). If the value is “all” or no value is provided, the threads are started on every node.
- **Type**: string
- **Restart required**: no
- **System**: no

- **Optional**: yes
- **Example value**: all
- **Since**: 6.12.0.0

mailSender.retryInterval.seconds

- **Module**: cmas-core-server
- **Description**: Determines the period in seconds before another attempt to send an email is made, if the previous attempt failed.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.12.0.0

mailSender.threads.number

- **Module**: cmas-core-server
- **Description**: Determines the number of threads for sending emails.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 1
- **Since**: 6.12.0.0

mailSender.txTimeout.seconds

- **Module**: cmas-core-server
- **Description**: Determines the transaction timeout for sending emails in seconds.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.12.0.0

max.licences.perUser

- **Module**: cmas-core-server
- **Description**: Sets the maximum number of licenses which a single user can use (e.g., logging in from different browsers). By default this value is not restricted.
- **Type**: integer

- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 10
- **Since**: 6.8.4.5

monitoring.engineer.login

- **Module**: cmas-core-server
- **Description**: Login of monitoring engineer.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: nagios
- **Since**: 6.9.3.0

monitoring.unit.login

- **Module**: cmas-core-server
- **Description**: Login of monitoring unit.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: nagios
- **Since**: 6.9.3.0

nimh.enabled

- **Module**: cmas-core-server
- **Description**: Enables the NIMH service. Must be suffixed with the cluster node ID, e.g., `nimh.enabled.NODEID` = “true”.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: false
- **Since**: 6.9.4.0

number.of.tasks

- **Module**: cmas-core-server
- **Description**: Number of threads to use by the Task Execution Framework (TEF).

- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 1
- **Since**: 6.9.4.0

recent.items.cleanup.cluster.node.id

- **Module**: cmas-core-server
- **Description**: Value of a `-Dcmas.clusternode.id` designating the node which will clean up recent items.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 1 (assuming the cluster node started with `-Dcmas.clusternode.id=1` parameter)
- **Since**: 6.11.0.1

recent.items.cleanup.interval.minutes

- **Module**: cmas-core-server
- **Description**: Controls the time interval (in minutes) in which recent items should be checked for removal.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 60
- **Since**: 6.11.0.1

recent.items.max.per.engineer

- **Module**: cmas-core-server
- **Description**: Maximum number of preserved recent items per engineer while cleaning up (older recent items will be deleted).
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 500
- **Since**: 6.11.0.1

recent.items.persistence.enabled

- **Module**: cmas-core-server
- **Description**: Enables persistence of recent items. If the property is set to “false”, it prevents storing new recent items. If a recently visited widget (`recentlyVisitedWidget`) is used on a dashboard, this property needs to be set to “true”. Otherwise, you can set it to “false” to save system resources.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true
- **Since**: 6.11.1.0

resource.replace.batchSize

- **Module**: cmas-core-server
- **Description**: Defines the number of objects to be processed in a resource replace action.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 5
- **Since**: 6.10.0.0

resource.replace.timeout

- **Module**: cmas-core-server
- **Description**: Transaction timeout (in seconds) of a resource replacement action step.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 120
- **Since**: 6.10.0.0

script.evict.unused.after.hours

- **Module**: cmas-core-server
- **Description**: Determines the number of hours for which unused scripts remain in the cache. After this time, the compiled class of the script is removed. The ConSol CM server checks for scripts to evict every hour.
- **Type**: integer
- **Restart required**: no
- **System**: yes

- **Optional**: no
- **Example value**: 24 (default)
- **Since**: 6.11.1.14

script.logging.threshold.seconds

- **Module**: cmas-core-server
- **Description**: When this time, in seconds, is exceeded during script execution, a warning is emitted in the logs.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 10 (default)
- **Since**: 6.10.1.0

serial.mods.tracking.enabled

- **Module**: cmas-core-server
- **Description**: Low level technical flag deciding whether serial diff tracking for entities is enabled. If enabled, there will be no `StackOverflow` error in case a dependency between two entities (for example engineer and ticket) causes an infinite loop first and then as a result, the StackOverflow. The property must be added to the configuration manually. It will not be added to a system configuration during setup or update.

  > ⚠ Please enable the restricted ticket change behavior described in this section only when advised by a ConSol representative! It is a low level technical flag with intricate consequences for system behavior and thus should not be used without thorough scrutiny.

- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: false (default value)
- **Since**: 6.10.7.0, 6.11.0.5

server.instance.task.crash.period.seconds

- **Module**: cmas-core-server
- **Description**: Determines the period (in seconds) for detecting live and crashed ConSol CM server instances.
- **Type**: integer
- **Restart required**: no
- **System**: no

- **Optional**: yes
- **Example value**: 15 (default value)
- **Since**: 6.11.1.6

server.instance.task.period.seconds

- **Module**: cmas-core-server
- **Description**: Determines the interval (in seconds) between repeated task executions. The task will be executed repeatedly with the given number of seconds between each execution.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 5 (default value)
- **Since**: 6.11.1.6

server.session.archive.reaper.interval

- **Module**: cmas-core-server
- **Description**: Determines the interval (in seconds) when the reaper for archived server sessions is executed, refers to the database table `cmas_user_session`.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.7.1

server.session.archive.timeout

- **Module**: cmas-core-server
- **Description**: Server sessions archive validity timeout (in days). After this time session info is removed from the database (refers to the database table `cmas_user_session`).
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 3 or 7
- **Since**: 6.7.1

server.session.reaper.interval

- **Module**: cmas-core-server
- **Description**: Determines the interval (in seconds) when the reaper for inactive (ended) server sessions is executed, refers to the database table `cmas_user_session`.

- **Type**: integer
- **Restart required**: only Session Service
- **System**: yes
- **Optional**: no
- **Example value**: 10800 (3 hours)
- **Since**: 6.6.1, 6.7.1

server.session.timeout

- **Module**: cmas-core-server
- **Description**: Server session timeout (in seconds) for connected clients (database table `cmas_user_session`). Each client can overwrite this timeout with custom value using its ID (ADMIN_TOOL, WEB_CLIENT, WORKFLOW_EDITOR, TRACK (before 6.8, please use PORTER), ETL, REST) appended to property name, e.g., `server.session.timeout.ADMIN_TOOL`.
  Please see also the Page Customization attributes *updateTimeServerSessionActivityEnabled* and *updateTimeServerSessionActivity*, both of type *cmApplicationCustomization*.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 1800
- **Since**: 6.6.1, 6.7.1

Detailed explanation for the Admin Tool:

- server.session.timeout.ADMIN_TOOL
  Defines the time interval how long the server considers a session valid while there is no activity from the Admin Tool holding the session. The Admin Tool is not aware of this value, it only suffers having an invalid session, if the last activity has been longer in the past.
- admin.tool.session.check.interval
  Defines the time between two checks done by the Admin Tool, if the server still considers its session valid.

For example, if admin.tool.session.check.interval = 60, the Admin Tool queries the server every minute if its session is still active/valid. In case server.session.timeout.ADMIN_TOOL = 600 the Admin Tool will get the response that the session is now invalid after ten minutes of inactivity.

skip.wfl.transfer.cleanup

- **Module**: cmas-core-server
- **Description**: If set to “true”, skips workflow cleanup after transfer.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: false (default value)
- **Since**: 6.9.4.1

skip.wfl.transfer.translations.cleanup

- **Module**: cmas-core-server
- **Description**: Enables skipping the cleanup of localized properties of removed workflow elements.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: false (default value)
- **Since**: 6.10.5.5

strict.utf.bmp.enabled

- **Module**: cmas-core-server
- **Description**: In ConSol CM versions lower than 6.10.6, incoming emails with a subject line containing four-byte UTF8 characters could not be handled by some installations using the MySQL database engine. The reason is the encoding/collation configuration of the database using a two-byte BMP (Basic Multilingual Plane) 0 plane which cannot be changed in some installations for technical reasons. Other database engines were unaffected. Emails with this encoding could not be imported into the system at all in CM versions lower than 6.10.6. In order to accommodate this issue this system property for configuration is available.
  Setting it to “true” will filter out all four-byte UTF8 characters before any database interaction, so the problems mentioned above will not occur.
  The property value is “true” by default for MySQL databases, and “false” for any other database where it should not be necessary at all. Change it for a MySQL database only, if the settings positively will support four-byte characters.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.10.6.0

task.execution.interval.seconds

- **Module**: cmas-core-server
- **Description**: Time in seconds between the end of an accomplished task in the TEF (Task Execution Framework) and the start of the next task.
- **Type**: Integer
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: 5
- **Since**: 6.9.4.0

task.execution.node.id

- **Module**: cmas-core-server
- **Description**: Only relevant in clustered environments. The ID of the node where scripts of the TEF (Task Execution Framework) will be executed. This applies to both scripts called from the workflow and scripts called manually using the Admin Tool. The Admin Tool can be started from any node.
- **Type**: Integer
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: 2
- **Since**: 6.11.0.1

tickets.delete.size

- **Module**: cmas-core-server
- **Description**: Defines the number of tickets deleted per transaction.
- **Type**: integer
- **Restart required**: only Session Service
- **System**: yes
- **Optional**: no
- **Example value**: 10 (default value)
- **Since**: 6.8.1

ticket.delete.timeout

- **Module**: cmas-core-server
- **Description**: Transaction timeout (in seconds) for deleting tickets.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 60
- **Since**: 6.1.3

ticket.from.incoming.message.accepted.links

- **Module**: cmas-core-server
- **Description**: List of domains to which links in incoming emails and links in comments added via REST API are clickable in the ticket history. Regular expressions can be used to specify the allowed URLs. It is possible to add several URLs by using a whitespace as delimiter. The URL must start with one of the allowed protocols (http, https, ftp, ftps, file, mailto). All other

links are removed, i.e., the link is displayed in the ticket history as text but it cannot be clicked. If the property is left empty, all links are removed. The regular expression `.+` can be used to allow all domains.

- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: https://.*\.consol\.de (allows links to “https://<any>.consol.de”)
- **Since**: 6.11.1.7

> ⚠ Please note that whitelisting domains might make ConSol CM vulnerable to cross-site scripting and other attacks. Choose the domains you whitelist carefully!

transaction.timeout.minutes

- **Module**: cmas-core-server
- **Description**: Sets the transaction timeout for the task execution service, i.e., one run of a task must finish before this timeout is reached. The changes are visible only for new tasks, the execution of which started after the configuration change.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 10*3600 (10 hours - default value)
- **Since**: 6.10

unit.description.mode

- **Module**: cmas-core-server
- **Description**: Defines whether unit (contact) descriptions in the ticket history are taken from the database or dynamically rendered using templates. The value, “DYNAMIC”, is a bit more costly from the performance perspective, while “PROTOCOL” is faster but returns historical names which might be outdated. Use “PROTOCOL” if you have lots of history entries from many different units. This is also the default value in CM versions 6.11.1.1 and up. In CM versions up to 6.11.1.0, “DYNAMIC” is the default.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: PROTOCOL
- **Since**: 6.11.0

unit.replace.batchSize

- **Module**: cmas-core-server
- **Description**: Defines the number of objects to be processed in a unit replace action.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 5
- **Since**: 6.8.2

unit.replace.timeout

- **Module**: cmas-core-server
- **Description**: Transaction timeout (in seconds) of a unit replacement action step.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 120
- **Since**: 6.8.2

unused.content.remover.cluster.node.id

- **Module**: cmas-core-server
- **Description**: Value of a `cmas.clusternode.id` designating which node will remove unused ticket attachments and unit content entries.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 1 (assuming cluster node started with the parameter `-Dcmas.clusternode.id=1`)
- **Since**: 6.9.0.0

unused.content.remover.enabled

- **Module**: cmas-core-server
- **Description**: Specifies whether a removal of unused ticket attachments and unit content entries should take place.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no

- **Example value**: true
- **Since**: 6.9.0.0

unused.content.remover.polling.minutes

- **Module**: cmas-core-server
- **Description**: Determines how often unused ticket attachments and unit content entries should be checked for removal.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 15
- **Since**: 6.9.0.0

unused.content.remover.ttl.minutes

- **Module**: cmas-core-server
- **Description**: Minimum interval, in minutes, after which unused ticket attachments and unit content entries can be removed.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 1440
- **Since**: 6.9.0.0

warmup.executor.enabled

- **Module**: cmas-core-server
- **Description**: Specifies whether the server should asynchronously warm up during startup (e.g., fill some of the internal caches).
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true
- **Since**: 6.9.4.2

wfl.sticky.transfer.disabled

- **Module**: cmas-core-server
- **Description**: Enables using preserved original names of workflow elements.
- **Type**: boolean
- **Restart required**: no

- **System**: no
- **Optional**: yes
- **Example value**: false
- **Since**: 6.10.1.0

workflow.deploy.cache.eviction.disabled

- **Module**: cmas-core-server
- **Description**: Determines if the infinispan cache should be cleared after deployment ("true") or not ("false"). If the property is set to "true", all mappings are removed from the cache.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: false (default value)
- **Since**: 6.11.1.4

G.2.2.8 cmas-core-shared (module)

cluster.mode

- **Module**: cmas-core-shared
- **Description**: Specifies whether ConSol CM is running in a cluster.
- **Type**: boolean
- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.1.0

cluster.unicast

- **Module**: cmas-core-shared
- **Description**: Flag to activate jgroups unicast mode for ConSol CM clusters (as opposed to the default multicast mode causing problems in some data center environments). If set to "true", remember to set the JVM start parameters: `jgroups.bind.port`, `jgroups.bind.address` and `jgroups.initial_hosts`.
- **Type**: boolean
- **Restart required**: yes
- **System**: yes
- **Optional**: yes
- **Example value**: false (default value)
- **Since**: 6.11.0.0
- **Removed in**: 6.12.0.0

data.directory

- **Module**: cmas-core-shared
- **Description**: Directory for ConSol CM data (e.g., index)
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: C:\Users\user\cmas
- **Since**: 6.0

expert.mode

- **Module**: cmas-core-shared
- **Description**: Switches expert mode on/off thereby unblocking/blocking expert features. For example, the CM system property `initialized` is only available in the expert mode.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.0

G.2.2.9 cmas-dwh-server (module)

autocommit.cf.changes

- **Module**: cmas-dwh-server
- **Description**: Defines whether DWH tasks which result from configurational changes on ticket fields are executed automatically without manual interaction in the Admin Tool. Can be also set in the Admin Tool in the navigation item *DWH*. The default and recommended value is “false”.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false (default value)
- **Since**: 6.7.0

batch-commit-interval

- **Module**: cmas-dwh-server
- **Description**: Number of objects in a DWH message. Larger values mean better transfer performance at the cost of higher memory usage.
  Starting with ConSol CM version 6.11, this property is only used if the package size of a DWH operation is not set. This can only happen when the command is directly addressed to

the Java MBean `consol.cmas.global.dwh.synchronizationService`, e.g. using the `update()` method. When a DWH operation is started using the Admin Tool, there is always a value for the package size. If not explicitly set, the default value of 1000 is used as value for `batch.commit.interval`.

