



ConSol Software GmbH

ConSol CM Process Designer Manual

Version 6.10.5.3

Contents

Contents	2
A - Introduction	8
A.1 ConSol CM for Business Process Management	9
A.2 List of Manuals	10
A.3 This Manual	11
A.3.1 Before You Read this Book	11
A.3.2 The Book's Structure	11
A.4 Legal Notice	12
A.5 Gender Disclaimer	12
A.6 Copyright	12
A.7 Layout Explanations	13
A.8 Business Processes	14
A.9 Introduction to Workflows in ConSol CM	15
A.10 The ConSol CM Process Designer at a Glance	16
A.10.1 Modeling Workflows	17
A.10.2 Tickets and Activities	18
A.10.3 Drag & Drop Modeling of Workflow Components	19
A.10.4 Scopes and Nesting of Scopes	20
A.10.5 Modeling Escalation Mechanisms (Triggers and Wait States)	20
A.10.6 Modeling Interrupts and Exceptions	21
A.10.7 Scripting Capabilities	22
A.10.8 Versioning of Workflows	22
A.11 Basic Components of ConSol CM Processes	24
A.11.1 General Objects	25
A.11.2 Data Fields	28
A.11.3 Standard Ticket Data Fields	29

B - Work with the Process Designer Application	30
B.1 Steps to Perform for a New Process	31
B.2 Start of the Process Designer	31
B.3 Process Designer GUI	33
B.3.1 Introduction to the Process Designer GUI Elements	34
B.3.2 The Script Editor	49
B.3.3 GUI Tips and Tricks	50
C - Components of ConSol CM Workflows	51
C.1 Introduction	51
C.2 Workflow Components: START Node	52
C.2.1 Properties of a Start Node	52
C.3 Workflow Components: END Nodes	54
C.3.1 The following actions are still possible for a closed ticket	54
C.3.2 The following actions are not possible for a closed ticket	55
C.3.3 Properties of an End Node	56
C.4 Workflow Components: Scopes	58
C.4.1 Introduction to Scopes	59
C.4.2 Defining a New Scope	61
C.4.3 Properties of a Scope	62
C.4.4 Scopes and Views	63
C.4.5 Scope Sort Index and its Side Effects	63
C.5 Workflow Components: Activities	64
C.5.1 Introduction to Activities	65
C.5.2 Properties of an Activity	67
C.5.3 Process Logic of Activities	69
C.5.4 Examples for Activities	70
C.6 Workflow Components: Decision Nodes	76
C.6.1 Introduction to Decision Nodes	77
C.6.2 Properties of a Decision Node	78

C.6.3 Example for a Decision Node	79
C.7 Adornments (Triggers and ACFs)	82
C.7.1 Time Triggers	83
C.7.2 Mail Triggers	94
C.7.3 Business Event Triggers	103
C.7.4 Activity Control Forms (ACFs)	116
C.8 Jump-out and Jump-in Nodes	125
C.8.1 Introduction	126
C.8.2 Jump-out Nodes	128
C.8.3 Jump-in Nodes	130
D - Introduction to Workflow Programming	132
D.1 Programming CM scripts	134
D.1.1 Some Short Examples of Java vs. Groovy-style Coding	134
D.2 CM API Documentation	135
D.3 CM Script Types in Workflows	136
D.4 Script Interactions	137
D.5 Scripts in ConSol CM in General	137
D.6 Process Logic	138
D.6.1 Introduction	139
D.6.2 Activities	139
D.6.3 Interrupts and Exceptions	141
D.6.4 Loops (Errors in Workflows)	143
D.6.5 Process Logic of Time Triggers	143
D.6.6 Process Logic of Business Event Triggers	143
D.7 Important Classes and Objects	144
D.7.1 Introduction	145
D.7.2 Important Objects	145
D.7.3 Convenience Classes and Methods	146

D.8 Working With Data Fields	150
D.8.1 Introduction to Data Fields	151
D.8.2 Data Types for Data Fields	152
D.8.3 Details about String Fields: Use Annotations to Fine-Tune Strings	155
D.8.4 Custom Fields for Ticket Data	157
D.8.5 Data Fields for Customer Data	170
D.8.6 Resource Data	179
D.8.7 Using Data Fields for (Invisible) Variables	180
D.9 Sending E-Mails	182
D.9.1 Introduction to Sending E-Mails	183
D.9.2 Important Methods	183
D.9.3 Examples	183
D.9.4 Writing E-Mails from Scripts when Engineer Representation Rules Apply	192
D.10 Working with Path Information	195
D.10.1 Introduction	196
D.10.2 Retrieve Path Information for a Workflow Element	197
D.10.3 Examples for the Use of Path Information	197
D.11 Working with Calendars and Times	198
D.11.1 Introduction	199
D.11.2 Calculating with Dates and Times without a CM Business Calendar	200
D.11.3 Calculating with Dates and Times Using a CM Business Calendar	200
D.12 Working with Object Relations	202
D.12.1 Working with Ticket Relations	203
D.12.2 Working with Customer Relations (Data Object Relations)	213
D.12.3 Working With Resource Relations	220
D.13 Working With Text Classes	223
D.13.1 Introduction	224
D.13.2 Example: Checking if a Solution Exists Before the Ticket can be Closed	225
D.13.3 Example: Adding a Text as Ticket Comment and Setting a Class of Text	227

D.14 Working With Attachments	230
D.14.1 Introduction	231
D.14.2 Example 1: Attaching all attachments of a ServiceDesk ticket to the child ticket	231
D.15 Searching for Tickets, Customers, and Resources Using the ConSol CM Workflow API	237
D.15.1 Introduction	238
D.15.2 Searching for Tickets	239
D.15.3 Searching for Units (Contacts and Companies)	244
D.15.4 Searching for Resources	246
D.16 Debug Information	249
D.16.1 Introduction	250
D.16.2 Using Statements for Debug Output	251
E - Best Practices	252
E.1 The Basic Organization of a Workflow: Using Scopes	253
E.1.1 Variant A: Use of a Global Scope	254
Variant B: Use of Three or More Main Scopes	256
E.2 The Position of the START Node	258
E.3 Store Some Workflow Scripts in the Admin Tool	259
E.3.1 When to Use Admin Tool Workflow Scripts	261
E.3.2 How to Use Admin Tool Workflow Scripts	262

E.4 Consider the Use of Trigger Combinations Well	263
E.5 Do Not Trigger Ticket Update Events If Not Really Required	266
E.6 How to Use the Disable Auto Update Parameter	267
E.7 Avoid Self-Triggering Business Event Triggers	269
F - Deploying Workflows	270
F.1 Introduction and Workflow Life Cycle	271
F.2 Engineer Permissions Required for Workflow Deployment	272
F.3 Actions During Workflow Deployment	273
G - Appendix	275
G.1 Annotations	276
G.1.1 List of Field Annotations	277
G.1.2 List of Group Annotations	290
G.2 System Properties	294
G.2.1 Alphabetical List of System Properties	295
G.2.2 List of System Properties by Module	377
G.2.3 List of System Properties by Area	453
G.3 Administrator and Notification E-Mail Addresses	501
G.3.1 Introduction	502
G.3.2 Default Configuration	502
G.3.3 Subsystem-Specific Notification E-Mail Addresses	503
G.4 List of Code Examples	507
G.5 Trademarks	511
G.6 Glossary	512

A - Introduction

This chapter discusses the following:

A.1 ConSol CM for Business Process Management	9
A.2 List of Manuals	10
A.3 This Manual	11
A.3.1 Before You Read this Book	11
A.3.2 The Book's Structure	11
A.4 Legal Notice	12
A.5 Gender Disclaimer	12
A.6 Copyright	12
A.7 Layout Explanations	13
A.8 Business Processes	14
A.9 Introduction to Workflows in ConSol CM	15
A.10 The ConSol CM Process Designer at a Glance	16
A.10.1 Modeling Workflows	17
A.10.2 Tickets and Activities	18
A.10.3 Drag & Drop Modeling of Workflow Components	19
A.10.4 Scopes and Nesting of Scopes	20
A.10.5 Modeling Escalation Mechanisms (Triggers and Wait States)	20
A.10.6 Modeling Interrupts and Exceptions	21
A.10.7 Scripting Capabilities	22
A.10.8 Versioning of Workflows	22
A.11 Basic Components of ConSol CM Processes	24
A.11.1 General Objects	25
A.11.2 Data Fields	28
A.11.3 Standard Ticket Data Fields	29

A.1 ConSol CM for Business Process Management

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

A.2 List of Manuals

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

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

A.3 This Manual

A.3.1 Before You Read this Book ...

When you read this manual, your company is presumably using ConSol CM as a business process management tool and it is your job to administer the system and to implement your company's processes in the application. The book 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.3.2 The Book's Structure

1. First, some basic components of business processes in general are explained (see this section).
2. Then, an overview of the implementation of the processes in ConSol CM is given (see section [Basic Components of ConSol CM Processes](#)).
3. Following this, the Process Designer is explained in detail (see sections [Work with the Process Designer Application](#) and [Components of ConSol CM Workflows](#)).
4. The sections [Process Logic](#), [Introduction to Workflow Programming](#), and [Best Practices](#) provide expert knowledge about workflow development.
5. Since every workflow has to be deployed to become active, the section [Deploying Workflows](#) treats this topic.
6. In the appendices, you find lists of all important terms that are used in the book ([Glossary](#)), of all annotations ([Annotations](#) (important for the GUI design), and properties ([System Properties](#)) (important for the CM system management). Please see also the trademarks page ([Trademarks](#)).

A.4 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 (e.g., LDAP, Kerberos), 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.5 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.6 Copyright


© 2017 ConSol Consulting & Solutions Software GmbH - All rights are reserved.


A.7 Layout Explanations

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

 This is an additional information.

 This is an important note. Be careful here!

 This is a warning!

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

A.8 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 [Basic Components of ConSol CM 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.

A.9 Introduction to Workflows 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.

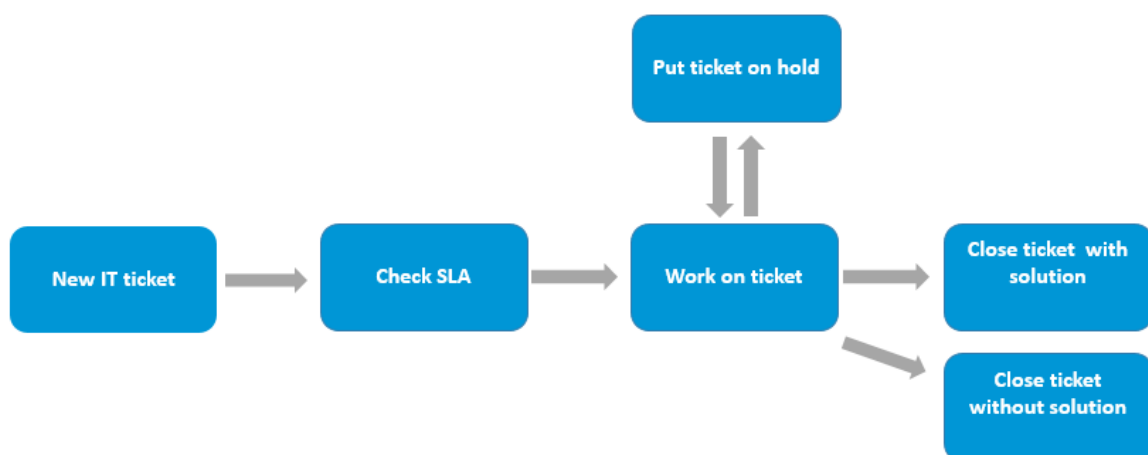


Figure 1: Graphical representation of a simple business process (IT helpdesk)


A.10 The ConSol CM Process Designer at a Glance

This chapter discusses the following:

A.10.1 Modeling Workflows	17
A.10.2 Tickets and Activities	18
A.10.3 Drag & Drop Modeling of Workflow Components	19
A.10.4 Scopes and Nesting of Scopes	20
A.10.5 Modeling Escalation Mechanisms (Triggers and Wait States)	20
A.10.6 Modeling Interrupts and Exceptions	21
A.10.7 Scripting Capabilities	22
A.10.8 Versioning of Workflows	22

A.10.1 Modeling 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.

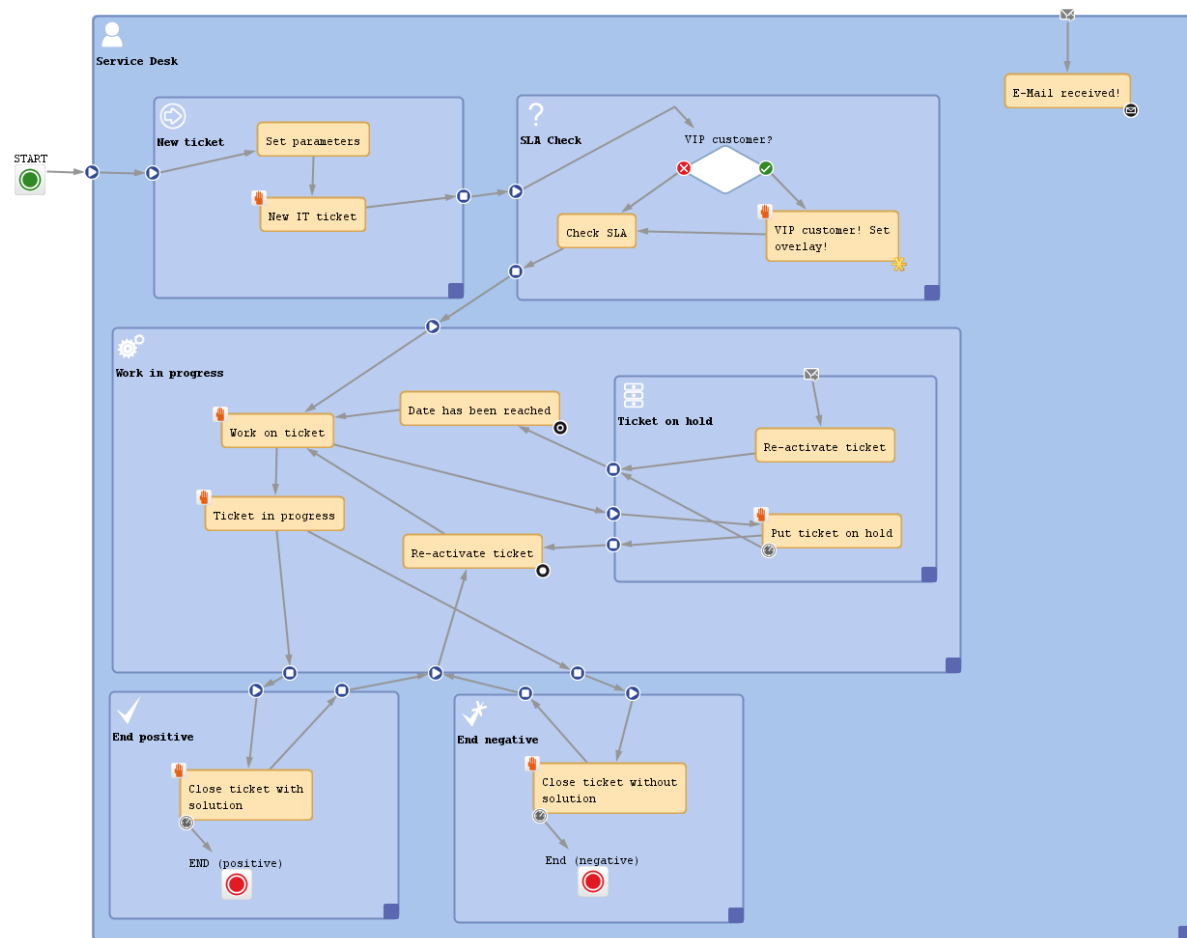


Figure 2: ConSol CM Process Designer - Workflow modeling, example process

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.

A.10.2 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.

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.

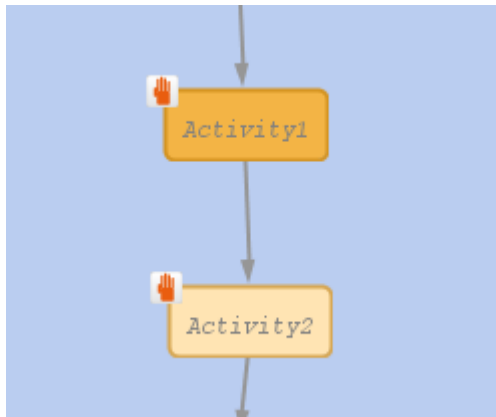


Figure 3: ConSol CM Process Designer - Two sequential manual activities

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.

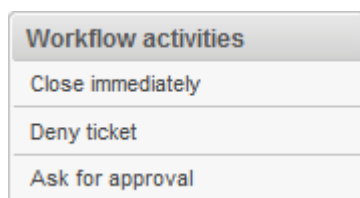


Figure 4: ConSol CM Web Client - Workflow activities

A.10.3 Drag & 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.

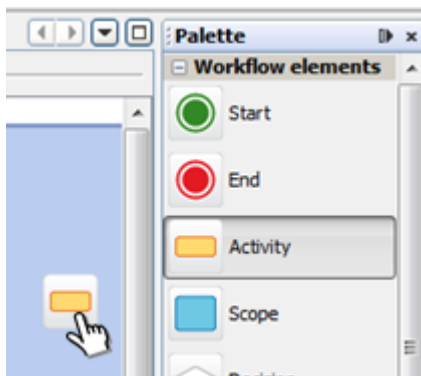


Figure 5: ConSol CM Process Designer - Drag & drop activities

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

A.10.4 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.

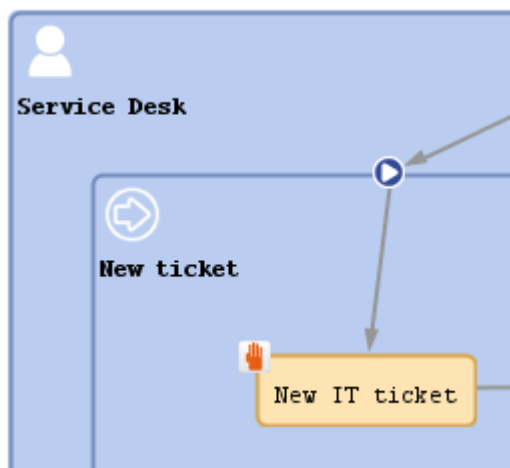


Figure 6: ConSol CM Process Designer - Nesting scopes

A.10.5 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.

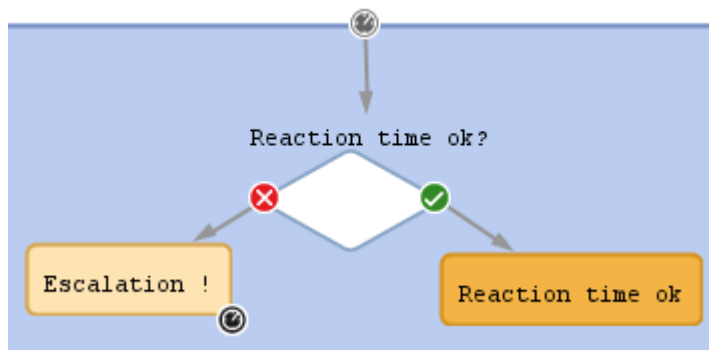


Figure 7: ConSol CM Process Designer - Triggering processes

A.10.6 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 8: ConSol CM Process Designer - Modeling interrupts

A.10.7 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, e-mails 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.

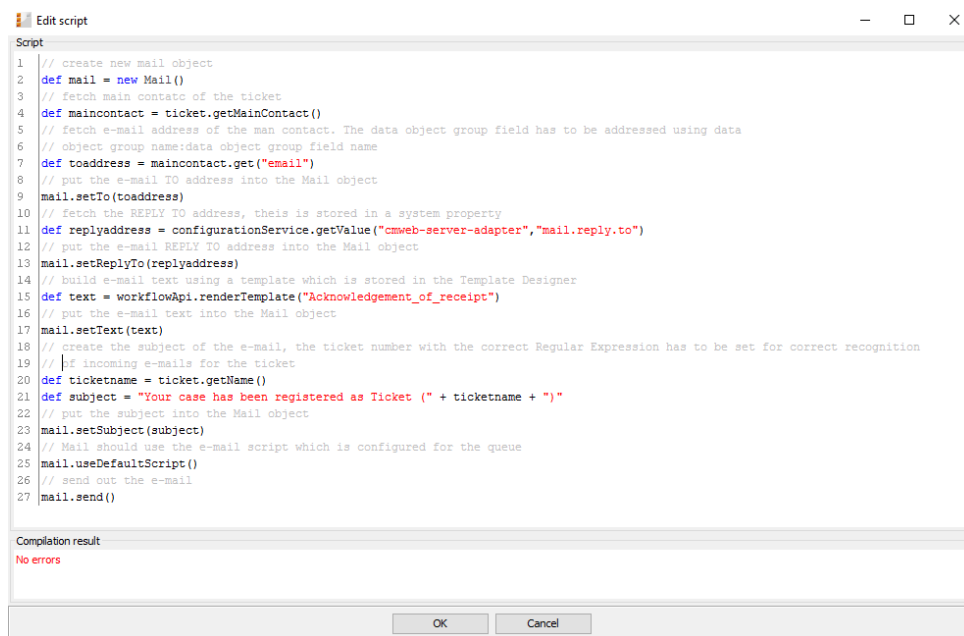
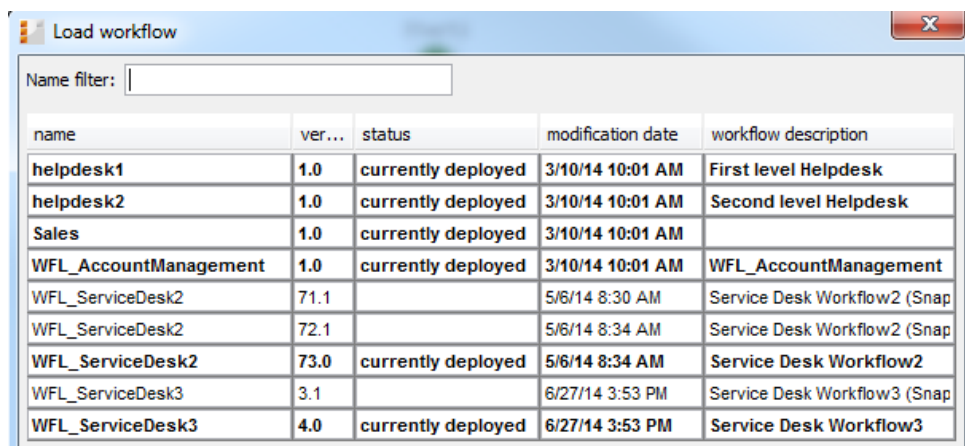


Figure 9: ConSol CM Process Designer - Script of an activity

A.10.8 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 10: ConSol CM Process Designer - Workflow versions

A.11 Basic Components of ConSol CM Processes

This chapter discusses the following:

A.11.1 General Objects	25
A.11.2 Data Fields	28
A.11.3 Standard Ticket Data Fields	29

A.11.1 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.
- **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.
- **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*.
- **Custom 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. Custom Fields are never defined on a single-field basis, but always in *Custom Field Groups*.
- **Data Object Group 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.

Data Object Group Fields are never defined on a single-field basis, but always in *Data Object Groups*.

- **Resource fields** (CM version 6.10 and up, 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 (version 6.9 and higher:) 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.

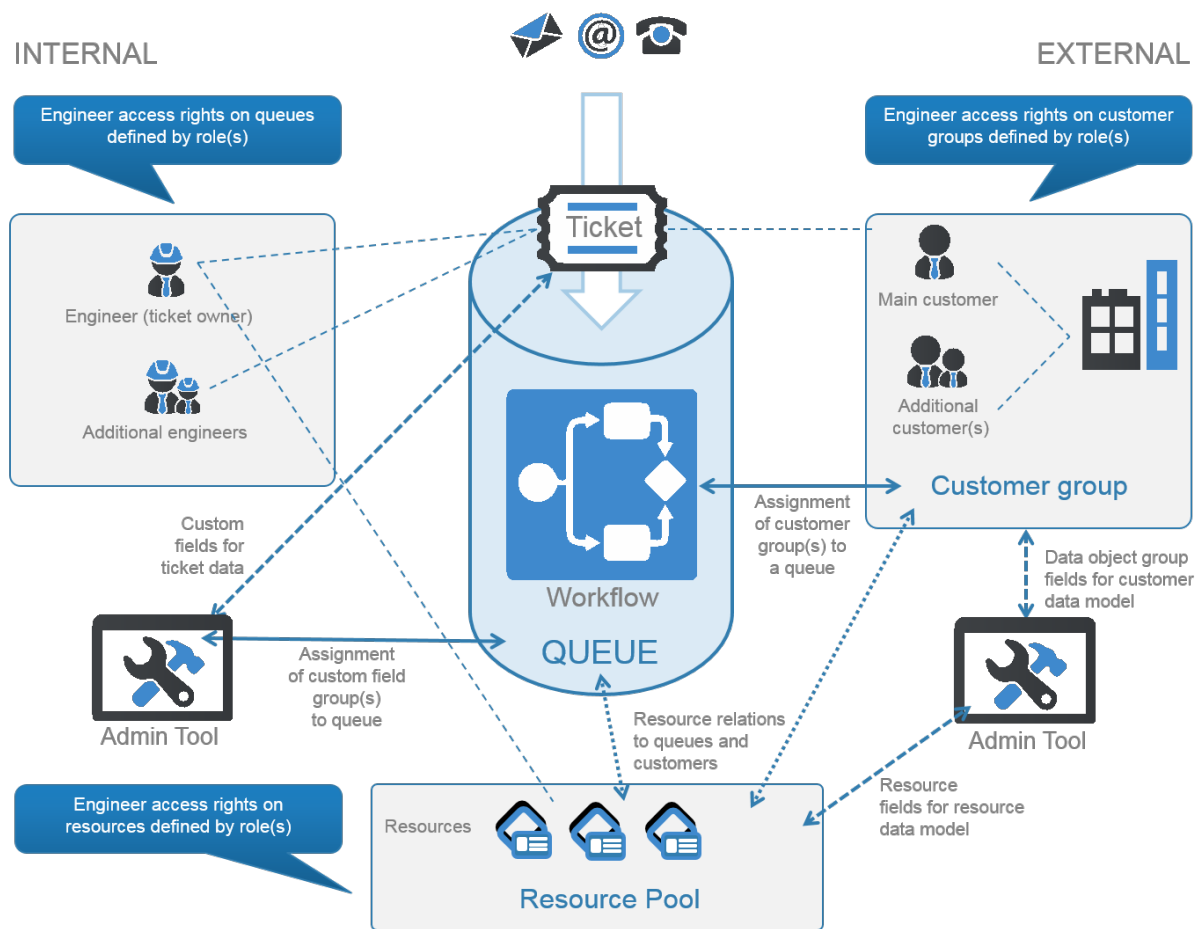


Figure 11: ConSol CM - Basic principle

A.11.2 Data Fields

For a detailed explanation of this topic, please see section [Working With Data Fields](#).

There are three types of data fields:

- **Custom Fields**
Used to define ticket data, managed in Custom Fields groups, as known from previous CM versions.
- **Data Object Group Fields**
Used to define customer data as part of the FlexCDM, the new customer data model. Managed in Data Object 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 Custom Field, Data Object Group 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*

A.11.3 Standard Ticket Data Fields

Some fields do not have to be defined as Custom 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.

B - Work with the Process Designer Application

This chapter discusses the following:

B.1 Steps to Perform for a New Process	31
B.2 Start of the Process Designer	31
B.3 Process Designer GUI	33
B.3.1 Introduction to the Process Designer GUI Elements	34
B.3.2 The Script Editor	49
B.3.3 GUI Tips and Tricks	50

B.1 Steps to Perform for 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 Custom 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*.

i 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.2 Start of the Process Designer

You can start the Process Designer on every PC or laptop where a standard web browser is installed (please see *System Requirements*) and which has network access to the ConSol CM server.

To start the Process Designer, open the ConSol CM start page and click on the Process Designer hyperlink. Java Web Start (JWS) is required to start the Process Designer application which runs on the local machine. However, JWS is an integral part of all Java distributions nowadays so that should not be a problem.

i 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.

On one machine, only one instance of the Process Designer should be running. You can control the *javaw* processes on the machine to be sure.

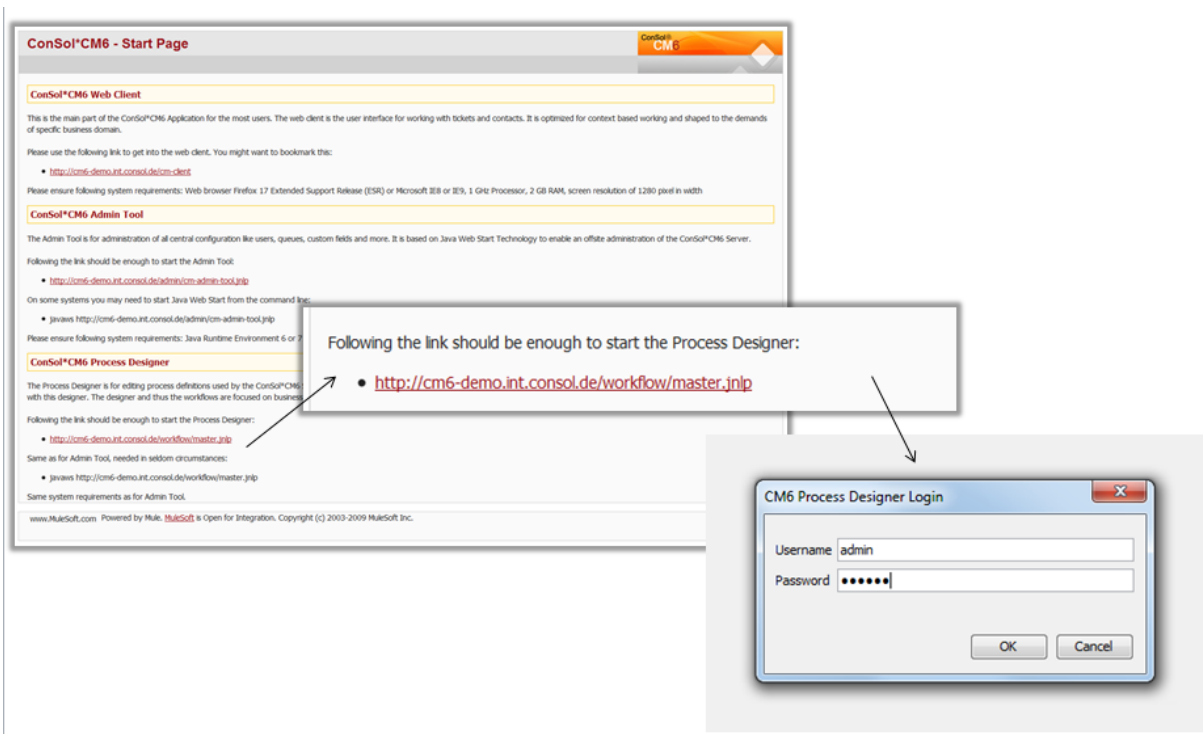


Figure 12: ConSol CM - Start of the Process Designer

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 *Role Administration*, for details.

B.3 Process Designer GUI

This chapter discusses the following:

- [Overview: GUI Sections](#)
- [Main Menu](#)
- [Workflow Editing Panel](#)
- [Palette for Elements and Adornments](#)
- [The Properties Editor \(Example: Activity\)](#)
- [The Script Editor](#)
- [GUI Tips and Tricks](#)

B.3.1 Introduction to the Process Designer GUI Elements

B.3.1.1 Overview: GUI Sections

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

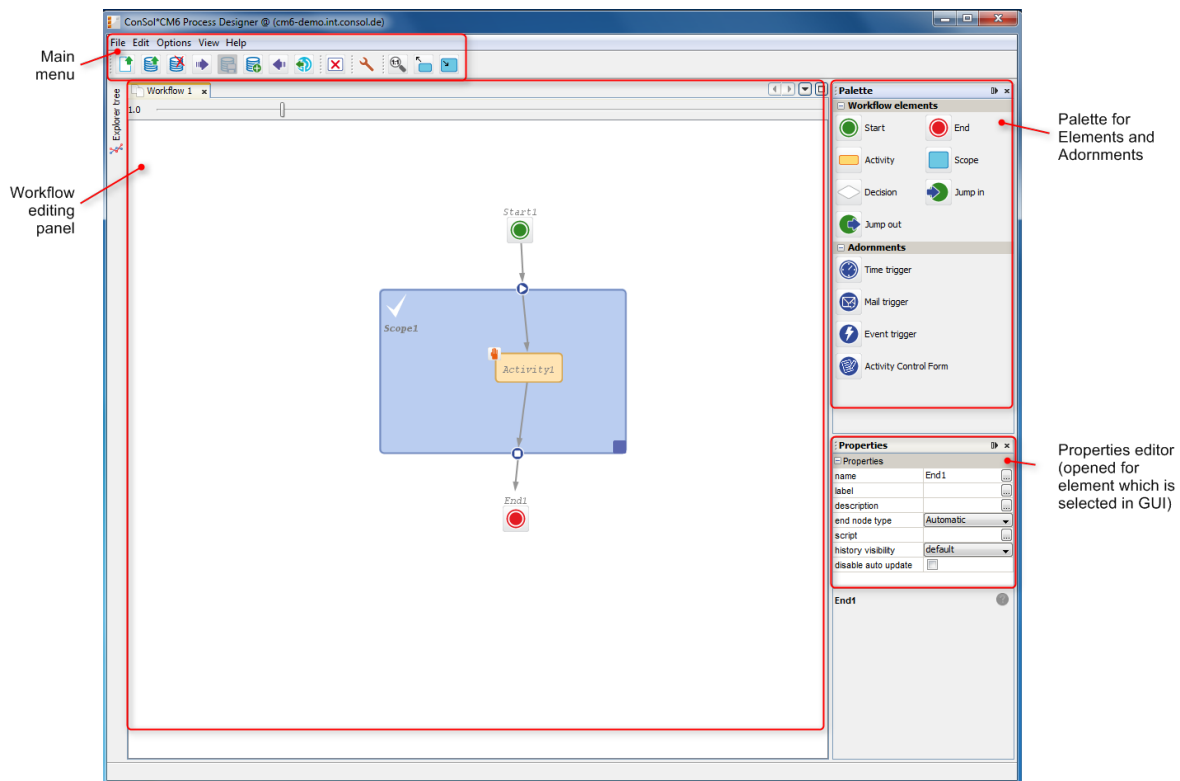















Figure 13: ConSol CM Process Designer - GUI elements

B.3.1.2 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 .

Menu main entry	Menu sub entry	Icon	Note
	Import ...		Import a workflow from a (proprietary workflow format) file.
	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 ...		<p>(Save as new version and) deploy the workflow, i.e. install the workflow in the system. The system might prompt you for a decision:</p> <ul style="list-style-type: none"> • Keep position of the tickets in the process (see section Actions During Workflow Deployment). • Start at START node again.
	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 <i>ConSol CM Administrator Manual</i> , section <i>Role Administration</i>) 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.
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 <i>Global Configuration</i> in the <i>ConSol CM Administrator Manual</i>) are available. The labels in the workflow in the main editing panel will be displayed in the selected language.
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		<p>Opens a pop-up window where the parameters for the ticket transfer during the deployment of a new workflow are displayed:</p> <ul style="list-style-type: none"> • 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 <i>Deploy</i> 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		<p>Opens the Log File Editor in the lower half of the screen and displays the user-specific log file of the Process Designer:</p> <p><USER_HOME_DIR>\.cmas\wfeditorR1\var\log</p>

Menu main entry	Menu sub entry	Icon	Note
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.3.1.3 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.