- **Default value**: 1000
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 100
- **Since**: 6.0.0

communication.channel

- **Module**: cmas-dwh-server
- **Description**: Communication channel. Only possible value since CM version 6.11.0.0: DIRECT
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: DIRECT
- **Since**: 6.8.5.0
- **Removed in**: 6.11.0.0 (DIRECT mode is the only available mode and is set automatically)

dwh.mode

- **Module**: cmas-dwh-server
- **Description**: Current mode for DWH data transfer. Possible values are OFF, ADMIN, LIVE
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: OFF
- **Since**: 6.0.1

ignore-queues

- **Module**: cmas-dwh-server
- **Description**: A comma-separated list of queue names which are not not transferred to the DWH.
- **Type**: string
- **Restart required**: no
- **System**: yes

- **Optional**: yes
- **Example value**: QueueName1,QueueName2,QueueName3
- **Since**: 6.6.19
- **Removed in**: 6.8.1

is.cmrf.alive

- **Module**: cmas-dwh-server
- **Description**: As a starting point, the time the last message was sent to CMRF should be used. If a response from CMRF is not received after the set value (in seconds), it should create a DWH operation status with an error message indicating that CMRF is down.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 1200
- **Since**: 6.7.0

java.naming.factory.initial

- **Module**: cmas-dwh-server
- **Description**: Factory class for the DWH context factory.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: org.jnp.interfaces.NamingContextFactory
- **Since**: 6.0.1
- **Removed in**: 6.11.0.0

java.naming.factory.url.pkgs

- **Module**: cmas-dwh-server
- **Description**:
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: org.jboss.naming:org.jnp.interfaces
- **Since**: 6.0.1
- **Removed in**: 6.11.0.0

java.naming.provider.url

- **Module**: cmas-dwh-server
- **Description**: URL of naming provider.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: localhost
- **Since**: 6.0.1
- **Removed in**: 6.11.0.0

last.ping.timestamp

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property shows the status of CMRF and is filled automatically by the CMRF. It contains the date of the last response on the ping from the ConSol CM server to the CMRF.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 32323214
- **Since**: 6.11.0.1

live.start

- **Module**: cmas-dwh-server
- **Description**: When the DWH synchronization mode is set to LIVE using the Admin Tool (navigation group *Data Warehouse*, navigation item *Administration*, *Configuration* button), this property is created and set to the current date.
  If LIVE mode is not enabled and there is no data in `cmas_dwh_ser_sync_object`, the property `live.start` is deleted.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes (automatically added in DWH “LIVE” mode)
- **Example value**: 15028802377645
- **Since**: 6.7.0

notification.error.description

- **Module**: cmas-dwh-server
- **Description**: Text for error emails from the DWH.
- **Type**: string

- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: Error occurred
- **Since**: 6.0.1

notification.error.from

- **Module**: cmas-dwh-server
- **Description**: From address for error emails from the DWH
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: myuser@consol.de
- **Since**: 6.0.1

notification.error.subject

- **Module**: cmas-dwh-server
- **Description**: Subject for error emails from the DWH
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: Error occurred
- **Since**: 6.0.1

notification.error.to

- **Module**: cmas-dwh-server
- **Description**: To address for error emails from the DWH
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: myuser@consol.de
- **Since**: 6.0.1

notification.finished_successfully.description

- **Module**: cmas-dwh-server
- **Description**: Text for emails from the DWH when a transfer finishes successfully.
- **Type**: string

- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: Transfer finished successfully
- **Since**: 6.0.1

notification.finished_successfully.from

- **Module**: cmas-dwh-server
- **Description**: From address for emails from the DWH when a transfer finishes successfully.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: myuser@consol.de
- **Since**: 6.0.1

notification.finished_successfully.subject

- **Module**: cmas-dwh-server
- **Description**: Subject for emails from the DWH when a transfer finishes successfully.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: Transfer finished successfully
- **Since**: 6.0.1

notification.finished_successfully.to

- **Module**: cmas-dwh-server
- **Description**: To address for emails from the DWH when a transfer finishes successfully.
- **Type**: string
- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: myuser@consol.de
- **Since**: 6.0.1

notification.finished_unsuccessfully.description

- **Module**: cmas-dwh-server
- **Description**: Text for emails from the DWH when a transfer finishes unsuccessfully.
- **Type**: string

- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: Transfer finished unsuccessfully
- **Since**: 6.0.1

notification.finished_unsuccessfully.from

- **Module**: cmas-dwh-server
- **Description**: From address for emails from the DWH when a transfer finishes unsuccessfully.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: myuser@consol.de
- **Since**: 6.0.1

notification.finished_unsuccessfully.subject

- **Module**: cmas-dwh-server
- **Description**: Subject for emails from the DWH when a transfer finishes unsuccessfully.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: Transfer finished unsuccessfully
- **Since**: 6.0.1

notification.finished_unsuccessfully.to

- **Module**: cmas-dwh-server
- **Description**: To address for emails from the DWH when a transfer finishes unsuccessfully.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: myuser@consol.de
- **Since**: 6.0.1

notification.host

- **Module**: cmas-dwh-server
- **Description**: Email (SMTP) server hostname for sending DWH emails.
- **Type**: string

- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: myserver.consol.de
- **Since**: 6.0.1

notification.password

- **Module**: cmas-dwh-server
- **Description**: Password for sending DWH emails (optional).
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.0.1

notification.port

- **Module**: cmas-dwh-server
- **Description**: SMTP port for sending DWH emails.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 25
- **Since**: 6.0.1

notification.protocol

- **Module**: cmas-dwh-server
- **Description**: The protocol used for sending emails from the DWH.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: pop3

notification.tls.enabled

- **Module**: cmas-dwh-server
- **Description**: Activates SMTP via SSL/TLS (SMTPS) for sending notification emails from the DWH. The default value is “false”. If it is set to “true”, SMTPS is activated for sending notifications from the DWH.
- **Type**: string

- **Restart required**: yes
- **System**: no
- **Optional**: yes
- **Example value**: false (default value)
- **Since**: 6.11.1.6

notification.username

- **Module**: cmas-dwh-server
- **Description**: (SMTP) user name for sending DWH emails.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: myuser
- **Since**: 6.0.1

recoverable.exceptions

- **Module**: cmas-dwh-server
- **Description**: Comma-separated list of exception definitions: CLASS[+][:REGEX]. The exceptions included in the list do not stop CM from sending to the CMRF process, but force it to try again. If optional '+' after CLASS is present, classes which extend CLASS are matched.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: java.sql.SQLRecoverableException,java.lang.RuntimeException+:.*T.1\,2T.*
- **Since**: 6.8.4.6

skip-ticket

- **Module**: cmas-dwh-server
- **Description**: Tickets are not transferred during transfer/update.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.6.19
- **Removed in**: 6.8.1

skip-ticket-history

- **Module**: cmas-dwh-server
- **Description**: History of ticket is not transferred during transfer/update.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.6.19
- **Removed in**: 6.8.1

skip-unit

- **Module**: cmas-dwh-server
- **Description**: Units are not transferred during transfer/update.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.6.19
- **Removed in**: 6.8.1

skip-unit-history

- **Module**: cmas-dwh-server
- **Description**: History of unit is not transferred during transfer/update.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.6.19
- **Removed in**: 6.8.1

split.history

- **Module**: cmas-dwh-server
- **Description**: Changes the SQL that fetches the history for the tickets during DWH transfer so that the history is not fetched for all tickets at once but only for one ticket per SQL.
- **Type**: boolean
- **Restart required**: no

- **System**: yes
- **Optional**: yes
- **Example value**: false
- **Since**: 6.8.0

statistics.calendar

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.client.group

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.contact.role

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.content.entry

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.content.entry.class

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.content.entry.history

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.customer.definition

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.

- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.engineer

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.enum.group

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.field.definition

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes

- **Example value**: 0
- **Since**: 6.11.0.1

statistics.group.definition

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.locale

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.localized.property

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.mla

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.project

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.queue

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.resource

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.

- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.resource.group

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.resource.history

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.resource.relation.definition

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes

- **Example value**: 0
- **Since**: 6.11.0.1

statistics.resource.type

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.ticket

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.ticket.function

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.ticket.history

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.time.booking

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.timestamp

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.unit

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.

- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.unit.history

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.unit.relation.definition

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.1

statistics.workflow

- **Module**: cmas-dwh-server
- **Description**: Internal DWH property, not to be changed manually. This property is filled automatically by the CMRF to store statistical data from the DWH. This data is displayed in the Admin Tool, section *Data Warehouse -> Monitor -> CMRF Statistics*.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes

- **Example value**: 0
- **Since**: 6.11.0.1

time.buffer

- **Module**: cmas-dwh-server
- **Description**: Number of minutes to extend date of start live mode.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 5
- **Since**: 6.8.1.11

unit.transfer.order

- **Module**: cmas-dwh-server
- **Description**: Defines in which order customer field groups should be transferred to the DWH.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: company;customer
- **Since**: 6.6.19
- **Removed in**: 6.8.1

G.2.2.10 cmas-nimh (module)

filesystem.polling.threads.number

- **Module**: cmas-nimh
- **Description**: Number of threads started for database emails' queue polling. The default value is 1
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 10
- **Since**: 6.4.0

filesystem.polling.threads.shutdown.timeout.seconds

- **Module**: cmas-nimh
- **Description**: Waiting time after the shutdown signal. When the timeout is reached, the thread will be terminated.

- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60 (default value)
- **Since**: 6.4.0

filesystem.polling.threads.watchdog.interval.seconds

- **Module**: cmas-nimh
- **Description**: Determines the interval in seconds to execute the watchdog which checks the activity of the threads of the file system poller.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 30 (default value)
- **Since**: 6.4.0

filesystem.task.enabled

- **Module**: cmas-nimh
- **Description**: With this property the service thread related to a given poller can be disabled.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true (default value)
- **Since**: 6.4.0

filesystem.task.interval.seconds

- **Module**: cmas-nimh
- **Description**: Default interval for polling mailboxes in seconds.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60 (default value)
- **Since**: 6.4.0

filesystem.task.polling.folder

- **Module**: cmas-nimh
- **Description**: Polling folder location which will be scanned for emails in the format of eml files. The default value is the `mail` subdirectory of the ConSol CM data directory
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: c://cmas//mail
- **Since**: 6.4.0

filesystem.task.timeout.seconds

- **Module**: cmas-nimh
- **Description**: After this time (of inactivity) the service thread is considered damaged and automatically restarted. Default: 120 seconds
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.4.0

filesystem.task.transaction.timeout.seconds

- **Module**: cmas-nimh
- **Description**: Default transaction timeout (in seconds) for email fetching transactions. Should be correlated with number of messages fetched at once.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60 (default value)
- **Since**: 6.4.0

mailbox.<NUMBER>.name

- **Module**: cmas-nimh
- **Description**: Identifier (name) of the mailbox.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes

- **Example value**: myEmailAccount
- **Since**: 6.11.2.0

mailbox.1.connection.host

- **Module**: cmas-nimh
- **Description**: Host (server) for first configured mailbox. Will overwrite the default parameter `mailbox.default.connection.host`.

mailbox.1.connection.password

- **Module**: cmas-nimh
- **Description**: Password for first configured mailbox. Will overwrite the default parameter `mailbox.default.connection.password`.

mailbox.1.connection.port

- **Module**: cmas-nimh
- **Description**: Port for first configured mailbox. Will overwrite the default parameter `mailbox.default.connection.port`.

mailbox.1.connection.protocol

- **Module**: cmas-nimh
- **Description**: Protocol (e.g., IMAP or POP3) for first configured mailbox. Will overwrite the default parameter `mailbox.default.connection.protocol`.

mailbox.1.connection.username

- **Module**: cmas-nimh
- **Description**: User name for first configured mailbox. Will overwrite the default parameter `mailbox.default.connection.username`.

mailbox.2.connection.host

- **Module**: cmas-nimh
- **Description**: Host (server) for second configured mailbox. Will overwrite the default parameter `mailbox.default.connection.host`.

mailbox.2.connection.password

- **Module**: cmas-nimh
- **Description**: Password for second configured mailbox. Will overwrite the default parameter `mailbox.default.connection.password`.

mailbox.2.connection.port

- **Module**: cmas-nimh
- **Description**: Port for second configured mailbox. Will overwrite the default parameter `mailbox.default.connection.port`.

mailbox.2.connection.protocol

- **Module**: cmas-nimh
- **Description**: Protocol (e.g., IMAP or POP3) for second configured mailbox. Will overwrite the default parameter `mailbox.default.connection.protocol`.

mailbox.2.connection.username

- **Module**: cmas-nimh
- **Description**: User name for second configured mailbox. Will overwrite the default parameter `mailbox.default.connection.username`.

> For all NIMH-related mailbox properties, the following principle is used: a default property is defined (e.g. `mailbox.default.connection.port`). If no mailbox-specific value is configured, this default value will be used.

mailbox.default.connection.host

- **Module**: cmas-nimh
- **Description**: Host (server name) of a given mailbox from which the poller reads emails.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 10.10.1.157
- **Since**: 6.4.0

mailbox.default.connection.password

- **Module**: cmas-nimh
- **Description**: Password for given mailbox from which the poller reads emails.
- **Type**: password
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: consol
- **Since**: 6.4.0

mailbox.default.connection.port

- **Module**: cmas-nimh
- **Description**: Port for a given mailbox from which the poller reads emails.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes

- **Example value**: 143
- **Since**: 6.4.0

mailbox.default.connection.protocol

- **Module**: cmas-nimh
- **Description**: Poller's protocol e.g., IMAP or POP3. No default value
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: imap
- **Since**: 6.4.0

mailbox.default.connection.username

- **Module**: cmas-nimh
- **Description**: User name for a given mailbox from which the poller reads emails.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: username
- **Since**: 6.4.0

mailbox.default.session.mail.debug

- **Module**: cmas-nimh
- **Description**: JavaMail property, see JavaMail API documentation for further information. Allows for more detailed JavaMail session debugging.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true
- **Since**: 6.4.0

mailbox.default.session.mail.mime.address.strict

- **Module**: cmas-nimh
- **Description**: JavaMail property, see JavaMail API documentation for further information. Counterpart of the old `mule mail.mime.strict`, allows to set not so strict email header parsing.
- **Type**: boolean
- **Restart required**: no

- **System**: no
- **Optional**: yes
- **Example value**: true
- **Since**: 6.4.0

mailbox.default.session.mail.<PROTOCOL>.connectiontimeout

- **Module**: cmas-nimh
- **Description**: JavaMail property, see JavaMail API documentation for further information. Determines the connection timeout in milliseconds.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 3000
- **Since**: 6.4.0 (IMAP and POP3) / 6.11.2.0 (IMAPS and POP3S)

mailbox.default.session.mail.<PROTOCOL>.fetchsize

- **Module**: cmas-nimh
- **Description**: JavaMail property, see JavaMail API documentation for further information. Determines the size of partial fetch in bytes for the indicated protocol.
  For IMAP systems: in CM versions 6.10.7.0 and up, the value of `mailbox.default.session.mail.imap.fetchsize` is set to 1048576 (equals 1 MB) during the initial setup of a ConSol CM system. During an update of an existing ConSol CM system, the value of the property is left unchanged, if the property is already present. In case the property is not yet present, it is added with the default value.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 1048576
- **Since**: 6.9.4.0 (IMAP and POP3) / 6.11.2.0 (IMAPS and POP3S)

mailbox.default.session.mail.<PROTOCOL>.partialfetch

- **Module**: cmas-nimh
- **Description**: JavaMail property, see JavaMail API documentation for further information. Determines whether the protocol's partial fetch capability should be used.
  For IMAP systems: in CM versions 6.10.7.0 and up, the value of `mailbox.default.session.mail.imap.partialfetch` is set to "false" during the initial setup of a ConSol CM system. During an update of an existing ConSol CM system, the value of the property is left unchanged, if the property is already present. In case the property is not yet present, it is added with the default value.
- **Type**: boolean
- **Restart required**: no