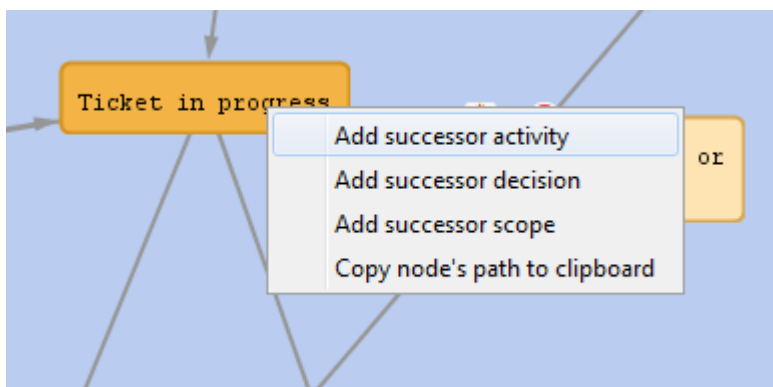


Figure 14: ConSol CM Process Designer - Context menu for a workflow activity

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.

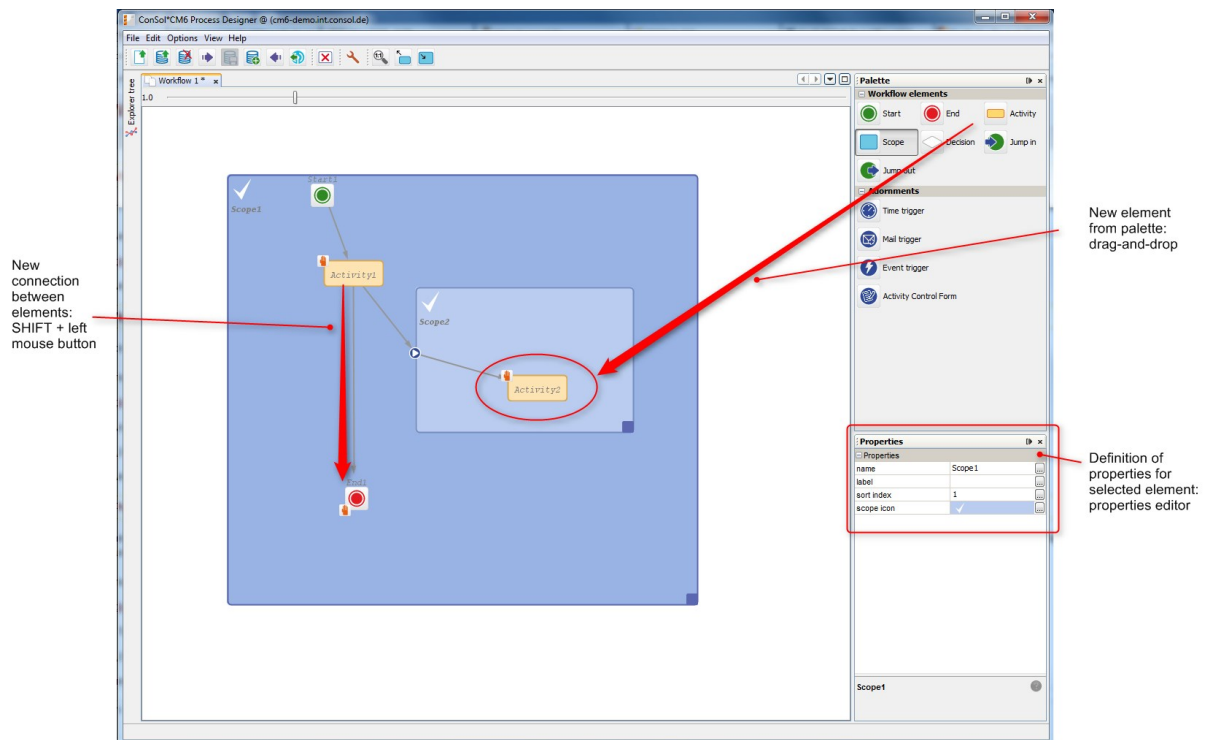


Figure 15: ConSol CM Process Designer - Adding new elements and connections

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.

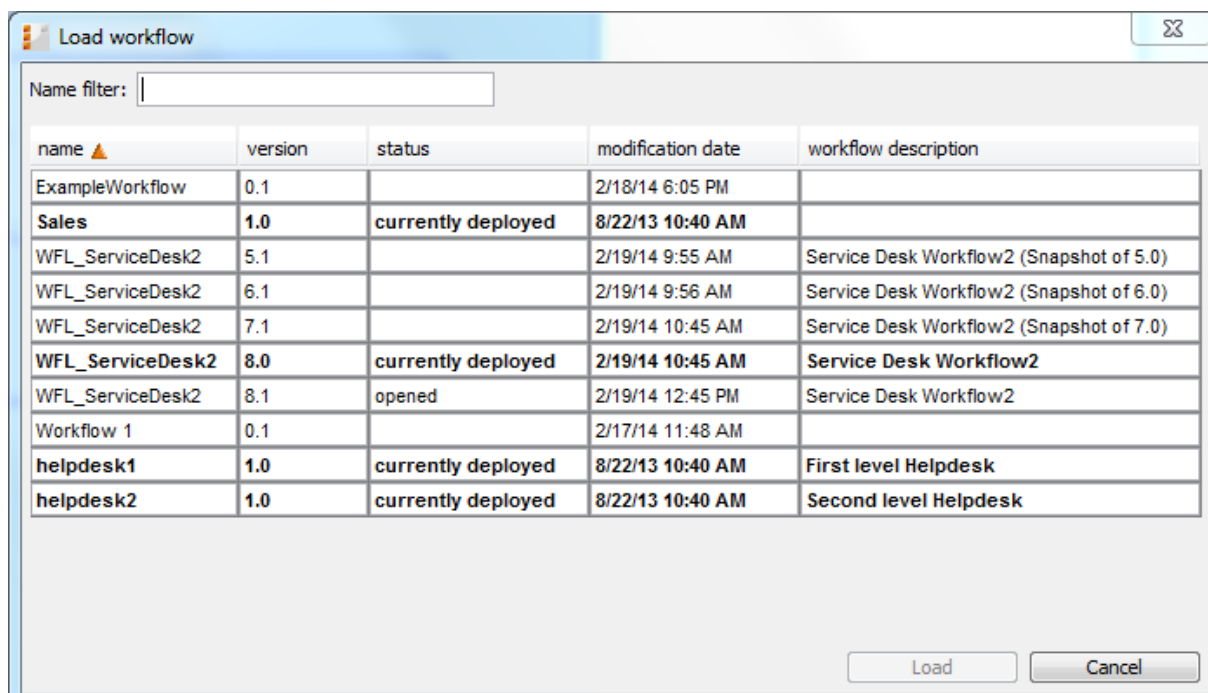


Figure 16: ConSol CM Process Designer - Load a workflow

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.

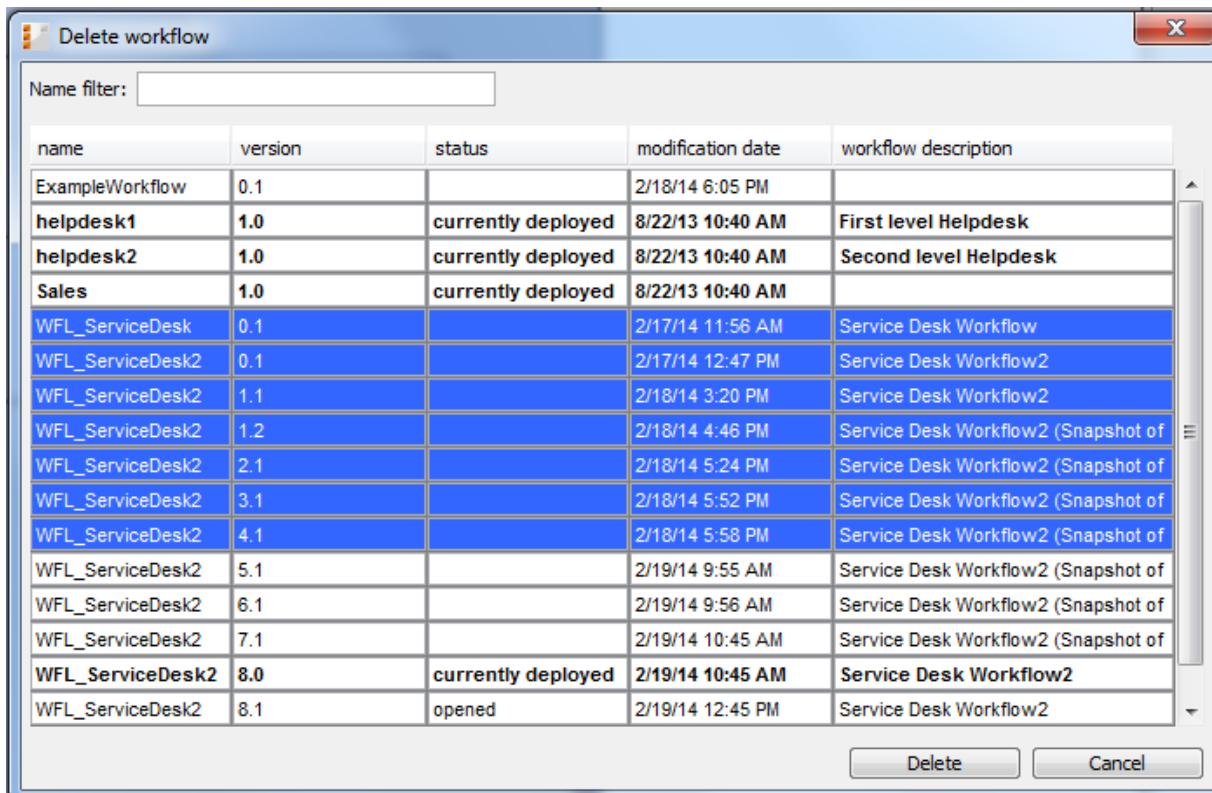



Figure 17: ConSol CM Process Designer - Delete a workflow

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*.

B.3.1.4 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.	Workflow Components: START Node
	End node	A workflow can contain one or more end nodes.	Workflow Components: END Nodes
	Activity	The actions in the workflow, manual or automatic.	Workflow Components: Activities
	Scope	The highest hierarchy level in workflows	Workflow Components: Scopes
	Decision	Decision node which has a <i>true</i> and a <i>false</i> exit point.	Workflow Components: 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



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 e-mail 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.

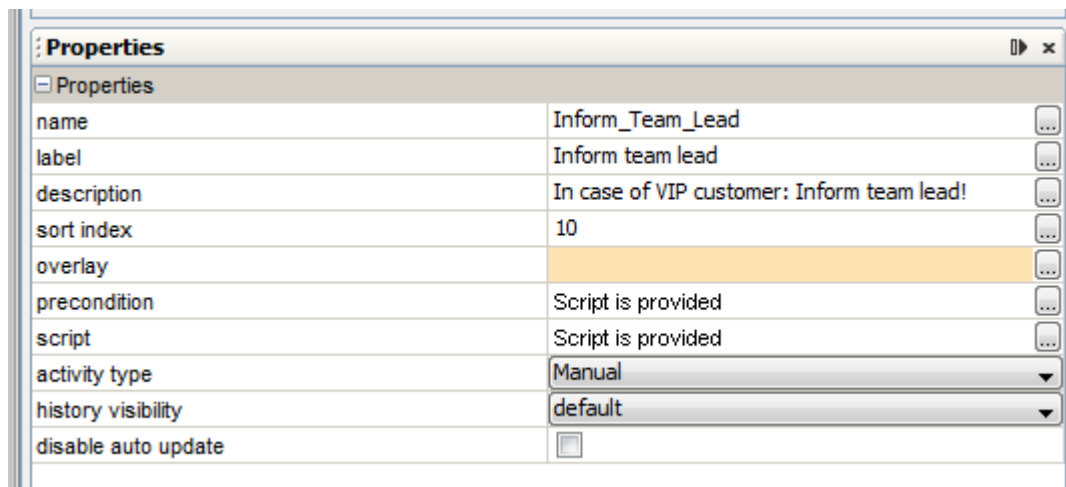
Icon	Name	Description	Section
	Manual	Indicates that the activity is a manual activity, i.e. the engineer has to click it in the Web Client.	Activity type
	Precondition	Indicates that the activity has a condition script attached. The condition script is executed when the previous activity has been performed. The activity with the precondition icon is only displayed if the condition script returns <i>true</i> .	Precondition

B.3.1.5 The Properties Editor (Example: Activity)

The Properties Editor is opened for the element which has been selected in the main editing panel and contains component-specific parameters. Some general parameters are present for all components, some are present only for a certain type of component.



Figure 18: ConSol CM Process Designer - Selected activity in workflow



Properties	
name	Inform_Team_Lead
label	Inform team lead
description	In case of VIP customer: Inform team lead!
sort index	10
overlay	
precondition	Script is provided
script	Script is provided
activity type	Manual
history visibility	default
disable auto update	<input type="checkbox"/>

Figure 19: ConSol CM Process Designer - Properties editor

Properties:

- **name**

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

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

- **label**

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

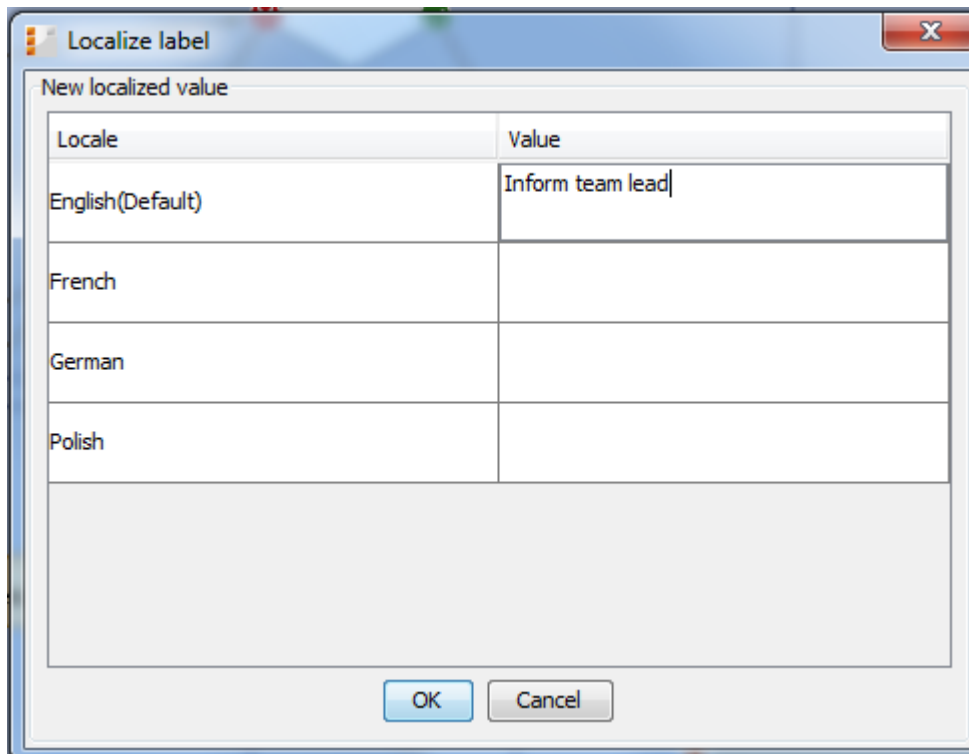


Figure 20: ConSol CM Process Designer - Localization for labels

- **description**

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

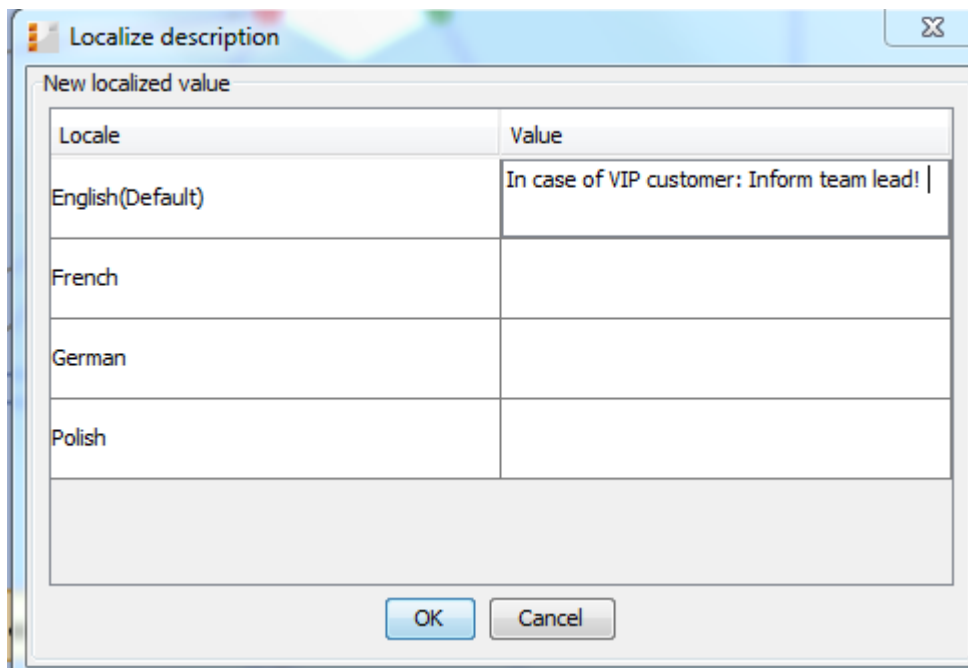


Figure 21: ConSol CM Process Designer - Localized description of an activity

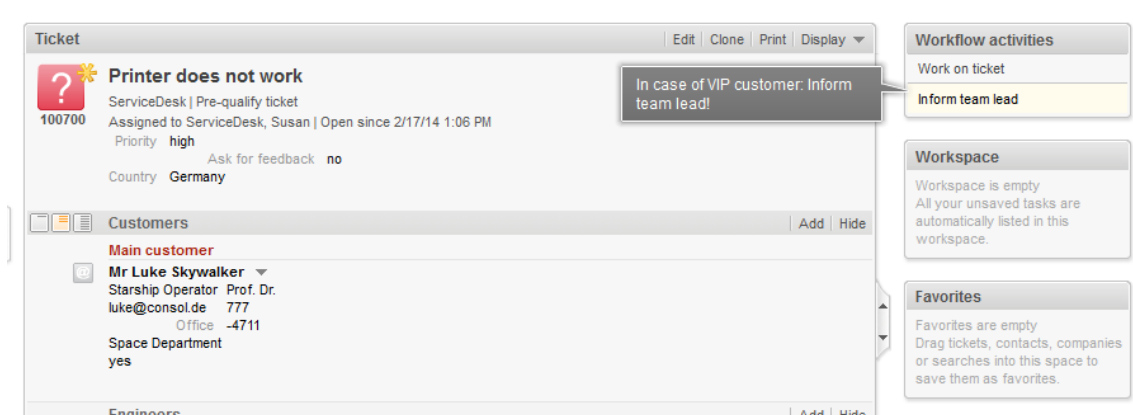


Figure 22: ConSol CM Web Client - Localized description of an activity as mouse-over

- **sort index**

Defines:

- **For activities:**
The order of the activities in the list of *Workflow activities* in the Web Client. The higher the number the more at the bottom of the list the activity is offered in the Web Client.
- **For scopes:**
The order of the tickets in the ticket list (Web Client) in views. The higher the scope index the more at the bottom of the list the tickets are displayed.

- **overlay**

Optional, for activities. Click into the orange space to define a standard ConSol CM overlay or one that has already been uploaded. Click on the file explorer (...) button to open the file explorer of the operation system for an upload of a new icon. When the ticket passes through an activity the overlay is added to the ticket icon in the Web Client. As a maximum, 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 e-mail:**

The ticket has received an e-mail.

- **A note for the engineer:**

E.g. another engineer has added a comment to *my* ticket.



Figure 23: ConSol CM Process Designer - Properties editor: Standard overlays and one customer-defined overlay



Figure 24: 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. 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.

- **precondition**

Optional, for activities. A script can be entered using the Script Editor (see section [The Script Editor](#)) which has to return *true* or *false*. The script 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/-precondition*.

- **script**

Optional, for activities. A script can be entered using the Script Editor (see section [The Script Editor](#)) which is executed when the ticket enters the activity.

- **activity type**

Mandatory, for activities. *Manual* or *Automatic* has to be selected. A manual activity is displayed in the Web Client and has to be explicitly selected/executed by an engineer. In the Process Designer it is marked by the icon *hand/manual*.

An automatic activity is executed without any engineer interaction. For a detailed explanation of the ConSol CM process logic, please see section [Process Logic](#).

- **history visibility**

Mandatory, but default value has been set (*default*). The value defines the display levels of the Web Client GUI where the action (that the activity has been performed) should be displayed:

- 2nd level and 3rd level
- only 3rd level
- on every level
- default

This refers to the value defined in the Admin Tool under *Ticket History* for the activity configuration. Depending on the type of activity, one of the following parameters is used:

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

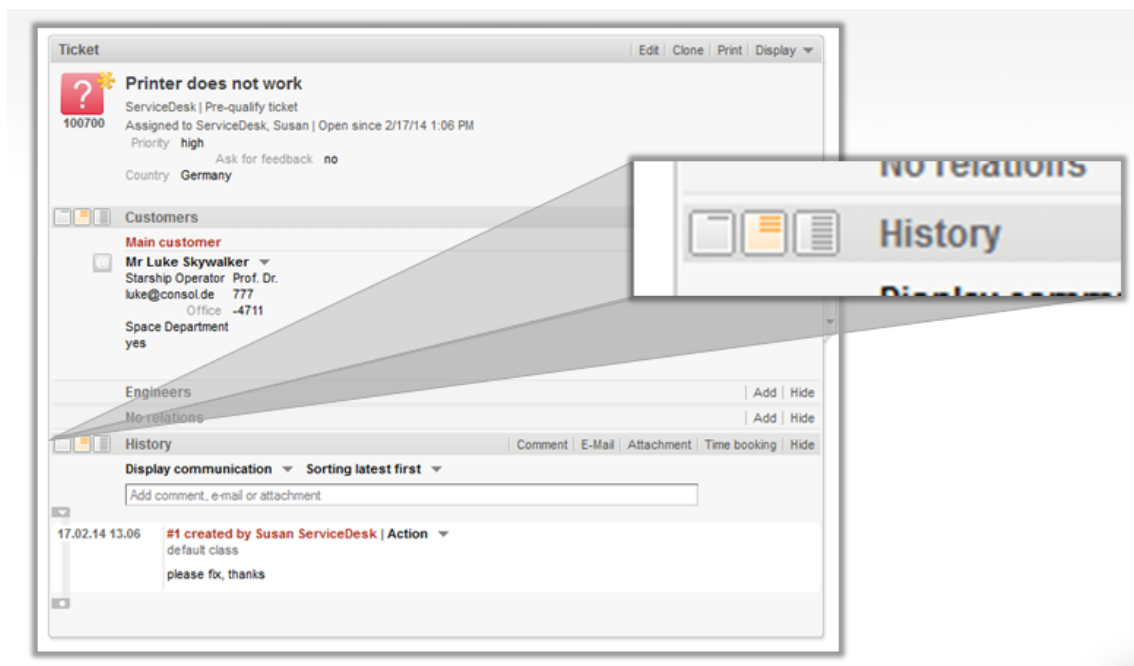


Figure 25: ConSol CM Web Client - Display levels in ticket history

- **disable auto update**

Defines ticket 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, only after the last event, the ticket is updated.

B.3.2 The Script Editor

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 [Introduction to Workflow Programming](#).

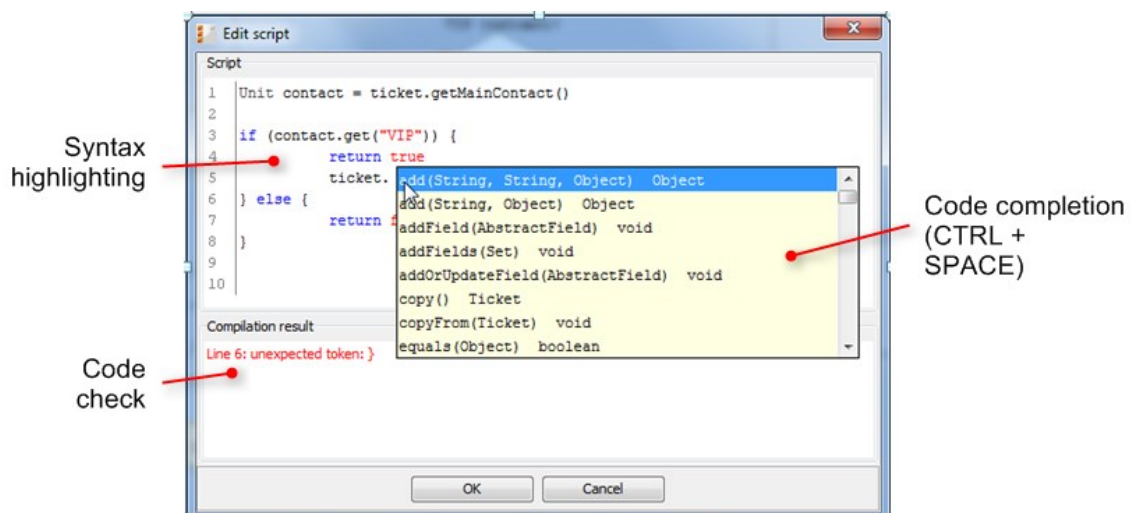



Figure 26: ConSol CM Process Designer - Script editor

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.3.3 GUI Tips and Tricks

 Make sure you save the workflow on a regular basis while you work on it. Currently, there is no UNDO button. In case you have accidentally removed elements which you still need, please load one of the predecessors of the current workflows and continue work with it.

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 e-mail trigger to a scope	grab the e-mail trigger symbol in the palette with the mouse and drop the newly created e-mail 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

C - Components of ConSol CM Workflows

C.1 Introduction

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 Workflow Components: START Node .
END Node(s)	One or more end nodes of the process. The ticket is closed. See section Workflow Components: END Nodes .
Scopes	Realms of a process, see section Workflow Components: Scopes .
Activities	The steps of a process. Can be automatic or manual, see section Workflow Components: Activities .
Decision Nodes	Workflow element which represents a <i>true/false</i> decision, see section Workflow Components: 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.2 Workflow Components: START Node

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.

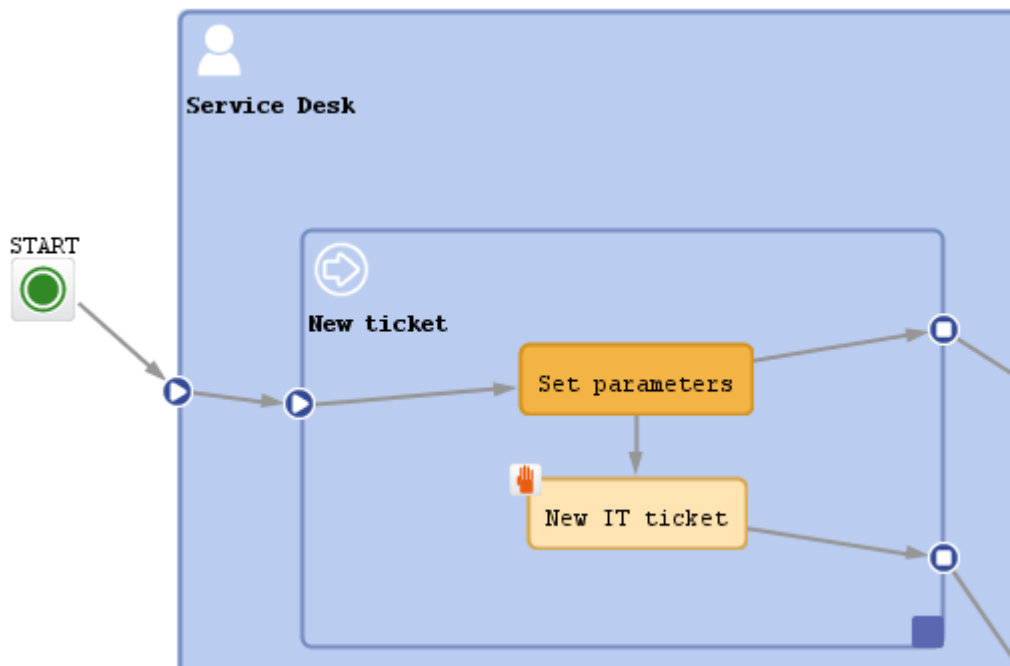



Figure 27: ConSol CM Process Designer - Start node

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.2.1 Properties of a Start Node

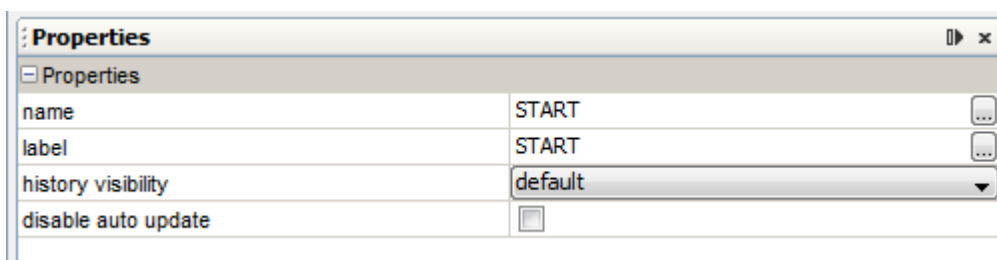


Figure 28: ConSol CM Process Designer - Start node properties

Properties:

- **name**
Technical object name.
- **label**
Localized name which will be displayed on the GUI.
- **history visibility**
See section [history visibility](#).
- **disable auto update**
See section [disable auto update](#).

C.3 Workflow Components: END Nodes

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

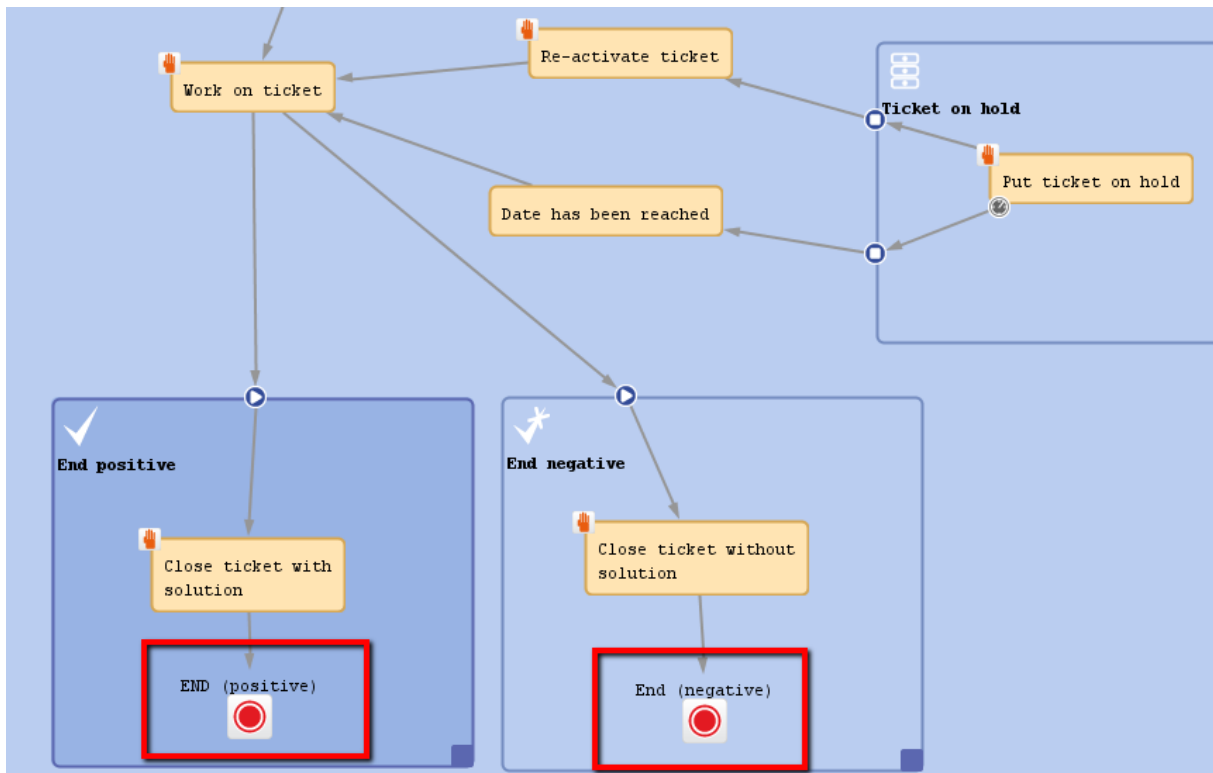


Figure 29: ConSol CM Process Designer - End nodes

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.3.1 The following actions are still possible for a closed ticket

- The ticket can be re-opened 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.
- 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.3.2 The following actions are not possible for a closed ticket

- 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).

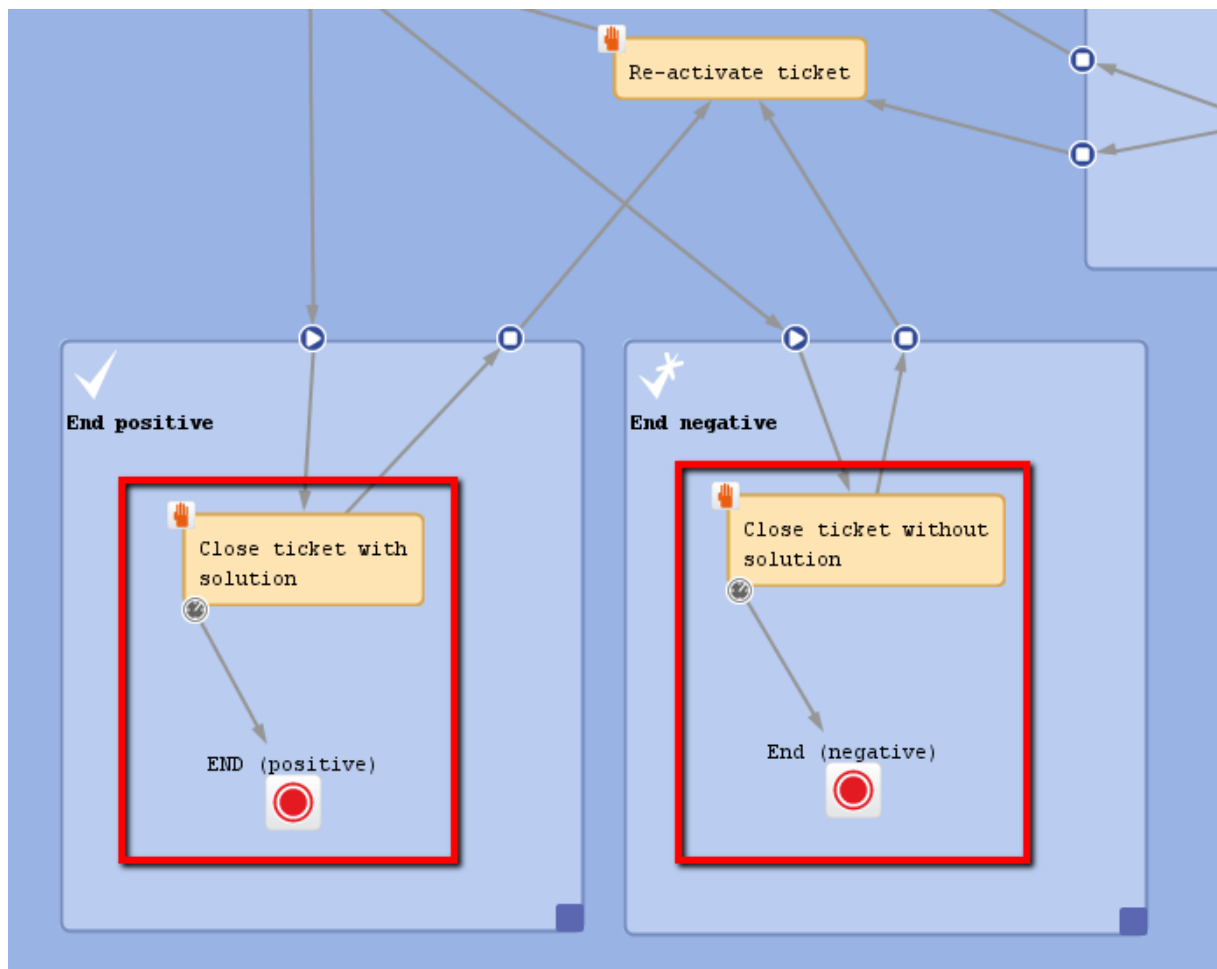


Figure 30: ConSol CM Process Designer - End nodes reached via time trigger

C.3.3 Properties of an End Node

Properties	
[-] Properties	
name	End_negative
label	End (negative)
description	The ticket is closed. There is no solution!
end node type	Automatic
script	
history visibility	default
disable auto update	<input type="checkbox"/>

Figure 31: 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.
- **end node type**
Automatic/Manual.
- **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**
See section [history visibility](#).
- **disable auto update**
See section [disable auto update](#).

C.4 Workflow Components: Scopes

This chapter discusses the following:

C.4.1 Introduction to Scopes	59
C.4.2 Defining a New Scope	61
C.4.3 Properties of a Scope	62
C.4.4 Scopes and Views	63
C.4.5 Scope Sort Index and its Side Effects	63

C.4.1 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.

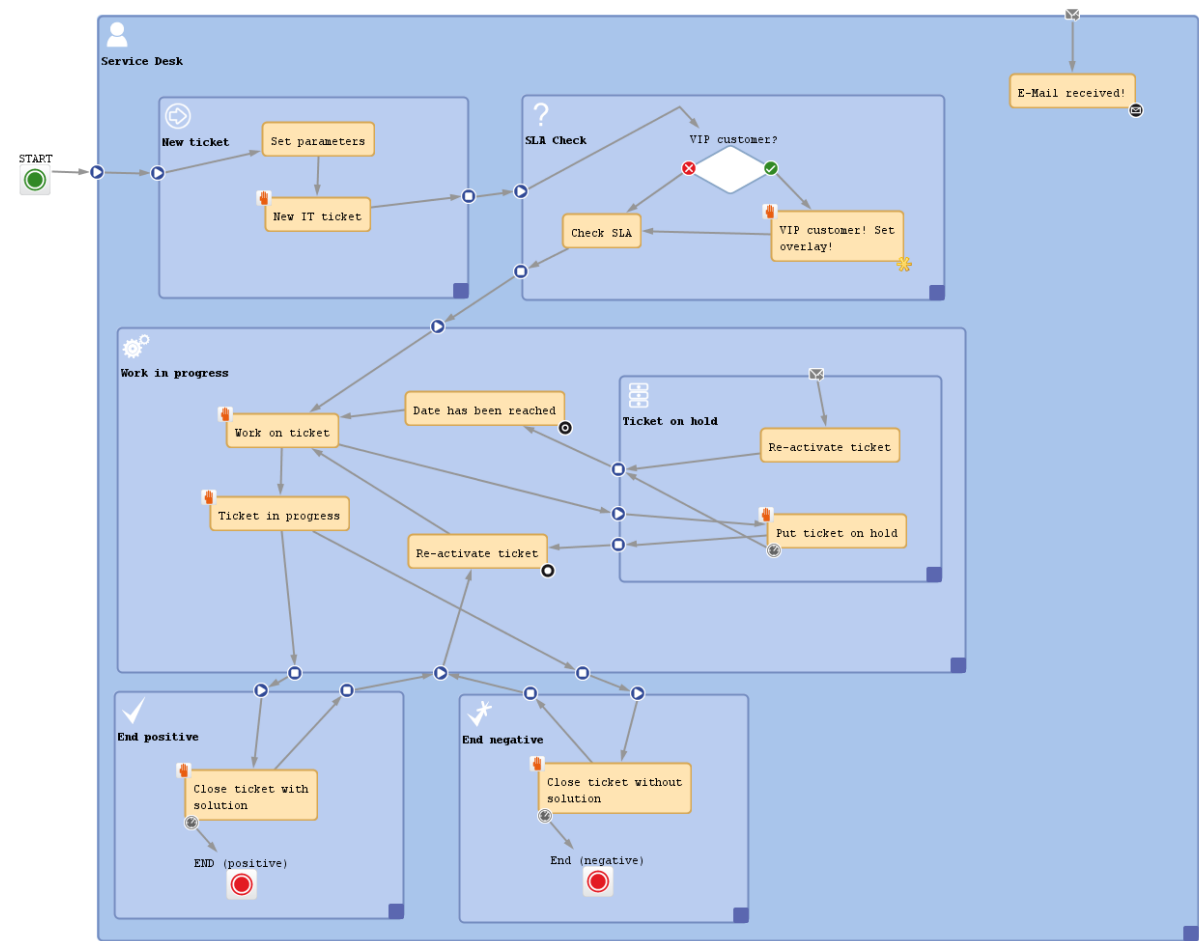


Figure 32: ConSol CM Process Designer - Workflow with scopes

Within each larger process step (i.e. within each scope), there can be one or more activities, e.g. during pre-qualification, first the VIP customer check is performed, then the SLA is checked. Those activities are described in detail in the section [Workflow Components: 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 e-mail 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.4.2 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.

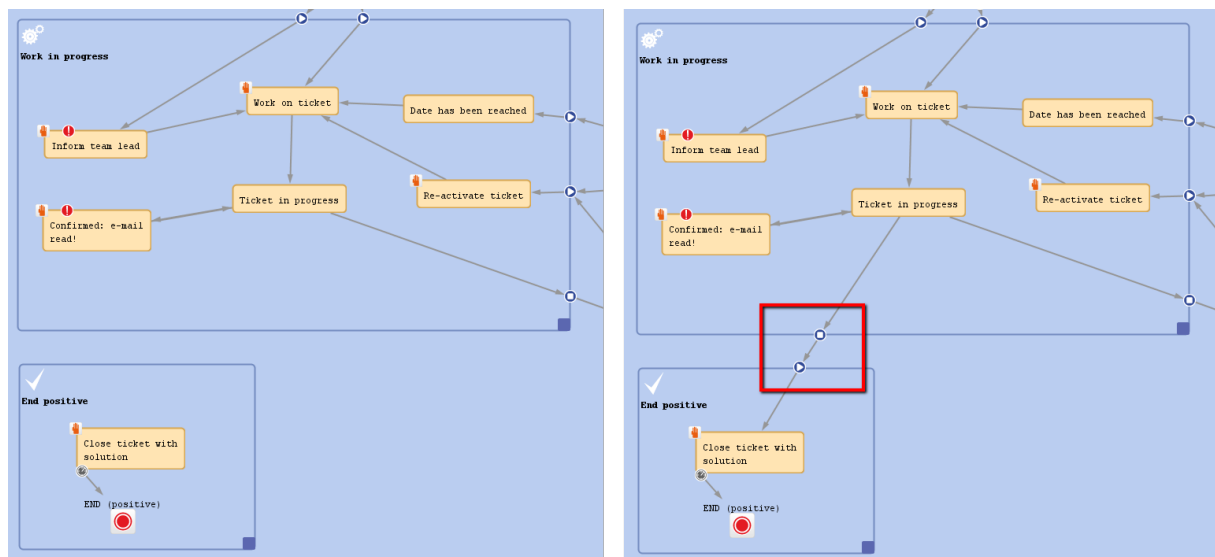


Figure 33: ConSol CM Process Designer - Automatically generated exit and entry points in scopes

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

C.4.3 Properties of a Scope

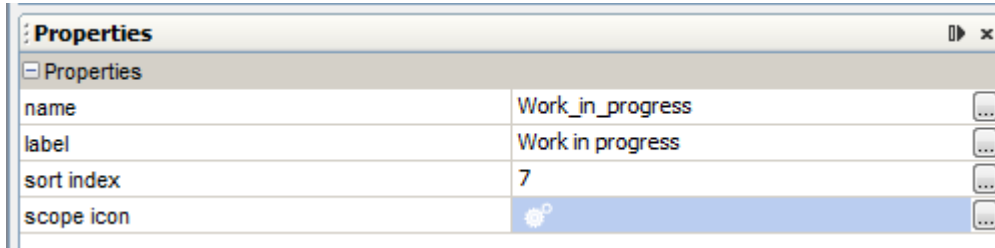


Figure 34: ConSol CM Process Designer - Scope properties

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.
- **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 and its Side Effects](#).
- **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.

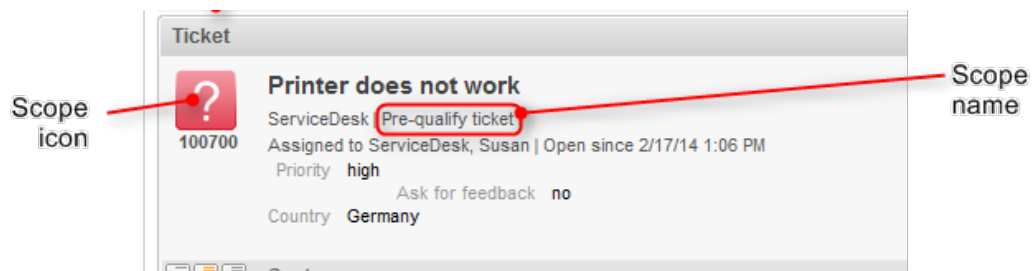


Figure 35: ConSol CM Web Client - Scope icon



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.4.4 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.4.5 Scope Sort Index and its Side Effects

The scope index defines

- the scope sort order in CM.Track V2 (available in CM version 6.10.5 and up)

C.5 Workflow Components: Activities

This chapter discusses the following:

C.5.1 Introduction to Activities	65
C.5.2 Properties of an Activity	67
C.5.3 Process Logic of Activities	69
C.5.4 Examples for Activities	70

C.5.1 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

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 *hand/manual* icon.

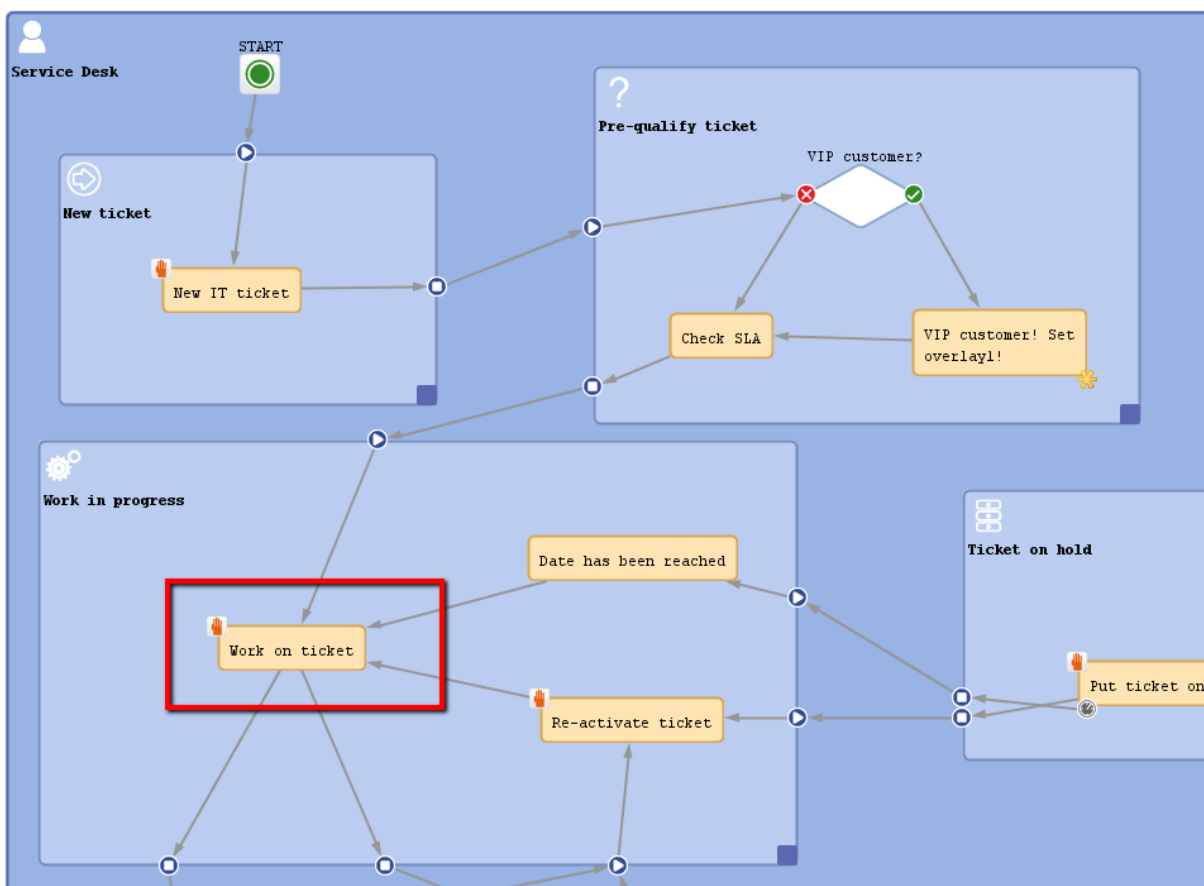


Figure 36: ConSol CM Process Designer - Manual activity in workflow

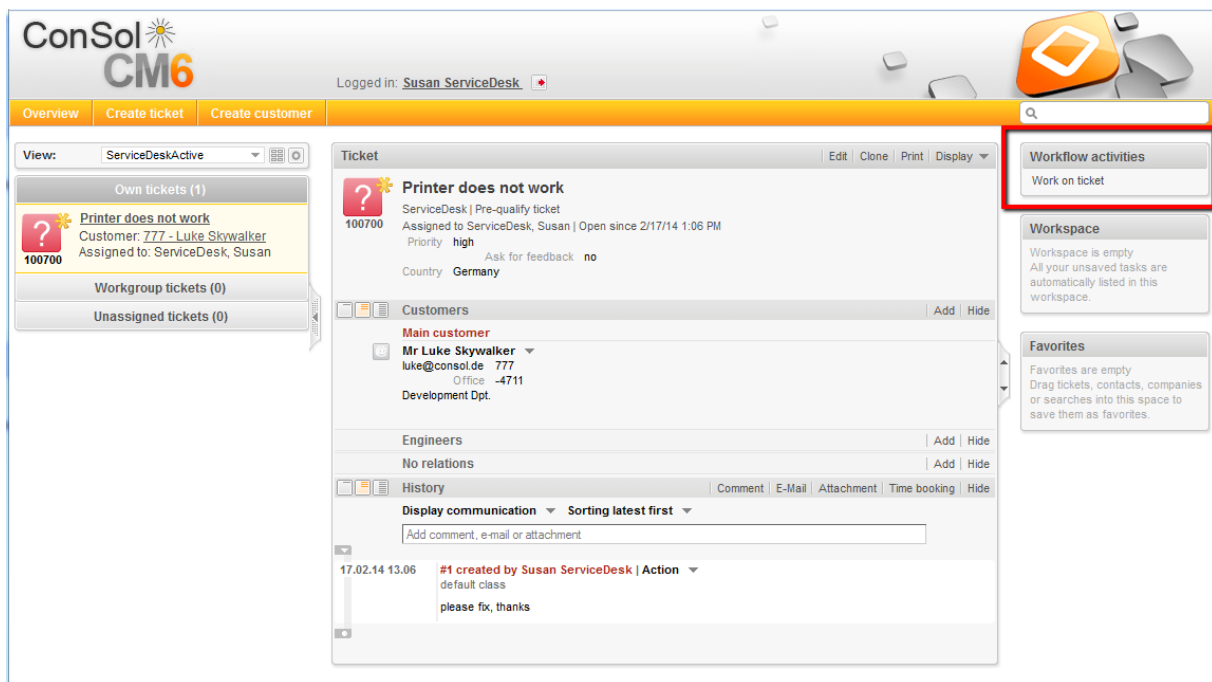


Figure 37: ConSol CM Web Client - Manual activity

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.

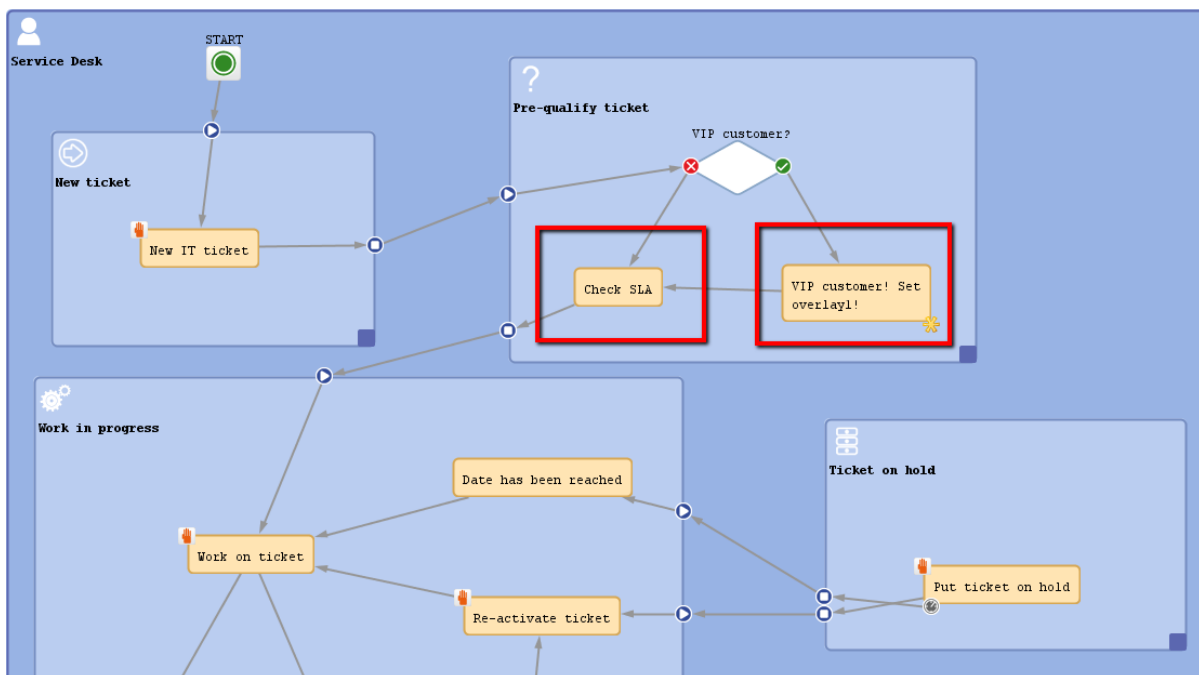


Figure 38: ConSol CM Process Designer - Automatic activities

C.5.2 Properties of an Activity

In order to display and edit the properties of an activity, mark the activity in the Process Designer.

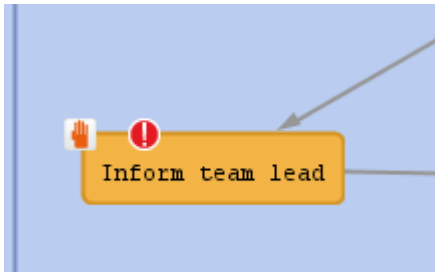


Figure 39: ConSol CM Process Designer - Activity

The Properties Editor will be opened for this activity.

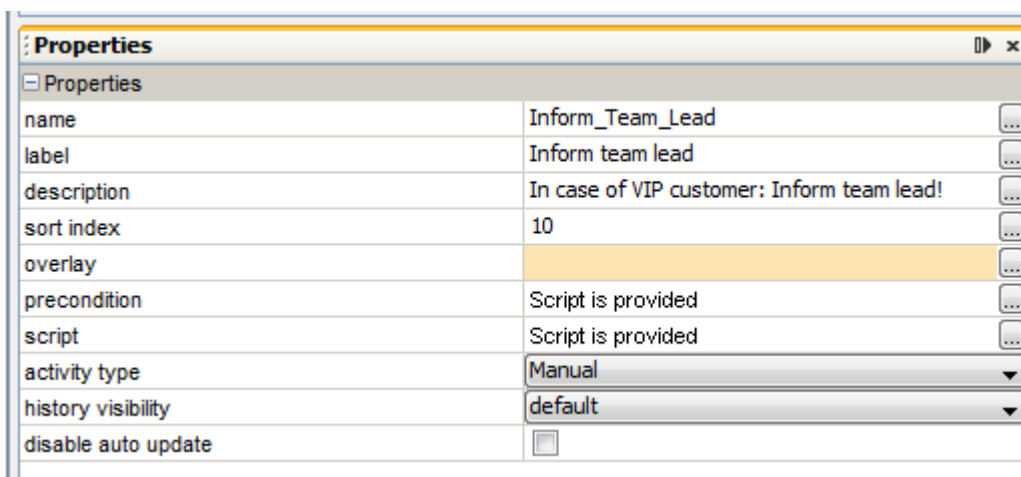


Figure 40: ConSol CM Process Designer - Properties of an activity

An activity can have the following properties:

- **name**
Mandatory, technical object name.
- **label**
Optional (if not set, the technical name is used). Localized name which will be displayed in the Web Client. The language which is configured in the web browser is used.
- **description**
Optional. Will be displayed as mouse-over in the Web Client.
- **sort index**
Defines the order of the activities in the Web Client.
- **overlay**
Optional. Click into the orange space to load standard ConSol CM overlay icons or use the file browser (...) to upload another icon from the file system.

- **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. 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.

- **precondition**

Optional. A script can be entered which is executed when the activity should be offered in the Web Client GUI. The script has to return *true* or *false*. If a precondition has been defined for an activity, the activity is marked by the *exclamation mark/precondition* icon (see figure above).

- Return value is **true**.

The activity is displayed. If it is a manual activity it can be selected/performed by the engineer in the Web Client GUI.

- Return value is **false**.

The activity is not displayed in the Web Client GUI



CM version 6.9 and higher:

When you work with Data Object Group 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 defined which is executed when the ticket passes through the activity.

- **activity type**

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

- **history visibility**

See section [history visibility](#).

- **disable auto update**

See section [disable auto update](#).

C.5.3 Process Logic of Activities

This is the process logic of activities:

1. When a ticket has passed through an activity it always waits behind this activity (and not before the next one!).
2. 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.
3. 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.
4. 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.
5. If the engineer selects the activity in the Web Client GUI, the script of the activity is executed.
6. If there is a *postActivityScript*, this script is executed immediately after the execution of the activity script.
7. 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.



A ticket always waits behind the last activity which has been executed and not before the new one!!

C.5.4 Examples for Activities

C.5.4.1 Example 1: Precondition for Displaying Activity "Inform team lead"

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 should not be offered. The Data Object Group Field *vip* which is part of the customer data model is checked for this purpose.

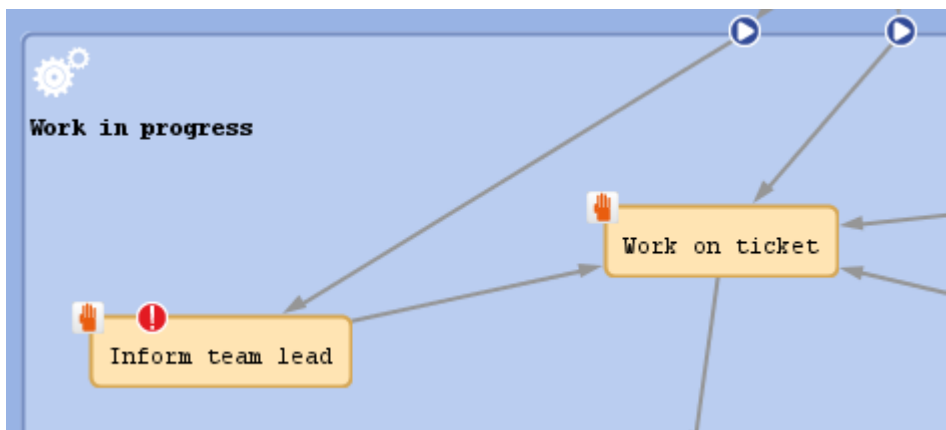


Figure 41: ConSol CM Process Designer - Workflow activities (one with precondition script)

```
// Get the main contact of the ticket. The unit object (can be a customer or a
// company) is provided;
// here it has to be a customer, i.e. a contact:

Unit contact = ticket.mainContact

// Check the Custom Field "vip" of the main contact. (see next image)
// If it is set to true, return true, i.e. the condition is TRUE.
// Else return false, i.e. the condition is FALSE:

if (contact.get("vip")) {
    return true
} else {
    return false
}
```

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

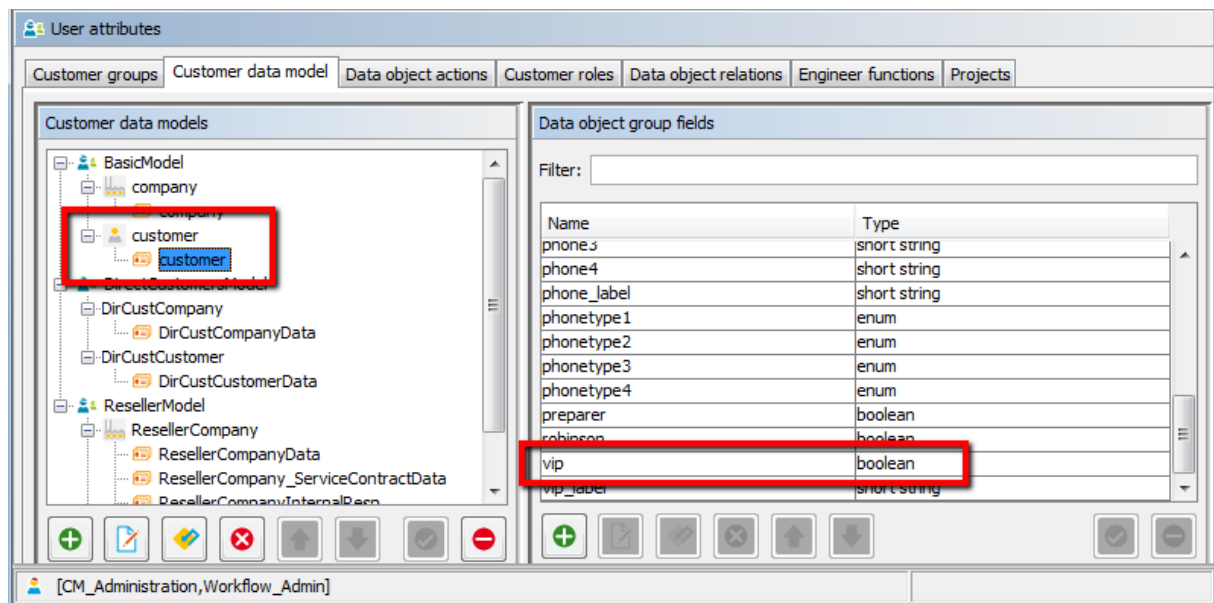


Figure 42: ConSol CM Admin Tool - Data Object Group Field "vip" (CM version 6.9)

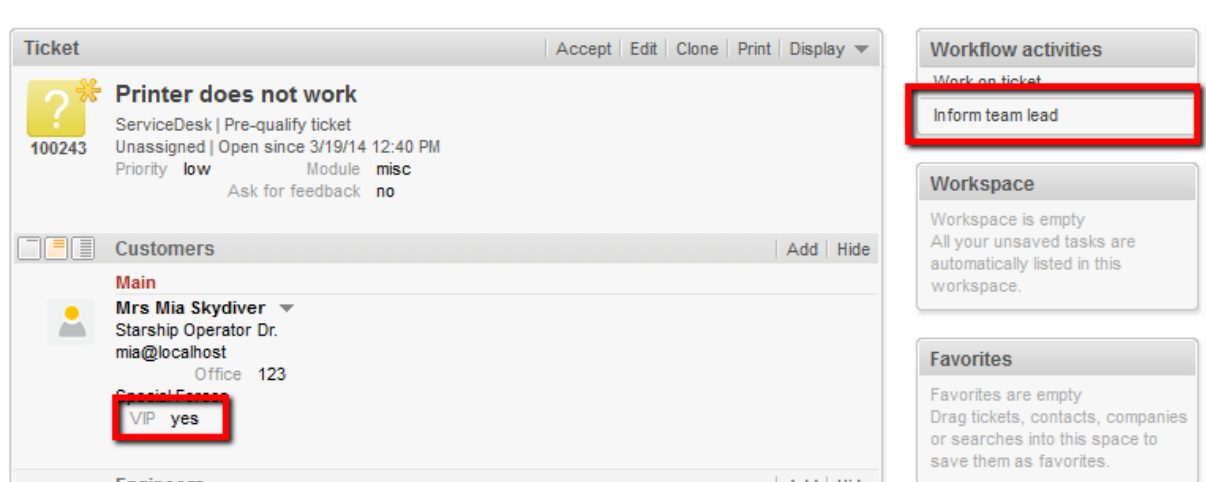


Figure 43: ConSol CM Web Client - Precondition: return value TRUE

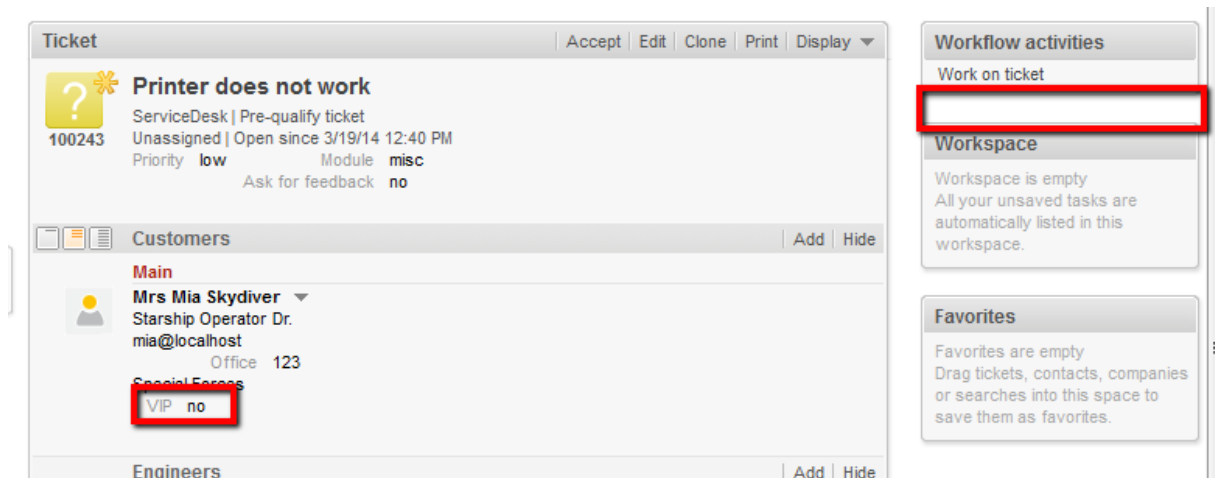


Figure 44: ConSol CM Web Client - Precondition: return value FALSE

C.5.4.2 Example 2: Send an E-Mail to the Main Contact When a Ticket Has Been Opened

When a ticket has been opened, an e-mail should be sent to the main contact of the ticket.

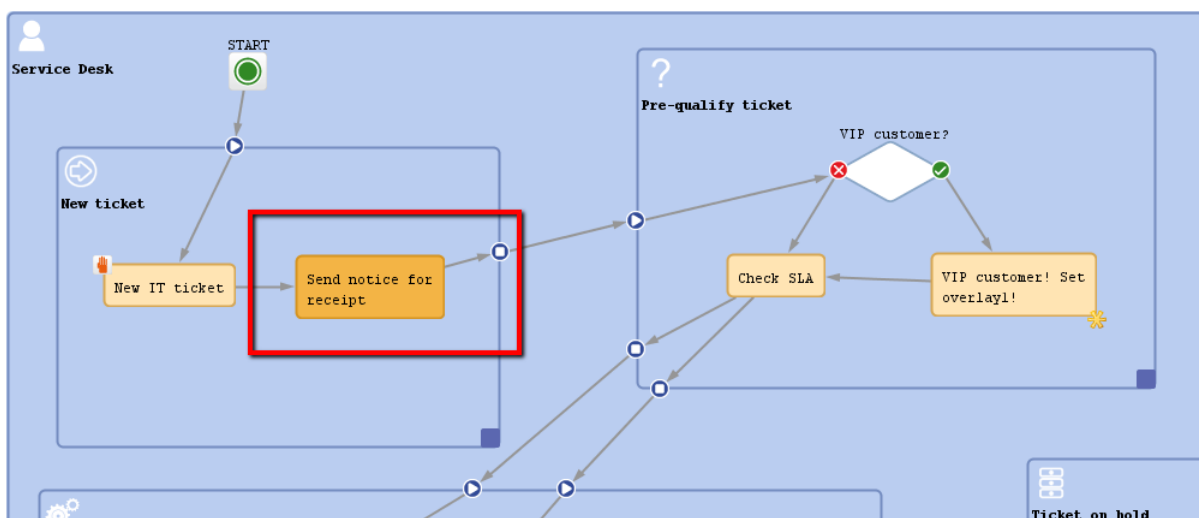


Figure 45: ConSol CM Process Designer - Automatic activity where receipt note is sent

```
// Get the main contact of the ticket:
def mycontact = ticket.mainContact

// Get the value of the Custom Field "email" of the main contact:
def mycontact_e = mycontact.get("email")

// Use as text the e-mail template with name "receipt_notice_ServiceDesk".
// Can be located in the Template Designer or in the Admin-Tool.
// Usually e-mail templates are stored in the Template Designer:
def text = workflowApi.renderTemplate("receipt_notice_ServiceDesk")

// Get the reply-to address for the e-mail.
// This is stored in the system property "cmweb-server-adapter","mail.reply.to":
def replyto = configurationService.getValue("cmweb-server-adapter","mail.reply.to")

// Build the string for the ticket subject.
// Keep in mind that the regular expression which defines the ticket identifier has
// to be in this subject.
// Otherwise, an e-mail cannot be assigned to the correct ticket.
def subj = "Your request has been received: ticket (" + ticket.getId() + ")"

//Send out the e-mail
workflowApi.sendEmail(contact_e,subj,text,replyto,null)
```

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

```
// all lines of code identical to variant 1 except for the last line:

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.5.4.3 Example 3: Assign the Ticket to the Current Engineer

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

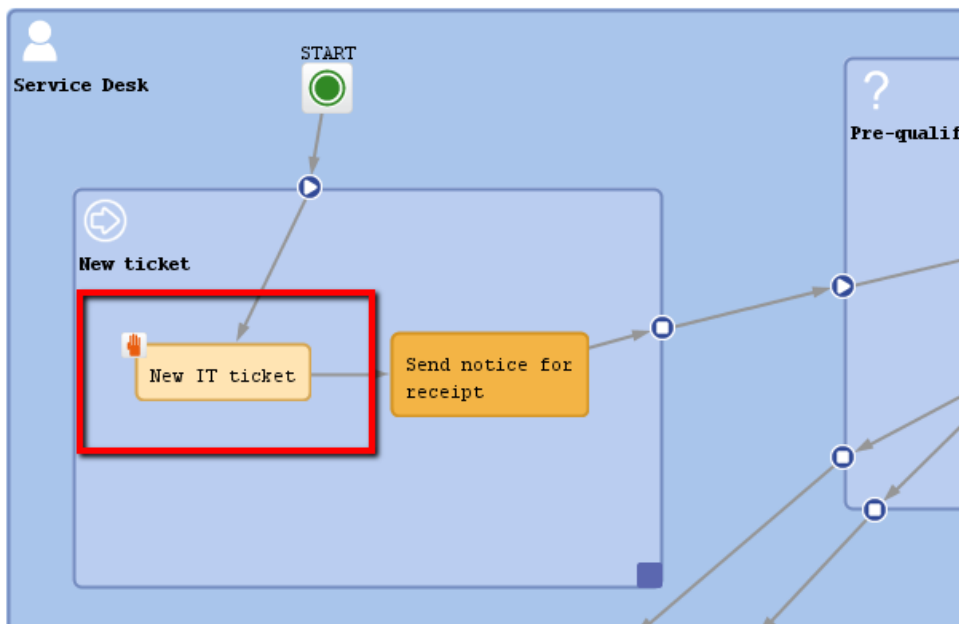


Figure 46: ConSol CM Process Designer - Workflow activity where engineer should be assigned

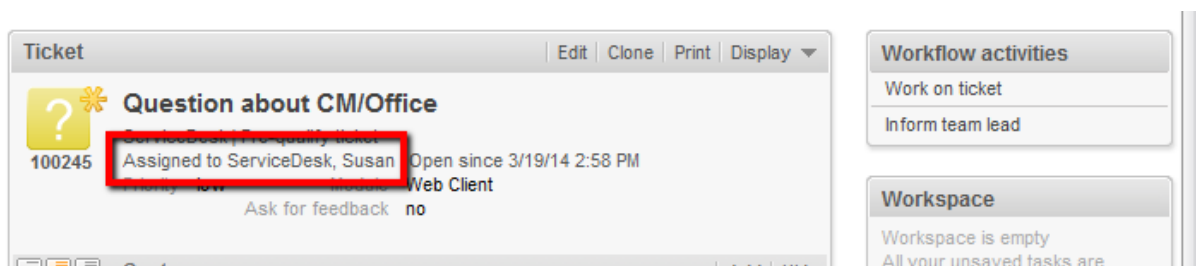


Figure 47: ConSol CM Web Client - Ticket passed through activity where engineer was assigned

```
// Get the engineer who is executing the activity:
def curr_eng = workflowApi.getCurrentEngineer()

// Assign the ticket to the current engineer
ticket.setEngineer(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 the following method:

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

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

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.6 Workflow Components: Decision Nodes

This chapter discusses the following:

C.6.1 Introduction to Decision Nodes	77
C.6.2 Properties of a Decision Node	78
C.6.3 Example for a Decision Node	79

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.