- **System**: no
- **Optional**: yes
- **Example value**: true
- **Since**: 6.9.4.0 (IMAP and POP3) / 6.11.2.0 (IMAPS and POP3S)

mailbox.default.session.mail.<PROTOCOL>.timeout

- **Module**: cmas-nimh
- **Description**: JavaMail property, see JavaMail API documentation for further information. Determines the I/O timeout in milliseconds.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 120 for IMAP(S) and 3000 for POP3(S)
- **Since**: 6.4.0 (IMAP and POP3), 6.11.2.0 (IMAPS and POP3S)

mailbox.default.task.delete.read.messages

- **Module**: cmas-nimh
- **Description**: This defines whether messages should be removed from the mailbox after processing. For IMAP protocol messages are marked as SEEN by default. For POP3 protocol, when flag is set to “true” the message is removed, otherwise remains on server and will result in infinite reads.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: false (default value)
- **Since**: 6.4.0

mailbox.default.task.enabled

- **Module**: cmas-nimh
- **Description**: With this property, the service thread related to the given poller can be disabled.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true (default value)
- **Since**: 6.4.0

mailbox.default.task.interval.seconds

- **Module**: cmas-nimh
- **Description**: Default interval for polling mailboxes in seconds.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60 (default value)
- **Since**: 6.4.0

mailbox.default.task.max.message.size

- **Module**: cmas-nimh
- **Description**: Maximum size of email messages (i.e., email plus attachment). Emails exceeding the size limit will not be automatically processed by NIMH but will be stored in the database (table `cmas_nimh_archived_mail`) and will therefore appear in the email backups in the Admin Tool (see section *Email Backups* in the *ConSol CM Administrator Manual*). From there they can be resent, downloaded to the file system, or deleted. For those operations the message size is not relevant.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 10485760 (default value, 10 MB)
- **Since**: 6.4.0

mailbox.default.task.max.messages.per.run

- **Module**: cmas-nimh
- **Description**: Number of messages fetched at once from mailbox. Must be correlated with transaction timeout. The default value is 20
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.4.0

mailbox.default.task.timeout.seconds

- **Module**: cmas-nimh
- **Description**: After this time (of inactivity) the service thread is considered damaged and automatically restarted. The default value is 120 seconds.
- **Type**: integer

- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.4.0

mailbox.default.task.transaction.timeout.seconds

- **Module**: cmas-nimh
- **Description**: Default transaction timeout (in seconds) for email fetching transactions. Should be correlated with number of messages fetched at once.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60 (default value)
- **Since**: 6.4.0

mailbox.polling.threads.mail.log.enabled

- **Module**: cmas-nimh
- **Description**: Enables email logging which is especially crucial in cluster environment (used as semaphore there)
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true (default value)
- **Since**: 6.9.4.1

mailbox.polling.threads.number

- **Module**: cmas-nimh
- **Description**: Number of threads for accessing mailboxes.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 1 (default value)
- **Since**: 6.4.0

queue.polling.threads.number

- **Module**: cmas-nimh
- **Description**: Number of threads started for polling the email queue in the database.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 1 (default value)
- **Since**: 6.4.0

queue.polling.threads.shutdown.timeout.seconds

- **Module**: cmas-nimh
- **Description**: Waiting time after the shutdown signal. When the timeout is reached, the thread will be terminated.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60 (default value)
- **Since**: 6.4.0

queue.polling.threads.watchdog.interval.seconds

- **Module**: cmas-nimh
- **Description**: Determines the interval in seconds to execute the watchdog which checks the activity of the threads of the email queue poller.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 30 (default value)
- **Since**: 6.4.0

queue.task.error.pause.seconds

- **Module**: cmas-nimh
- **Description**: Maximum number of seconds, the queue poller waits after infrastructure (e.g. database) error.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes

- **Example value**: 180 (default value)
- **Since**: 6.4.0

queue.task.interval.seconds

- **Module**: cmas-nimh
- **Description**: Main emails' queue polling thread interval.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 15 (default value)
- **Since**: 6.4.0

queue.task.max.retries

- **Module**: cmas-nimh
- **Description**: Maximum number of email processing retries after an exception. When reached, the email is moved to the email archive. This email can be rescheduled again using NIMH API (or the Admin Tool).
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 10
- **Since**: 6.4.0

queue.task.timeout.seconds

- **Module**: cmas-nimh
- **Description**: After this time of inactivity (in seconds), the service thread is considered damaged and automatically restarted.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 600 (default value)
- **Since**: 6.4.0

queue.task.transaction.timeout.seconds

- **Module**: cmas-nimh
- **Description**: Transaction timeout for email processing in the pipe.
- **Type**: integer
- **Restart required**: no

- **System**: no
- **Optional**: yes
- **Example value**: 60 (default value)
- **Since**: 6.4.0

G.2.2.11 cmas-nimh-extension (module)

mail.attachments.validation.info.sender

- **Module**: cmas-nimh-extension
- **Description**: Sets the From header for error notification emails which are sent when the validation of the attachment type fails.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: admin@mail.com
- **Since**: 6.7.5

mail.attachments.validation.info.subject

- **Module**: cmas-nimh-extension
- **Description**: Sets the subject for error notification emails which are sent when the validation of the attachment type fails.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: Email was not processed because its attachments were rejected!
- **Since**: 6.7.5

mail.db.archive

- **Module**: cmas-nimh-extension
- **Description**: If property is set to “true”, incoming emails are archived in the database.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: false (default value)
- **Since**: 6.8.5.5

mail.error.from.address

- **Module**: cmas-nimh-extension
- **Description**: From address for error emails from NIMH
- **Type**: email
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: myuser@consol.de
- **Since**: 6.4.0

mail.error.to.address

- **Module**: cmas-nimh-extension
- **Description**: To address for error emails from NIMH. As a default the email address of the administrator which you have entered during system setup is used.
- **Type**: email
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: myuser@consol.de
- **Since**: 6.4.0

mail.on.error

- **Module**: cmas-nimh-extension
- **Description**: If set to “true” an error email is sent to the above configured address in case the email message could not be processed.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true (default)
- **Since**: 6.4.0

mail.ticketname.pattern

- **Module**: cmas-nimh-extension
- **Description**: Regular expression pattern used to identify the ticket name in the subject of incoming mails.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no

- **Example value**: .*?Ticket\s+\((\S+)\).*
- **Since**: 6.4.0

G.2.2.12 cmas-restapi-core (module)

comment.authors.disabled

- **Module**: cmas-restapi-core
- **Description**: Disables the display of the content's author via REST API.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: false (default value)
- **Since**: 6.11.0

csrf.domain.allow.none

- **Module**: cmas-restapi-core
- **Description**: Determines whether empty `Origin/Referer` headers are accepted. By default, the property is set to “false”, so that existing REST requests without `Origin/Referer` headers work.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false (default value)
- **Since**: 6.11.2.2

csrf.domain.white.list

- **Module**: cmas-restapi-core
- **Description**: The list of domains (separated with “|”) which are allowed in `Origin/Referer` headers and will not be blocked by the CSRF (cross-site request forgery) filter. By default, the property is empty so that cross-site requests are blocked.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: https://example.com:80 | http://www.consol.de:8080
- **Since**: 6.11.2.2

csrf.request.filter.enabled

- **Module**: cmas-restapi-core
- **Description**: It allows to disable the CSRF (cross-site request forgery) request filter for the REST API.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true (default value)
- **Since**: 6.11.2.2

diff.tracking.disabled

- **Module**: cmas-restapi-core
- **Description**: Fallback property for disabling diff tracking for CM/Track, which is history-based so it can be heavy.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: false (default value)
- **Since**: 6.10.5.6

domain.map.for.client.config.<DOMAIN_NAME>

- **Module**: cmas-restapi-core
- **Description**: Enables mapping multiple instances of CM/Track to a specific domain. DOMAIN_NAME is the name of the client configuration in the Admin Tool (if your CM/Track configuration is called “trackV2customized”, the name of the property is `domain.map.for.client.config.trackV2customized`). You can provide several URLs separated by a comma.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: https://www.consol.de, https://www.consol.com
- **Since**: 6.10.7.0

security.fields.customer.exposure.check.enabled

- **Module**: cmas-restapi-core
- **Description**: Enables customer exposure annotation checks for ticket fields.
- **Type**: boolean
- **Restart required**: no

- **System**: no
- **Optional**: yes
- **Example value**: true (default value)
- **Since**: 6.10.5.4

security.restrict.unit.access.to.own.data

- **Module**: cmas-restapi-core
- **Description**: If set to “true”, an additional check is performed when a user logs in as a customer using the REST API, e.g. CM/Track. When requesting customer data, only the company of the user or other contacts of the user’s company are returned. If set to “false”, no additional security check is performed and the former security rules apply.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true (default value)
- **Since**: 6.9.2.14

G.2.2.13 cmas-restapi-http-headers (module)

This module contains properties which represent header fields which should be sent with REST API requests and responses.

Each property in the module `cmas-restapi-http-headers` represents one header field. The property name/key identifies the HTTP header field and the value of the property is the field value sent in this header.

Please be aware that additional HTTP response headers must be correctly defined with the exact spelling as officially specified. Please note also that the correct interpretation and application of these headers is fully in the realm and responsibility of the HTTP client.

For example, if you add the property `Access-Control-Allow-Origin` with the value “http://www.example.com” to the `cmas-restapi-http-headers` module, REST API requests from *http://www.example.com* can be allowed in your domain even if the general policy is to allow only requests from the same domain.

As there are no default values for the headers in this module, the module is not displayed when creating the first property in it. The module name needs to be typed in manually in this case.

G.2.2.14 cmas-setup-hibernate (module)

cmas.dropSchemaBeforeSetup

- **Module**: cmas-setup-hibernate
- **Description**: Flag if schema is to be (was) dropped during setup
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no

- **Example value**: true
- **Since**: 6.0

connection.release.mode

- **Module**: cmas-setup-hibernate
- **Description**: Describes the JEE connection handling strategy for transactions. If set to “AFTER_TRANSACTION”, the connection will be cached during the transaction and released at the end. If set to “AFTER_STATEMENT”, the connection will be released to the pool after each statement execution. Please do not change the default here unless advised by ConSol.
- **Type**: string
- **Restart required**: yes
- **System**: no
- **Optional**: yes
- **Example value**: AFTER_STATEMENT (default for JEE environment)
- **Since**: 6.0

hibernate.dialect

- **Module**: cmas-setup-hibernate
- **Description**: The dialect used by hibernate. Usually set during initial set-up (depending on the database system).
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: org.hibernate.dialect.MySQL5InnoDBDialect
- **Since**: 6.0

update.6.11.0.0.sleep

- **Module**: cmas-setup-hibernate
- **Description**: Helper property for the update preparation scripts introduced in context of CM database refactoring in version 6.11. This is an optional setting allowing a delay (in milliseconds) after each loop iteration of the preparation scripts. Setting the delay should lower the database load, for example during working hours. This property may be removed after the update preparation tasks finish.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 0
- **Since**: 6.11.0.0, for use in 6.10.5.x

update.6.11.0.0.timezone

- **Module**: cmas-setup-hibernate
- **Description**: Helper property for the ticket history migration (the new way of counting history groups). Since 6.11.0.0 the groups are constant (2h time span), but before 6.11.0.0 groups were not constant and depended on the customer's time zone. Migration scripts use an old algorithm to calculate groups and therefore need information about the time zone. The property should be set to the timezone which is most commonly used by the customers. If the property is not set, the default server time zone is used (`TimeZone.getDefault()`). The property should be set before updating to 6.11.0.0 and will be removed automatically after migration. The list of accepted timezones can be found for example here: http://joda-time.sourceforge.net/timezones.html.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: Europe/Berlin
- **Since**: 6.11.0.0, for use before updating to this version

G.2.2.15 cmas-setup-manager (module)

initialized

- **Module**: cmas-setup-manager
- **Description**: Flag if ConSol CM is initialized. If this value is missing or not "true", the setup will be performed. Starting with ConSol CM version 6.11, this property is only available in expert mode.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true
- **Since**: 6.0

> ⚠ Be careful with using this property! When you set the value to "false", the ConSol CM server will perform the system setup at the next start, i.e. all data of the existing system is lost, including system properties!

G.2.2.16 cmas-setup-scene (module)

scene

- **Module**: cmas-setup-scene
- **Description**: Scene file which was imported during setup (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

G.2.2.17 cmas-web-notifications (module)

notifications.enabled

- **Module**: cmas-web-notifications
- **Description**: Determines whether the notifications feature is enabled ("true") or disabled ("false"). The default value is "true" for ConSol CM systems running on JBoss application servers, and "false" for ConSol CM systems running on WebLogic application servers.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: true
- **Since**: 6.12.0.0

ttl.days

- **Module**: cmas-web-notifications
- **Description**: Determines the time (in days) after which unpinned notifications are removed.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: 15
- **Since**: 6.12.0.0

G.2.2.18 cmas-workflow-engine (module)

jobExecutor.adminMail

- **Module**: cmas-workflow-engine
- **Description**: Email address which will get notified about job execution problems (when retry counter is exceeded).
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes

- **Example value**: admin@consol.de
- **Since**: 6.8.0

jobExecutor.idleInterval.seconds

- **Module**: cmas-workflow-engine
- **Description**: Determines how often the job executor thread will look for new jobs to execute.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 45 (default value up to CM version 6.10.5.2. The default value for CM versions 6.10.5.3 and up is 5)
- **Since**: 6.8.0

jobExecutor.jobMaxRetries

- **Module**: cmas-workflow-engine
- **Description**: Controls the number of retry attempts the job executor will do before declaring a job as failed.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 5 (default value)
- **Since**: 6.8.0

jobExecutor.jobMaxRetriesReachedSubject

- **Module**: cmas-workflow-engine
- **Description**: The subject used in the notification mail which administrators receive about failed job executors.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: Job maximum retries reached. Job was removed!!! (default value)
- **Since**: 6.8.0

jobExecutor.lockingLimit

- **Module**: cmas-workflow-engine
- **Description**: Number of jobs locked at once (marked for execution) by the job executor thread.
- **Type**: integer
- **Restart required**: no

- **System**: yes
- **Optional**: yes
- **Example value**: 5 (default value since CM version 6.10.5.3)
- **Since**: 6.8.0

jobExecutor.lockTimeout.seconds

- **Module**: cmas-workflow-engine
- **Description**: Determines how long the job can be locked (marked for execution) by the job executor.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 360 (default value)
- **Since**: 6.8.0

jobExecutor.mailFrom

- **Module**: cmas-workflow-engine
- **Description**: Email address which will be set as From header for notifications to the administrator.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: jobexecutor@consol.de
- **Since**: 6.8.0

jobExecutor.maxInactivityInterval.minutes

- **Module**: cmas-workflow-engine
- **Description**: Number of minutes of allowed job executor inactivity (e.g. when it is blocked by long timer execution). After this time executors threads are restarted.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 30 (default value)
- **Since**: 6.9.2.0

jobExecutor.threads

- **Module**: cmas-workflow-engine
- **Description**: Number of job execution threads.

- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 1 (default value)
- **Since**: 6.8.0

jobExecutor.timerRetryInterval.seconds

- **Module**: cmas-workflow-engine
- **Description**: Determines how long the job executor thread will wait after job execution error.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 10 (default up to CM version 6.10.5.2. The default value for CM versions 6.10.5.3 and up is 30)
- **Since**: 6.8.0

jobExecutor.txTimeout.seconds

- **Module**: cmas-workflow-engine
- **Description**: Transaction timeout used for job execution.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 60 (default value)
- **Since**: 6.8.0

G.2.2.19 cmas-workflow-jbpm (module)

fetchLock.interval

- **Module**: cmas-workflow-jbpm
- **Description**:
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 5000
- **Removed in**: 6.8.0

jobExecutor.idleInterval

- **Module**: cmas-workflow-jbpm
- **Description**:
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 45000
- **Removed in**: 6.8.0
- **Replaced by**: jobExecutor.idleInterval.seconds

jobExecutor.jobExecuteRetryNumber

- **Module**: cmas-workflow-jbpm
- **Description**:
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 5
- **Removed in**: 6.8.0
- **Replaced by**: jobExecutor.jobMaxRetries

jobExecutor.timerRetryInterval

- **Module**: cmas-workflow-jbpm
- **Description**:
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 10000
- **Removed in**: 6.8.0
- **Replaced by**: jobExecutor.timerRetryInterval.seconds

mail.sender.address

- **Module**: cmas-workflow-jbpm
- **Description**: From address for emails from the workflow engine.
- **Type**: string
- **Restart required**: no
- **System**: yes

- **Optional**: no
- **Example value**: myuser@consol.de
- **Removed in**: 6.8.0
- **Replaced by**: jobExecutor.mailFrom

outdated.lock.age

- **Module**: cmas-workflow-jbpm
- **Description**:
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 60000
- **Removed in**: 6.8.0
- **Replaced by**: cmas-workflow-engine, jobExecutor.lockTimeout.seconds

refreshTimeInCaseOfConcurrentRememberMeRequests

- **Module**: cmas-workflow-jbpm
- **Description**: It sets the refresh time (in seconds) after which the 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

G.2.2.20 cmweb-server-http-headers (module)

X-Frame-Options

- **Module**: cmweb-server-http-headers
- **Description**: Example property to illustrate the configuration of HTTP headers. In this case the delivered HTTP header contains the field *X-Frame-Options* with the value “SAMEORIGIN”.

  Each property in the module `cmweb-server-http-headers` represents one header field. The property name/key identifies the response header field and the value of the property is the field value sent in this header.

> ⚠ Please be aware that additional HTTP response headers must be correctly defined with the exact spelling as officially specified! Please note also that the correct interpretation and application of these headers is fully in the realm and responsibility of the client browser which requested the page!

- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: SAMEORIGIN
- **Since**: 6.10.8

G.2.2.21 cmweb-server-adapter (module)

attachment.upload.timeout

- **Module**: cmweb-server-adapter
- **Description**: Defines the transaction timeout in minutes for adding attachments to a ticket, resource or customer. Counts the time for the upload of all attachments of one transaction. When the timeout occurs, all files which have been temporarily stored on the server are deleted. No file is uploaded.
- **Type**: Integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 3
- **Since**: 6.10.5.3

automatic.booking.enabled

- **Module**: cmweb-server-adapter
- **Description**: If enabled, time spend on creating comment/email will be measured and automatic time booking will be added.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: true
- **Since**: 6.9.4.2

checkUserOnlineIntervalInSeconds

- **Module**: cmweb-server-adapter
- **Description**: The interval in seconds to check which users are online (default 180sec = 3min).

- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 180
- **Since**: 6.0
- **Removed in**: 6.5 / 6.11.0.1

cmoffice.enabled

- **Module**: cmweb-server-adapter
- **Description**: Flag if CM/Doc (former CM/Office) is enabled.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.4.0

cmoffice.oo.path.NUMBER

- **Module**: cmweb-server-adapter
- **Description**: Possible location of the OpenOffice installation. The properties are numbered starting with 0.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: c:\Program Files (x86)\LibreOffice 3.6\program
- **Since**: 6.10.1.0
- **Removed in**: 6.12.0.0

cmoffice.strict.versioning.enabled

- **Module**: cmweb-server-adapter
- **Description**: Controls if the SAVE operation in Microsoft Word / OpenOffice documents creates a new attachment (“true”) or overwrites the existing attachment (“false”). This concerns the behavior within one session using the text editing program. If the program is stopped, the overwrite mechanism will not work anymore.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes

- **Example value**: true
- **Since**: 6.10.5.4

cmoffice.websocket.port

- **Module**: cmweb-server-adapter
- **Description**: Determines the port on which the CM/Doc application should be started and connected to.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 30333
- **Since**: 6.12.0.0

commentRequiredForTicketCreation

- **Module**: cmweb-server-adapter
- **Description**: Flag if the comment is a required field for ticket creation.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true (default value)
- **Since**: 6.2.0

csrf.domain.white.list

- **Module**: cmweb-server-adapter
- **Description**: The list of domains (separated with "|") which are allowed and will not be checked by CSRF (cross-site request forgery) filter
- **Type**: String
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: example.com | consol.de
- **Since**: 6.10.7.0

csrf.request.filter.enabled

- **Module**: cmweb-server-adapter
- **Description**: It allows to disable CSRF (Cross-site request forgery) request filter
- **Type**: Boolean
- **Restart required**: no

- **System**: no
- **Optional**: yes
- **Example value**: true
- **Since**: 6.10.7.0

customizationVersion

- **Module**: cmweb-server-adapter
- **Description**: UID representing the latest web customization version. Used only internally, please do not change the value.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: cd58453e-f3cc-4538-8030-d15e8796a4a7
- **Since**: 6.5.0

data.optimization

- **Module**: cmweb-server-adapter
- **Description**: Defines optimization to be applied on response data. So far, the following values are supported (for setting more than one value, separate values by '|'): MINIFICATION and COMPRESSION. MINIFICATION minifies HTML data by e.g. stripping whitespaces and comments. COMPRESSION applies gzip compression to HTTP response. (Note: If you are running in cluster mode and want to test different configurations in parallel, you can set different values for each cluster node by specifying property `data.optimization.nodeId` to override default property.)
- **Type**: string
- **Restart required**: COMPRESSION can be switched on/off without restart, MINIFICATION requires restart.
- **System**: yes
- **Optional**: yes
- **Example value**: MINIFICATION|COMPRESSION

defaultAttachmentEntryClassName

- **Module**: cmweb-server-adapter
- **Description**: The default content entry class used to classify an attachment if no other class was set explicitly.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: DefaultTextElement
- **Since**: 6.9.2.0

defaultContentEntryClassName

- **Module**: cmweb-server-adapter
- **Description**: Default text class for new ACIMs.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: default_class
- **Since**: 6.3.0

defaultNumberOfCustomFieldsColumns

- **Module**: cmweb-server-adapter
- **Description**: Default number of columns for ticket fields.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 3
- **Since**: 6.2.0

diffTrackingEnabled

- **Module**: cmweb-server-adapter
- **Description**: Removed in ConSol CM version 6.11.
  Defines if parallel editing of a ticket by different engineers should be possible.
  “false”: Previous way of handling changes when editing a ticket. If the ticket has been changed in the meantime, the current engineer will not be able to submit his changes without being forced to reload the page before submitting.
  “true”: New changes handling mode. If the ticket has been changed, this will not block the submission of other changes anymore. If the part of the ticket that was changed was exactly the part that is changed by the submitting engineer, then an information message will be displayed, but the ticket change will be persisted/stored anyway.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true (default)
- **Since**: 6.10.1
- **Removed in**: 6.11.0

diffTrackingEnabledForUnitAndResource

- **Module**: cmweb-server-adapter
- **Description**: Enables the prevention of concurrent modifications on units / resources.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 3
- **Since**: 6.11.0.0

favoritesSizeLimit

- **Module**: cmweb-server-adapter
- **Description**: Maximum number of items in Favorites list.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 10
- **Since**: 6.0

forward.mails.to.representatives

- **Module**: cmweb-server-adapter
- **Description**: Determines if emails which are manually sent from the Web Client are also sent to representing engineers. The default value of the property is “false”, meaning that this kind of emails are not forwarded to the representing engineer. Set the property to “true” if you want to restore the previous behavior, i.e., all emails which are sent to the represented engineer are automatically forwarded to the representing engineer. Please take into account that this might not be desired if the same person is an engineer and a customer in the CM system.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: no
- **Example value**: false (default value)
- **Since**: 6.11.1.7

> This property only configures the handling of manually sent emails. The handling of automatically sent emails depends on the used Java method.

globalSearchResultSizeLimit

- **Module**: cmweb-server-adapter
- **Description**: Maximum number of items in Quick Search result.

- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 10
- **Since**: 6.0

helpFilePath

- **Module**: cmweb-server-adapter
- **Description**: URL for online help. If not empty, the *Help* link is displayed in the Web Client.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: http://www.consol.de
- **Since**: 6.2.1

hideTicketSubject

- **Module**: cmweb-server-adapter
- **Description**: If set to “true”, the ticket subject is hidden.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.2.1

mail.from

- **Module**: cmweb-server-adapter
- **Description**: This email address is used instead of the engineer’s email address during email conversations.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: consolcm@example.com
- **Since**: 6.1.2

mail.reply.to

- **Module**: cmweb-server-adapter
- **Description**: When set, the Ticket Email Editor in the Web Client displays a Reply-To field pre-filled with this value.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: consolcm@example.com
- **Since**: 6.0.1

> ⚠ Please read the detailed information about ConSol CM Reply-To addresses in section *Scripts of Type Email* in the *ConSol CM Administrator Manual*.

mailTemplateAboveQuotedText

- **Module**: cmweb-server-adapter
- **Description**: Determines the behavior of the email template in the Ticket Email Editor when another email is quoted, i.e. forwarded or replied to. Often used to place the signature correctly.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.2.4

maxSizePerPagemapInMegaBytes

- **Module**: cmweb-server-adapter
- **Description**: The parameter defines the size (in MB) of the file which is created by the Wicket framework per user session. i.e. for each engineer which is currently logged in. The file is used to save pages during the running session. When the defined size limit has been reached and new entries are added, the oldest entries are removed. In the Web Client, due to this behavior, an engineer who works with an "old" page will be redirected to the *Overview*/Start page (usually the dashboard page) when the "old" page is removed from the file.
  So in case engineers who work with a great number of open tabs in ConSol CM and complain about being redirected to the *Overview* page, it might be useful to increase this parameter. In large systems, you could use e.g. a value of 45 or 50. Since this is the size of the file which is saved on disk, the maximum value depends on the available disk space, however, a value which is too large is not recommended either.
- **Type**: integer
- **Restart required**: yes
- **System**: yes

- **Optional**: no
- **Example value**: 15
- **Since**: 6.3.5

pagemapLockDurationInSeconds

- **Module**: cmweb-server-adapter
- **Description**: Number of seconds to pass before pagemap is considered to be locked for too long.
- **Type**: integer
- **Restart required**: yes
- **System**: yes
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.7.3

postActivityExecutionScriptName

- **Module**: cmweb-server-adapter
- **Description**: Defines the name for the script which should be executed after every workflow activity, see section *PostActivityExecutionScript* in the *ConSol CM Administrator Manual*. If no script should be executed, leave the value empty.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: postActivityExecutionHandler
- **Since**: 6.2.0

queuesExcludedFromGS

- **Module**: cmweb-server-adapter
- **Description**: Comma-separated list of queue names which are excluded from Quick Search.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: Helpdesk,FAQ
- **Since**: 6.0

rememberMeLifetimeInMinutes

- **Module**: cmweb-server-adapter
- **Description**: Lifetime for *remember me* in minutes.
- **Type**: integer

- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: 1440
- **Since**: 6.0

request.scope.transaction

- **Module**: cmweb-server-adapter
- **Description**: It allows to disable request scope transaction. By default one transaction is used per request. Setting this property to “false” there will cause one transaction per service method invocation.
- **Type**: boolean
- **Restart required**: yes
- **System**: yes
- **Optional**: yes
- **Example value**: true
- **Since**: 6.8.1

searchPageSize

- **Module**: cmweb-server-adapter
- **Description**: Default page size for search results.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 20
- **Since**: 6.0

searchPageSizeOptions

- **Module**: cmweb-server-adapter
- **Description**: Options for the page size for search results.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 10|20|30|40|50|75|100
- **Since**: 6.0

serverPoolingInterval

- **Module**: cmweb-server-adapter
- **Description**: Defines the time in seconds for pooling server to invalidate caches on the web layer.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 5
- **Since**: 6.1.0

supportEmail

- **Module**: cmweb-server-adapter
- **Description**:
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.0
- **Removed in**: 6.11.0.1

themeOverlay

- **Module**: cmweb-server-adapter
- **Description**: Name of the folder containing the skin used to customize the layout of the Web Client.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: consolINT
- **Since**: 6.0, functionality changed in 6.11.2.0

ticketListRefreshIntervalInSeconds

- **Module**: cmweb-server-adapter
- **Description**: Refresh interval for the ticket list (in seconds).
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no

- **Example value**: 180
- **Since**: 6.0

ticketListSizeLimit

- **Module**: cmweb-server-adapter
- **Description**: Maximum number of tickets in ticket list.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 100
- **Since**: 6.0

tx.read.only.mode.enabled

- **Module**: cmweb-server-adapter
- **Description**: Enables read-only transactions for faster page loading. This transactional behavior was introduced in 6.11.0, and this property acts as a safety guard to restore the old behaviors. Do not change this value unless facing tx problems and advised by ConSol.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true (default value)
- **Since**: 6.11

unitIndexSearchResultSizeLimit

- **Module**: cmweb-server-adapter
- **Description**: Maximum number of units in unit search result (e.g. when searching for contact).
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 5
- **Since**: 6.0

urlLogoutPath

- **Module**: cmweb-server-adapter
- **Description**: URL which is opened when the user logs out. There are three configuration possibilities:
  - Empty value: The user is redirected to the login page.
  - <CM URL>/cm-client/logout: The user is redirected to the logout page, which includes a link to log in again.
  - Any URL: The user is redirected to the given URL outside ConSol CM.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: https://mycmserver/cm-client/logout
- **Since**: 6.3.1, behavior changed in 6.11.2.0

voCacheEnabled

- **Module**: cmweb-server-adapter
- **Description**: This property enables additional caching for the Web Client, voCaching, of complete objects, thus improving performance.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: true
- **Since**: 6.11.1.0

Notes:

- Since ConSol CM version 6.11.1.1, the default value is “true” for non-clustered environments. The value is set to “true” automatically during the setup or update of ConSol CM 6.11.1.1.
- When voCaching is enabled and lazy loading is used for folding ticket history entries, once the engineer unfolded an entry, he cannot fold it again by reloading the page or opening the ticket from the workspace.
- When using the dynamic mode for displaying engineer and customer names in the ticket history (as configured in the system properties cmas-core-server, engineer.description.mode and cmas-core-server, unit.description.mode), the new version of the engineer and/or customer name is only displayed after the ticket has been changed.
- This system property is ignored for clustered environments (environments with cmas-core-shared, cluster.mode set to “true”). In clustered environments, voCaching is always disabled to avoid problems that changes made to objects on one node are not visible on the other nodes.

webSessionTimeoutInMinutes

- **Module**: cmweb-server-adapter
- **Description**: Session timeout in minutes.
- **Type**: integer
- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: 180
- **Removed in**: 6.7.1
- **Replaced by**: cmas-core-server, server.session.timeout

wicketAjaxRequestHeaderFilterEnabled

- **Module**: cmweb-server-adapter
- **Description**: This enables filter for Wicket AJAX requests, coming from stale pages with Wicket 1.4 scripting (CM pre-6.8.0), after update to CM6 post-6.8.0.
- **Type**: boolean
- **Restart required**: yes
- **System**: yes
- **Optional**: yes
- **Example value**: false
- **Since**: 6.8.1

G.2.3 List of System Properties by Area

This chapter lists the system properties which are relevant for the following areas.