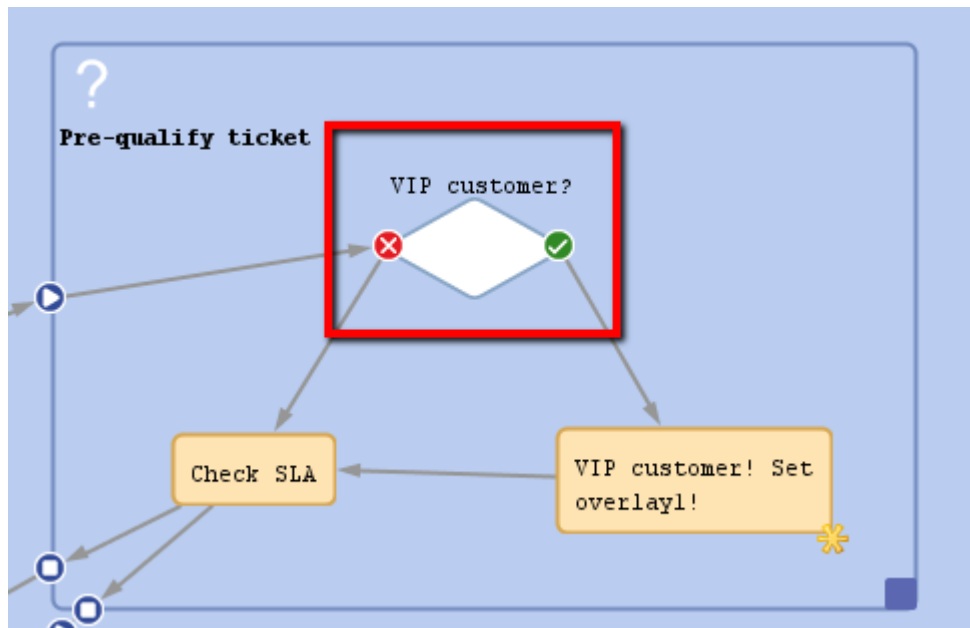
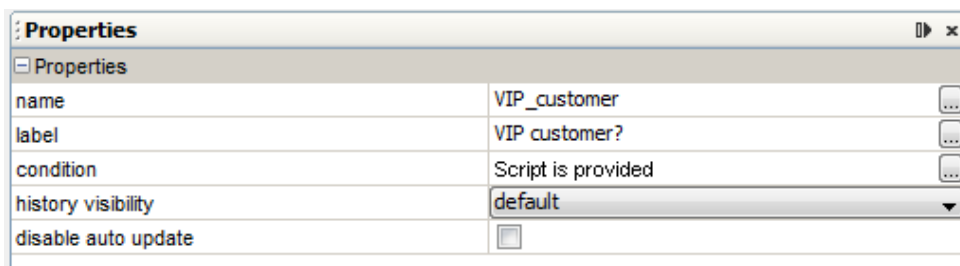


Figure 48: ConSol CM Process Designer - Decision node

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**
See section [history visibility](#).
- **disable auto update**
See section [disable auto update](#).



The screenshot shows a 'Properties' dialog box with a table of properties for a decision node. The properties are:

Properties	
name	VIP_customer
label	VIP customer?
condition	Script is provided
history visibility	default
disable auto update	<input type="checkbox"/>

Figure 49: ConSol CM Process Designer - Decision node: Properties

C.6.3 Example for 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).

1. A Data Object Group 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*.

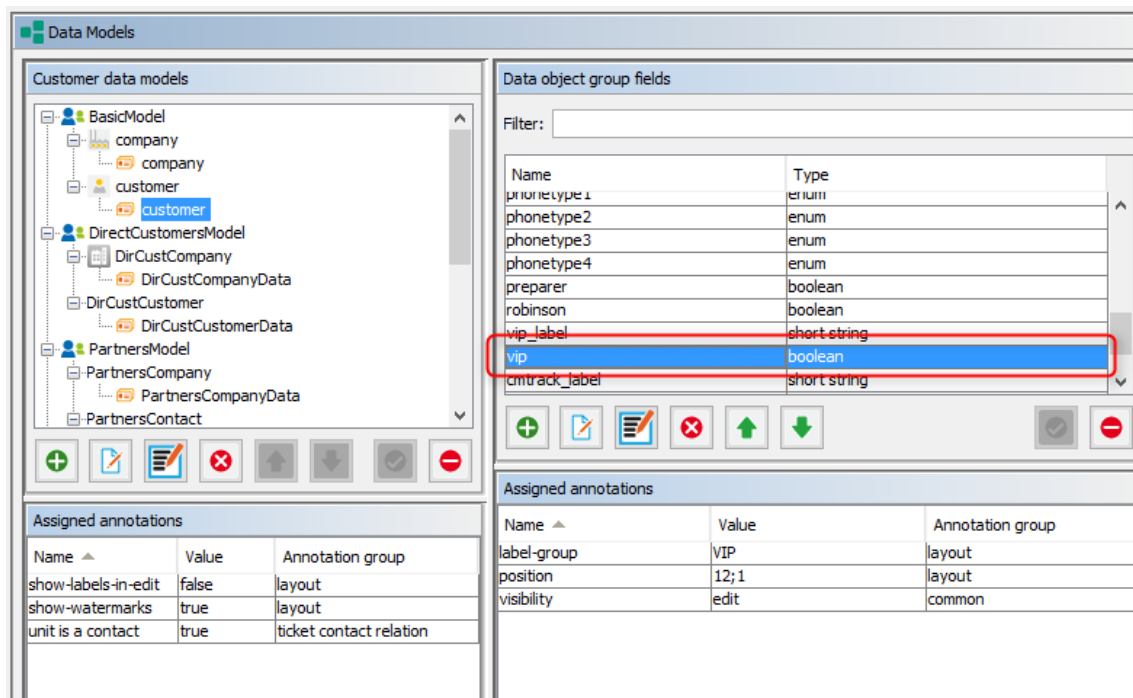


Figure 50: ConSol CM Admin Tool - Data Object Group Field "vip" in customer data

Figure 51: ConSol CM Web Client - Data Object Group Field "VIP" for customer/contact data

2. 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*).

```
// Get the main contact of the ticket. The unit object (can be a customer or
// a company) is provided;
// here it has to be a customer, i.e. a contact:

Unit mycontact = ticket.mainContact

// Check the Custom Field "vip" of the main contact. (see next image)
// If it is set to true, return true, i.e. the condition is TRUE.
// Else return false, i.e. the condition is FALSE:

if (mycontact.get("vip")) {
    return true
} else {
    return false
}
```

Code example 6: Working with customer data. Example: using a Data Object Group Field of type boolean

3. 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.

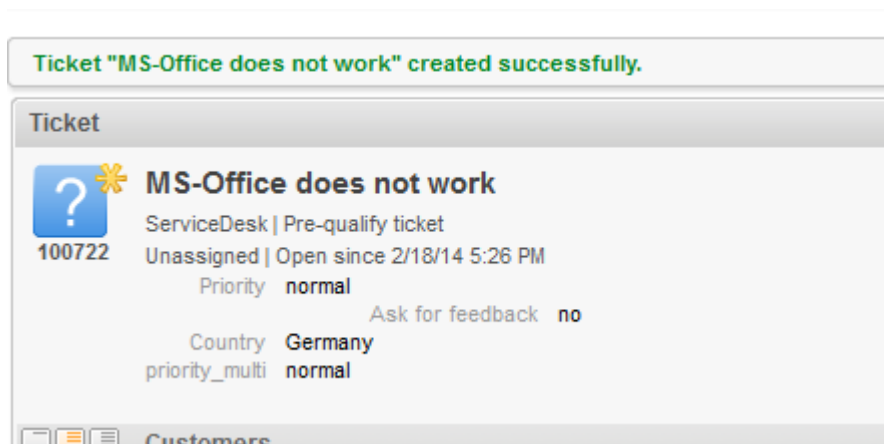


Figure 52: ConSol CM Web Client - Ticket icon with VIP overlay

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 e-mail 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 53: *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/she 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 e-mail 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.

- **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.

- **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.

- **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*).

Properties	
name	TimeTrigger2
minutes	0
hours	4
days	0
use calendar	<input type="checkbox"/>
repeatable	<input type="checkbox"/>
script after timer	
script on timer start	
retry interval	default value

Figure 54: 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 e-mail to a ticket has come in and after three hours, no engineer has read the e-mail and has taken care of the ticket, an alert should be triggered. To implement this, an incoming e-mail (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 e-mail 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.

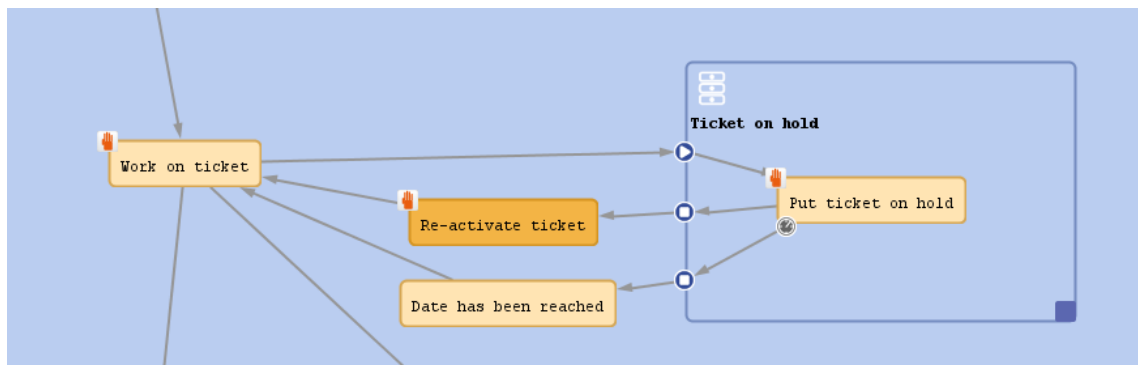


Figure 55: ConSol CM Process Designer - Use case 1: Workflow

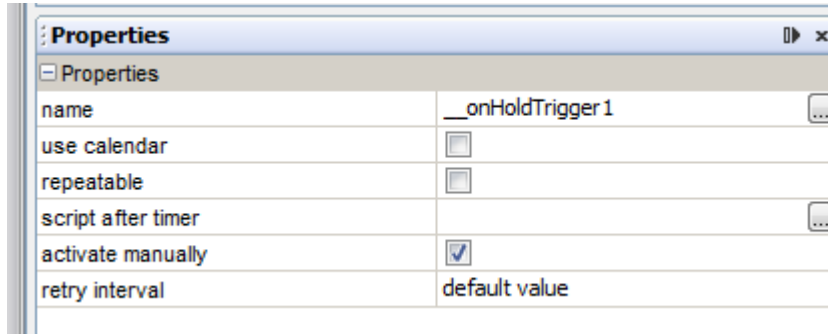


Figure 56: ConSol CM Process Designer - Use case 1: Properties editor for time trigger

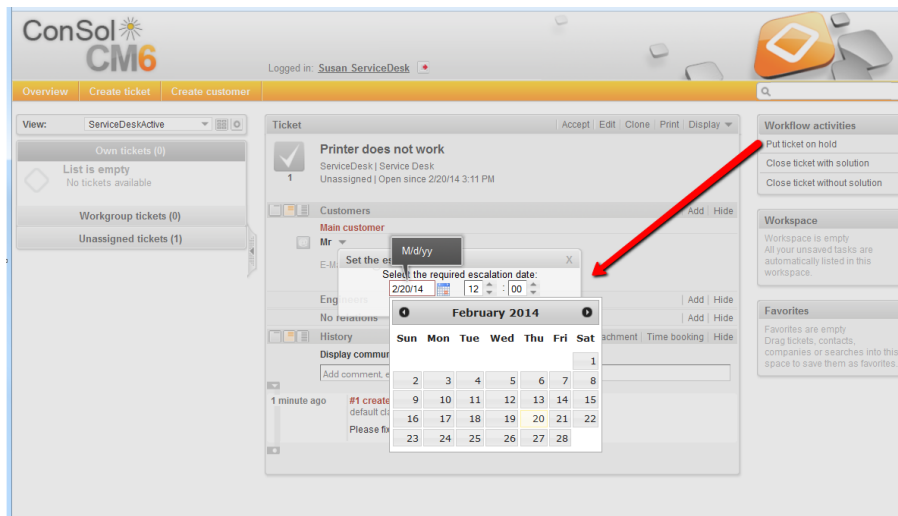


Figure 57: ConSol CM Web Client - Use case 1: Date picker

- **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 e-mail is sent to the team lead.

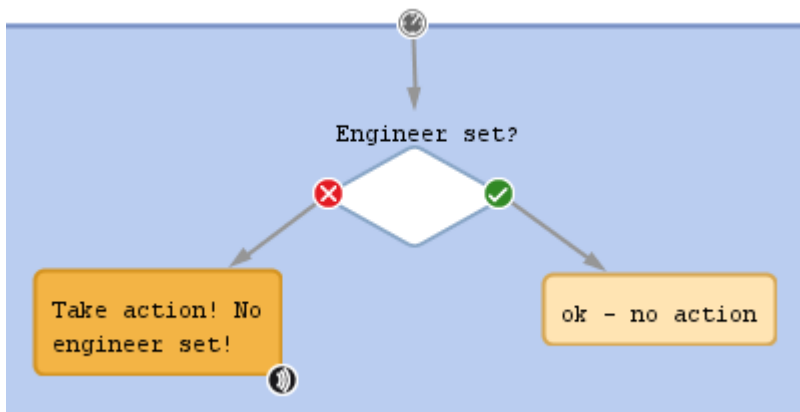


Figure 58: ConSol CM Process Designer - Use case 2: Workflow

The screenshot shows a 'Properties' dialog box for a time trigger. The properties are as follows:

Property	Value
name	EscalationTrigger_1
minutes	0
hours	4
days	0
use calendar	<input checked="" type="checkbox"/>
repeatable	<input type="checkbox"/>
script after timer	
script on timer start	
retry interval	default value

Figure 59: ConSol CM Process Designer - Use case 2: Properties editor for time trigger

The screenshot shows a list of unassigned tickets (5) in the ConSol CM Web Client. The first ticket is highlighted with a red box:

Ticket ID	Subject	Customer	Category	Date/Time
100265	Check invoice # 998877	Skywalker,Lea		7/31/14 1:15 PM
100260	Sell a printer to each special end customer	Skywalker,Luke	misc	5/5/14 3:07 PM
100251	Customer question regarding product documentation	Skywalker,Lea	CM/Phone	3/28/14 2:30 PM
100244	Application ABC not available	Skywalker,Lea	Web Client	3/19/14 1:10 PM
100243	Call back Customer from New York	Mia Skydiver	misc	3/19/14 12:40 PM

Figure 60: 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.

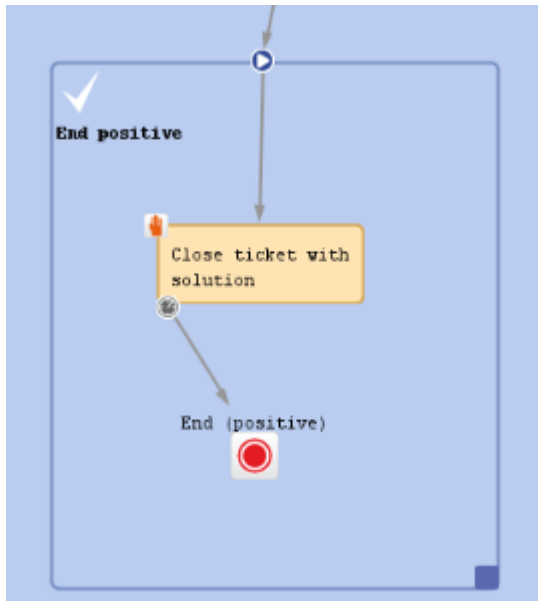


Figure 61: CM Process Designer - Use case 3: Workflow

The image shows a 'Properties' dialog box with a table of configuration options for a time trigger.

Properties	
name	__escalationTrigger
minutes	0
hours	0
days	5
use calendar	<input type="checkbox"/>
repeatable	<input type="checkbox"/>
script after timer	<input type="text"/>
script on timer start	<input type="text"/>
activate manually	<input type="checkbox"/>
retry interval	default value

Figure 62: ConSol CM Process Designer - Use case 3: Properties editor for time trigger

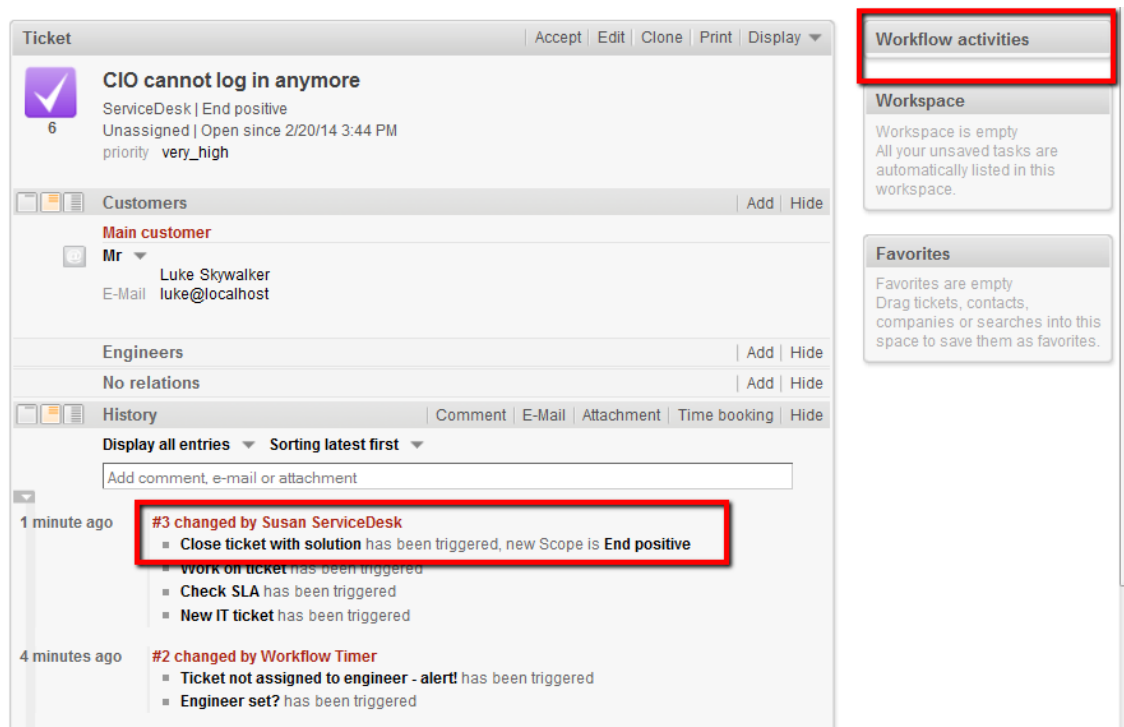


Figure 63: ConSol CM Web Client - Closed ticket

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](#).

Example 1: Set the Due Time of a Time Trigger Depending on the Queue

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*.

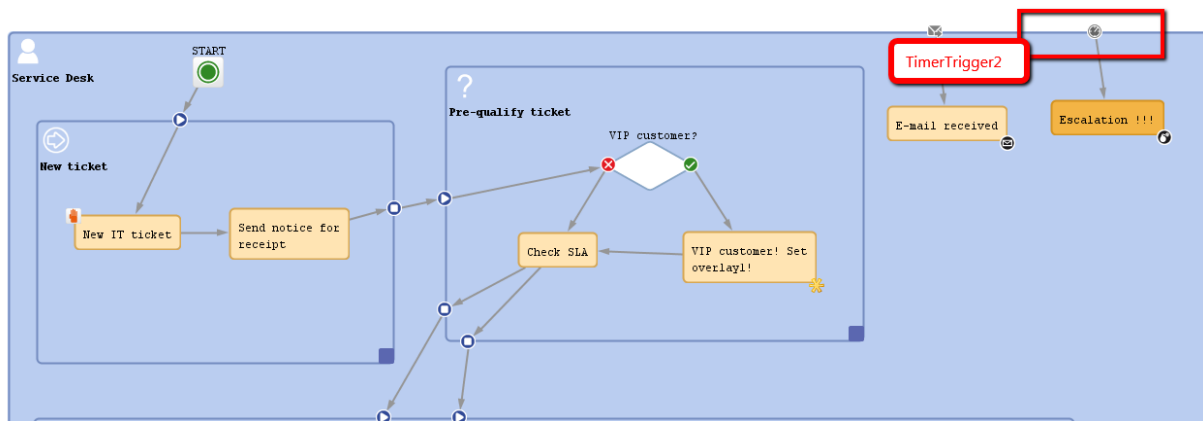


Figure 64: ConSol CM Process Designer - TimerTrigger in ServiceDesk workflow


```
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 7: 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*. That 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.

```
2014-03-28 15:32:42,156 INFO [w.DefaultWorkflowEventListener] Ticket's 100253 timer defaultScope/Service_Desk/TimeTrigger2 was activated with escalation time Fri Mar 28 15:32:42 CET 2014
2014-03-28 15:32:42,166 INFO [w.DefaultWorkflowEventListener] Ticket's 100253 timer defaultScope/Service_Desk/TimeTrigger2 was activated with escalation time Fri Mar 28 15:32:42 CET 2014
2014-03-28 15:32:58,866 INFO [ket.resource.PropertiesFactory] Loading properties files from vfszip:/usr/local/...
2.5.jar/cm/console/cmweb/client/components/ticket/list/filter/FilterSelectionPanel.properties with loader org.apache.wicket.resource.IsoPropertiesFilePropertiesLoader
2014-03-28 15:32:58,907 INFO [ket.resource.PropertiesFactory] Loading properties files from vfszip:/usr/local/...
2.5.jar/cm/console/cmweb/client/components/ticket/list/ticketListOptionsPanel.properties with loader org.apache.wicket.resource.IsoPropertiesFilePropertiesLoader
2014-03-28 15:33:03,345 INFO [orkflow.engine.job.JobExecutor] Removing timer after regular execution: workflow instance id: 249, timer name: defaultScope/Service_Desk/Ti
2014-03-28 15:33:03,345 INFO [engine.exe.event.TimerManager] Removing timer of WorkflowInstance id: 249
2014-03-28 15:33:03,353 INFO [w.DefaultWorkflowEventListener] Ticket's 100253 timer defaultScope/Service_Desk/TimeTrigger1 was deactivated
```

Figure 65: File *server.log* with calculated time trigger *DueTime*

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: Calculate an Escalation as Warning 2 Days before Desired End Date

```
def now = new Date()
def wunschTermin = ticket.get("helpdesk_standard", "date_test")
def twoWorkDays = -2*8*60*60*1000L

// calculate escalation date
def escalDate = BusinessCalendarUtil.getBusinessTime(wunschTermin, twoWorkDays,
  ticket.queue.calendar)
// calculate and set due time
trigger.setDueTime(escalDate.time - now.time)
```

Code example 8: Calculate and set time for *TimerTrigger* using *BusinessCalendar*

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 e-mail infrastructure. This makes it possible for the engineer to send manual e-mails and for the system to send automatic e-mails to customers and to engineers, as required in the respective process step. Obviously, ConSol CM has also to receive e-mails. This is done by retrieving e-mails 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*. Here, only the workflow interactions are explained.



Figure 66: ConSol CM Process Designer - Mail trigger

Mail Trigger at a Scope

When an e-mail is received which belongs to an existing and active (open) 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.



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 e-mail in a specific mailbox, is to attach this e-mail 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 E-Mail*.

All other actions, which should be executed when an e-mail 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 e-mail has been received ...

- the engineer of the ticket (the ticket owner) should also get an e-mail 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 e-mail.
- the sender and the subject of the e-mail are checked and parsed. If the e-mail 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 e-mail 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 e-mail is received.



Figure 67: 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

Use Case 1: Overlay for Ticket Icon

When an e-mail 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*.

That way, the ticket is marked with the overlay when the e-mail 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.

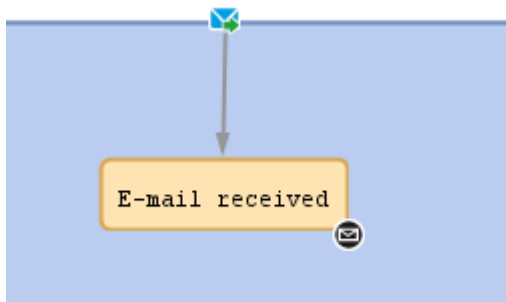


Figure 68: ConSol CM Process Designer - Use case 1: Scope with mail trigger

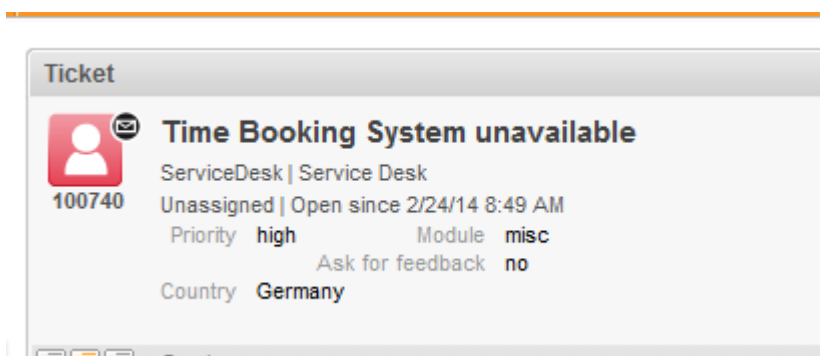


Figure 69: ConSol CM Web Client - Use case 1: Ticket with overlay icon

Use Case 2: Overlay for Ticket Icon and E-Mail Confirmation by Engineer

When an e-mail 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*. Additionally, the ticket should be transferred to a position where it waits until the engineer has confirmed that he/she has read the e-mail.

The mail trigger is attached to the scope and the overlay is attached to the adjacent automatic activity. The overlay range is *activity*. That way, the ticket is marked with the overlay when the e-mail 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 *Confirmed: e-mail read!* is offered wherever 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 e-mail than to use only the overlay. The engineer has to confirm the e-mail by executing the workflow activity *Confirmed: e-mail read!* explicitly. Within this workflow activity the value of the boolean field *mail_to_read* is set back to *false*. Now the ticket is ready to receive another e-mail 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 e-mail 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.

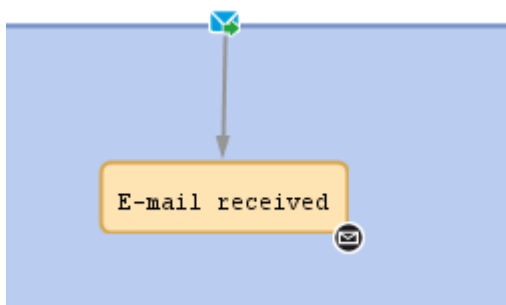


Figure 70: ConSol CM Process Designer - Use case 2: Scope with mail trigger

Properties	
Properties	
name	Email_received
label	E-mail received
description	
sort index	11
overlay	
overlay range	Activity
precondition	
script	Script is provided
activity type	Automatic
history visibility	default
disable auto update	<input type="checkbox"/>

Figure 71: ConSol CM Process Designer - Use case 2: Properties of activity "E-mail received"

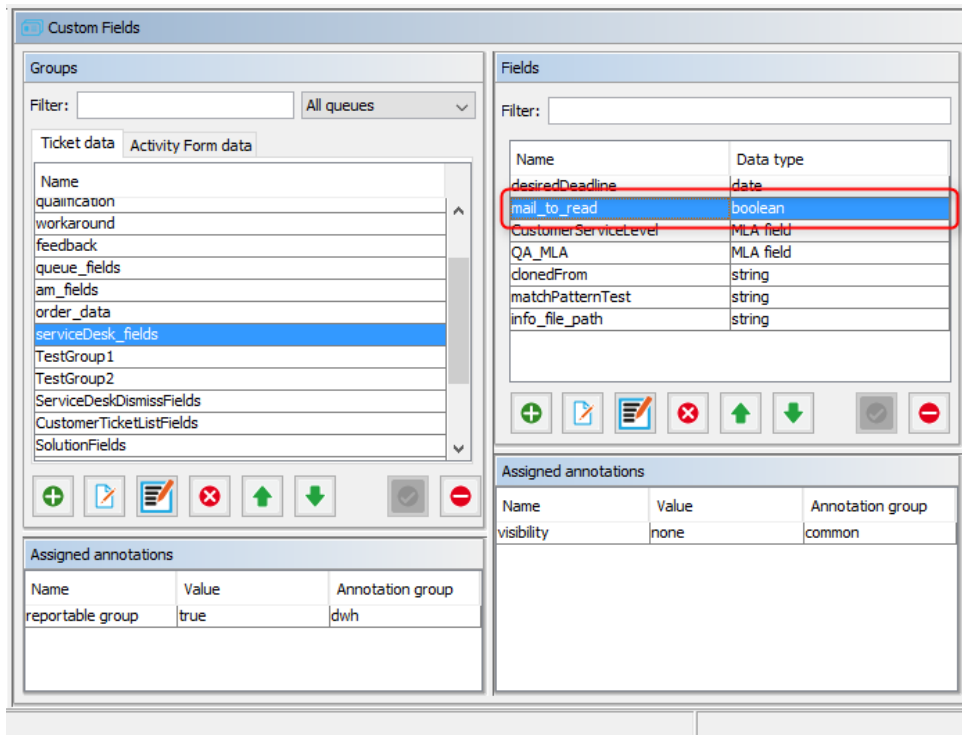


Figure 72: ConSol CM Admin Tool - Use case 2: New boolean field to register e-mail

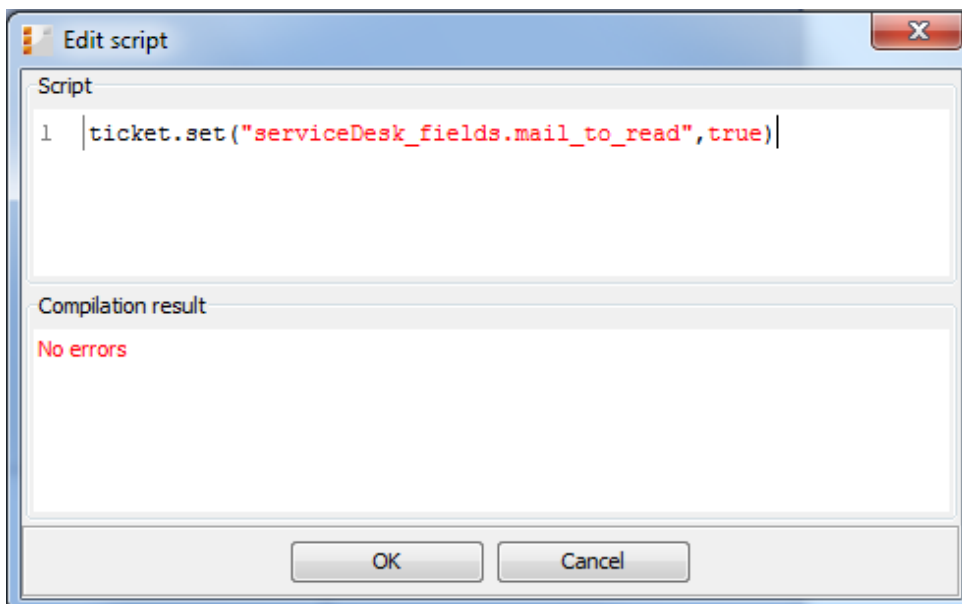


Figure 73: ConSol CM Process Designer - Use case 2: Script for activity "E-mail received"

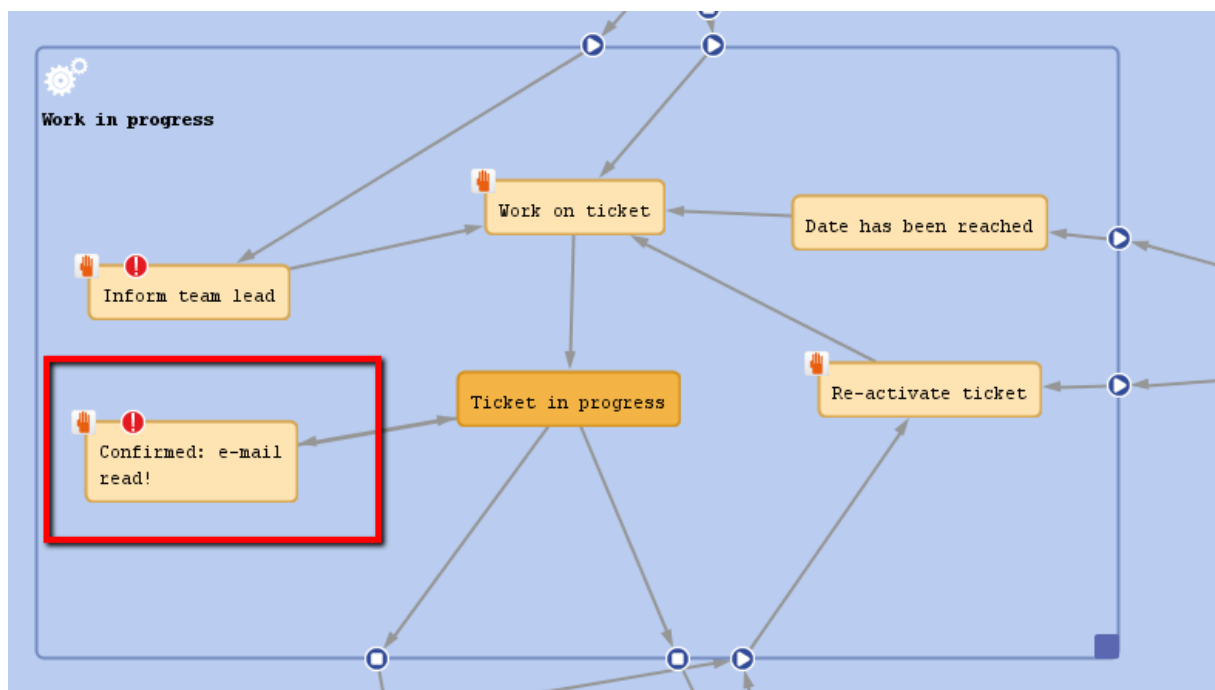


Figure 74: ConSol CM Process Designer - Use case 2: Activity for e-mail confirmation

Properties	
Properties	
name	Confirmed_email_read
label	Confirmed: e-mail read!
description	
sort index	12
overlay	
precondition	Script is provided
script	Script is provided
activity type	Manual
history visibility	default
disable auto update	<input type="checkbox"/>

Figure 75: ConSol CM Process Designer - Use case 2: Properties of activity "Confirmed: e-mail read!"

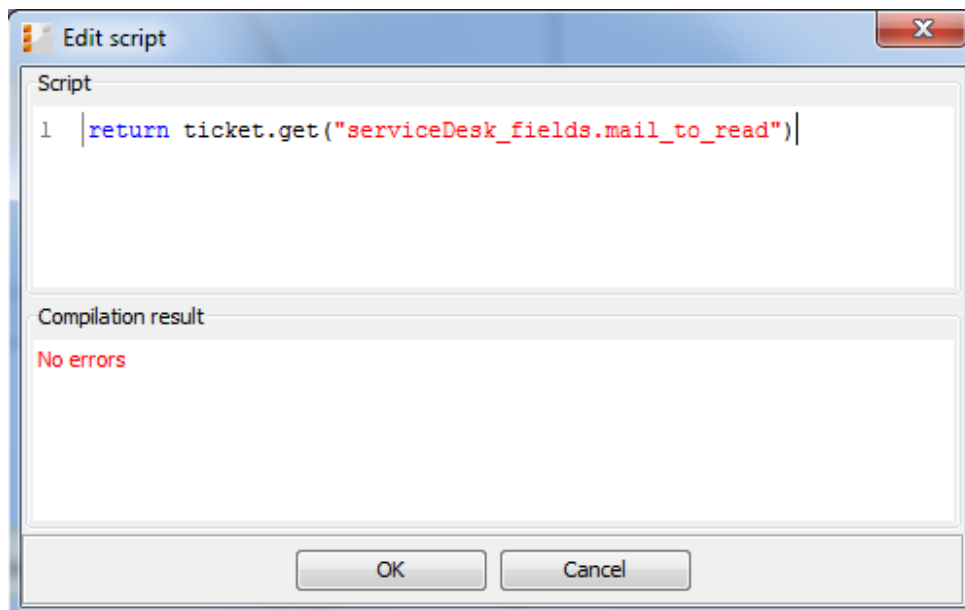


Figure 76: ConSol CM Process Designer - Use case 2: Precondition script for activity "Confirmed: e-mail read!"

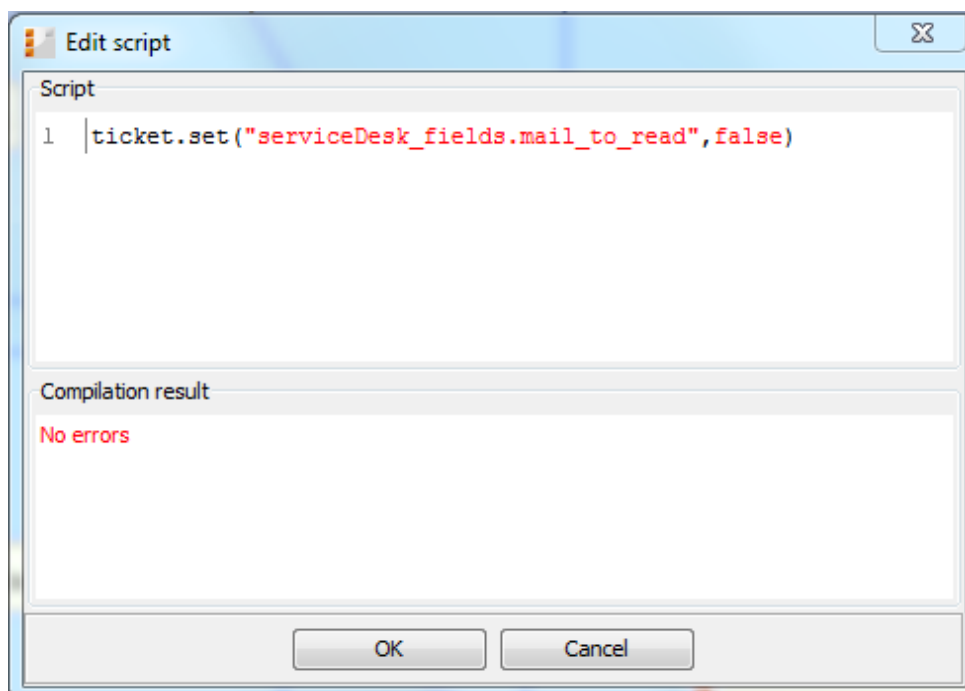


Figure 77: ConSol CM Process Designer - Use case 2: Script for activity "Confirmed: e-mail read!"

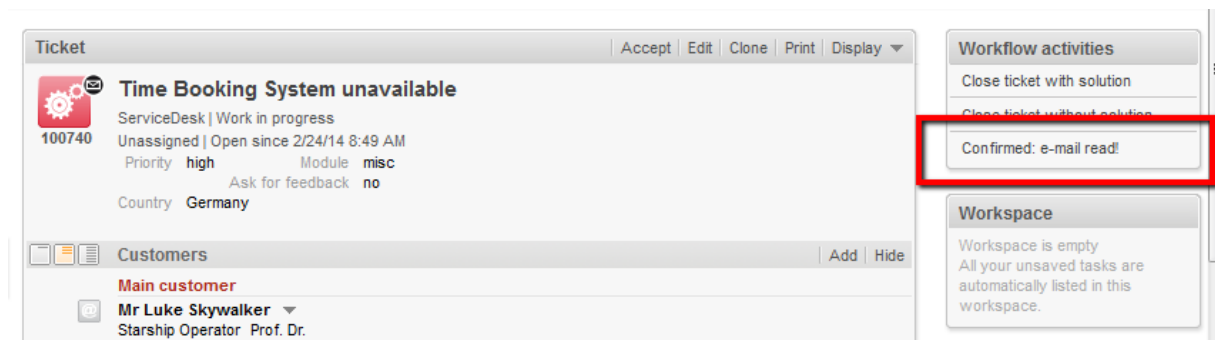


Figure 78: ConSol CM Web Client - Use case 2: Workflow activity "Confirmed: e-mail read!"

C.7.2.5 Process Logic with Mail Triggers

When an e-mail 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 e-mail comes in, the mail trigger for this scope fires **(2)** and, as a consequence, the ticket is moved to another scope **(3)**.

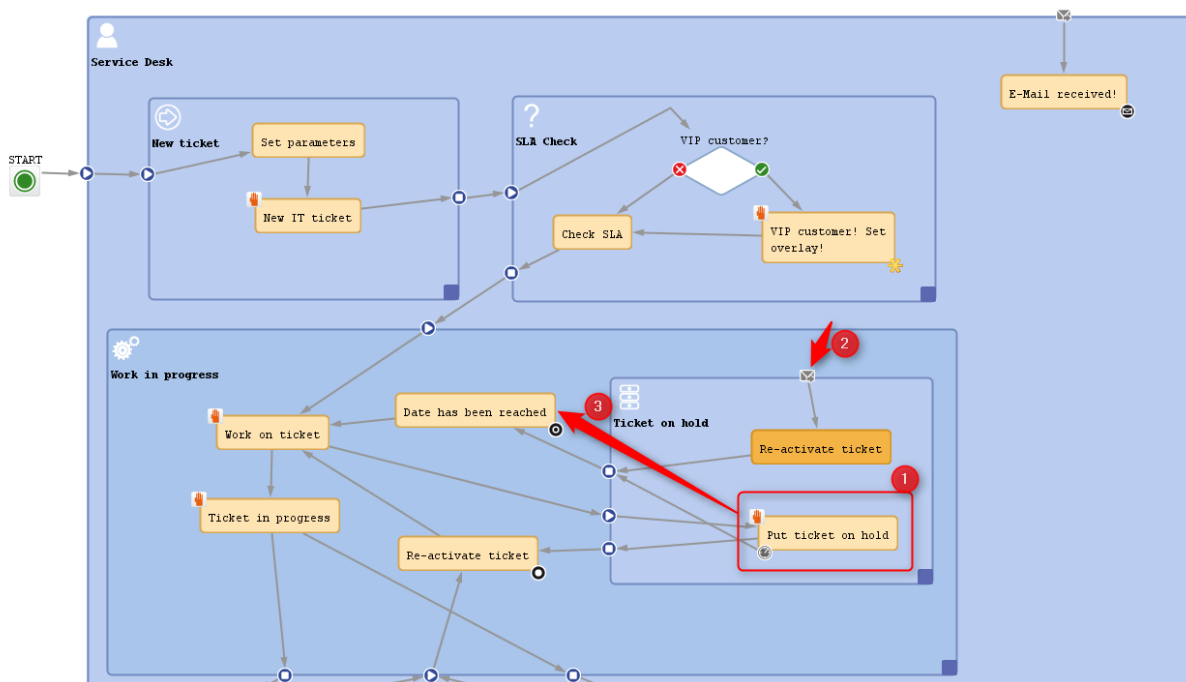


Figure 79: ConSol CM Process Designer - Example 1: Mail trigger of sub-scope active

Example 2:

The ticket is at position **(1)** in the *Work in progress* scope. When an e-mail 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)**.

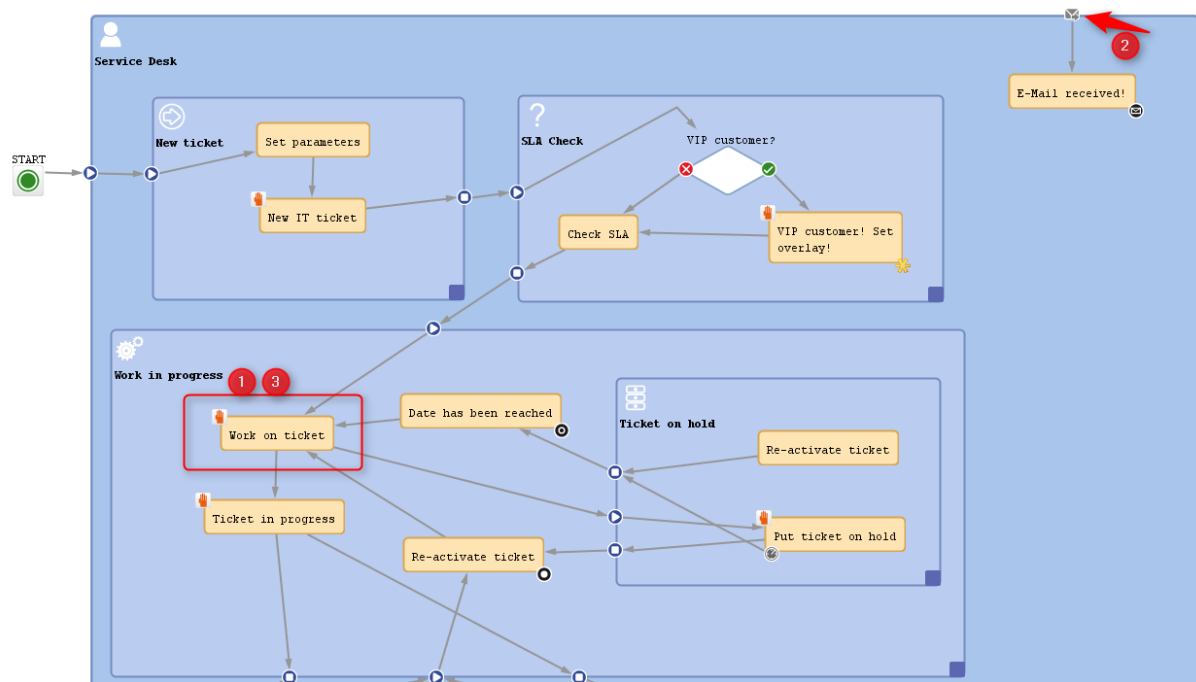


Figure 80: ConSol CM Process Designer - Example 2: Mail trigger of main scope active

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: 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 81: 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 e-mail)
- change of any Custom 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.

i 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

Properties	
queue	<input checked="" type="checkbox"/>
engineer	<input type="checkbox"/>
subject	<input type="checkbox"/>
comment	<input checked="" type="checkbox"/>
referenced engineer	<input type="checkbox"/>
related resource	<input type="checkbox"/>
time booking	<input type="checkbox"/>
custom field	serviceDesk_fields / CustomerServiceLevel
script after event	

Figure 82: 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 e-mail has been received for the ticket.
- An e-mail 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 Custom Field(s) 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 Custom Field definition (see *ConSol CM Administrator Manual*, section *Custom Field Administration*), you first have to select the Custom Field Group in the left pull-down menu and then you can choose one of the Custom Fields of this group in the right pull-down menu. You can select as many Custom Fields as you like.

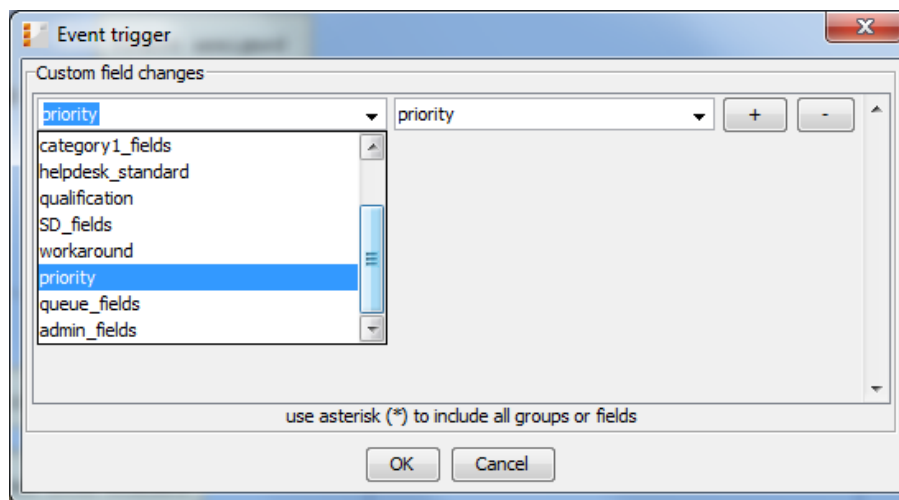


Figure 83: ConSol CM Process Designer - Property "Custom Field" of a business event trigger

- **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.

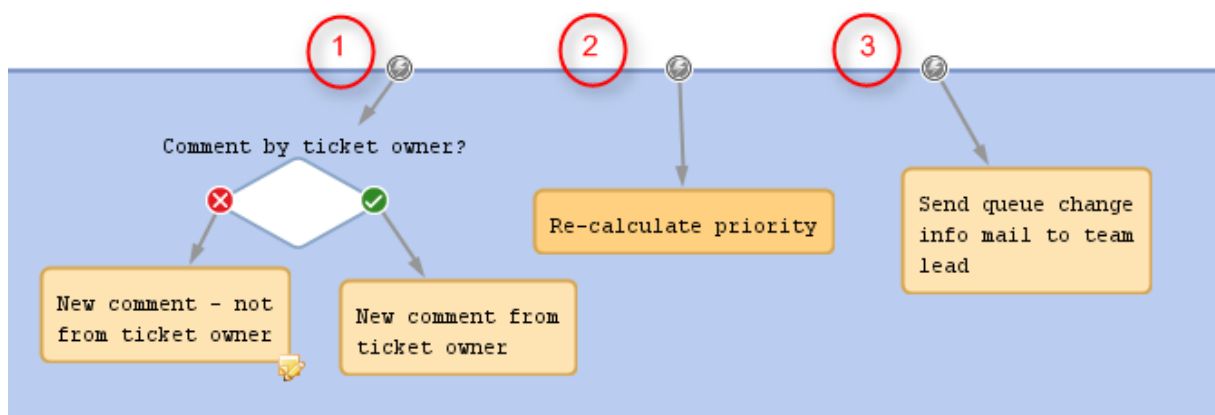


Figure 84: ConSol CM Process Designer - Firing order of business event triggers (1)

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 next figure, 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.

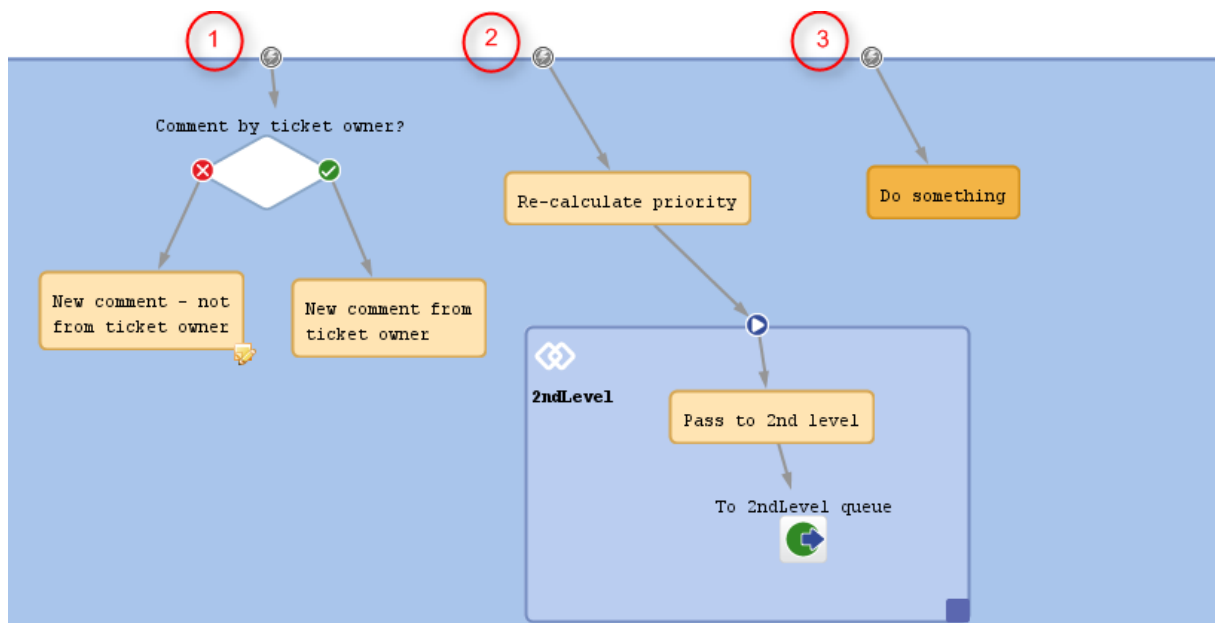


Figure 85: ConSol CM Process Designer - Firing order of business event triggers (2)

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

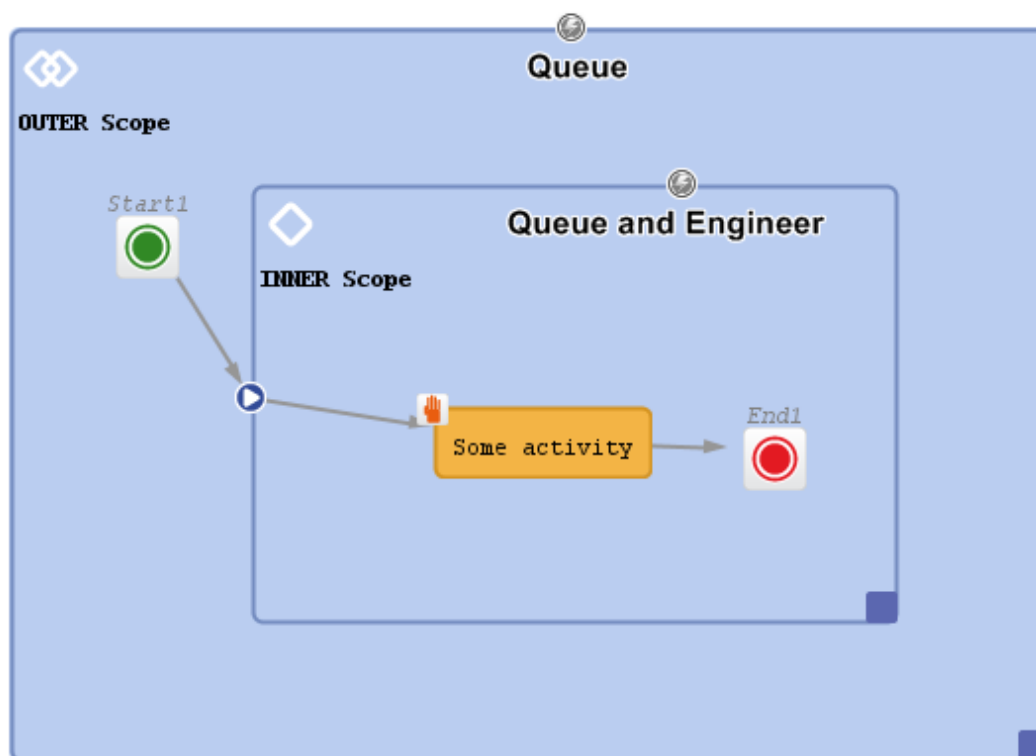


Figure 86: ConSol CM Process Designer - Hierarchical business event triggers (1)

Fired events:

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

Case 2

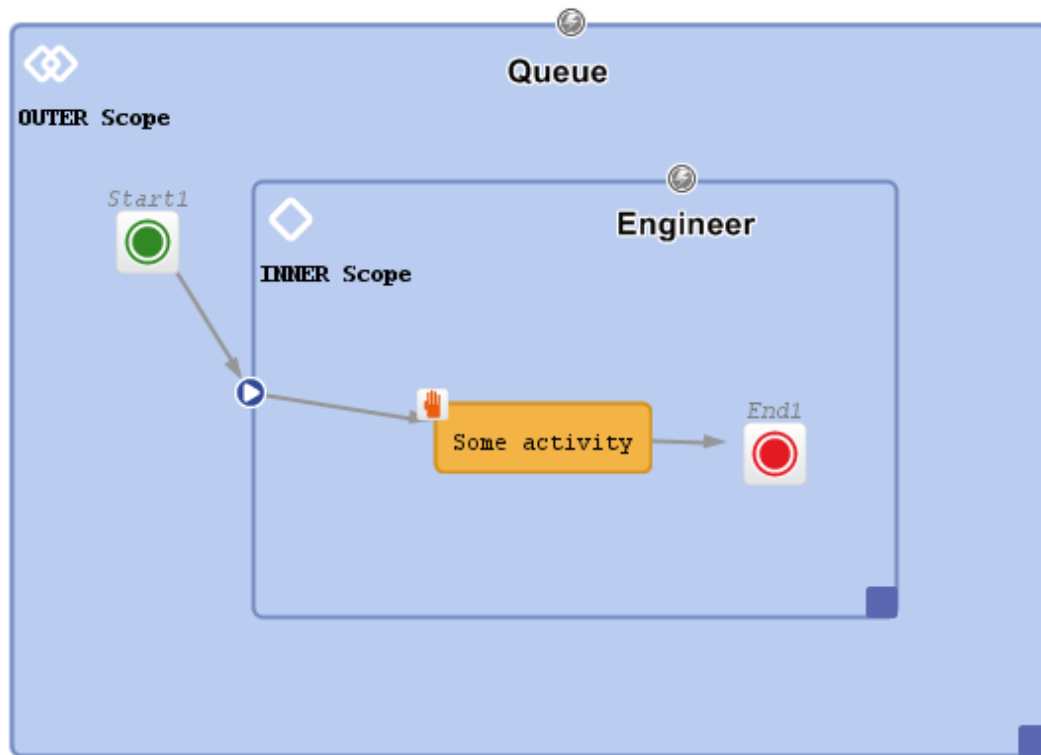


Figure 87: ConSol CM Process Designer - Hierarchical business event triggers (2)

Fired events:

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

Case 3

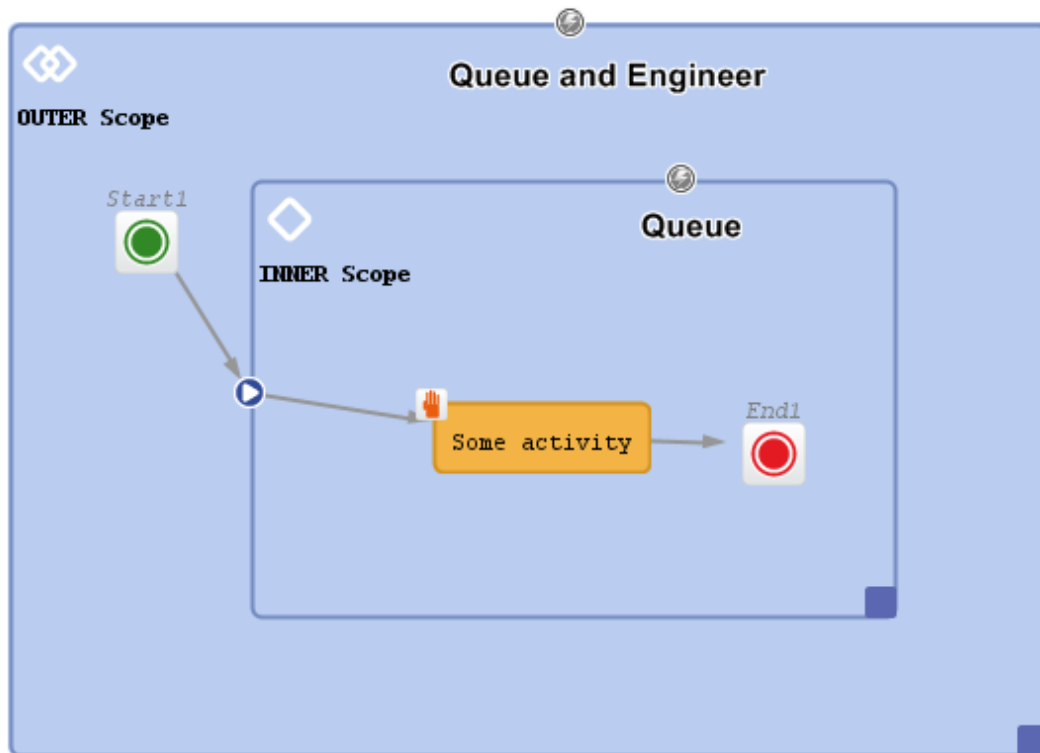


Figure 88: ConSol CM Process Designer - Hierarchical business event triggers (3)

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

Use Case 1: Check Engineer Comment

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. That 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 e-mail might have been received.

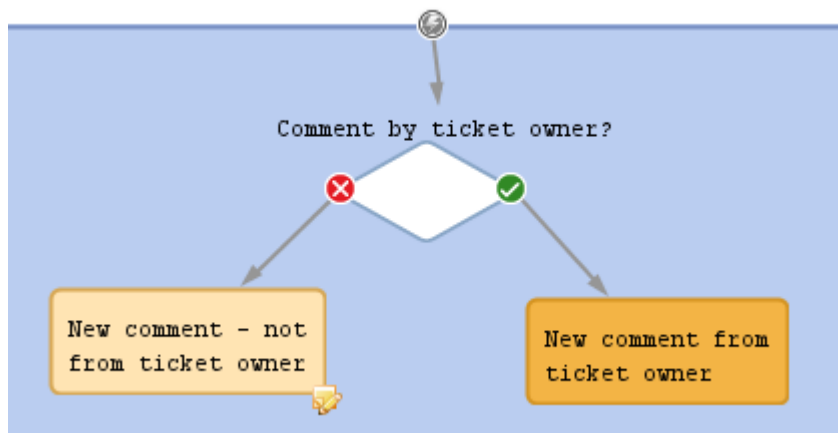


Figure 89: ConSol CM Process Designer - Business event trigger with following activities

Properties	
Properties	
queue	<input type="checkbox"/>
engineer	<input type="checkbox"/>
subject	<input type="checkbox"/>
comment	<input checked="" type="checkbox"/>
referenced engineer	<input type="checkbox"/>
related resource	<input type="checkbox"/>
time booking	<input type="checkbox"/>
custom field	...
script after event	...

Figure 90: ConSol CM Process Designer - Properties of a business event trigger (1)