- CMRF & DWH Configuration
- Indexer and Search Configuration
- LDAP Configuration
- Email Configuration
- Activity Interval Configuration
- Administrator Email Addresses
- HTTP Header Configuration

CM/ is a registered trademark of ConSol Consulting & Solutions Software GmbH

G.2.3.1 CMRF & DWH Configuration

autocommit.cf.changes

- **Module**: cmas-dwh-server
- **Description**: Defines whether DWH tasks which result from configurational changes on ticket fields are executed automatically without manual interaction in the Admin Tool. Can be also set in the Admin Tool in the navigation item *DWH*. The default and recommended value is “false”.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false (default value)
- **Since**: 6.7.0

batch-commit-interval

- **Module**: cmas-dwh-server
- **Description**: Number of objects in a DWH message. Larger values mean better transfer performance at the cost of higher memory usage.
  Starting with ConSol CM version 6.11, this property is only used if the package size of a DWH operation is not set. This can only happen when the command is directly addressed to the Java MBean `consol.cmas.global.dwh.synchronizationService`, e.g. using the `update()` method. When a DWH operation is started using the Admin Tool, there is always a value for the package size. If not explicitly set, the default value of 1000 is used as value for `batch.commit.interval`.
- **Default value**: 1000
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 100
- **Since**: 6.0.0

communication.channel

- **Module**: cmas-dwh-server
- **Description**: Communication channel. Only possible value since CM version 6.11.0.0: DIRECT
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: DIRECT

- **Since**: 6.8.5.0
- **Removed in**: 6.11.0.0 (DIRECT mode is the only available mode and is set automatically)

dwh.mode

- **Module**: cmas-dwh-server
- **Description**: Current mode for DWH data transfer. Possible values are OFF, ADMIN, LIVE
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: OFF
- **Since**: 6.0.1

ignore-queues

- **Module**: cmas-dwh-server
- **Description**: A comma-separated list of queue names which are not not transferred to the DWH.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: QueueName1,QueueName2,QueueName3
- **Since**: 6.6.19
- **Removed in**: 6.8.1

is.cmrf.alive

- **Module**: cmas-dwh-server
- **Description**: As a starting point, the time the last message was sent to CMRF should be used. If a response from CMRF is not received after the set value (in seconds), it should create a DWH operation status with an error message indicating that CMRF is down.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 1200
- **Since**: 6.7.0

java.naming.factory.initial

- **Module**: cmas-dwh-server
- **Description**: Factory class for the DWH context factory.
- **Type**: string

- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: org.jnp.interfaces.NamingContextFactory
- **Since**: 6.0.1
- **Removed in**: 6.11.0.0

java.naming.factory.url.pkgs

- **Module**: cmas-dwh-server
- **Description**:
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: org.jboss.naming:org.jnp.interfaces
- **Since**: 6.0.1
- **Removed in**: 6.11.0.0

java.naming.provider.url

- **Module**: cmas-dwh-server
- **Description**: URL of naming provider.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: localhost
- **Since**: 6.0.1
- **Removed in**: 6.11.0.0

notification.error.description

- **Module**: cmas-dwh-server
- **Description**: Text for error emails from the DWH.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: Error occurred
- **Since**: 6.0.1

notification.error.from

- **Module**: cmas-dwh-server
- **Description**: From address for error emails from the DWH
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: myuser@consol.de
- **Since**: 6.0.1

notification.error.subject

- **Module**: cmas-dwh-server
- **Description**: Subject for error emails from the DWH
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: Error occurred
- **Since**: 6.0.1

notification.error.to

- **Module**: cmas-dwh-server
- **Description**: To address for error emails from the DWH
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: myuser@consol.de
- **Since**: 6.0.1

notification.finished_successfully.description

- **Module**: cmas-dwh-server
- **Description**: Text for emails from the DWH when a transfer finishes successfully.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: Transfer finished successfully
- **Since**: 6.0.1

notification.finished_successfully.from

- **Module**: cmas-dwh-server
- **Description**: From address for emails from the DWH when a transfer finishes successfully.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: myuser@consol.de
- **Since**: 6.0.1

notification.finished_successfully.subject

- **Module**: cmas-dwh-server
- **Description**: Subject for emails from the DWH when a transfer finishes successfully.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: Transfer finished successfully
- **Since**: 6.0.1

notification.finished_successfully.to

- **Module**: cmas-dwh-server
- **Description**: To address for emails from the DWH when a transfer finishes successfully.
- **Type**: string
- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: myuser@consol.de
- **Since**: 6.0.1

notification.finished_unsuccessfully.description

- **Module**: cmas-dwh-server
- **Description**: Text for emails from the DWH when a transfer finishes unsuccessfully.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: Transfer finished unsuccessfully
- **Since**: 6.0.1

notification.finished_unsuccessfully.from

- **Module**: cmas-dwh-server
- **Description**: From address for emails from the DWH when a transfer finishes unsuccessfully.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: myuser@consol.de
- **Since**: 6.0.1

notification.finished_unsuccessfully.subject

- **Module**: cmas-dwh-server
- **Description**: Subject for emails from the DWH when a transfer finishes unsuccessfully.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: Transfer finished unsuccessfully
- **Since**: 6.0.1

notification.finished_unsuccessfully.to

- **Module**: cmas-dwh-server
- **Description**: To address for emails from the DWH when a transfer finishes unsuccessfully.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: myuser@consol.de
- **Since**: 6.0.1

notification.host

- **Module**: cmas-dwh-server
- **Description**: Email (SMTP) server hostname for sending DWH emails.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: myserver.consol.de
- **Since**: 6.0.1

notification.password

- **Module**: cmas-dwh-server
- **Description**: Password for sending DWH emails (optional).
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.0.1

notification.port

- **Module**: cmas-dwh-server
- **Description**: SMTP port for sending DWH emails.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 25
- **Since**: 6.0.1

notification.protocol

- **Module**: cmas-dwh-server
- **Description**: The protocol used for sending emails from the DWH.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: pop3

notification.tls.enabled

- **Module**: cmas-dwh-server
- **Description**: Activates SMTP via SSL/TLS (SMTPS) for sending notification emails from the DWH. The default value is “false”. If it is set to “true”, SMTPS is activated for sending notifications from the DWH.
- **Type**: string
- **Restart required**: yes
- **System**: no
- **Optional**: yes
- **Example value**: false (default value)
- **Since**: 6.11.1.6

notification.username

- **Module**: cmas-dwh-server
- **Description**: (SMTP) user name for sending DWH emails.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: myuser
- **Since**: 6.0.1

skip-ticket

- **Module**: cmas-dwh-server
- **Description**: Tickets are not transferred during transfer/update.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.6.19
- **Removed in**: 6.8.1

skip-ticket-history

- **Module**: cmas-dwh-server
- **Description**: History of ticket is not transferred during transfer/update.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.6.19
- **Removed in**: 6.8.1

skip-unit

- **Module**: cmas-dwh-server
- **Description**: Units are not transferred during transfer/update.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no

- **Example value**: false
- **Since**: 6.6.19
- **Removed in**: 6.8.1

skip-unit-history

- **Module**: cmas-dwh-server
- **Description**: History of unit is not transferred during transfer/update.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.6.19
- **Removed in**: 6.8.1

split.history

- **Module**: cmas-dwh-server
- **Description**: Changes the SQL that fetches the history for the tickets during DWH transfer so that the history is not fetched for all tickets at once but only for one ticket per SQL.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: false
- **Since**: 6.8.0

unit.transfer.order

- **Module**: cmas-dwh-server
- **Description**: Defines in which order customer field groups should be transferred to the DWH.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: company;customer
- **Since**: 6.6.19
- **Removed in**: 6.8.1

G.2.3.2 Indexer and Search Configuration

Indexer

big.task.minimum.size

- **Module**: cmas-core-index-common
- **Description**: Indicates the minimum size of index task (in parts, each part has 100 entities) to qualify this task as a big one. Big tasks have a lower priority than normal tasks.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 15 (default value)
- **Since**: 6.8.3

database.notification.enabled

- **Module**: cmas-core-index-common
- **Description**: Indicates whether the notification channel “database” should be used for index updates instead of JMS. The default value is “true.”
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true
- **Since**: 6.8.4.7
- **Removed in**: 6.12.0.0

database.notification.redelivery.delay.seconds

- **Module**: cmas-core-index-common
- **Description**: If the notification channel “database” is used for index updates, this property indicates the delay for notification redelivery when an exception occurs.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 60
- **Since**: 6.8.4.7

database.notification.redelivery.max.attempts

- **Module**: cmas-core-index-common
- **Description**: In case of index update database notification channel, indicates maximum redelivery attempts when an exception occurs.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 60
- **Since**: 6.8.4.7

disable.admin.task.auto.commit

- **Module**: cmas-core-index-common
- **Description**: All tasks created for index update will be automatically executed right after creation.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false
- **Since**: 6.6.1

index.attachment

- **Module**: cmas-core-index-common
- **Description**: Specifies whether the content of attachments is indexed.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true
- **Since**: 6.4.3

index.history

- **Module**: cmas-core-index-common
- **Description**: Specifies whether unit and ticket history are indexed.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: false

- **Since**: 6.1.0
- **Removed in**: 6.11.0

index.status

- **Module**: cmas-core-index-common
- **Description**: Status of the Indexer, possible values RED, YELLOW, GREEN, will be displayed in the Admin Tool.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: GREEN
- **Since**: 6.6.1

index.task.worker.threads

- **Module**: cmas-core-index-common
- **Description**: How many threads will be used to execute index tasks (synchronization, administrative and repair tasks). We recommend to use a value not larger than 2.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 1 (default value)
- **Since**: 6.6.14, 6.7.3. Since 6.8.0 and exclusively in 6.6.21 also normal (live) index updates are affected by this property.

index.version.current

- **Module**: cmas-core-index-common
- **Description**: Holds information about the current (possibly old) index version.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 1 (default value)
- **Since**: 6.7.0

index.version.newest

- **Module**: cmas-core-index-common
- **Description**: Holds information about which index version is considered newest.
- **Type**: integer
- **Restart required**: no

- **System**: yes
- **Optional**: no
- **Example value**: 1 (default value)
- **Since**: 6.7.0

indexed.assets.per.thread.in.memory

- **Module**: cmas-core-index-common
- **Description**: Determines how many assets should be loaded into memory at once, per thread, during indexing.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 200 (default value)
- **Since**: 6.8.0

indexed.engineers.per.thread.in.memory

- **Module**: cmas-core-index-common
- **Description**: Determines how many engineers should be loaded into memory at once, per thread, during indexing.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 300 (default value)
- **Since**: 6.6.14, 6.7.3

indexed.resources.per.thread.in.memory

- **Module**: cmas-core-index-common
- **Description**: Determines how many resources should be loaded into memory at once, per thread, during indexing.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 200 (default value)
- **Since**: 6.10.0.0

indexed.tickets.per.thread.in.memory

- **Module**: cmas-core-index-common
- **Description**: Determines how many tickets should be loaded into memory at once, per thread, during indexing.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 100 (default value)
- **Since**: 6.6.14, 6.7.3

indexed.units.per.thread.in.memory

- **Module**: cmas-core-index-common
- **Description**: Determines how many units should be loaded into memory at once, per thread, during indexing.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 200 (default value)
- **Since**: 6.6.14, 6.7.3

synchronize.master.address

- **Module**: cmas-core-index-common
- **Description**: Value of `-Dcmas.http.host.port` specifying how to connect to the indexing master server. The default value is null. Since 6.6.17 this value is configurable in set-up to designate the initial indexing master server. Since 6.12.0.0 it is possible to switch the node acting as master node by entering the address of the new master node. The index updates are archived for the time configured in the property `cmas-core-index-common`, `synchronize.archive.timeout.minutes`, so the new master can recover missing data in case it had not been fully synchronized with the previous master indexing server at the moment of switch.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 127.0.0.1:80
- **Since**: 6.6.0

synchronize.master.security.token

- **Module**: cmas-core-index-common
- **Description**: The password for accessing the index snapshot via URL, e.g., for index synchronization or for backups.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: token
- **Since**: 6.6.0

synchronize.master.security.user

- **Module**: cmas-core-index-common
- **Description**: The user name for accessing the index snapshot via URL, e.g., for index synchronization or for backups.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: user
- **Since**: 6.6.0

synchronize.master.timeout.minutes

- **Module**: cmas-core-index-common
- **Description**: Determines how long the master server may continually fail until a new master gets elected. Since 6.6.17, this value is configurable in setup, where 0 means that master server will never change (failover is disabled).
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 5 (default value)
- **Since**: 6.6.0

synchronize.megabits.per.second

- **Module**: cmas-core-index-common
- **Description**: Determines how much bandwidth the master server may consume when transferring index changes to all slave servers. Do not use all available bandwidth to transfer index changes between hosts, as doing so will most probably partition the cluster due to some subsystems being unable to communicate.
- **Type**: integer

- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 85 (default value)
- **Since**: 6.6.0

synchronize.sleep.millis

- **Module**: cmas-core-index-common
- **Description**: Determines how often each slave server polls the master server for index changes.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 1000 (default value)
- **Since**: 6.6.0

Search Results

globalSearchResultSizeLimit

- **Module**: cmweb-server-adapter
- **Description**: Maximum number of items in Quick Search result.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 10
- **Since**: 6.0

searchPageSize

- **Module**: cmweb-server-adapter
- **Description**: Default page size for search results.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 20
- **Since**: 6.0

searchPageSizeOptions

- **Module**: cmweb-server-adapter
- **Description**: Options for the page size for search results.

- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 10|20|30|40|50|75|100
- **Since**: 6.0

unitIndexSearchResultSizeLimit

- **Module**: cmweb-server-adapter
- **Description**: Maximum number of units in unit search result (e.g. when searching for contact).
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 5
- **Since**: 6.0

G.2.3.3 LDAP Configuration

LDAP Configuration (if LDAP is Used as Authentication Mode in the CM Web Client)

LDAP parameters apply only if the authentication mode for the CM Web Client has been set to “LDAP”:

authentication.method

- **Module**: cmas-core-security
- **Description**: User authentication method (internal CM database or LDAP authentication). Allowed values are LDAP or DATABASE.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: DATABASE
- **Since**: 6.0

ldap.authentication

- **Module**: cmas-core-security
- **Description**: Authentication method used when using LDAP authentication. Possible values are “anonymous” and “simple”.
- **Type**: string
- **Restart required**: yes
- **System**: yes

- **Optional**: no
- **Example value**: simple (default value)
- **Since**: 6.0

ldap.basedn

- **Module**: cmas-core-security
- **Description**: Base DN used for looking up LDAP user accounts when using LDAP authentication.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: ou=accounts,dc=consol,dc=de
- **Since**: 6.0

ldap.initialcontextfactory

- **Module**: cmas-core-security
- **Description**: Class name for the initial context factory of the LDAP implementation when using LDAP authentication. If it is not set, `com.sun.jndi.ldap.LdapCtxFactory` is used.
- **Type**: string
- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: com.sun.jndi.ldap.LdapCtxFactory
- **Since**: 6.0

ldap.password

- **Module**: cmas-core-security
- **Description**: Password for connecting to LDAP to look up users when using LDAP authentication. Only needed if look-up cannot be performed anonymously.
- **Type**: password
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.1.2

ldap.providerurl

- **Module**: cmas-core-security
- **Description**: LDAP provider when using LDAP authentication.
- **Type**: string
- **Restart required**: no

- **System**: yes
- **Optional**: no
- **Example value**: ldap://myserver.consol.de:389
- **Since**: 6.0

ldap.searchattr

- **Module**: cmas-core-security
- **Description**: Search attribute for looking up LDAP entry associated with a CM login.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: uid
- **Since**: 6.0

ldap.userdn

- **Module**: cmas-core-security
- **Description**: LDAP user for connecting to LDAP to look up users when using LDAP authentication. Only needed if look-up cannot be performed anonymously.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.1.2

LDAP Configuration (if LDAP is Used as Authentication Mode in CM/Track)