```
return (workflowApi.getCurrentEngineer() == ticket.getEngineer())
```

Code example 9: Code of decision node script

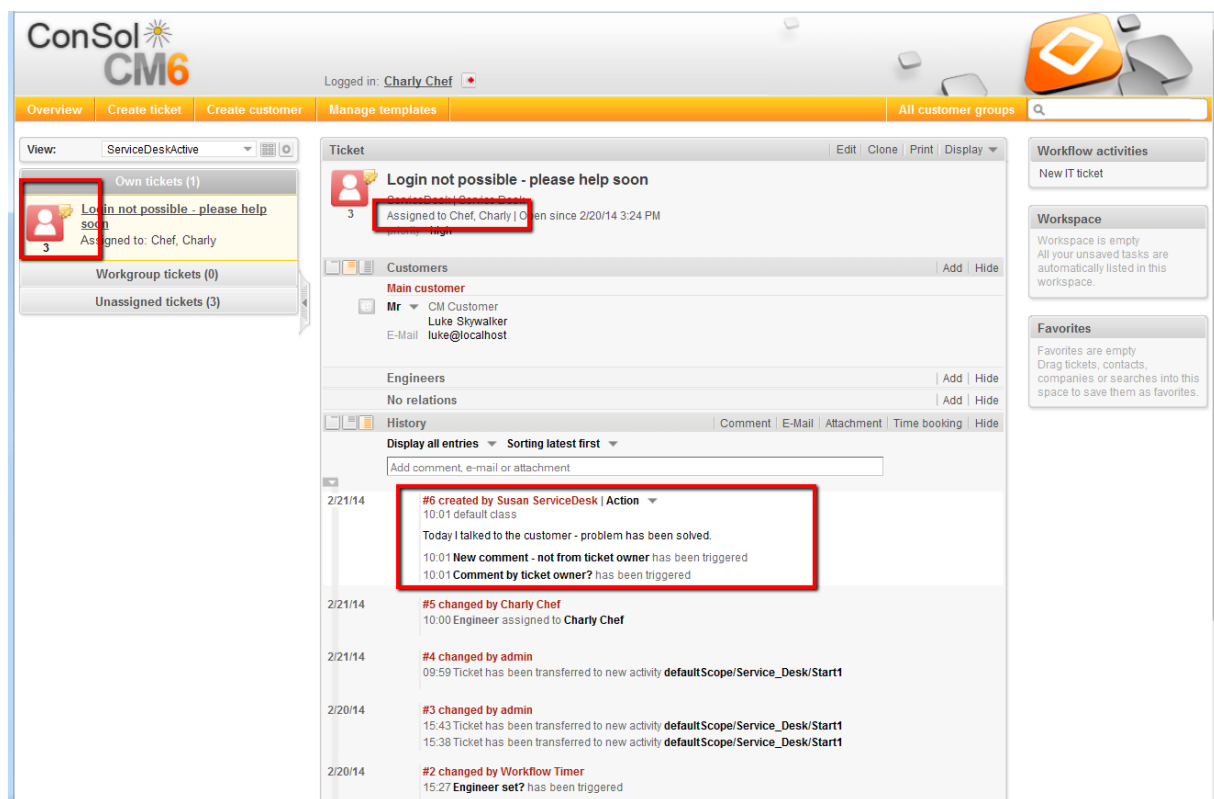


Figure 91: ConSol CM Web Client- Ticket marked with new overlay

Use Case 2: Re-Calculate the Ticket Priority if Impact and/or Urgency Have Been Changed

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 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.

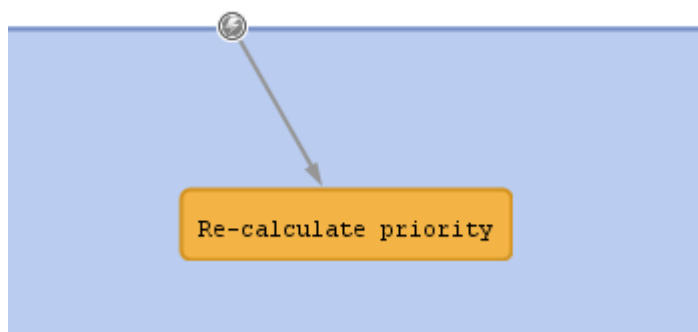


Figure 92: ConSol CM Process Designer - Business event trigger with following automatic activity

Properties	
queue	<input type="checkbox"/>
engineer	<input type="checkbox"/>
subject	<input type="checkbox"/>
comment	<input type="checkbox"/>
referenced engineer	<input type="checkbox"/>
related resource	<input type="checkbox"/>
time booking	<input type="checkbox"/>
custom field	multiple attributes ...
script after event	...

Figure 93: ConSol CM Process Designer - Properties of a business event trigger (2)

Event trigger

Custom field changes

service_desk_fields	urgency	+	-
service_desk_fields	impact	+	-

use asterisk (*) to include all groups or fields

OK Cancel

Figure 94: 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 10: Code of automatic activity script Re-calculate priority

Use Case 3: Continue Delivery Process When Shipment for the Order Has Arrived

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*.

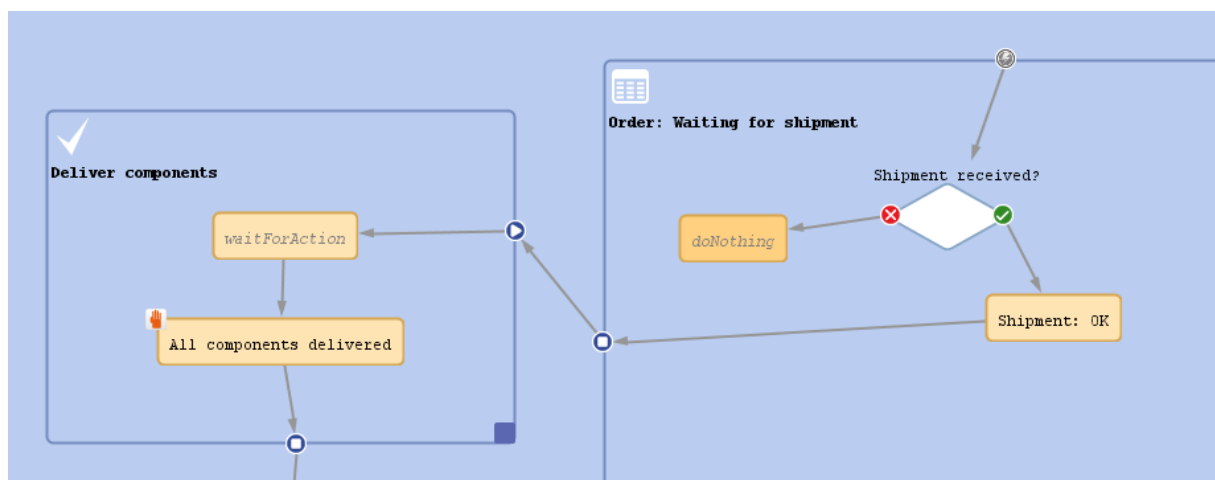


Figure 95: ConSol CM Process Designer - Workflow for use case 3

C.7.3.6 Best Practices: Using Business Event Triggers

See section [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](#)

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.

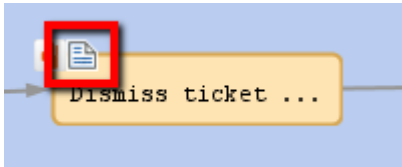


Figure 96: ConSol CM Process Designer - Activity Control Form (ACF)

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.

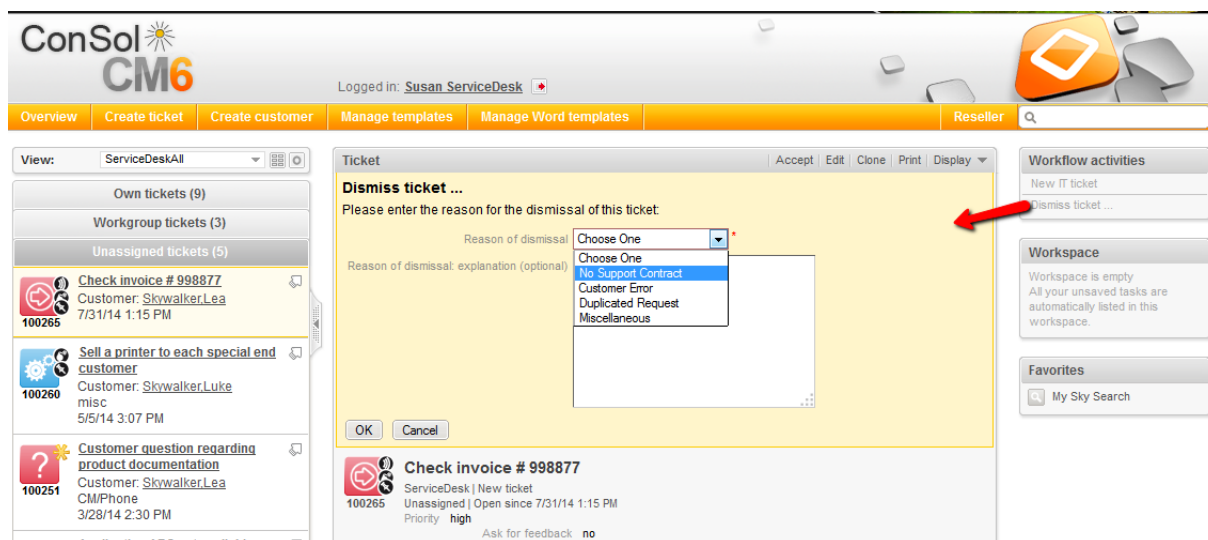


Figure 97: ConSol CM Web Client - Opened ACF

C.7.4.2 Adding an ACF to a Workflow

Variant A: Starting the ACF Definition Using 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 *Custom 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 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 Using 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 Custom 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 Custom 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*).

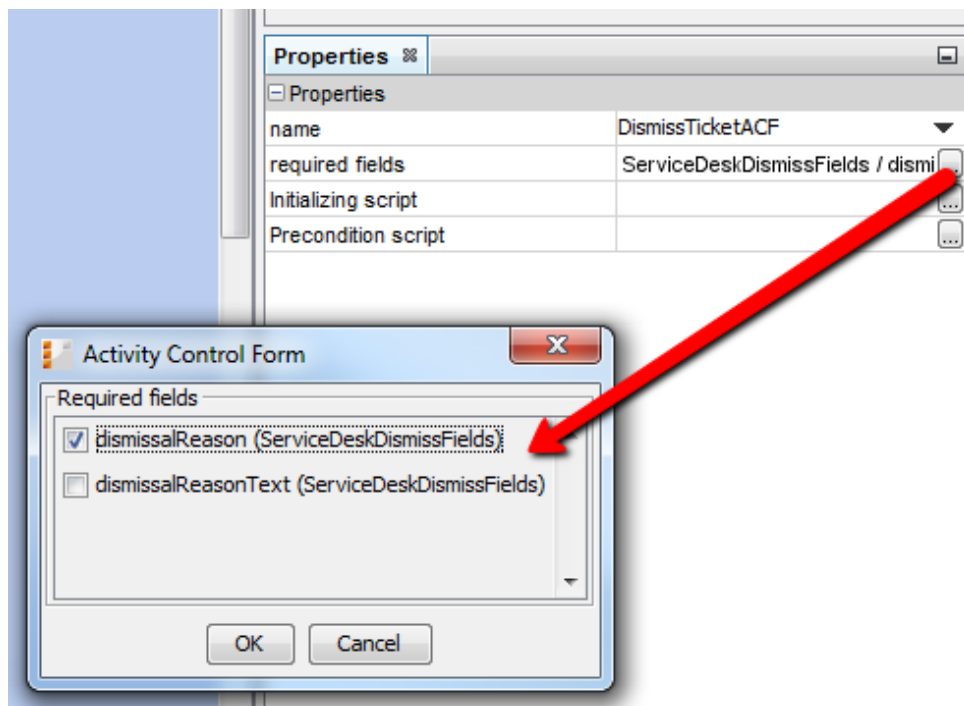


Figure 98: ConSol CM Process Designer - Properties of an ACF



All Custom Fields which are part of an ACF have to be available in the target queue, i.e. the respective Custom Field Group (CF group) has to be assigned to the queue where the workflow is used! There are two possibilities to achieve that:

1. You assign the CF group to a queue manually.
2. You just create the ACF and use it in a workflow. When you deploy the workflow, ConSol CM will automatically assign the required CF 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 Activity

ACFs are only possible for manual 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 Custom Field Group(s) which are *invisible* at the start of the process (Custom Field Group annotation *group-visibility = false*). In the step after the activity with the ACF, the Custom 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.

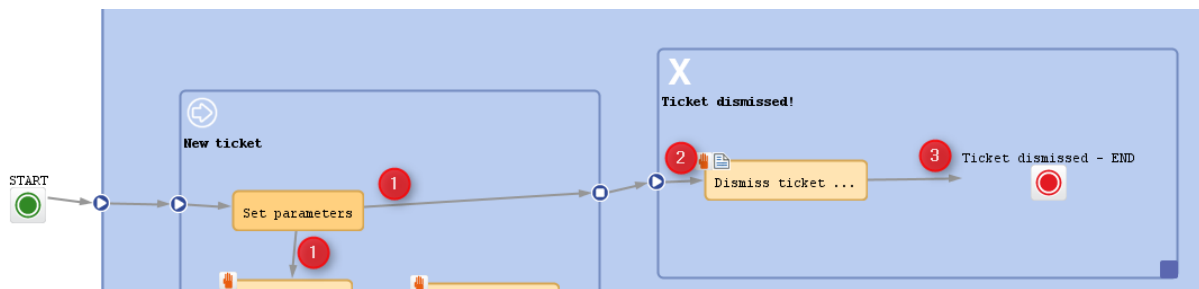


Figure 99: ConSol CM Process Designer - ACF process logic

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*.

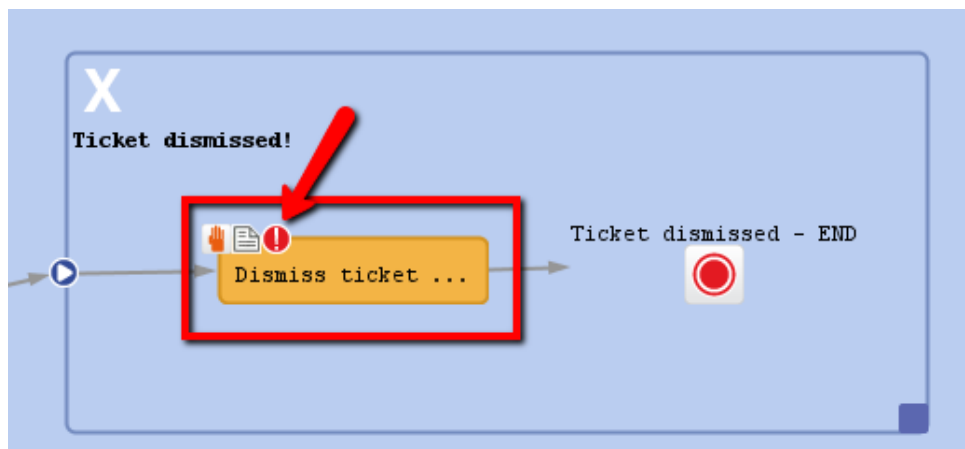


Figure 100: ConSol CM Process Designer - Manual activity with ACF and condition

C.7.4.5 Examples for the Use of ACFs

Use Case 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.

Name	Data type
dismissalReason	enum
dismissalReasonText	long string

Figure 101: ConSol CM Admin Tool - ACF definition

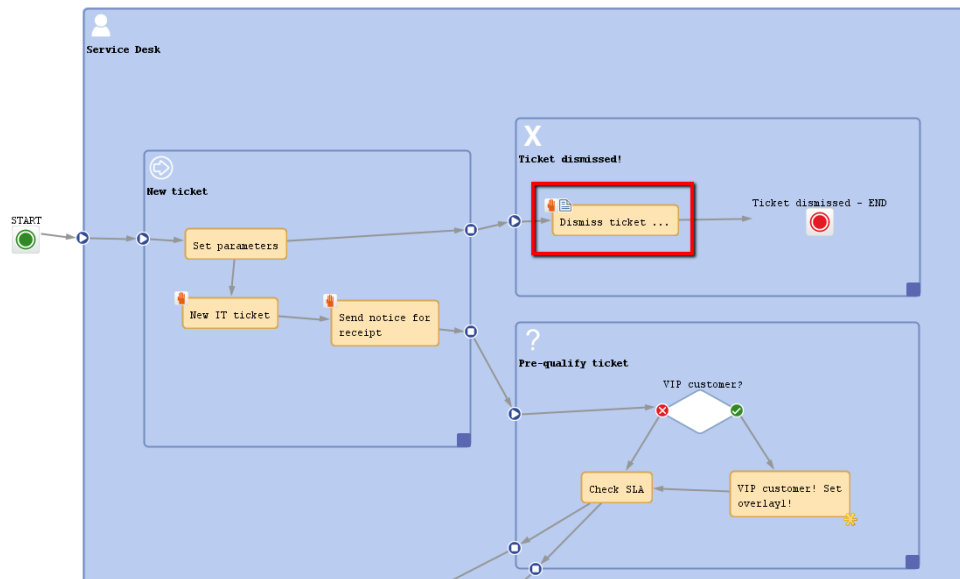


Figure 102: ConSol CM Process Designer - ACF in workflow

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

Use Case 2: Fill-in Sales Information when Bid is Created

When a sales representative selects the workflow activity *Create bid* in the Web Client GUI, 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.

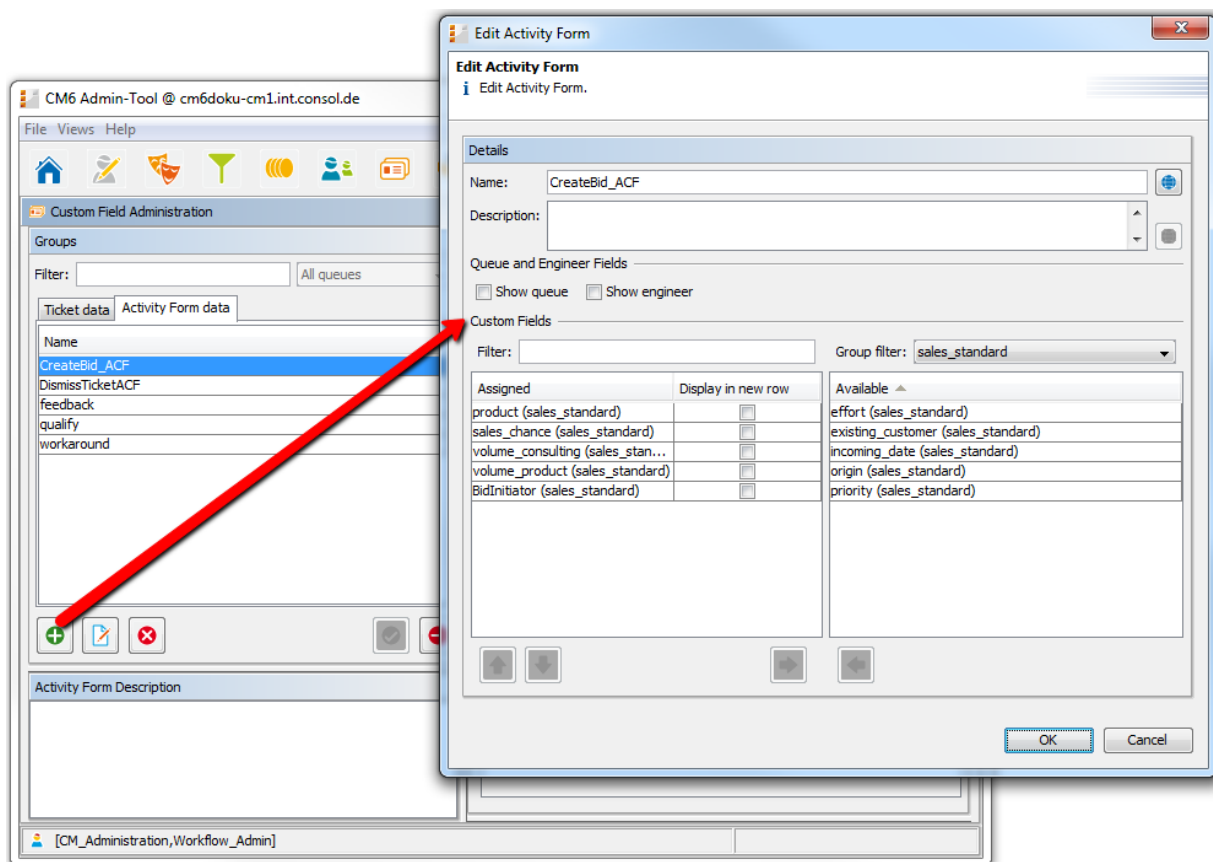


Figure 103: ConSol CM Admin Tool - ACF for Sales workflow

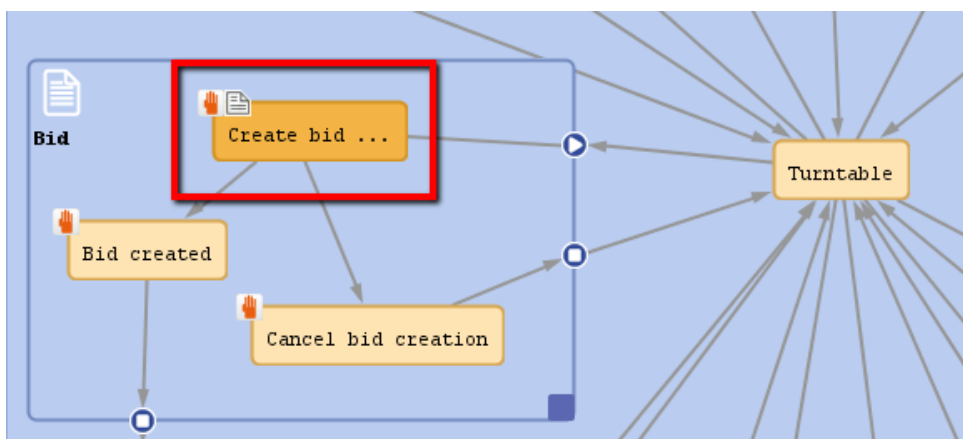


Figure 104: ConSol CM Process Designer - ACF in Sales workflow

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

Code example 11: Process Designer: Initializing script for Create bid ACF

The screenshot displays the 'Ticket' window in the ConSol CM Web Client. The main form is titled 'Create bid ...' and contains the following fields:

- Product:** A dropdown menu with 'Others' selected.
- Sales chance:** A dropdown menu with '50% - Product and budget' selected.
- Volume consulting:** An empty text input field.
- Volume product:** An empty text input field.
- Initiator of this bid:** A text input field containing 'Mr. Miller'.

Below the form are 'OK' and 'Cancel' buttons. At the bottom of the window, a notification bar shows a blue icon and the text: 'New Sales Opportunity in Bordeaux: Call asap (CM/Phone) Sales | Sales'. To the right of the main form is a 'Workflow activities' panel with a list of activities: 'Increase chance', 'Put on phone list', 'Make appointment', 'Create bid ...', 'Pickle', 'Win', and 'Loss'. Below this is a 'Workspace' panel.

Figure 105: ConSol CM Web Client - Sales process ACF

C.8 Jump-out and Jump-in Nodes

This chapter discusses the following:

C.8.1 Introduction	126
C.8.2 Jump-out Nodes	128
C.8.3 Jump-in Nodes	130

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.

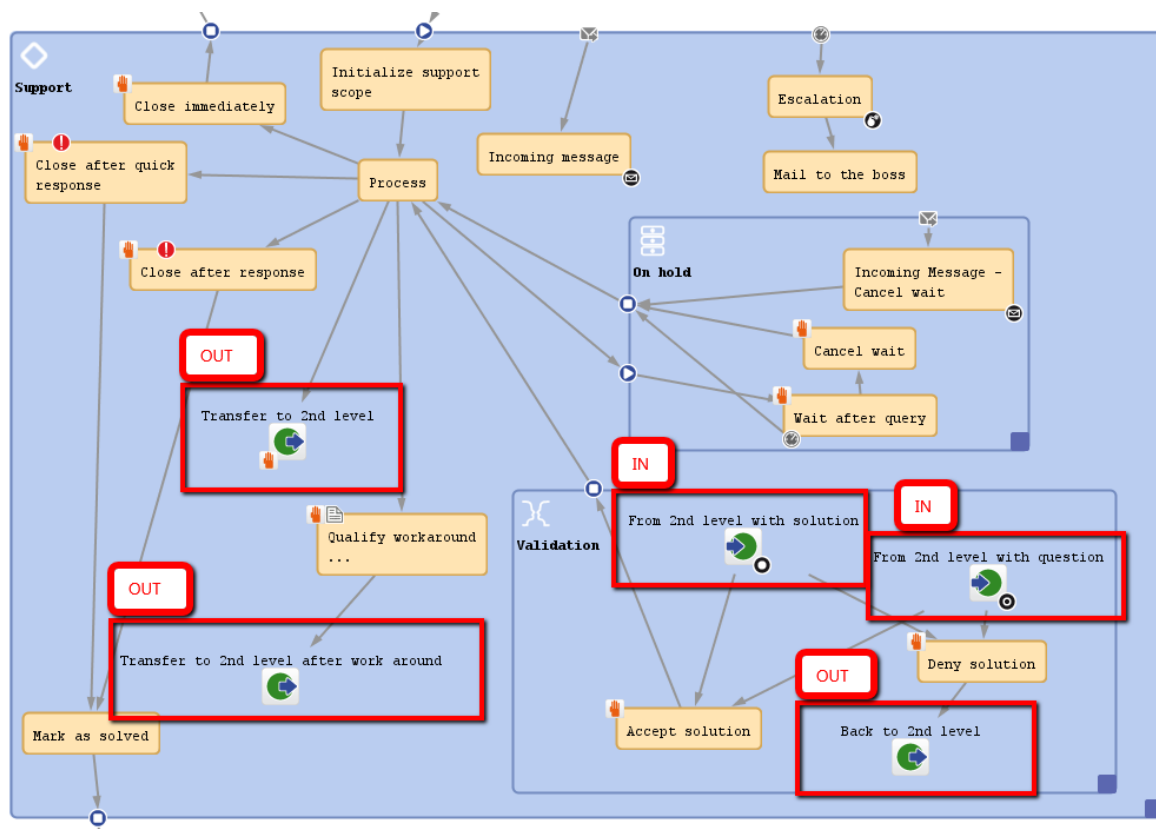


Figure 106: ConSol CM Process Designer - Example for jump-out and jump-in nodes

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

i 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.

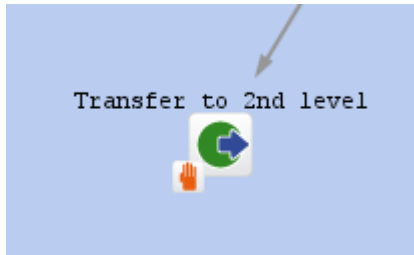



Figure 107: ConSol CM Process Designer - Jump-out node

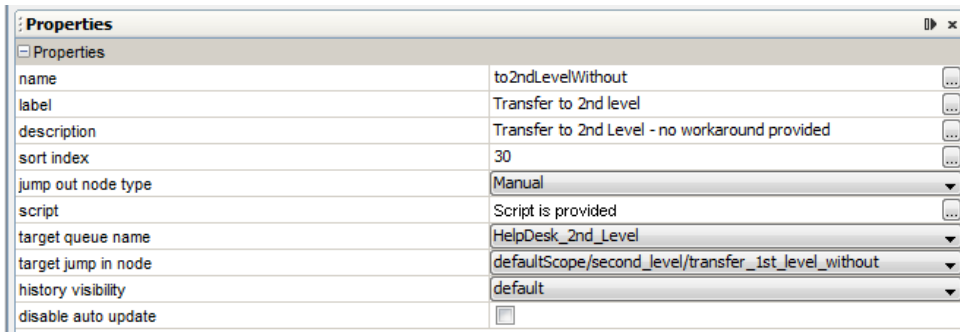
C.8.2.1 Properties of a Jump-out Node

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**
Selection. See section [history visibility](#)
- **disable auto update**
Boolean. See section [disable autoupdate](#)



Properties	
Properties	
name	to2ndLevelWithout
label	Transfer to 2nd level
description	Transfer to 2nd Level - no workaround provided
sort index	30
jump out node type	Manual
script	Script is provided
target queue name	HelpDesk_2nd_Level
target jump in node	defaultScope/second_level/transfer_1st_level_without
history visibility	default
disable auto update	<input type="checkbox"/>

Figure 108: 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 109: 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**
Selection. See section [history visibility](#).
- **disable auto update**
Boolean. See section [disable auto update](#).

The screenshot shows a 'Properties' dialog box with a table of properties. The 'overlay' property is highlighted with a yellow background. The 'overlay range' and 'history visibility' properties are dropdown menus. The 'disable auto update' property is a checkbox.


Properties	
[-] Properties	
name	from_2nd_level_solution
label	From 2nd level with solution
description	Back from 2nd Level, solution is provided
script	
overlay	
overlay range	Activity
history visibility	default
disable auto update	<input type="checkbox"/>

Figure 110: ConSol CM Process Designer - Jump-in node: Properties editor

D - Introduction to Workflow Programming

This chapter discusses the following:

D.1 Programming CM scripts	134
D.1.1 Some Short Examples of Java vs. Groovy-style Coding	134
D.2 CM API Documentation	135
D.3 CM Script Types in Workflows	136
D.4 Script Interactions	137
D.5 Scripts in ConSol CM in General	137
D.6 Process Logic	138
D.6.1 Introduction	139
D.6.2 Activities	139
D.6.3 Interrupts and Exceptions	141
D.6.4 Loops (Errors in Workflows)	143
D.6.5 Process Logic of Time Triggers	143
D.6.6 Process Logic of Business Event Triggers	143
D.7 Important Classes and Objects	144
D.7.1 Introduction	145
D.7.2 Important Objects	145
D.7.3 Convenience Classes and Methods	146
D.8 Working With Data Fields	150
D.8.1 Introduction to Data Fields	151
D.8.2 Data Types for Data Fields	152
D.8.3 Details about String Fields: Use Annotations to Fine-Tune Strings	155
D.8.4 Custom Fields for Ticket Data	157
D.8.5 Data Fields for Customer Data	170
D.8.6 Resource Data	179
D.8.7 Using Data Fields for (Invisible) Variables	180
D.9 Sending E-Mails	182
D.9.1 Introduction to Sending E-Mails	183
D.9.2 Important Methods	183

ConSol CM Process Designer Manual (Version 6.10.5.3) - D - Introduction to Workflow Programming	133
D.9.3 Examples	183
D.9.4 Writing E-Mails from Scripts when Engineer Representation Rules Apply	192
D.10 Working with Path Information	195
D.10.1 Introduction	196
D.10.2 Retrieve Path Information for a Workflow Element	197
D.10.3 Examples for the Use of Path Information	197
D.11 Working with Calendars and Times	198
D.11.1 Introduction	199
D.11.2 Calculating with Dates and Times without a CM Business Calendar	200
D.11.3 Calculating with Dates and Times Using a CM Business Calendar	200
D.12 Working with Object Relations	202
D.12.1 Working with Ticket Relations	203
D.12.2 Working with Customer Relations (Data Object Relations)	213
D.12.3 Working With Resource Relations	220
D.13 Working With Text Classes	223
D.13.1 Introduction	224
D.13.2 Example: Checking if a Solution Exists Before the Ticket can be Closed	225
D.13.3 Example: Adding a Text as Ticket Comment and Setting a Class of Text	227
D.14 Working With Attachments	230
D.14.1 Introduction	231
D.14.2 Example 1: Attaching all attachments of a ServiceDesk ticket to the child ticket	231
D.15 Searching for Tickets, Customers, and Resources Using the ConSol CM Workflow API	237
D.15.1 Introduction	238
D.15.2 Searching for Tickets	239
D.15.3 Searching for Units (Contacts and Companies)	244
D.15.4 Searching for Resources	246
D.16 Debug Information	249
D.16.1 Introduction	250
D.16.2 Using Statements for Debug Output	251

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.

D.1.1 Some Short Examples of 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 Getter Methods Can Often Be Omitted

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.	<code>String mysubject = ticket.getSubject()</code>	<code>def mysubject = ticket.subject</code>
Get the engineer of a ticket.	<code>Engineer myeng = ticket.getEngineer()</code>	<code>def myeng = ticket.engineer</code>
Get the main contact of a ticket.	<code>Unit mymaincontact = ticket.getMainContact()</code>	<code>def mymaincontact = ticket.mainContact</code>
Get the value of a certain Custom Field from a ticket.	<code>String myprio = ticket.get("helpdesk_fields", "prio")</code>	<code>def myprio = ticket.get("helpdesk_fields.prio")</code>

Use Case	Java-like syntax (extended version)	Groovy syntax (short version)
Get the unit type for the primary contact.	<pre>Unit mycustomer = workflowApi.getPrimaryContact() UnitDefinition myunitdef = mycustomer.getDefinition() UnitDefinitionType mydeftype = myunitdef.getType()</pre>	<pre>def mycustomer = workflowApi.primaryContact def myunitdef = mycustomer.definition def mydeftype = mycustomer.definition.type</pre>

Access to Custom 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 Custom Fields.

D.1.1.2 Setter Methods Can Often Be Omitted

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.	<code>ticket.setSubject("asd")</code>	<code>ticket.subject = "asd"</code>

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.

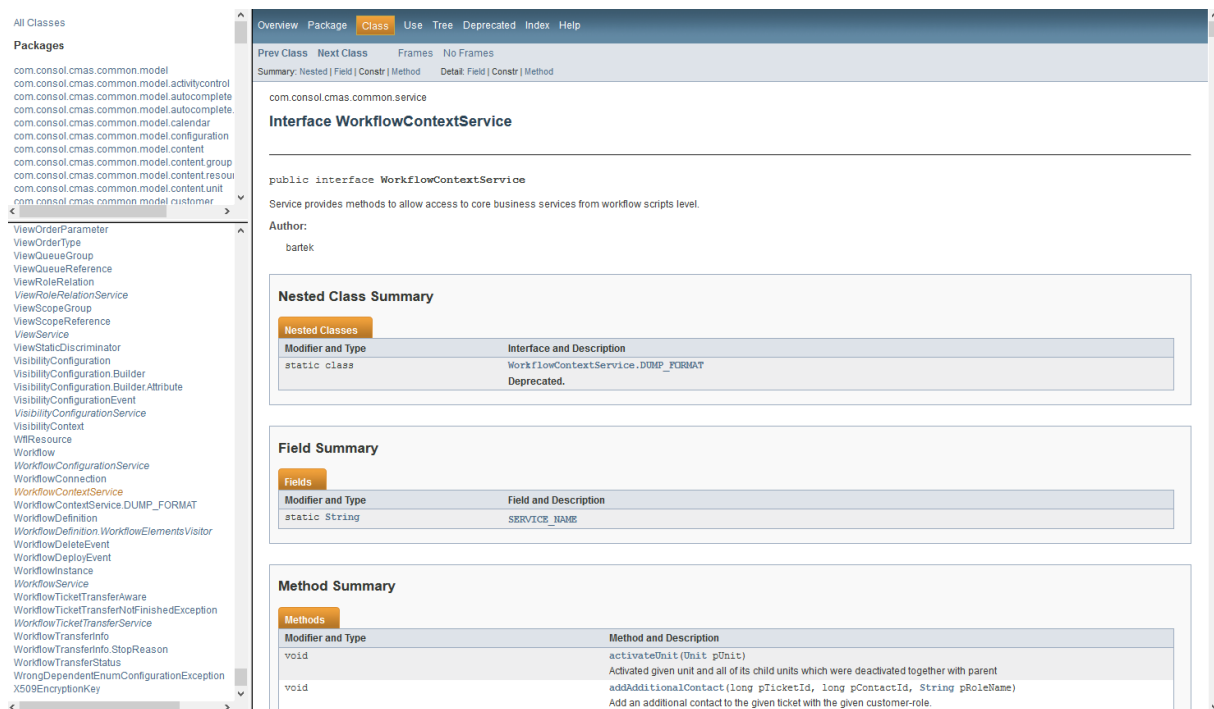


Figure 111: ConSol CM Groovy API Doc

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 [Store Some Workflow Scripts in the Admin Tool](#) 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*.

D.6 Process Logic

This chapter discusses the following:

D.6.1 Introduction	139
D.6.2 Activities	139
D.6.3 Interrupts and Exceptions	141
D.6.4 Loops (Errors in Workflows)	143
D.6.5 Process Logic of Time Triggers	143
D.6.6 Process Logic of Business Event Triggers	143

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 (number **(1)** and **(2)** in the following figure).
- If the next possible activity is an automatic activity, the activity is executed, i.e. the ticket passes through this activity (number **(3)** in the following figure).
- 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.

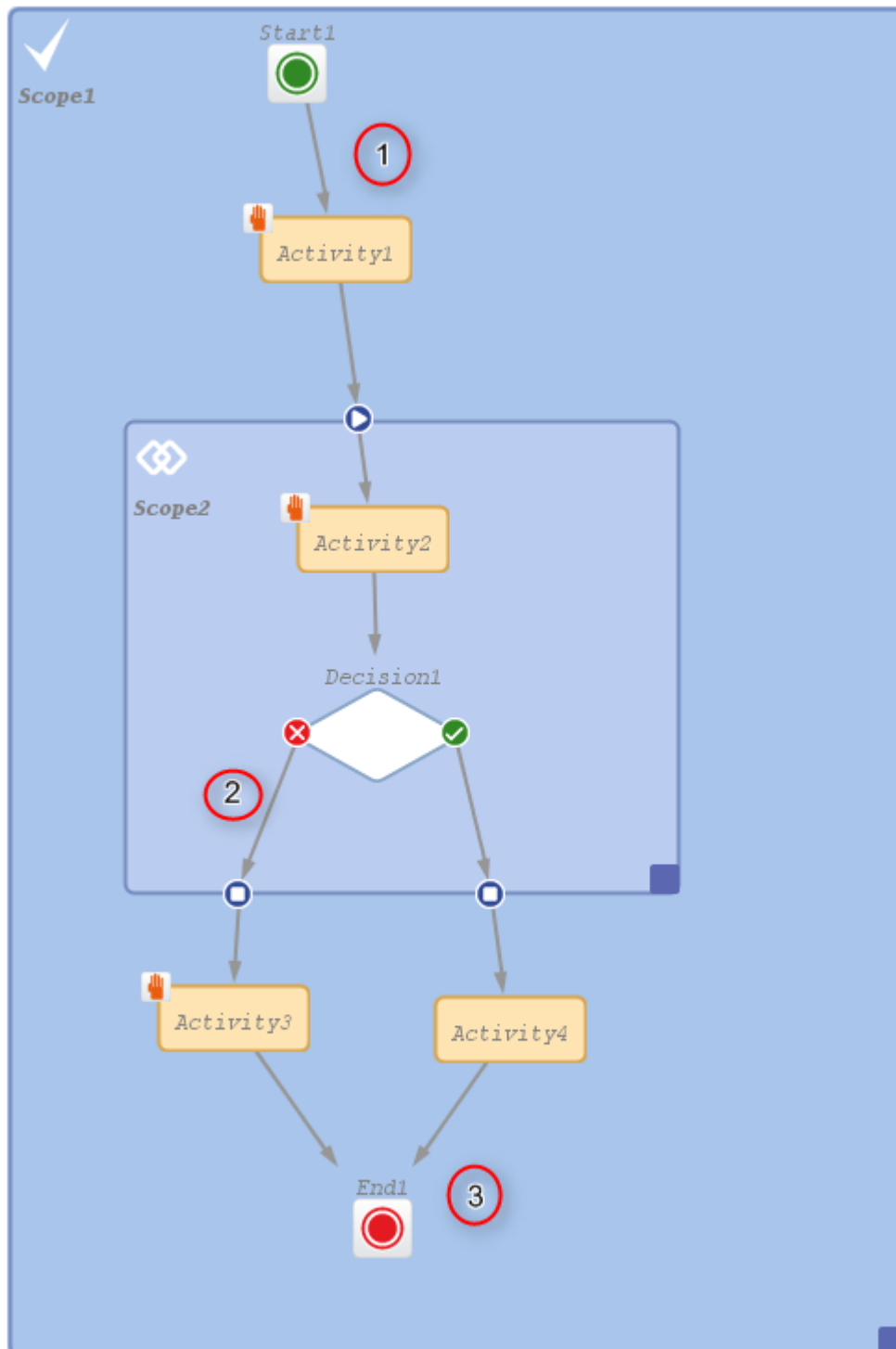


Figure 112: ConSol CM Process Designer - Process logic 1

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 e-mail 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 e-mail has been received (see figure below) or when an escalation time has been reached.

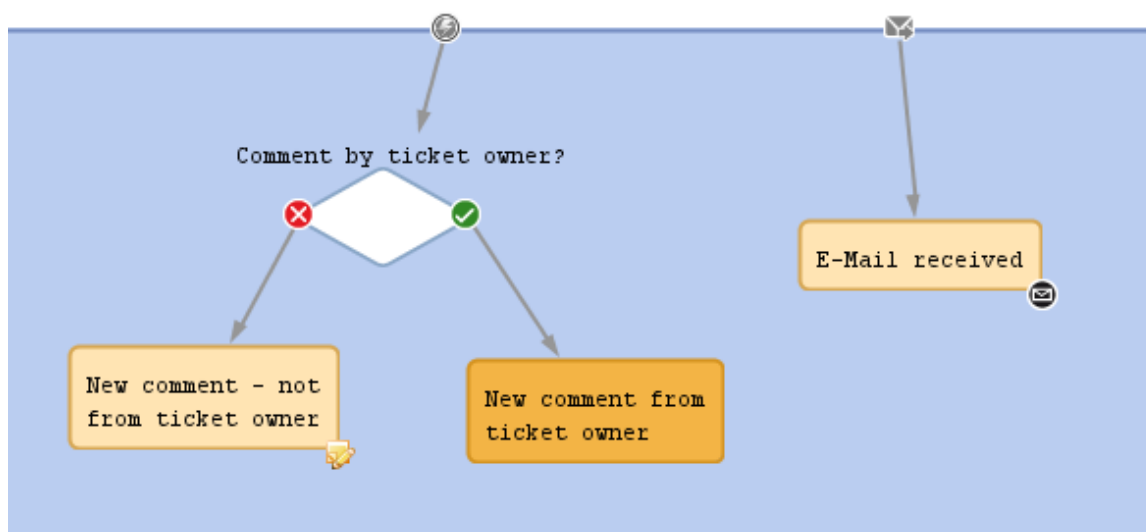


Figure 113: ConSol CM Process Designer - Two interrupts

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.

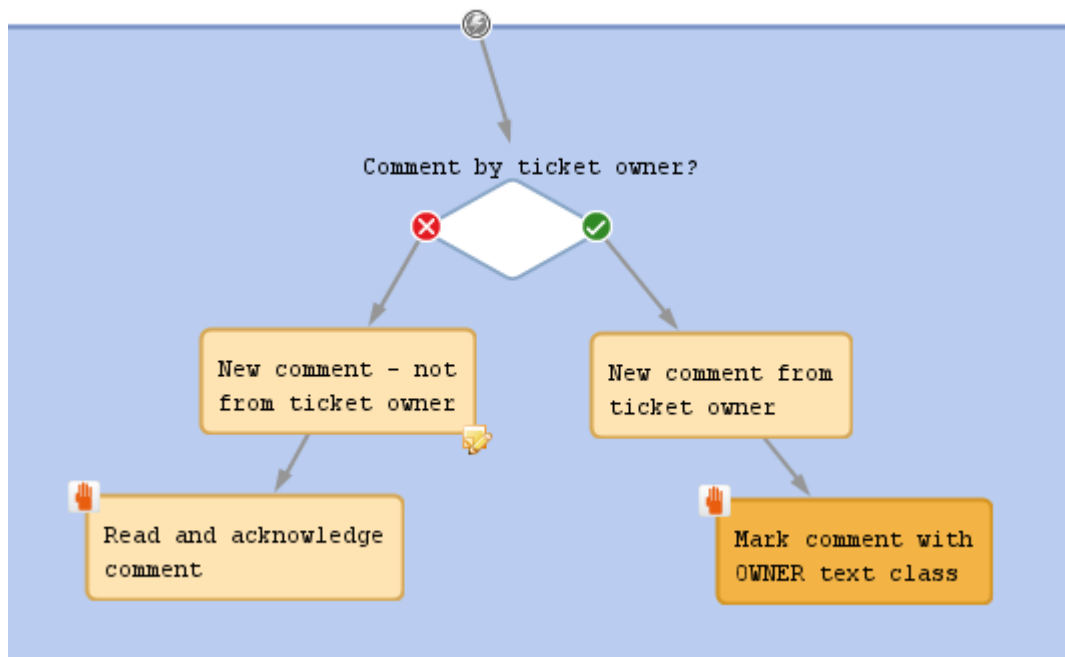


Figure 114: ConSol CM Process Designer - Exception

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.

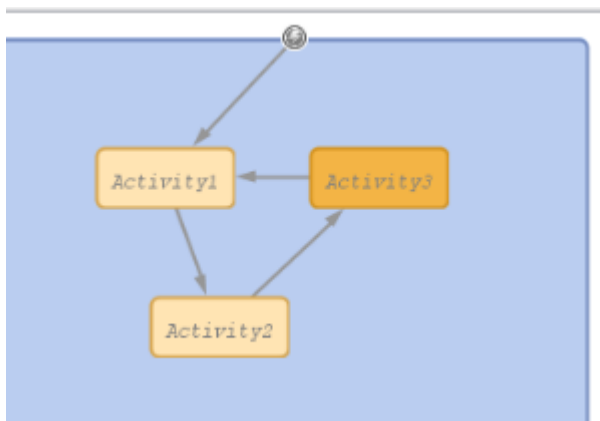


Figure 115: ConSol CM Process Designer - Loop in workflow

An error has occurred on 2/18/14 at 5:54 PM. Please contact your Administrator.

Figure 116: ConSol CM Web Client - Error message when loop was detected

```

2014-02-18 17:52:18,277 WARN [xkflowConfigurationServiceImpl] [admin-] Missing translation for process element, key: defaultScope.Service
Desk.Prequalify.ticket.VIP.customer.info, bundle locale: null
2014-02-18 17:54:11,997 ERROR [com.consol.cmas.workflow.engine.exe.WorkflowElementExecutorImpl] [defaultScope/Service_Desk/Activity1-defaultScope/Service_Desk/Activity2] Path: defaultScope/Service_Desk/Activity1-defaultScope/Service_Desk/Activity2 was already executed
at com.consol.cmas.workflow.engine.exe.WorkflowElementExecutorImpl.checkLoops(WorkflowElementExecutorImpl.java:142)
at com.consol.cmas.workflow.engine.exe.WorkflowElementExecutorImpl.doExecuteWithEvents(WorkflowElementExecutorImpl.java:87)
at com.consol.cmas.workflow.engine.exe.WorkflowElementExecutorImpl.executeInInterrupt(WorkflowElementExecutorImpl.java:78)
at sun.reflect.GeneratedMethodAccessor5197.invoke(Unknown Source)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
at java.lang.reflect.Method.invoke(Method.java:597)
at org.springframework.aop.support.AopUtils.invokeJoinpointUsingReflection(AopUtils.java:318)
at org.springframework.aop.framework.ReflectiveMethodInvocation.invokeJoinpoint(ReflectiveMethodInvocation.java:183)
at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:150)

```

Figure 117: Console - File server.log: Error message caused by workflow loop

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 [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	145
D.7.2 Important Objects	145
D.7.3 Convenience Classes and Methods	146

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 12: *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.sendEmail(contact_e, subj, text, replyto, null)
```

Code example 13: *Using workflowApi to send an e-mail (older variant. Current variant would mean using an object of class Mail)*

```
def curr_eng = workflowApi.getCurrentEngineer()
ticket.setEngineer(curr_eng)
```

Code example 14: *Using workflowApi to assign a ticket to current engineer*

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

Code example 15: *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 16: *Using workflowApi 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 17: *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 the 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 18: *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 19: *Using the ConfigurationService to retrieve base URL of the system*

D.7.3.2 Example 2: Using the 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 parameters 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 20: 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.7.3.3 Example 3: Using EnumService to Retrieve an Enum Value by Name

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

Code example 21: Using EnumService to retrieve an enum value by name

D.7.3.4 Example 4: Using the 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 22: *Using TicketService to find ticket of a view*

D.7.3.5 Example 5: Using the EngineerRoleRelationService to Send an E-Mail 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 23: *Using the EngineerRoleRelationService to send an e-mail to all engineers of a role*

D.8 Working With Data Fields

This chapter discusses the following:

D.8.1 Introduction to Data Fields	151
D.8.2 Data Types for Data Fields	152
D.8.3 Details about String Fields: Use Annotations to Fine-Tune Strings	155
D.8.4 Custom Fields for Ticket Data	157
D.8.5 Data Fields for Customer Data	170
D.8.6 Resource Data	179
D.8.7 Using Data Fields for (Invisible) Variables	180

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.

D.8.1.1 ConSol CM Version 6.9 and Higher

Starting with ConSol CM version 6.9.0, there are two types of data fields:

- **Custom Fields**
Used to define ticket data, managed in Custom Field Groups, as known from previous CM versions.
- **Data Object Group Fields**
Used to define customer data as part of the *FlexCDM*, the new customer data model. Managed in Data Object Groups.

The work with data fields of the new (version 6.9 and higher) customer data model (*FlexCDM*) is explained in detail in the *ConSol CM Administrator Manual - Customer Data Model 6.9: FlexCDM* and in the *ConSol CM Administrator Manual (Version 6.9)*, section *The CM Customer Data Model: FlexCDM*.

Rules for work with data fields CM 6.9 and higher:

When you work with Custom Fields and Data Object Group Fields, there are three main rules you have to keep in mind:

1. Custom Fields are always managed and referenced in Custom Field Groups, e.g. when you want to retrieve the value of a CF, you use `<CF GroupName>.<CF FieldName>`
2. Data Object Group Fields are always managed and referenced in Data Object Groups, e.g. when you want to retrieve the value of a Data Object Group Field, you use `<Data Object GroupName>:<Data Object Group FieldName>`
3. You always use the technical unique name to reference a Data Object Group Field or a Data Object Group, not the localized value.

D.8.1.2 ConSol CM Version 6.10 and Higher

In ConSol CM version 6.10, the module CM.Resource Pool was introduced. A detailed explanation of all objects in the new module is provided in the *ConSol CM Administrator Manual*, section *CM.Resource Pool*. Thus, in addition to the data fields already known from version 6.9, there is a new type of data fields: resource fields. In CM versions 6.10 and up, we work with the following data fields:

- **Custom Fields**
Used to define ticket data, managed in Custom Field Groups, as known from previous CM versions.


- **Data Object Group Fields**
Used to define customer data as part of the *FlexCDM*, the customer data model. Managed in Data Object Groups.
- **Resource fields**
Used to define resource data as part of the resource data model. Managed in resource fields groups.

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 Custom Fields, Data Object Group Fields and Resource Fields.

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

 If you work with scripts, either in CM workflows or in the Admin Tool, please note that the behavior of boolean fields which are represented as checkboxes, i.e. with annotation *boolean-type = checkbox* (default) is different depending on the CM version!

- **In CM versions prior to 6.9.4.0:**
If a boolean field has not been touched, its value is *false*. If it is checked, its value is *true*, and if it is unchecked again, its value is *false* again.
- **In version 6.9.4.0 and up:**
If a boolean field has not been touched, its value is *NULL*. If it is checked, its value is *true*, and if it is unchecked again, its value is *false*.

Fields which have already been filled with values in the database will not be changed during an update from a version prior to 6.9.4.0 to a version 6.9.4.0 and up.

Boolean fields represented as radio buttons (annotation *boolean-type = radio*) or drop-down menu (annotation *boolean-type = select*) are always shown and behave as described for versions 6.9.4.0 and up, i.e., with *NULL* value if untouched.

- **date**
Format and accuracy can be set by annotations.
- **enum**
For sorted lists. The engineer can choose one of the enum values in the Web Client. Enums and values have to be created previously within the Enum Administration in the Admin Tool. Select the desired *Enum type* and *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*.

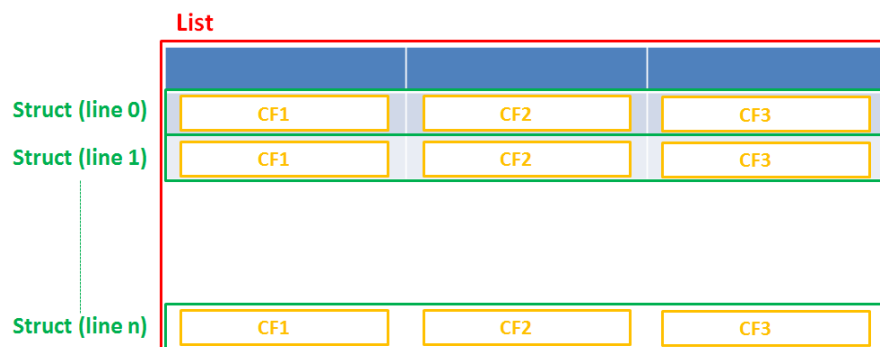


Figure 118: Scheme: List of Structs

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

List = array

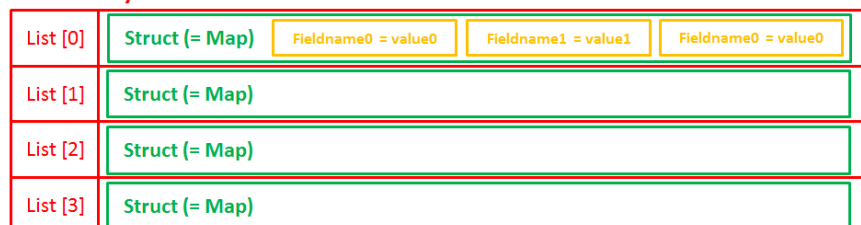


Figure 119: List of structs, technical principle

- **number**

For integer values.

- **fixed point number**

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

- **string**

For up to 4000 alphanumeric characters.



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

- **long string**

For large objects.



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

- **short string**

For up to 255 alphanumeric characters.



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

- **contact data reference**

Special data type used internally for referencing the contacts associated with a ticket. In FlexCDM (i.e., starting with CM version 6.9.0), this data type is no longer displayed but only used internally in the CM system.

- **MLA field**

This data type is used for fields that contain hierarchical lists with a tree structure called *MLA* (Multi Level Attributes). The name of the field is the name of the new MLA that has to be defined within the MLA Administration 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 *"/loc-alhost/"*) 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*

... 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 **CM.Track login**?

Value for annotation *username*: **true**

Will be used for authentication against CM.Track server. Only for Data Object Group 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 Data Object Group Fields in a contact object.

... define a field for the **valid e-mail addresses**?

Value for annotation *email*: **true**

The field may only contain valid e-mail addresses. Input will be validated according to standard e-mail 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*.

D.8.4 Custom Fields for Ticket Data

In the Admin Tool, the Custom Fields for ticket data are defined in the *Custom Field Administration*: navigation group *Tickets*, navigation item *Custom Fields*.

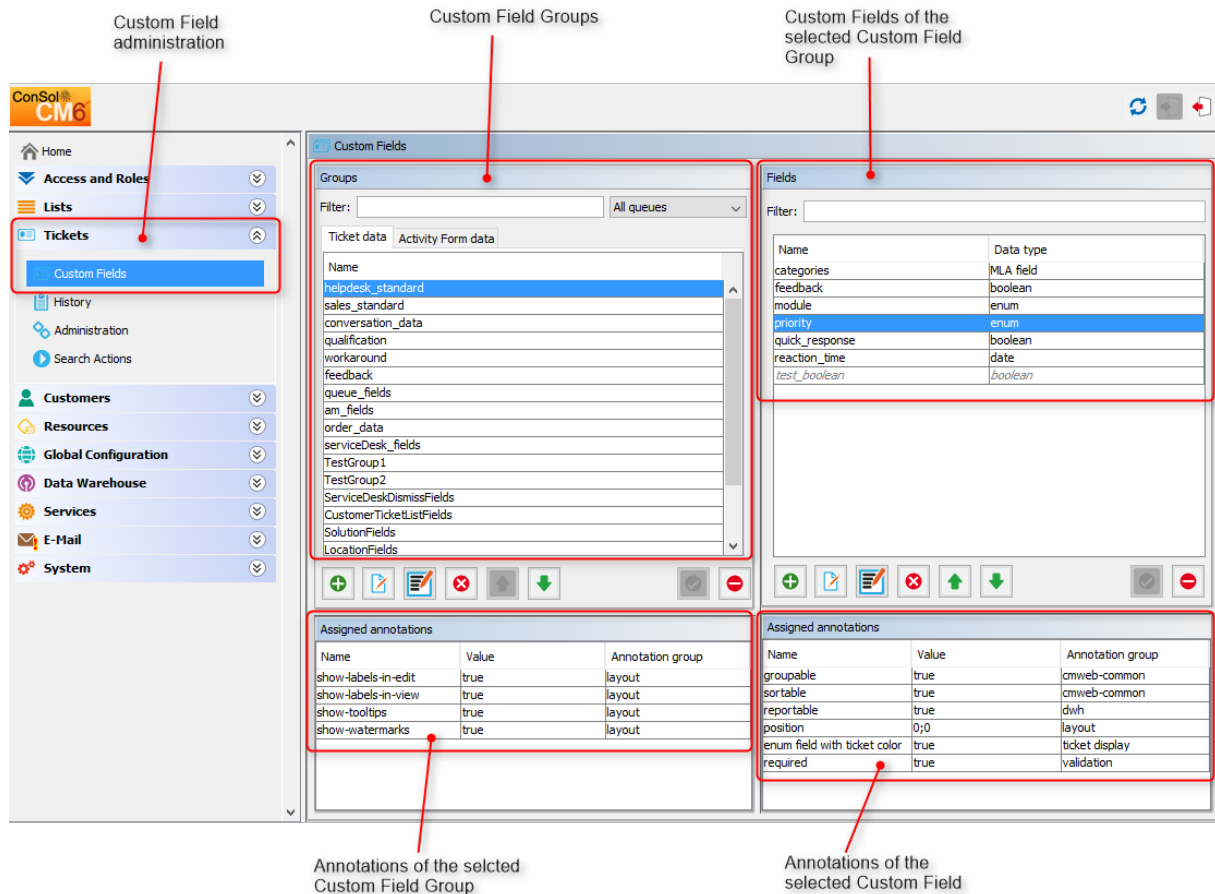


Figure 120: ConSol CM Admin Tool: Custom Field administration for ticket data (CM version 6.10)

D.8.4.1 Most Important Methods for Access to Ticket Custom Fields

Three methods are of major importance for programming CF access in CM scripts. They all are methods of the class *Ticket*:

- **Ticket.get()**
 - For retrieving data from a CF.
- **Ticket.set()**
 - For setting data in an already existing CF.
- **Ticket.add()**
 - For calculating with a value within a CF, 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 Custom Field Values for Ticket Data

To retrieve data from a Custom Field in a script, you have to reference it by using the technical names of the Custom Field Group and of the Custom Field. The method which has to be used can vary depending on the data type of the CF.

Simple Data Types

The following examples refer to the Custom Fields in the figure above. The method which should be used (because it is the most convenient way) is:

```
ticket.get("<Group_name>.<CF_name>")
```



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
println 'Display enum from helpdesk CF group ... '

def myfield = ticket.get("helpdesk_standard.priority")
println 'Class of helpdesk_standard.priority field is ' + myfield.getClass()
println 'Value of helpdesk_standard.priority field is ' + myfield.getName()
println 'Retrieve value directly, method 1, using getter method ... '

def my_new_field1 = ticket.get("helpdesk_standard.priority").getName()
println 'Result is ' + my_new_field1
println 'Retrieve value directly, method 2, using direct access to attribute ... '

def my_new_field2 = ticket.get("helpdesk_standard.priority").name
println 'Result is ' + my_new_field2
println 'Display value of simple data field ... '

def fb = ticket.get("helpdesk_standard.feedback")
println 'Value of feedback boolean field is ' + fb
```

Code example 24: *Admin Tool script used to display Custom 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 Custom 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 25: *Precondition script where boolean value is checked*

Or shorter:

```

return ticket.get("am_fields.vip")

```

Code example 26: *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")
println "Priority is now " + prio.getName()
//or shorter:
println "Priority is now " + prio.name

```

Code example 27: *Retrieving an enum value for a CF*

MLAs

Work with one MLA value (the selected "leaf" of the MLA tree)

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.

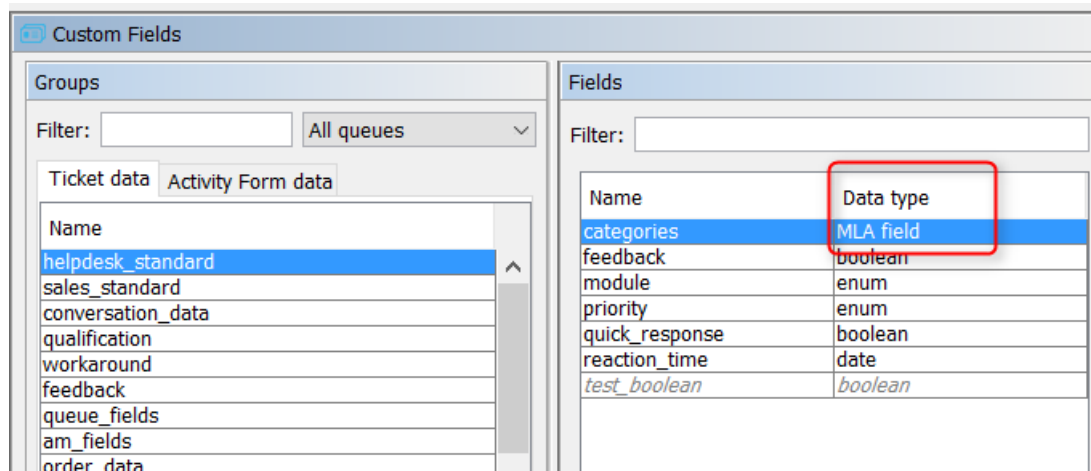


Figure 121: Admin Tool: Definition of a Custom Field of type MLA

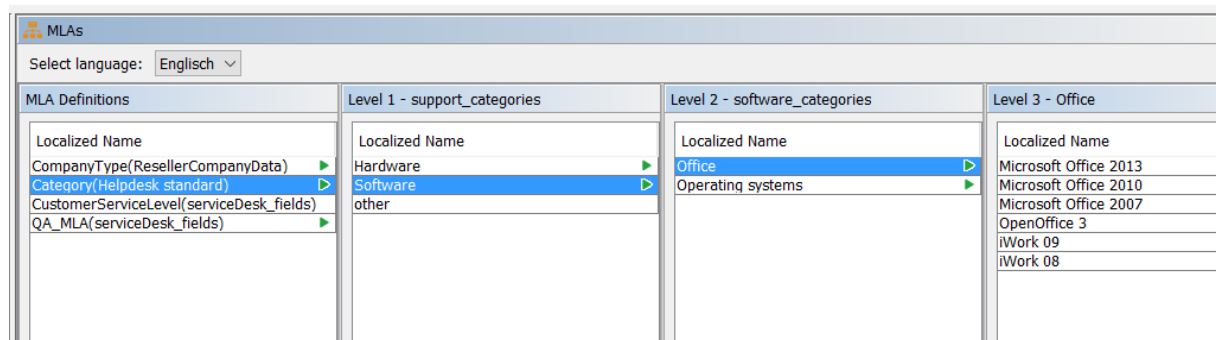


Figure 122: Admin Tool: MLA definition

Figure 123: 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

println 'Displaying MLA value ...'
def my_mla_field = ticket.get("helpdesk_standard.categories")
println '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())
println 'MLA value categories (localized name) is now ' + my_mla_field_localized
```

Code example 28: 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
stdout] [Susan-] MLA value categories (localized description) is now Category
```

Figure 124: 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:

```
println 'Displaying MLA branch ...'
List<EnumValue> mlaPathElements = mlaService.getAssignedMla(ticket, "helpdesk_
standard", "categories")
def mlaPath1=""
mlaPathElements.each() {elem ->
    mlaPath1 = elem.name + ' -- ' + mlaPath1
}
println 'mlaPath1 is now ' + mlaPath1

//or shorter:
def mlaPath2 = mlaService.getAssignedMla(ticket, "helpdesk_standard",
"categories").reverse().name.join(" -- ")
println 'mlaPath2 is now ' + mlaPath2
```

Code example 29: 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
----- [Susan-] 1-1-2013 15:55:23 [11-0-2013 15:55:23] X: 1-1-2013 15:55:23
```

Figure 125: 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 CF of type *date* has to have the parameter *Belongs to* which points to the list.

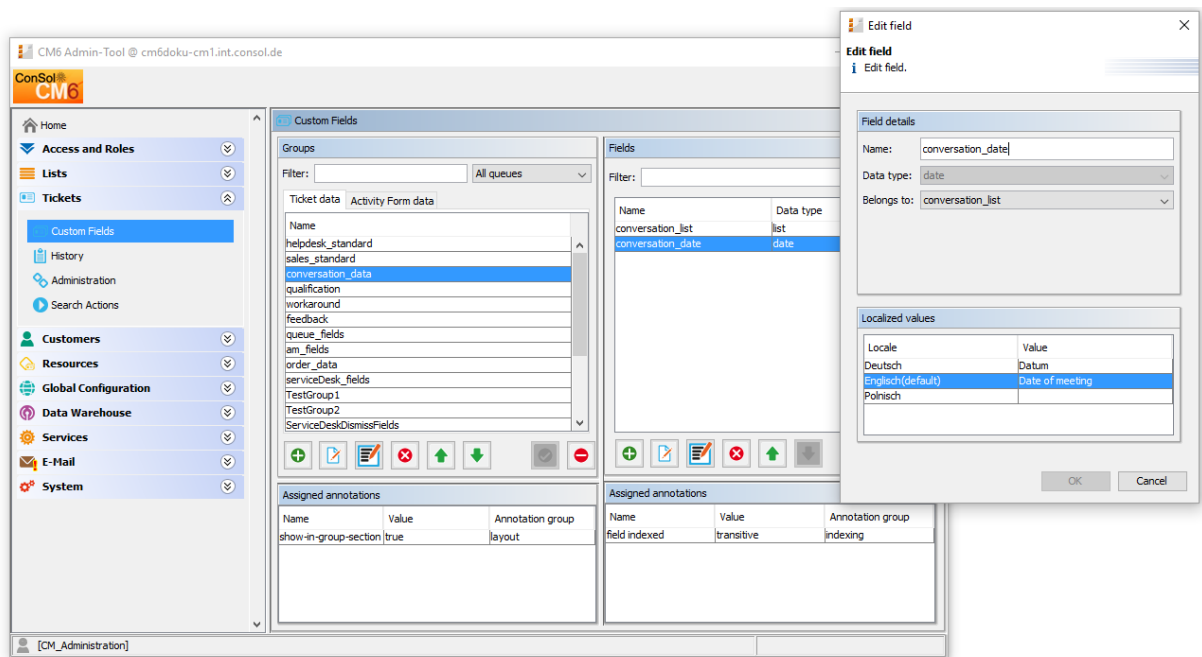


Figure 126: ConSol CM Admin Tool - CFs for a list of date fields

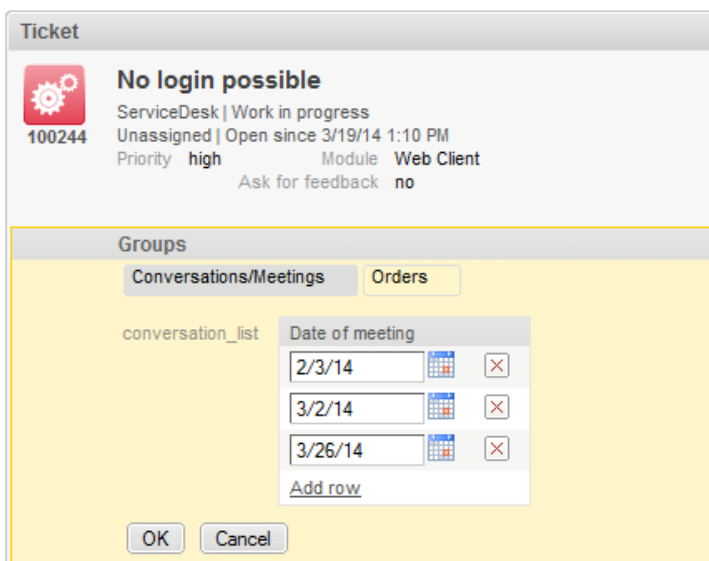


Figure 127: ConSol CM Web Client - List of date fields in a ticket (edit mode)

For access to each *date* CF within a list use the following lines of code:

```
def convs = ticket.get("conversation_data.conversation_list").each() { conv ->
  println "NEXT DATE is :" + conv
  println "CLASS of NEXT DATE is " + conv.getClass()
}
```

Code example 30: *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 128: *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 31: *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 Custom 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 CF *orders_list* represents the list.
- the CF *orders_list* is located within the CF 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*).

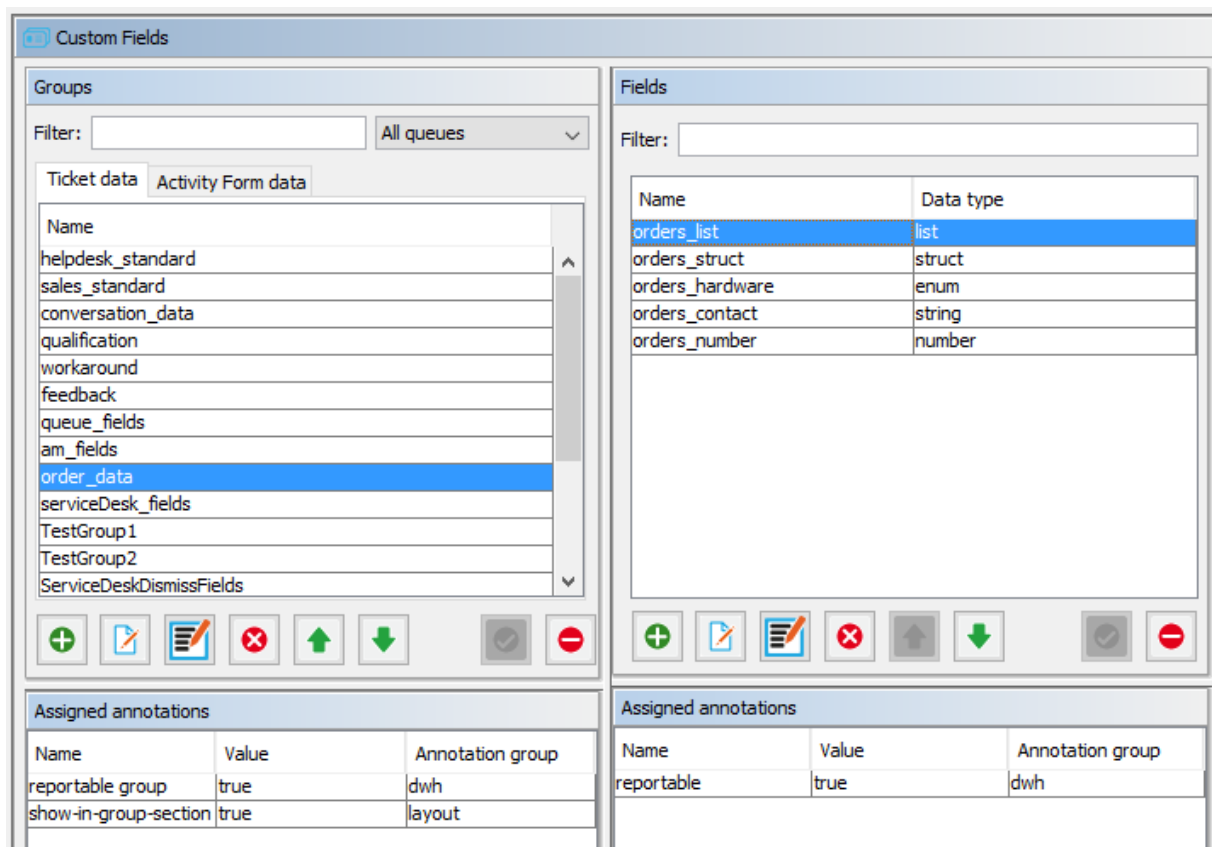


Figure 129: ConSol CM Admin Tool - Custom Fields for list

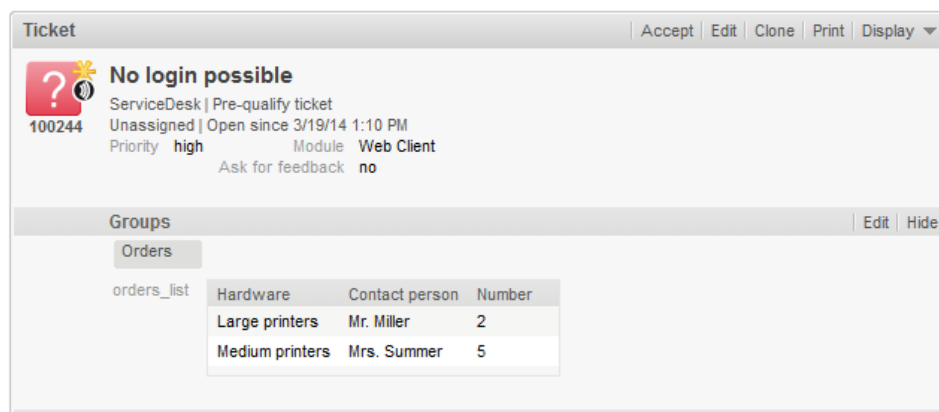
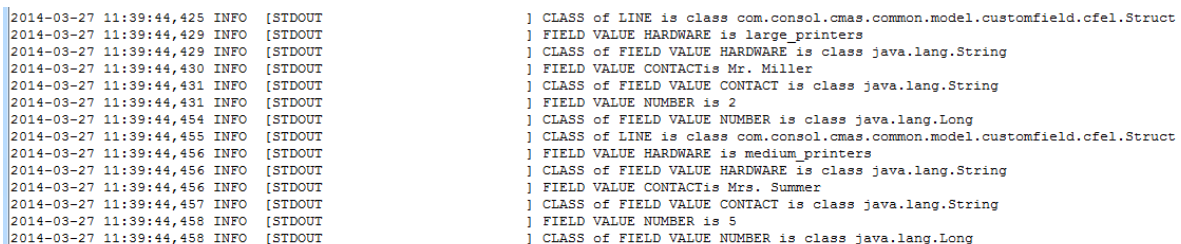


Figure 130: ConSol CM Web Client - Ticket with filled-in table

```
def structs = ticket.get("order_data.orders_list").each() { str ->
    println("CLASS of LINE is " + str.getClass())
    println("FIELD VALUE HARDWARE is " + str.orders_hardware.getName())
    println("CLASS of FIELD VALUE HARDWARE is " + str.orders_hardware.getName()
        ().getClass())
    println("FIELD VALUE CONTACT is " + str.orders_contact)
    println("CLASS of FIELD VALUE CONTACT is " + str.orders_contact.getClass())
    println("FIELD VALUE NUMBER is " + str.orders_number)
    println("CLASS of FIELD VALUE NUMBER is " + str.orders_number.getClass())
}
```

Code example 32: 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 CONTACT is 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 CONTACT is 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 131: Log File - Script Output

D.8.4.3 Setting Custom Field Values for Ticket Data

To set values for ticket CFs, you follow the same principle as for getting data: use the CF group name and the technical name of the CF 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("<Group_name>.<CF_name>", <value>)
```

Setting Values for Custom Fields with Simple Data Types

```
ticket.set("fields.reaction_time", new Date());
```

Code example 33: Set a CF value for a Date CF

When you work with *number* or *date* fields, you can even calculate with the CF 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 34: Calculate with value of date CF

Setting a value to *null* (i.e. emptying the field) is the same as removing the value:

```
ticket.set("fields.numberOfEmployees", null)
```

Code example 35: *Setting a CF value to null*

Or shorter:

```
ticket.remove("fields.numberOfEmployees" )
```

Code example 36: *Setting a CF 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 CF.

```
ticket.set("Group_name.CF_name",<technical name of value>)
```

```
ticket.set("fields.priority", "URGENT");
```

Code example 37: *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 38: *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 39: *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 40: *Adding a new line in a list of structs*

D.8.4.4 Fading-in and -out of Custom Field Groups

A Custom Field Group (CF group) can be faded-in (made visible) and faded-out (made invisible) using a *workflowApi* method. This works for CF groups which are displayed in the main ticket data section as well as for CF groups which are displayed in the tabbed section.

A typical use case is a CF group which is invisible at first (CF group annotation *group-visibility = false*) and is faded-in when the engineer needs to work with the data in the process. For example, a CF 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 41: *Fade-in a CF group*

To fade-out some CF groups, e.g. when the ticket has been qualified and some of the CF 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 42: *Fade-out CF groups*

D.8.5 Data Fields for Customer Data

D.8.5.1 Data Object Group Fields for Customer Data (CM Version 6.9 and Higher)

In CM version 6.9 and higher, the customer Data Object Groups are part of the new customer data model (*FlexCDM*). In version 6.9 they are defined in the Admin Tool, section *User attributes*, file card *Customer data model*. In version 6.10 they are defined in the Admin Tool, navigation group *Customers*, navigation item *Data Models* (also see [Data Object Group Fields for Customer Data \(CM Version 6.10 and Higher\)](#)).

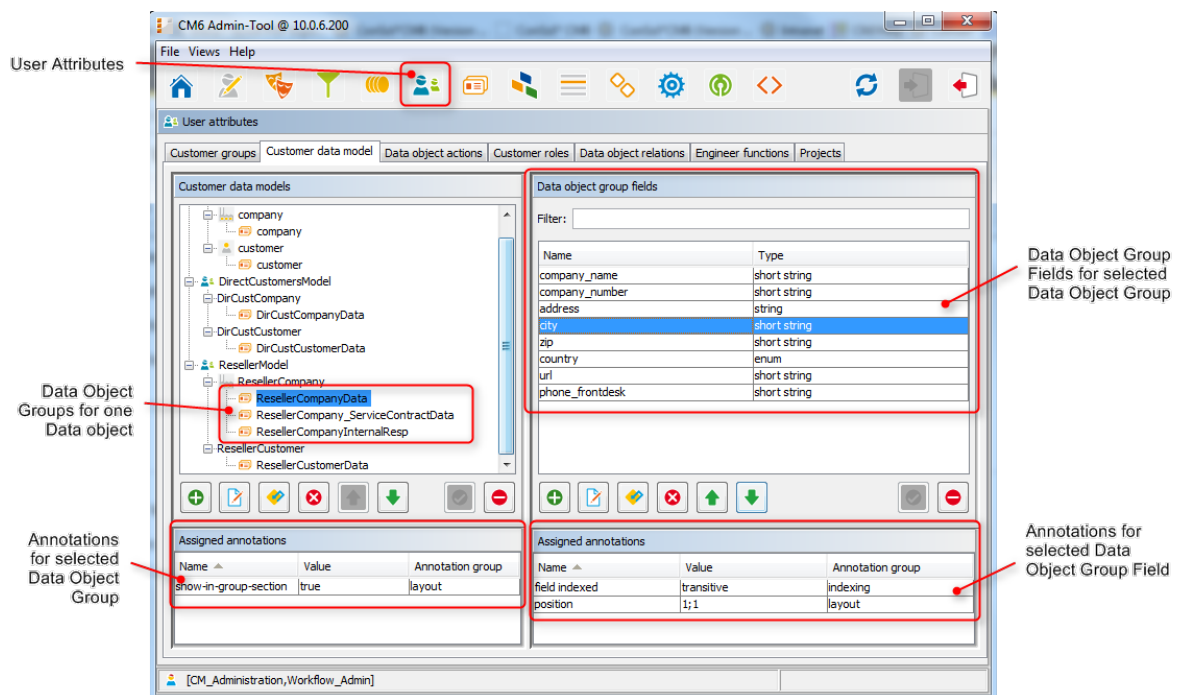


Figure 132: ConSol CM Admin Tool - Data Object Group Field administration for customer data (CM Version 6.9)

The fields, which were called Custom Fields in the customer data model of previous versions, are now called **Data Object Group Fields**. However, the principle you use to retrieve and set values for the data fields is principally the same as in CM version 6.8 and older.