LDAP parameters apply only if the authentication mode for CM/Track has been set to “LDAP”:

contact.authentication.method

- **Module**: cmas-core-security
- **Description**: Indicates contact authentication method, where possible values are DATABASE or LDAP or LDAP,DATABASE or DATABASE,LDAP.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: LDAP
- **Since**: 6.9.3.0

ldap.contact.name.basedn

- **Module**: cmas-core-security
- **Description**: Base path to search for contact DN by LDAP ID.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: ou=accounts,dc=consol,dc=de
- **Since**: 6.9.3.0

ldap.contact.name.password

- **Module**: cmas-core-security
- **Description**: Password to look up contact DN by LDAP ID. If not set, the anonymous account is used.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Since**: 6.9.3.0

ldap.contact.name.providerurl

- **Module**: cmas-core-security
- **Description**: Address of the LDAP server (ldap[s]://host:port).
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: ldap://ldap.consol.de:389
- **Since**: 6.9.3.0

ldap.contact.name.searchattr

- **Module**: cmas-core-security
- **Description**: Attribute to search for contact DN by LDAP ID.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: uid
- **Since**: 6.9.3.0

ldap.contact.name.userdn

- **Module**: cmas-core-security
- **Description**: User DN to look up contact DN by LDAP ID. If not set, the anonymous account is used.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Since**: 6.9.3.0

ldap.initialcontextfactory

- **Module**: cmas-core-security
- **Description**: Class name for the initial context factory of the LDAP implementation when using LDAP authentication. If it is not set, `com.sun.jndi.ldap.LdapCtxFactory` is used.
- **Type**: string
- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: com.sun.jndi.ldap.LdapCtxFactory
- **Since**: 6.0

G.2.3.4 Email Configuration

Outgoing Email

mail.smtp.email

- **Module**: cmas-core-server
- **Description**: SMTP email URL for outgoing emails
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: smtp://mail.mydomain.com:25
- **Since**: 6.0

mail.smtp.envelopesender

- **Module**: cmas-core-server
- **Description**: Email address used as sender in SMTP envelope. If not set, the From address of the email is used.
- **Type**: string
- **Restart required**: no

- **System**: yes
- **Optional**: no
- **Example value**: mysender@mydomain.com
- **Since**: 6.5.7

mail.from

- **Module**: cmweb-server-adapter
- **Description**: This email address is used instead of the engineer's email address during email conversations.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: consolcm@example.com
- **Since**: 6.1.2

mail.reply.to

- **Module**: cmweb-server-adapter
- **Description**: When set, the Ticket Email Editor in the Web Client displays a Reply-To field prefilled with this value.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: consolcm@example.com
- **Since**: 6.0.1

> ⚠ Please read the detailed information about ConSol CM Reply-To addresses in section *Scripts of Type Email* in the *ConSol CM Administrator Manual*.

mailTemplateAboveQuotedText

- **Module**: cmweb-server-adapter
- **Description**: Determines the behavior of the email template in the Ticket Email Editor when another email is quoted, i.e. forwarded or replied to. Often used to place the signature correctly.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no

- **Example value**: false
- **Since**: 6.2.4

mail.sender.address

- **Module**: cmas-workflow-jbpm
- **Description**: From address for emails from the workflow engine.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: myuser@consol.de
- **Removed in**: 6.8.0
- **Replaced by**: jobExecutor.mailFrom

mail.smtp.tls.enabled

- **Module**: cmas-core-server
- **Description**: Activates SMTP via SSL/TLS (SMTPS) for sending emails from the Web Client and scripts. The default value is “false”. If it is set to “true”, SMTPS is activated for sending emails.
- **Type**: boolean
- **Restart required**: yes
- **System**: no
- **Optional**: yes
- **Example value**: true
- **Since**: 6.11.1.6

Incoming Email

nimh.enabled

- **Module**: cmas-core-server
- **Description**: Enables the NIMH service. Must be suffixed with the cluster node ID, e.g., `nimh.enabled.NODEID` = “true”.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: false
- **Since**: 6.9.4.0

filesystem.polling.threads.number

- **Module**: cmas-nimh
- **Description**: Number of threads started for database emails' queue polling. The default value is 1
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 10
- **Since**: 6.4.0

filesystem.polling.threads.shutdown.timeout.seconds

- **Module**: cmas-nimh
- **Description**: Waiting time after the shutdown signal. When the timeout is reached, the thread will be terminated.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60 (default value)
- **Since**: 6.4.0

filesystem.polling.threads.watchdog.interval.seconds

- **Module**: cmas-nimh
- **Description**: Determines the interval in seconds to execute the watchdog which checks the activity of the threads of the file system poller.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 30 (default value)
- **Since**: 6.4.0

filesystem.task.enabled

- **Module**: cmas-nimh
- **Description**: With this property the service thread related to a given poller can be disabled.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes

- **Example value**: true (default value)
- **Since**: 6.4.0

filesystem.task.interval.seconds

- **Module**: cmas-nimh
- **Description**: Default interval for polling mailboxes in seconds.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60 (default value)
- **Since**: 6.4.0

filesystem.task.polling.folder

- **Module**: cmas-nimh
- **Description**: Polling folder location which will be scanned for emails in the format of eml files. The default value is the `mail` subdirectory of the ConSol CM data directory
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: c://cmas//mail
- **Since**: 6.4.0

filesystem.task.timeout.seconds

- **Module**: cmas-nimh
- **Description**: After this time (of inactivity) the service thread is considered damaged and automatically restarted. Default: 120 seconds
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.4.0

filesystem.task.transaction.timeout.seconds

- **Module**: cmas-nimh
- **Description**: Default transaction timeout (in seconds) for email fetching transactions. Should be correlated with number of messages fetched at once.
- **Type**: integer
- **Restart required**: no

- **System**: no
- **Optional**: yes
- **Example value**: 60 (default value)
- **Since**: 6.4.0

mailbox.<number>.name

- **Module**: cmas-nimh
- **Description**: Identifier (name) of the mailbox.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: myEmailAccount
- **Since**: 6.11.2.0

mailbox.1.connection.host

- **Module**: cmas-nimh
- **Description**: Host (server) for first configured mailbox. Will overwrite the default parameter `mailbox.default.connection.host`.

mailbox.1.connection.password

- **Module**: cmas-nimh
- **Description**: Password for first configured mailbox. Will overwrite the default parameter `mailbox.default.connection.password`.

mailbox.1.connection.port

- **Module**: cmas-nimh
- **Description**: Port for first configured mailbox. Will overwrite the default parameter `mailbox.default.connection.port`.

mailbox.1.connection.protocol

- **Module**: cmas-nimh
- **Description**: Protocol (e.g., IMAP or POP3) for first configured mailbox. Will overwrite the default parameter `mailbox.default.connection.protocol`.

mailbox.1.connection.username

- **Module**: cmas-nimh
- **Description**: User name for first configured mailbox. Will overwrite the default parameter `mailbox.default.connection.username`.

mailbox.2.connection.host

- **Module**: cmas-nimh
- **Description**: Host (server) for second configured mailbox. Will overwrite the default parameter `mailbox.default.connection.host`.

mailbox.2.connection.password

- **Module**: cmas-nimh
- **Description**: Password for second configured mailbox. Will overwrite the default parameter `mailbox.default.connection.password`.

mailbox.2.connection.port

- **Module**: cmas-nimh
- **Description**: Port for second configured mailbox. Will overwrite the default parameter `mailbox.default.connection.port`.

mailbox.2.connection.protocol

- **Module**: cmas-nimh
- **Description**: Protocol (e.g., IMAP or POP3) for second configured mailbox. Will overwrite the default parameter `mailbox.default.connection.protocol`.

mailbox.2.connection.username

- **Module**: cmas-nimh
- **Description**: User name for second configured mailbox. Will overwrite the default parameter `mailbox.default.connection.username`.

> For all NIMH-related mailbox properties, the following principle is used: a default property is defined (e.g. `mailbox.default.connection.port`). If no mailbox-specific value is configured, this default value will be used.

mailbox.default.connection.host

- **Module**: cmas-nimh
- **Description**: Host (server name) of a given mailbox from which the poller reads emails.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 10.10.1.157
- **Since**: 6.4.0

mailbox.default.connection.password

- **Module**: cmas-nimh
- **Description**: Password for given mailbox from which the poller reads emails.
- **Type**: password
- **Restart required**: no
- **System**: no
- **Optional**: yes

- **Example value**: consol
- **Since**: 6.4.0

mailbox.default.connection.port

- **Module**: cmas-nimh
- **Description**: Port for a given mailbox from which the poller reads emails.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 143
- **Since**: 6.4.0

mailbox.default.connection.protocol

- **Module**: cmas-nimh
- **Description**: Poller's protocol e.g., IMAP or POP3. No default value
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: imap
- **Since**: 6.4.0

mailbox.default.connection.username

- **Module**: cmas-nimh
- **Description**: User name for a given mailbox from which the poller reads emails.
- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: username
- **Since**: 6.4.0

mailbox.default.session.mail.debug

- **Module**: cmas-nimh
- **Description**: JavaMail property, see JavaMail API documentation for further information. Allows for more detailed JavaMail session debugging.
- **Type**: boolean
- **Restart required**: no
- **System**: no

- **Optional**: yes
- **Example value**: true
- **Since**: 6.4.0

mailbox.default.session.mail.mime.address.strict

- **Module**: cmas-nimh
- **Description**: JavaMail property, see JavaMail API documentation for further information. Counterpart of the old `mule mail.mime.strict`, allows to set not so strict email header parsing.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true
- **Since**: 6.4.0

mailbox.default.session.mail.<PROTOCOL>.connectiontimeout

- **Module**: cmas-nimh
- **Description**: JavaMail property, see JavaMail API documentation for further information. Determines the connection timeout in milliseconds.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 3000
- **Since**: 6.4.0 (IMAP and POP3) / 6.11.2.0 (IMAPS and POP3S)

mailbox.default.session.mail.<PROTOCOL>.fetchsize

- **Module**: cmas-nimh
- **Description**: JavaMail property, see JavaMail API documentation for further information. Determines the size of partial fetch in bytes for the indicated protocol.
  For IMAP systems: in CM versions 6.10.7.0 and up, the value of `mailbox.default.session.mail.imap.fetchsize` is set to 1048576 (equals 1 MB) during the initial setup of a ConSol CM system. During an update of an existing ConSol CM system, the value of the property is left unchanged, if the property is already present. In case the property is not yet present, it is added with the default value.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: 1048576
- **Since**: 6.9.4.0 (IMAP and POP3) / 6.11.2.0 (IMAPS and POP3S)

mailbox.default.session.mail.<PROTOCOL>.partialfetch

- **Module**: cmas-nimh
- **Description**: JavaMail property, see JavaMail API documentation for further information. Determines whether the protocol's partial fetch capability should be used.
  For IMAP systems: in CM versions 6.10.7.0 and up, the value of `mailbox.default.session.mail.imap.partialfetch` is set to "false" during the initial setup of a ConSol CM system. During an update of an existing ConSol CM system, the value of the property is left unchanged, if the property is already present. In case the property is not yet present, it is added with the default value.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true
- **Since**: 6.9.4.0 (IMAP and POP3) / 6.11.2.0 (IMAPS and POP3S)

mailbox.default.session.mail.<PROTOCOL>.timeout

- **Module**: cmas-nimh
- **Description**: JavaMail property, see JavaMail API documentation for further information. Determines the I/O timeout in milliseconds.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 120 for IMAP(S) and 3000 for POP3(S)
- **Since**: 6.4.0 (IMAP and POP3), 6.11.2.0 (IMAPS and POP3S)

mailbox.default.task.delete.read.messages

- **Module**: cmas-nimh
- **Description**: This defines whether messages should be removed from the mailbox after processing. For IMAP protocol messages are marked as SEEN by default. For POP3 protocol, when flag is set to "true" the message is removed, otherwise remains on server and will result in infinite reads.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: false (default value)
- **Since**: 6.4.0

mailbox.default.task.enabled

- **Module**: cmas-nimh
- **Description**: With this property, the service thread related to the given poller can be disabled.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true (default value)
- **Since**: 6.4.0

mailbox.default.task.interval.seconds

- **Module**: cmas-nimh
- **Description**: Default interval for polling mailboxes in seconds.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60 (default value)
- **Since**: 6.4.0

mailbox.default.task.max.message.size

- **Module**: cmas-nimh
- **Description**: Maximum size of email messages (i.e., email plus attachment). Emails exceeding the size limit will not be automatically processed by NIMH but will be stored in the database (table `cmas_nimh_archived_mail`) and will therefore appear in the email backups in the Admin Tool (see section *Email Backups* in the *ConSol CM Administrator Manual*). From there they can be resent, downloaded to the file system, or deleted. For those operations the message size is not relevant.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 10485760 (default value, 10 MB)
- **Since**: 6.4.0

mailbox.default.task.max.messages.per.run

- **Module**: cmas-nimh
- **Description**: Number of messages fetched at once from mailbox. Must be correlated with transaction timeout. The default value is 20
- **Type**: integer
- **Restart required**: no

- **System**: no
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.4.0

mailbox.default.task.timeout.seconds

- **Module**: cmas-nimh
- **Description**: After this time (of inactivity) the service thread is considered damaged and automatically restarted. The default value is 120 seconds.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60
- **Since**: 6.4.0

mailbox.default.task.transaction.timeout.seconds

- **Module**: cmas-nimh
- **Description**: Default transaction timeout (in seconds) for email fetching transactions. Should be correlated with number of messages fetched at once.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60 (default value)
- **Since**: 6.4.0

mailbox.polling.threads.mail.log.enabled

- **Module**: cmas-nimh
- **Description**: Enables email logging which is especially crucial in cluster environment (used as semaphore there)
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true (default value)
- **Since**: 6.9.4.1

mailbox.polling.threads.number

- **Module**: cmas-nimh
- **Description**: Number of threads for accessing mailboxes.

- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 1 (default value)
- **Since**: 6.4.0

queue.polling.threads.number

- **Module**: cmas-nimh
- **Description**: Number of threads started for polling the email queue in the database.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 1 (default value)
- **Since**: 6.4.0

queue.polling.threads.shutdown.timeout.seconds

- **Module**: cmas-nimh
- **Description**: Waiting time after the shutdown signal. When the timeout is reached, the thread will be terminated.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60 (default value)
- **Since**: 6.4.0

queue.polling.threads.watchdog.interval.seconds

- **Module**: cmas-nimh
- **Description**: Determines the interval in seconds to execute the watchdog which checks the activity of the threads of the email queue poller.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 30 (default value)
- **Since**: 6.4.0

queue.task.error.pause.seconds

- **Module**: cmas-nimh
- **Description**: Maximum number of seconds, the queue poller waits after infrastructure (e.g. database) error.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 180 (default value)
- **Since**: 6.4.0

queue.task.interval.seconds

- **Module**: cmas-nimh
- **Description**: Main emails' queue polling thread interval.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 15 (default value)
- **Since**: 6.4.0

queue.task.max.retries

- **Module**: cmas-nimh
- **Description**: Maximum number of email processing retries after an exception. When reached, the email is moved to the email archive. This email can be rescheduled again using NIMH API (or the Admin Tool).
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 10
- **Since**: 6.4.0

queue.task.timeout.seconds

- **Module**: cmas-nimh
- **Description**: After this time of inactivity (in seconds), the service thread is considered damaged and automatically restarted.
- **Type**: integer
- **Restart required**: no
- **System**: no

- **Optional**: yes
- **Example value**: 600 (default value)
- **Since**: 6.4.0

queue.task.transaction.timeout.seconds

- **Module**: cmas-nimh
- **Description**: Transaction timeout for email processing in the pipe.
- **Type**: integer
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: 60 (default value)
- **Since**: 6.4.0

mail.attachments.validation.info.sender

- **Module**: cmas-nimh-extension
- **Description**: Sets the From header for error notification emails which are sent when the validation of the attachment type fails.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: admin@mail.com
- **Since**: 6.7.5

mail.attachments.validation.info.subject

- **Module**: cmas-nimh-extension
- **Description**: Sets the subject for error notification emails which are sent when the validation of the attachment type fails.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: Email was not processed because its attachments were rejected!
- **Since**: 6.7.5

mail.db.archive

- **Module**: cmas-nimh-extension
- **Description**: If property is set to “true”, incoming emails are archived in the database.
- **Type**: boolean

- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: false (default value)
- **Since**: 6.8.5.5

mail.error.from.address

- **Module**: cmas-nimh-extension
- **Description**: From address for error emails from NIMH
- **Type**: email
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: myuser@consol.de
- **Since**: 6.4.0

mail.error.to.address

- **Module**: cmas-nimh-extension
- **Description**: To address for error emails from NIMH. As a default the email address of the administrator which you have entered during system setup is used.
- **Type**: email
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: myuser@consol.de
- **Since**: 6.4.0

mail.on.error

- **Module**: cmas-nimh-extension
- **Description**: If set to “true” an error email is sent to the above configured address in case the email message could not be processed.
- **Type**: boolean
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: true (default)
- **Since**: 6.4.0

CM/ is a registered trademark of ConSol Consulting & Solutions Software GmbH

mail.ticketname.pattern

- **Module**: cmas-nimh-extension
- **Description**: Regular expression pattern used to identify the ticket name in the subject of incoming mails.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: .*?Ticket\s+\((\S+)\).*
- **Since**: 6.4.0

Attachments for Incoming Emails

attachment.allowed.types

- **Module**: cmas-core-server
- **Description**: Comma-separated list of allowed file name extensions (if no value is defined, all file extensions are allowed).
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: txt,zip,doc
- **Since**: 6.5.0

attachment.max.size

- **Module**: cmas-core-server
- **Description**: Maximum attachment size, in MB. This is a validation property of the CM API. It controls the size of attachments at tickets, units, and resources. It also controls the size of incoming (not outgoing!) email attachments. The value of this property needs to be aligned with the respective setting in the application server configuration.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 100 (default value)
- **Since**: 6.4.0

Email Encryption (Outgoing and Incoming)

These settings only apply if email encryption is active (true).

mail.encryption

- **Module**: cmas-core-server
- **Description**: If the property is set to “true”, the encrypt checkbox in the Ticket Email Editor is checked by default. The default value of the property is “false”.
- **Type**: boolean
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: true
- **Since**: 6.8.4.0

In case certificates are stored in an LDAP directory, the following settings have to be made:

ldap.certificate.basedn

- **Module**: cmas-core-server
- **Description**: Base DN for certificates location in the LDAP tree. If not provided, `cmas-core-security, ldap.basedn` is used.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: ou=accounts,dc=consol,dc=de
- **Since**: 6.8.4

ldap.certificate.content.attribute

- **Module**: cmas-core-server
- **Description**: LDAP attribute name used where certificate data is stored in the LDAP tree.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: usercertificate (default value)
- **Since**: 6.8.4

ldap.certificate.password

- **Module**: cmas-core-server
- **Description**: LDAP Certificates manager password. If not set, `cmas-core-security, ldap.password` is used.
- **Type**: string
- **Restart required**: no
- **System**: yes

- **Optional**: yes
- **Since**: 6.8.4

ldap.certificate.providerurl

- **Module**: cmas-core-server
- **Description**: LDAP Certificates provider URL. If not set, `cmas-core-security, ldap.providerurl` is used.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: ldap://ldap.consol.de:389
- **Since**: 6.8.4

ldap.certificate.searchattr

- **Module**: cmas-core-server
- **Description**: LDAP attribute name used to search for certificate in the LDAP tree.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Example value**: mail (default value)
- **Since**: 6.8.4

ldap.certificate.userdn

- **Module**: cmas-core-server
- **Description**: LDAP Certificates manager DN. If not set, `cmas-core-security, ldap.userdn` is used.
- **Type**: string
- **Restart required**: no
- **System**: yes
- **Optional**: yes
- **Since**: 6.8.4

G.2.3.5 Activity Interval Configuration

admin.tool.session.check.interval

- **Module**: cmas-app-admin-tool
- **Description**: Configures the time interval (in seconds) in which the system checks for inactive (ended) Admin Tool sessions.
- **Type**: integer

- **Restart required**: yes
- **System**: yes
- **Optional**: no
- **Example value**: 30
- **Since**: 6.7.5

server.session.timeout

- **Module**: cmas-core-server
- **Description**: Server session timeout (in seconds) for connected clients (database table `cmas_user_session`). Each client can overwrite this timeout with custom value using its ID (ADMIN_TOOL, WEB_CLIENT, WORKFLOW_EDITOR, TRACK (before 6.8, please use PORTER), ETL, REST) appended to property name, e.g., `server.session.timeout.ADMIN_TOOL`.
  Please see also the Page Customization attributes *updateTimeServerSessionActivityEnabled* and *updateTimeServerSessionActivity*, both of type *cmApplicationCustomization*.
- **Type**: integer
- **Restart required**: no
- **System**: yes
- **Optional**: no
- **Example value**: 1800
- **Since**: 6.6.1, 6.7.1

Detailed explanation for the Admin Tool:

- server.session.timeout.ADMIN_TOOL
  Defines the time interval how long the server considers a session valid while there is no activity from the Admin Tool holding the session. The Admin Tool is not aware of this value, it only suffers having an invalid session, if the last activity has been longer in the past.
- admin.tool.session.check.interval
  Defines the time between two checks done by the Admin Tool, if the server still considers its session valid.

For example, if admin.tool.session.check.interval = 60, the Admin Tool queries the server every minute if its session is still active/valid. In case server.session.timeout.ADMIN_TOOL = 600 the Admin Tool will get the response that the session is now invalid after ten minutes of inactivity.

G.2.3.6 Administrator Email Addresses

ConSol CM can use different administrator email addresses, depending on the subsystem. Please see *Administrator and Notification Email Addresses* in the *ConSol CM Administrator Manual* for detailed explanations concerning admin email addresses. If no specific admin email addresses are configured, the global admin email address (that you have defined during system set-up) is used.

G.2.3.7 HTTP Header Configuration

It is possible to configure the HTTP response header returned with the web page in the Web Client. This allows, for example, to define security-related response headers according to the requirements dictated by policy or environment. These properties are managed in the module `cmweb-server-http-headers`. The name of the property is the field of the response header and the value of the property is the field value sent in the header.

> Both name and value must match the exact spelling of the HTTP specification for the desired response header field. The correct interpretation and handling of the header lies in the responsibility of the client browser.

Example Property: X-Frame-Options

- **Module**: cmweb-server-http-headers
- **Description**: Example property to illustrate the configuration of HTTP headers. In this case the delivered HTTP header contains the field *X-Frame-Options* with the value “SAMEORIGIN”.

  Each property in the module `cmweb-server-http-headers` represents one header field. The property name/key identifies the response header field and the value of the property is the field value sent in this header.

  > Please be aware that additional HTTP response headers must be correctly defined with the exact spelling as officially specified! Please note also that the correct interpretation and application of these headers is fully in the realm and responsibility of the client browser which requested the page!

- **Type**: string
- **Restart required**: no
- **System**: no
- **Optional**: yes
- **Example value**: SAMEORIGIN
- **Since**: 6.10.8

G.3 Default Java Imports

In order to have all required classes and methods available in ConSol CM scripts, you might have to import Java/Groovy classes and packages. This is relevant for the Admin Tool and for the Process Designer. However, starting with ConSol CM version 6.11.1, you will only have to import classes and packages which are not used in most standard cases. All packages which contain classes which are used rather frequently are imported implicitly by the Admin Tool and by the Process Designer. In this way, the code validation during writing the scripts as well as the execution of the code during runtime is based on this implicit imports which makes coding CM scripts rather comfortable.

The following classes are implicitly imported in the Admin Tool:

- com.consol.cmas.common.model.*
- com.consol.cmas.common.model.calendar.*
- com.consol.cmas.common.model.configuration.*
- com.consol.cmas.common.model.content.*
- com.consol.cmas.common.model.content.unit.*
- com.consol.cmas.common.model.customer.*
- com.consol.cmas.common.model.customfield.*
- com.consol.cmas.common.model.customfield.cfel.*
- com.consol.cmas.common.model.customfield.enums.*
- com.consol.cmas.common.model.customfield.meta.*
- com.consol.cmas.common.model.event.*
- com.consol.cmas.common.model.event.configuration.*
- com.consol.cmas.common.model.event.content.*
- com.consol.cmas.common.model.event.content.support.*
- com.consol.cmas.common.model.event.customfield.*
- com.consol.cmas.common.model.event.delete.*
- com.consol.cmas.common.model.event.engineer.*
- com.consol.cmas.common.model.event.engineer.support.*
- com.consol.cmas.common.model.event.localization.*
- com.consol.cmas.common.model.event.localization.support.*
- com.consol.cmas.common.model.event.ticket.*
- com.consol.cmas.common.model.event.ticket.support.*
- com.consol.cmas.common.model.event.unit.*
- com.consol.cmas.common.model.event.unit.support.*
- com.consol.cmas.common.model.event.util.*
- com.consol.cmas.common.model.event.workflow.*
- com.consol.cmas.common.model.history.*
- com.consol.cmas.common.model.history.custom.*
- com.consol.cmas.common.model.history.ticket.*

CM/ is a registered trademark of ConSol Consulting & Solutions Software GmbH

- com.consol.cmas.common.model.inventory.*
- com.consol.cmas.common.model.inventory.history.*
- com.consol.cmas.common.model.inventory.meta.*
- com.consol.cmas.common.model.localization.*
- com.consol.cmas.common.model.mail.*
- com.consol.cmas.common.model.resource.*
- com.consol.cmas.common.model.resource.history.*
- com.consol.cmas.common.model.resource.meta.*
- com.consol.cmas.common.model.scripting.OperationMessage
- com.consol.cmas.common.model.scripting.OperationResponse
- com.consol.cmas.common.model.scripting.OperationType
- com.consol.cmas.common.model.search.*
- com.consol.cmas.common.model.task.*
- com.consol.cmas.common.model.ticket.*
- com.consol.cmas.common.model.ticket.user.*
- com.consol.cmas.common.model.ticket.user.function.*
- com.consol.cmas.common.model.ticket.view.*
- com.consol.cmas.common.security.authentication.UserType
- com.consol.cmas.workflow.common.model.*
- java.io.*
- java.util.*
- javax.activation.DataSource
- com.consol.cmas.common.model.form.*

The following classes are implicitly imported in the Process Designer:

- com.consol.cmas.common.model.*
- com.consol.cmas.common.model.calendar.*
- com.consol.cmas.common.model.configuration.*
- com.consol.cmas.common.model.content.*
- com.consol.cmas.common.model.content.unit.*
- com.consol.cmas.common.model.customer.*
- com.consol.cmas.common.model.customfield.*
- com.consol.cmas.common.model.customfield.cfel.*
- com.consol.cmas.common.model.customfield.enums.*
- com.consol.cmas.common.model.customfield.meta.*
- com.consol.cmas.common.model.event.*
- com.consol.cmas.common.model.event.configuration.*
- com.consol.cmas.common.model.event.content.*

- com.consol.cmas.common.model.event.content.support.*
- com.consol.cmas.common.model.event.customfield.*
- com.consol.cmas.common.model.event.delete.*
- com.consol.cmas.common.model.event.engineer.*
- com.consol.cmas.common.model.event.engineer.support.*
- com.consol.cmas.common.model.event.localization.*
- com.consol.cmas.common.model.event.localization.support.*
- com.consol.cmas.common.model.event.ticket.*
- com.consol.cmas.common.model.event.ticket.support.*
- com.consol.cmas.common.model.event.unit.*
- com.consol.cmas.common.model.event.unit.support.*
- com.consol.cmas.common.model.event.util.*
- com.consol.cmas.common.model.event.workflow.*
- com.consol.cmas.common.model.history.*
- com.consol.cmas.common.model.history.custom.*
- com.consol.cmas.common.model.history.ticket.*
- com.consol.cmas.common.model.inventory.*
- com.consol.cmas.common.model.inventory.history.*
- com.consol.cmas.common.model.inventory.meta.*
- com.consol.cmas.common.model.localization.*
- com.consol.cmas.common.model.mail.*
- com.consol.cmas.common.model.resource.*
- com.consol.cmas.common.model.resource.history.*
- com.consol.cmas.common.model.resource.meta.*
- com.consol.cmas.common.model.scripting.OperationMessage
- com.consol.cmas.common.model.scripting.OperationResponse
- com.consol.cmas.common.model.scripting.OperationType
- com.consol.cmas.common.model.search.*
- com.consol.cmas.common.model.task.*
- com.consol.cmas.common.model.ticket.*
- com.consol.cmas.common.model.ticket.user.*
- com.consol.cmas.common.model.ticket.user.function.*
- com.consol.cmas.common.model.ticket.view.*
- com.consol.cmas.common.security.authentication.UserType
- com.consol.cmas.workflow.common.model.*
- java.io.*
- java.util.*

- javax.activation.DataSource
- com.consol.cmas.common.model.permission.*
- com.consol.cmas.common.model.scripting.*
- com.consol.cmas.common.model.util.*

CM/ is a registered trademark of ConSol Consulting & Solutions Software GmbH

G.4 List of Code Examples

In this section, you can find a list of the code examples from this manual.

Code example 1: Access to content of data fields of the three main CM objects 30
Code example 2: Precondition script: activity should only be displayed for VIP customers 73
Code example 3: Scripts for automatic activity where receipt note is sent, variant 1 77
Code example 4: Script for automatic activity where receipt note is sent, variant 2 77
Code example 5: Script for assigning ticket to current engineer 78
Code example 6: NimhIncomingMailRouting.groovy adapted to reopen a ticket if an email is received 92
Code example 7: Precondition script: activity should only be displayed for VIP customers 97
Code example 8: Example for a script on timer start 110
Code example 9: Calculate and set time for TimerTrigger using BusinessCalendar 111
Code example 10: Removing an overlay, here: the email overlay, variant 1 119
Code example 11: Removing an overlay, here: the email overlay, variant 2 119
Code example 12: Removing an overlay, here: the email overlay, variant 3 119
Code example 13: Code of decision node script 131
Code example 14: Code of automatic activity script Re-calculate priority 134
Code example 15: Sending an email to the current engineer of the ticket if the 'Deadline' field in the ticket has been changed (Dates not formatted) 136
Code example 16: Process Designer: Initializing script for Create bid ACF 144
Code example 17: Using the ticket object 171
Code example 18: Using workflowApi to create a child ticket 171
Code example 19: Using workflowApi to deactivate a trigger 171
Code example 20: Using workflowAppi to display a GUI message for the engineer/user 172
Code example 21: Example for a script on timer start 172
Code example 22: Using the ConfigurationService to retrieve the number of the engineer management ticket 172
Code example 23: Using the ConfigurationService to retrieve base URL of the system 173
Code example 24: Assigning the ticket to the current engineer 173
Code example 25: Use of EngineerService 174
Code example 26: Using EnumService to retrieve an enum value by name 174
Code example 27: Using TicketService to find ticket of a view 175
Code example 28: Using the EngineerRoleRelationService to send an email to all engineers of a role 175
Code example 29: Admin Tool script used to display ticket fields 184
Code example 30: Precondition script where boolean value is checked 185
Code example 31: Precondition script where boolean value is checked, short version 185
Code example 32: Retrieving an enum value for a ticket field 185

Code example 33: Code (Workflow or Admin Tool script) for displaying MLA data 187
Code example 34: Retrieving the entire path to the selected MLA value 187
Code example 35: Displaying the content of a list of date objects 189
Code example 36: Retrieve a certain value from a list of simple data types 189
Code example 37: Retrieve data from a list of structs 191
Code example 38: Set a ticket field value for a field of type Date 191
Code example 39: Calculate with the value of a ticket field of type date 191
Code example 40: Setting a ticket field value to null 192
Code example 41: Setting a ticket field value to null via removing the value 192
Code example 42: Setting an enum value 192
Code example 43: Adding a new line in a list of strings 192
Code example 44: Setting a value in a list of strings 192
Code example 45: Adding a new line in a list of structs 193
Code example 46: Fade in a ticket field group 193
Code example 47: Fade out ticket field groups 193
Code example 48: Retrieving a field value for a company 195
Code example 49: Admin Tool script (called from workflow) for displaying customer data 197
Code example 50: Set and add values for a customer field of type integer and set a value for a customer field of type string 198
Code example 51: Creating a new list of structs, version 2 198
Code example 52: Adding a new line in a list of structs for company data 199
Code example 53: Setting a value in a list of structs using index notation 199
Code example 54: Removing a struct (= line) from a list of structs (= table) 199
Code example 55: Convenience methods for access to customer data 199
Code example 56: Simple resource action script which displays information about the resource in the log file 201
Code example 57: Excerpt from a script which adds a line to a list of tickets on a resource page and makes the resource field group which contains the list visible 202
Code example 58: Sending an automatic acknowledgment of receipt to the customer who has opened a ticket, using a Mail object 207
Code example 59: Sending an email to the engineer when a certain escalation level has been reached, using a Mail object 208
Code example 60: Sending an email to a customer integrating the queue-specific mail script, using a Mail object 209
Code example 61: Sending an email to all contacts of the ticket 210
Code example 62: Admin Tool script for sending a notification of receipt and inserting the email as ticket history entry 212
Code example 63: Script for sending an email in an asynchronous mode, please see also alternative code in the following example 215

Code example 64: Alternative code to switch statement: use a multi-dimensional hashmap for more efficient coding ... 216
Code example 65: Workflow or Admin Tool script used to send an email to an engineer, using the representation feature by using the setTargetEngineer() method ... 218
Code example 66: Deactivate a time trigger ... 220
Code example 67: Re-initialize time trigger ... 220
Code example 68: Setting time for a time trigger ... 222
Code example 69: Script for time trigger for escalation 4 hours before deadline (method BusinessCalendarUtil.getEscalationTime for CM versions 6.10.6 or 6.11.0.4 and up) ... 223
Code example 70: Creating a ticket relation of type REFERENCE using workflowAPI ... 227
Code example 71: Creating a ticket relation of type MASTER_SLAVE using workflowAPI ... 228
Code example 72: Finding all slave tickets of the current ticket ... 228
Code example 73: Finding the master ticket of the current ticket ... 228
Code example 74: Creating a child ticket ... 229
Code example 75: Finding the parent ticket of a ticket ... 229
Code example 76: Finding all child tickets of a ticket ... 229
Code example 77: Finding all brother tickets of a (child) ticket ... 230
Code example 78: Example Script, display IDs and names of slave tickets workflow version ... 232
Code example 79: Example Script, display IDs and names of slave tickets Admin Tool script version 232
Code example 80: Calling previous AT script from workflow activity ... 232
Code example 81: Adding a data object relation using a workflow script ... 237
Code example 82: Calculate ticket deadline from SLA. SLA as resource which is linked to the ticket. 241
Code example 83: Workflow script to check if a solution has been defined in the ticket ... 245
Code example 84: Excerpt from the postActivityExecutionScript ... 248
Code example 85: Admin Tool script called from a workflow activity: creating a child ticket and handing over the products list and all ticket attachments ... 252
Code example 86: Alternative (additional) solution: handing over only the important attachments 253
Code example 87: Search for tickets (pseudocode) ... 256
Code example 88: Find tickets with the same module as the current ticket and display result in server.log file ... 257
Code example 89: Search for tickets by unit ... 258
Code example 90: Script of activity "New IT ticket (Accept ticket)", engineer workload is checked ... 260
Code example 91: Search for customers by enum value (general syntax) ... 262
Code example 92: Search for contacts by first name and last name ... 262
Code example 93: Search for customers by enum value (example) ... 263
Code example 94: Print an IT asset (resource) list into the ticket ... 265
Code example 95: Checking if all child tickets are closed and displaying a message for the engineer, non-localized version ... 267

Code example 96: Script of activity "New IT ticket (Accept ticket)", engineer workload is checked 268
Code example 97: Checking if all child tickets are closed and displaying a message for the engineer .269
Code example 98: Workflow script to send a notification 270
Code example 99: Debug entry in ConSol CM standard email script 272
Code example 100: Calling an Admin Tool script from the workflow 288
Code example 101: Calling an Admin Tool script from the workflow with use of parameters 288
Code example 102: Code which triggers TicketUpdateEvent 292
Code example 103: Code which does not trigger TicketUpdateEvent 292

G.5 Trademarks

- The Apache Commons Codec TM library is a trademark of the Apache Software Foundation. See Apache Commons Codec web page.
- Apache OpenOfficeTM – Apache and the Apache feather logos are trademarks of The Apache Software Foundation. OpenOffice.org and the seagull logo are registered trademarks of The Apache Software Foundation. See Apache OpenOffice Trademarks web page.
- Google MapsTM – Google Maps is a trademark of Google Inc. See Google trademark web page for details.
- HAProxy – HAProxy is copyright of Willy Tarreau. See HAProxy website.
- Microsoft® – Microsoft and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. See Microsoft trademark web page.
- Microsoft® Active Directory® – Microsoft and Microsoft Active Directory are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. See Microsoft trademark web page.
- Microsoft® Exchange Server – Microsoft and Microsoft Exchange Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. See Microsoft trademark web page.
- Microsoft® Office – Microsoft and Microsoft Office are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. See Microsoft trademark web page.
- Microsoft® SQL Server® – Microsoft and Microsoft SQL Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. See Microsoft trademark web page.
- Microsoft® 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® 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.
- MongoDB® – Mongo, MongoDB, and the MongoDB leaf logo are registered trademarks of MongoDB, Inc.
- NGiNX – NGiNX is copyright of Igor Sysoev and Nginx, Inc. See NGiNX license page.
- OpenStreetMap - OpenStreetMap® is open data, licensed under the Open Data Commons Open Database License (ODbL) by the OpenStreetMap Foundation (OSMF). See OpenStreetMap Copyright and License page.
- Oracle® – Oracle is a registered trademark of Oracle Corporation and/or its affiliates. See Oracle trademarks web page.
- Oracle® WebLogic – Oracle is a registered trademark of Oracle Corporation and/or its affiliates. See Oracle trademarks web page.
- Pentaho® – Pentaho and the Pentaho logo are registered trademarks of Pentaho Inc. See Pentaho trademark web page.
- Vis.js – Vis.js is copyright of Almende B.V. See Vis.js license page.

- Wicket™ - Apache Wicket and Wicket, Apache, the Apache feather logo, and the Apache Wicket project logo are trademarks of The Apache Software Foundation. See, for example, the hint at the bottom of the Wicket home page

G.6 Glossary

A

ACF

ACF is the abbreviation of Activity Control Form. ACFs can be used in workflow activities to force the engineer to fill out certain fields before proceeding.

ACIM

Activity item - entry in the history section of a ticket (e.g., comment, email, attachment, time booking entry).

AD

Microsoft Active Directory - an LDAP-based directory service for Microsoft Windows domain networks.

additional customer

Additional customers are customers (companies or contacts) who are interested in the ticket. They are optional and usually have a role indicating the reason why they were added.

additional engineer

Additional engineers are engineers who have a specific purpose, which depends on your business process. Usually, they have to carry out certain tasks within the process.

Admin Tool

ConSol CM component, graphical application to configure and manage a ConSol CM system. Uses Java Web Start.

AJP

Apache JServ Protocol, see, for example https://en.wikipedia.org/wiki/Apache_JServ_Protocol

B

BI

Business Intelligence - methods, technologies, and architectures to transform data into useful information for business purposes.

C

CFEL

Custom Field Expression Language - Java classes and methods of the ConSol CM API to access data in ticket fields, customer fields and resource fields.

CIDR

Classless Inter-Domain Routing, notation for IP address ranges

CM/Doc

A standard module of ConSol CM which enables the engineer via ConSol CM Web Client to work with Microsoft Word or OpenOffice documents prefilled with ConSol CM ticket or customer parameters.

CM/Phone

The ConSol CM module which provides CTI for CM.

CM/Resource Pool

CM/Resource Pool is an optional add-on which allows to store different kinds of objects as resources in ConSol CM.

CM/Track

CM/Track is the portal of ConSol CM. Customers can access their tickets through CM/Track.

CMDB

ConSol CM database - the working database of the CM system.

CMRF

ConSol CM Reporting Framework - a JEE application which synchronizes data between the ConSol CM database and the DWH.

company

The company is the upper hierarchical level of a two-level customer model. A company can have several contacts.

contact

The contact is the lower hierarchical level of a two-level customer model. A contact can only belong to one company.

CRM

Customer Relationship Management. Approach to manage a company's customers, e.g., to collect data from different sources and integrate the data to generate information which allows, e.g., to optimize the services for the customers.

CTI

Computer Telephony Integration - a denomination for any technology that facilitates interaction between a telephone and a computer.

customer

The customer represents the external side of a ticket. It designates the person or object that gave the reason for creating a ticket. A customer can either be a company or a contact.

customer action

Part of the Action Framework. An action which is performed for a customer object, i.e., a contact or company object.

customer data model

The customer data model is the definition of the customers. It determines the available data fields and possible relations.

customer field

A field where data for customers (contacts or companies) can be stored. Similar to ticket fields for ticket data. Previously called Data Object Group Field.

customer field group

A group of fields where data for customers (contacts or companies) can be stored. Similar to ticket field group for ticket data. Previously called Data Object Group.

customer group

The customer group determines which customer data model is used for its customers and which actions are available.

customer object

A customer (a contact or a company). Formerly called Data Object. The term Unit is used in the programming context.

D

Dashboard

A type of report which integrates data from different sources providing an overall perspective of a certain topic. Often times graphical representation is used.

DWH

Data Warehouse - A database used for reporting and data analysis. In a standard ConSol CM distribution, a DWH is included and only has to be installed and configured.

E

engineer

Engineers are the users who work on the tickets in the Web Client

ERP system

Enterprise Resource Planning - often used for this type of enterprise management software.

ESB

Enterprise Service Bus - a software architecture used for communication between mutually interacting software applications in a service-oriented architecture (SOA).

ETL

Extract Transform Load - extracts data from one source (a database or other source), transforms it, and loads it into a database, e.g., a data warehouse.

F

FlexCDM

Flexible Customer Data Model - the customer data model introduced in ConSol CM in version 6.9. For each customer group, a specific customer data model can be defined.

G

GUI

Graphical User Interface

H

history

The history contains all changes which were carried out for the ticket, customer, or resource.

HMAC

Hash-based Message Authentication Code, message authentication function using hashes

I

IMAP

Internet Message Access Protocol - Internet standard protocol to access email on a remote email server. Can be used as plain IMAP or as secure IMAP (IMAPs). In the latter case, proper certificates are required.

J

Java EE

Java Enterprise Edition

JMS

Java Message Service - Java EE component used to send messages between JMS clients.

JRE

Java Runtime Environment. Provides a Java Virtual Machine for Clients.

K

Kerberos

A network authentication protocol based on (Kerberos) tickets which requires a special infrastructure.

KPI

Key Performance Indicator - parameter used for performance measurement for companies, projects, etc.

L

LDAP

LDAP is the abbreviation of Lightweight Directory Access Protocol. It is a protocol used to manage login information for several applications.

LDAPS

LDAP over SSL

M

mailbox

Destination to which email messages are delivered. Mailboxes are managed on an email server. ConSol CM can access one or more mailboxes to retrieve emails.

main customer

The main customer is the customer who gave the reason for creating the ticket. The main customer is mandatory for a ticket.

Mule

An open source Java-based Enterprise Service Bus (ESB).

N

NIMH

New Incoming Mail Handler - module for retrieving incoming emails.

P

PCDS

Page Customization Definition Section

Pentaho

PentahoTM is a business intelligence (BI) suite which is available in open source and as enterprise editions.

permission

Permissions determine which tickets an engineer can see in the Web Client and which actions he is allowed to perform. Permissions are always granted via roles, i.e., they are not assigned to a single user but to a group of users sharing a common role. Usually these users belong to the same team and/or have similar functions in the company.

POP

Post Office Protocol - Internet standard protocol to retrieve emails from a remote server via TCP/IP. Can be used as plain POP or as secure POP (POPs).

In the latter case, proper certificates are required.

portal

CM/Track - provides customer access to ConSol CM.

Process Designer

ConSol CM component used to design, develop, and deploy workflows.

Q

queue

The queue contains thematically related tickets which should be handled in the same way and follow the same business process (workflow). Permissions and other parameters are also defined based on queues.

R

RDBMS

Relational Database Management System - e.g. Oracle ®, MS SQL Server ®, MySQL.

relation

Relations are connections between different data objects in ConSol CM. This can be a relation between two objects of the same type, e.g., between tickets, customers, and resources, or a relation between objects of different types, e.g., between a ticket and a resource or a customer and a resource.

resource

Resources are objects managed in CM/Resource Pool.

resource action

Part of the Action Framework. An action performed for a resource object.

resource field

A field where resource data can be stored.

resource field group

A group of fields where data for resources can be stored. Similar to ticket field group for ticket data.

resource type

The resource type is the definition of the resources. It determines the available data fields and possible relations and actions.

REST

Representational State Transfer - conventions for transferring data over HTTP connections.

role

Roles are assigned to engineers. They define the engineers' access permissions and views.

S

script

Program written for a specific run-time environment that can interpret and automate the execution of tasks. In ConSol CM, scripts are stored in the Admin Tool and are stored as scripts for activities in workflows.

search action

Part of the Action Framework. An action performed for the result set of a

search.

SMTP

Simple Message Transfer Protocol - standard protocol for sending emails.

T

TAPI

Telephony Application Programming Interface - a Microsoft Windows API which provides computer/telephony integration and enables PCs running Microsoft Windows to use telephone services.

TEF

Task Execution Framework - a ConSol CM module which can execute tasks asynchronously.

template

Templates contain predefined and pre-formatted text. They can be used for comments, emails, and documents.

ticket

The ticket is the request of the customer which the engineer works on. It is the object which runs through the business process defined by the workflow.

ticket field

A field where ticket data can be stored. Previously called Custom Field

ticket field group

A group of ticket fields where ticket data can be stored. Previously called Custom Field Group.

time booking

Time bookings allow the engineers to register the time they worked on a ticket or project.

U

Unit

Java class which represents a customer object. i.e. a contact is an object of class Unit and a company is also an object of class Unit.

V

view

Views limit the tickets which are shown in the ticket list in the ConSol CM Web Client to those tickets matching specific criteria (scopes from one or more workflows). Views are assigned to roles.

W

Web Client

The Web Client is the primary access to the system for the engineers.

Wicket

Apache Wicket is an open source, component oriented, serverside, Java web application framework. See https://wicket.apache.org/ for details information.

workflow

The workflow is the implementation of the business process managed in ConSol CM. It contains a series of steps which are carried out by the engineers.