Most Important Methods for Access to Customer Data Data Object Group Fields

Three methods are of major importance for programming access to Data Object Group Fields (DOGF) in CM scripts. They all are methods of the class *Unit*:

- **Unit.get()**
 - For retrieving data from a DOGF.
- **Unit.set()**
 - For setting data in an already existing DOGF.

- **Unit.add()**

- For calculating with a value within a DOGF, 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*:

- **Unit.remove()**

- Sets the value of the field to *null*.

Retrieving Values for Customer Data in CM Version 6.9 and Higher

Because the name of a *Data Object Group Field* might appear in more than one *Data Object Group*, the name of the Data Object Group has to be provided when accessing the customer data. For example, in the customer data model shown in the figure above, the Data Object Groups *ResellerCompanyData* and *DirCustCompanyData* could have a Data Object Group 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 43: Retrieving a field value for a company

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.

Customer Groups

Name	Customer data model
DirectCustomers	DirectCustomersModel
MyCustomerGroup	BasicModel
OurPartnerCompanies	PartnerModel
Reseller	ResellerModel
RetailCompanies	RetailCompaniesModel
RetailCustomers	RetailCustomersModel

Data Models

Name	Type
company_name	short string
company_number	short string
address	string
city	short string
zip	short string
country	enum
city	short string
phone	short string

```

import com.consol.cmas.common.model.ticket.Ticket
import com.consol.cmas.common.model.customfield.meta.UnitDefinitionType

def ticket = workflowApi.getTicket()
def moont = ticket.getMainContact()

println "CustomerGroup of main contact is now " + moont.getCustomerGroup().getName()

println "Customer definition of main contact is now " + moont.getCustomerDefinition().getName()

println "UnitDefinition of main contact is now " + moont.getDefinition().getName()

def custmod = moont.getCustomerDefinition().getName()
println "CUSTMOD is now " + custmod

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;
}

println "CITYFIELD is now " + cityfield

def utype1 = moont.getDefinition().getType()
def utype2 = moont.definition.type

println "UTYPE1 is now " + utype1
println "UTYPE2 is now " + utype2

def company = moont

if (utype2 == UnitDefinitionType.CONTACT) {
  company = moont.get("company()")
}

println "CITY is now " + company.get(cityfield)
  
```

Figure 133: 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.getTicket()

def mcont = ticket.getMainContact()
println "CustomerGroup of main contact is now " + mcont.getCustomerGroup().getName()
println "Customer definition of main contact is now " + mcont.getCustomerDefinition().getName()
println "UnitDefinition of main contact is now " + mcont.getDefinition().getName()

def custmod = mcont.getCustomerDefinition().getName()
// println "CUSTMOD is now " + custmod

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;
}

println "CITYFIELD is now " + cityfield

def utype1 = mcont.getDefinition().getType()
def utype2 = mcont.definition.type

println "UTYPE1 is now " + utype1
println "UTYPE2 is now " + utype2

def company = mcont

if (utype2 == UnitDefinitionType.CONTACT) {
    company = mcont.get("company()")
}

def mycity = company.get(cityfield)
println " CITY is now " + mycity
```

Code example 44: 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.

The screenshot shows a web client interface for managing customer data. It is divided into two main sections: **ResellerCustomer** and **ResellerCompany**.

ResellerCustomer Section:

- Fields: Skywalker, Lea, lea@localhost.de, 123.
- Checkbox: ☒ vip_person.
- Dropdown: Track user.
- Buttons: OK, Cancel.

ResellerCompany Section:

Groups: ResellerCompanyData, Service Contract Data, Internal responsibilities.

ResellerCompanyData Group:

- Fields: ConSol*, Franziskanerstraße 38, Germany (dropdown), +49 89 45841 - 0.
- Fields: München, URL.
- Field: company_number (81669).
- Buttons: OK, Cancel.

Figure 134: ConSol CM Web Client - Customer data set

```

2014-03-27 16:09:21,739 INFO [STDOUT] ] CustomerGroup of main contact is now Reseller
2014-03-27 16:09:21,739 INFO [STDOUT] ] Customer definition of main contact is now ResellerModel
2014-03-27 16:09:21,740 INFO [STDOUT] ] UnitDefinition of main contact is now ResellerCustomer
2014-03-27 16:09:21,743 INFO [STDOUT] ] CUSTMOD is now ResellerModel
2014-03-27 16:09:21,744 INFO [STDOUT] ] CITYFIELD is now ResellerCompanyData:city
2014-03-27 16:09:21,750 INFO [STDOUT] ] UTYPE1 is now CONTACT
2014-03-27 16:09:21,751 INFO [STDOUT] ] UTYPE2 is now CONTACT
2014-03-27 16:09:21,759 INFO [STDOUT] ] CITY is now München

```

Figure 135: Log file - Script output FlexCDM

```
String firstName = company.get("responsibleConsultants[0].firstName");
```

Code example 45: Retrieving a value from a list of structs using index notation

Setting Values for Customer Data in CM Version 6.9 and Higher

Setting Values for Data Object Group Fields with Simple Data Types

The *set* and *add* methods work as described for ticket Custom Fields. For example:

```
//set number field
company.set("numberOfEmployees", 1);
//add 1 to field value, afterwards the value of the field is 2
company.add("numberOfEmployees", 1);
```

Code example 46: *Set and add values for a Data Object Group Field of type integer*

Lists

Setting Values in a List of Structs for Customer Data

```
company.set("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 47: *Creating a new list of structs, version 2*

```
company.add("responsibleConsultants", new Struct().set("lastName", " Nowitzki")
    .set("email", "dnowitzki@consol.us"));
```

Code example 48: *Adding a new line in a list of structs for company data*

```
company.set("responsibleConsultants[0].firstName", "John");
```

Code example 49: *Setting a value in a list of structs using index notation*

```
company.set("responsibleConsultants[last]", null);
```

Code example 50: *Removing a struct (= line) from a list of structs (= table)*

D.8.5.2 Convenience Methods for Access to Customer Data in CM Version 6.9 and Higher

```
Unit mainContact = ticket.getMainContact();

// "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 51: *Convenience methods for access to customer data*

D.8.5.3 Data Object Group Fields for Customer Data (CM Version 6.10 and Higher)

In CM version 6.10 and higher, as in version 6.9, the customer Data Object Groups are part of the customer data model (*FlexCDM*). They are defined in the Admin Tool, navigation group *Customers*, navigation item *Data Models*.

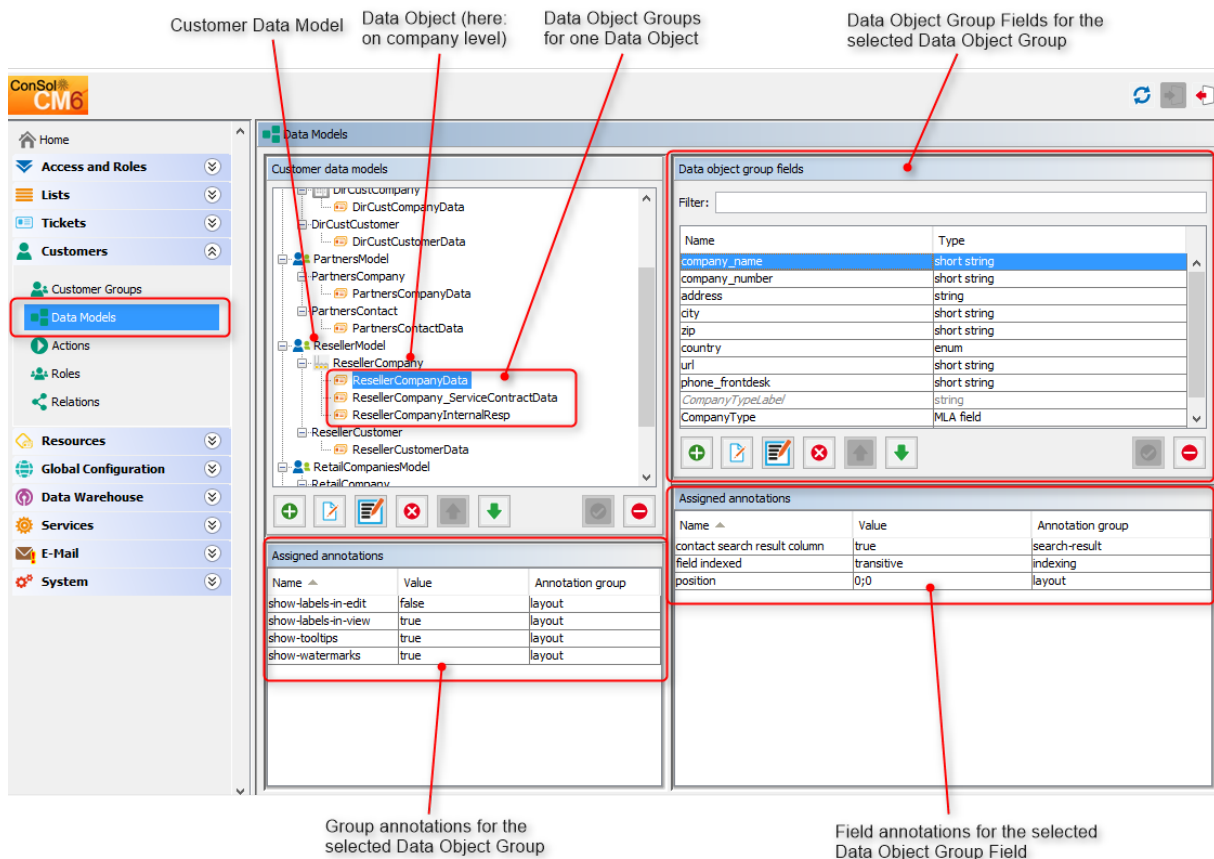


Figure 136: ConSol CM Admin Tool - Data Object Group Field administration for customer data (CM version 6.10)

Everything you have learned about Data Object Groups Fields in version 6.9 also applies to CM version 6.10. But some methods have become deprecated!

In CM version 6.9, it is still possible to use Unit methods which do not include the Data Object Group name as parameter, e.g.,

```
Unit.getField(String pFieldName)
```

This only works smoothly when a Data Object Group Field has the same name over all customer groups where the script is applied, or when the surrounding code ensures that it is not used in ambiguous runtime situations. In version 6.10, the Unit methods without Data Object Group name have become deprecated, and they will no longer be supported in version 6.11 and up. The following table lists all those methods:

CM versions up to 6.9, deprecated in 6.10, not in 6.11	CM version 6.10 and up
<code>getField(String pFieldName)</code>	<code>getField(String pGroupName, String pFieldName)</code>
<code>getFieldValue(String pFieldName)</code>	<code>getFieldValue(String pGroupName, String pFieldName)</code>
<code>setFieldValue(String pFieldName, Object pFieldValue)</code>	<code>setFieldValue(String pGroupName, String pFieldName, Object pValue)</code>
<code>removeField(String pFieldName)</code>	<code>removeField(String pGroupName, String pFieldName)</code>

D.8.6 Resource Data

Starting with version 6.10.0, ConSol CM can contain the optional module *CM.Resource Pool*. This module 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 set-up 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.

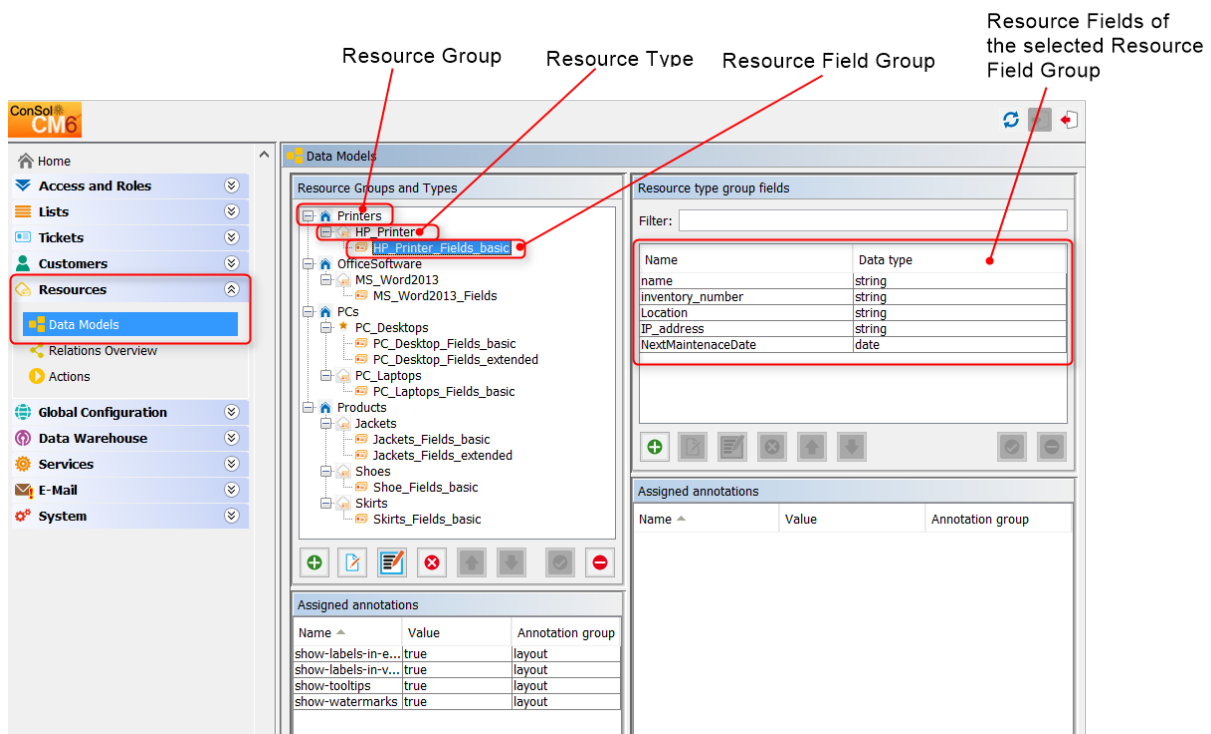


Figure 137: Admin Tool: Definition of the resource data model

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")
println 'Displaying resource information ...'
println ' Name is : ' + my_resource_name

def my_resource_inv_number = resource.get("HP_Printer_Fields_basic.inventory_
number")
println ' Inventory number is : ' + my_resource_inv_number
```

Code example 52: Simple resource action script which displays information about the resource in the log file

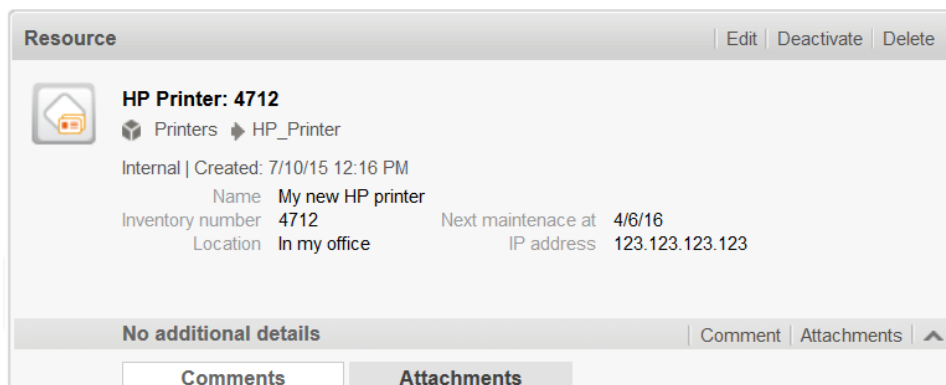


Figure 138: Web Client: Resource page of a HP printer resource

```
stdout] [Susan-] Displaying resource information ...
stdout] [Susan-] Name is : My new HP printer
stdout] [Susan-] Inventory number is : 4712
```

Figure 139: 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 Using Data Fields for (Invisible) Variables

Sometimes it is necessary to work with variables which are not used as values for Custom Fields, Data Object Group Fields or Resource Fields, which are not visible on the GUI, but which are only used as containers for internal programming variables.

Those of you who know how to program ConSol CM5 workflows know those containers as *global variables*. In ConSol CM6, you can achieve the same goal by creating regular Custom Fields (for ticket data), Data Object Group Fields (for customer data) or Resource Fields (for resource data) 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 E-Mails

This chapter discusses the following:

D.9.1 Introduction to Sending E-Mails	183
D.9.2 Important Methods	183
D.9.3 Examples	183
D.9.4 Writing E-Mails from Scripts when Engineer Representation Rules Apply	192

D.9.1 Introduction to Sending E-Mails

The capability of receiving and sending e-mails 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 e-mails 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 e-mail into the script but you work with e-mail 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

D.9.2.1 ConSol CM Version 6.9 and Higher

Use an object of class *Mail*.

Here you can define all required parameters for an e-mail and you can configure the *Mail* object to use the queue-specific e-mail default script. This is a script which processes the e-mail before it leaves the CM system. This kind of script can be assigned to a queue (*E-Mail 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).

D.9.3 Examples

D.9.3.1 Sending an Automatic Acknowledgment of Receipt to the Customer Who Has Opened a Ticket

ConSol CM Version 6.9 and Up

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 53: *Sending an automatic acknowledgment of receipt to the customer who has opened a ticket, using a Mail object*

D.9.3.2 Sending an E-Mail to the Engineer When a Certain 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.

ConSol CM Version 6.9 and Up

```
// 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 54: *Sending an e-Mail to the engineer when a certain escalation level has been reached, using a Mail object*

D.9.3.3 Sending an E-Mail to a Customer Integrating the Queue-Specific Mail 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 e-mail will pass through the script before it leaves the CM system. E-mail 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 55: *Sending an e-mail to a customer integrating the queue-specific mail script, using a Mail object*

D.9.3.4 Sending an E-Mail to All Contacts of the Ticket

This will send one e-mail with all customers (that have an e-mail 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 e-mail 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 56: *Sending an E-Mail to All Contacts of the Ticket*

D.9.3.5 Sending an E-Mail to Each Contact in a List of All Contacts of the Ticket

This will send one e-mail to every single customer (that has an e-mail 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 e-mail 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 Sending an E-Mail and Inserting it into the Ticket History (CM 6.9 and up)

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 e-mail is sent, because in the example system, companies do not have e-mail addresses. In the next step, the correct Data Object Group Field which contains the e-mail 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 e-mail 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 contact 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, 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 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 57: *Admin Tool script for sending a notification of receipt and inserting the email as ticket history entry*

D.9.3.7 Working with a Template for the Ticket Name in the Mail 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 e-mail subject guarantees that an incoming e-mail 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 *E-Mail* (compare following figure).

Figure 140: *Admin Tool: Pattern and template for the e-mail subject*

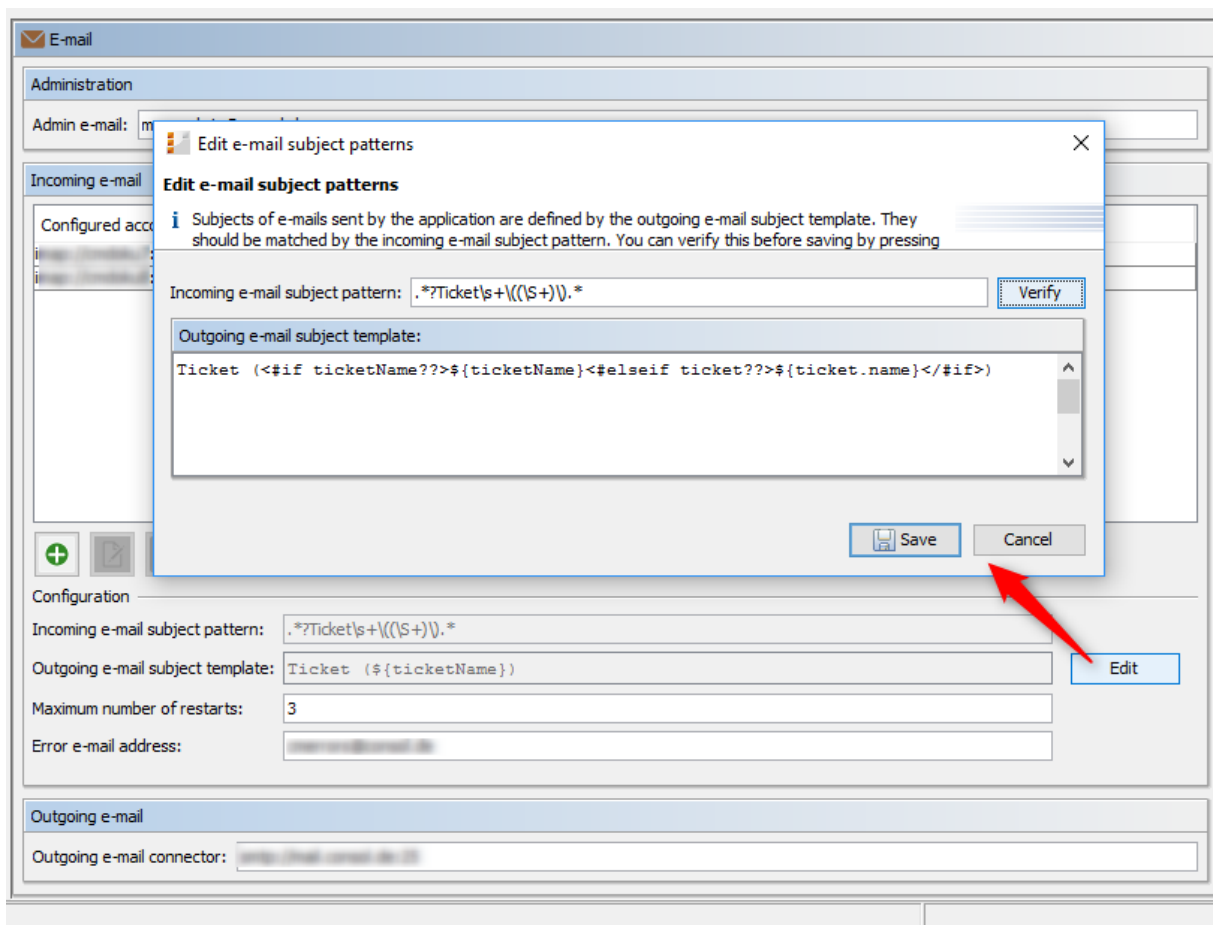
In order to avoid inconsistencies between workflow code and system configuration (Admin Tool), you can also work with the template name which is implicitly available in CM.

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 e-mails 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 e-mail 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.9.4 Writing E-Mails from Scripts when Engineer Representation Rules Apply

When you write scripts where e-mails 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 *Engineer Profile*).



Important information about representation configurations

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

1. An engineer writes an e-mail using the Ticket-E-Mail Editor: the representation rule is applied and the representing engineer receives a copy of the e-mail. This means all e-mails 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 e-mail 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 e-mail is sent automatically from the CM system: it depends on the specific configuration of the CM system which engineers receive a copy of the e-mail, the e-mail 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 e-mail 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 e-mail. Please read the following section for details!

For e-mails sent out using scripts, the CM system behavior concerning representation rules depends on the method which is used to send the e-mail.

All of the three methods listed here belong to the java Class *Mail*.

D.9.4.1 setTo(String pTo)

```
Mail.setTo(<e-mail 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 e-mail 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 e-mail to the engineer's e-mail address (if set) and to the representing engineer who is registered as representation for the original e-mail 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 e-mail to the e-mail address of each engineer in the list (if e-mail address is set) and to the representing engineers for all of the listed engineers' e-mail 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 e-mail. Please see the following example which uses the Mail.setTargetEngineer() method.

```
import com.consol.cmas.common.model.mail.Mail  
def ticket = workflowApi.ticket
```

```
import com.consol.cmas.common.model.mail.Mail

// Read engineer login from CF
// engineer login could also be retrieved, e.g., using ticket.engineer or
// engineerService.current, depending on what you need
def engineerName = ticket.get("main_data_fields.next_mail_to_engineer")
def engineer
if (engineerName) {
    engineer = engineerService.getByName(engineerName)
}
def engineerMail = engineer?.email
if (engineerMail) {
    def subject = "Your info mail about ticket " + ticket.getName() + " Subject: ' "
    + ticket.getSubject() + "' "

    // Retrieve e-mail text from template
    def text = workflowApi.renderTemplate("engineer_info")

    def mail = new Mail()
    mail.setTargetEngineer(engineer)
    mail.setSubject(subject)
    mail.setText(text)
    // Use outgoing mail script, queue-specific
    mail.useDefaultScript()

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

Code example 58: *Workflow or Admin Tool script used to send an e-mail 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	196
D.10.2 Retrieve Path Information for a Workflow Element	197
D.10.3 Examples for the Use of Path Information	197

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.

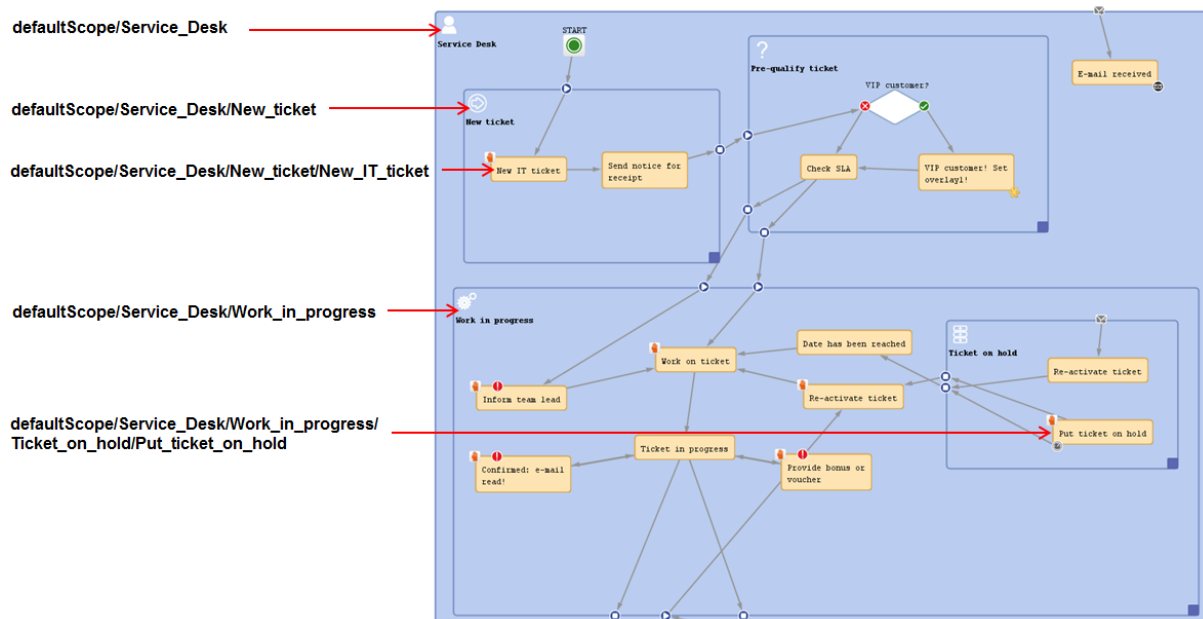


Figure 141: ConSol CM Process Designer - Path information (example: activities and scopes)

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*.

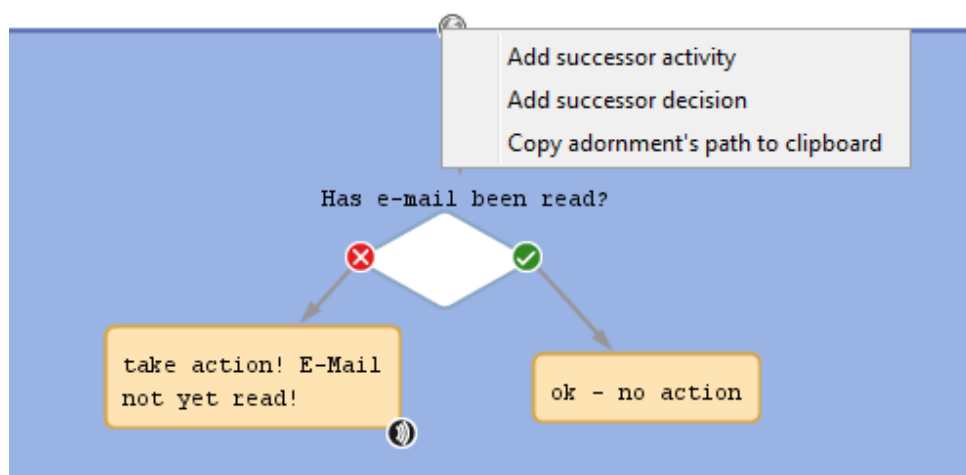


Figure 142: ConSol CM Process Designer - Copying the path of a workflow element

D.10.3 Examples 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 e-mail has been received and make sure that the e-mail 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 e-mail 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 59: *Deactivate a time trigger*

When an e-mail has been received, the trigger has to be re-initialized. The following code would be used:

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

Code example 60: *Re-initialize time trigger*

D.11 Working with Calendars and Times

This chapter discusses the following:

D.11.1 Introduction	199
D.11.2 Calculating with Dates and Times without a CM Business Calendar	200
D.11.3 Calculating with Dates and Times Using a CM Business Calendar	200

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 about 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.

i 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 CM Business Calendar

D.11.2.1 Example: Setting a Time Trigger Time with Dynamic Time Range

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 61: *Setting time for a time trigger*

D.11.3 Calculating with Dates and Times Using a CM Business Calendar

D.11.3.1 Example: Using a Time Trigger with a Business Calendar to Calculate Escalation Time (CM 6.9)

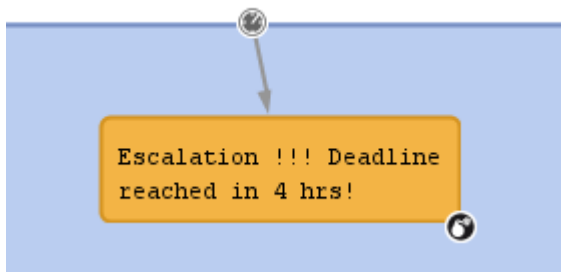


Figure 143: *ConSol CM Process Designer - Time trigger for escalation 4 hours before deadline*

```
def deadl = ticket.get("serviceDesk_fields.desiredDeadline")

// 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()

// 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

def dueTime = escalDate.time - now.time
trigger.setDueTime(dueTime)
```

Code example 62: *Script for time trigger for escalation 4 hours before deadline*

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 data objects, i.e. contacts and companies, see section Working with Customer Relations (Data Object 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 the Work 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.

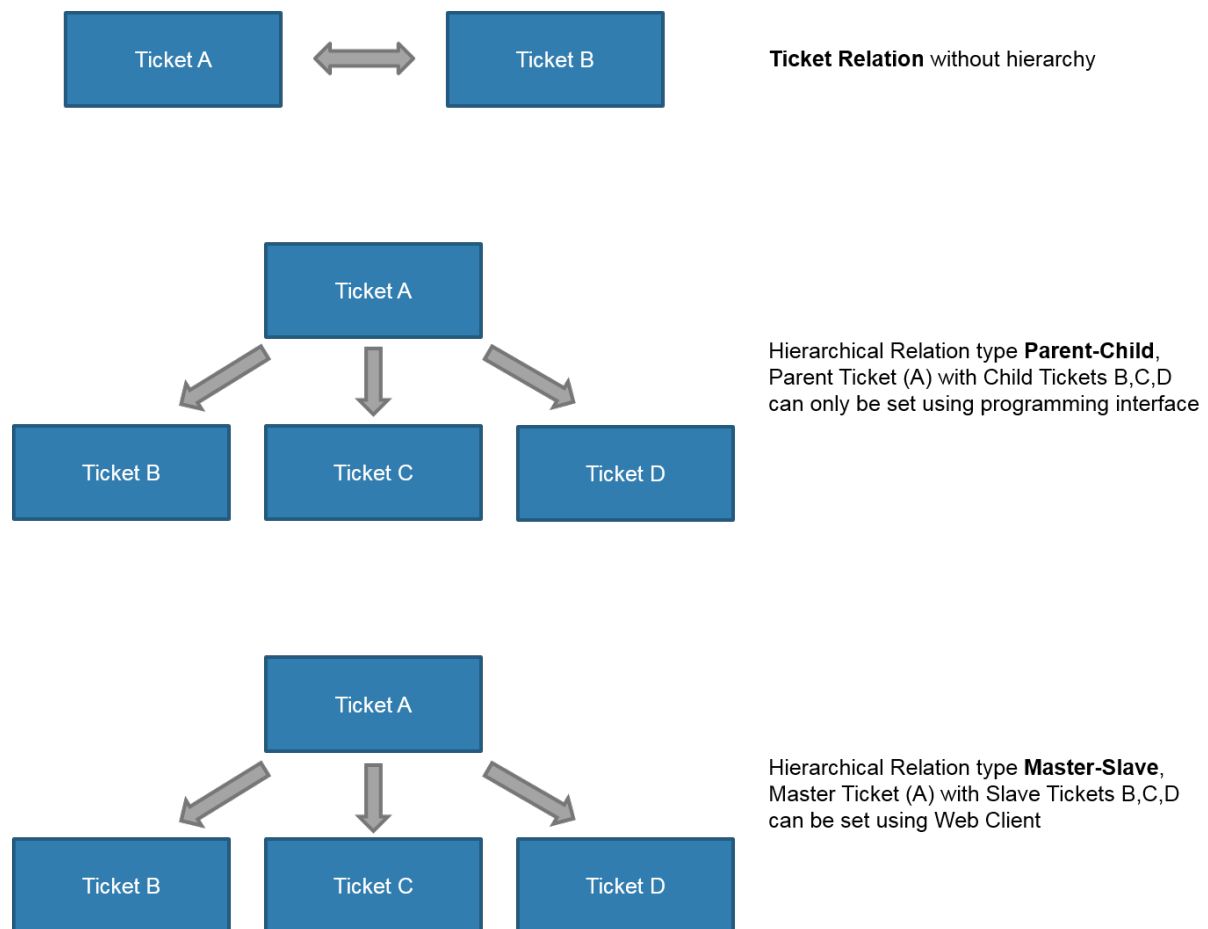


Figure 144: ConSol CM relation types

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 63: *Creating a ticket relation of type REFERENCE using workflowAPI*

Syntax: Find all Referenced Tickets

ConSol CM versions 6.10.4.0 and up

```
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 when 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 64: *Creating a ticket relation of type MASTER_SLAVE using workflowAPI*

Syntax: Finding All Slave Tickets

ConSol CM versions 6.10.3.x and below

```
// the ticket can be set, might be current ticket or another ticket
List<Ticket> mytickets = workflowApi.getTargetTickets(myTicket.getId(),
TicketRelationType.MASTER_SLAVE)
```

Code example 65: *Version A: Finding all target tickets (here: all slave tickets)*

```
// used for current ticket
List<Ticket> mytickets = workflowApi.getTargetTickets(TicketRelationType.MASTER_
SLAVE)
```

Code example 66: *Version B: Finding all target tickets (here: all slave tickets)*

ConSol CM versions 6.10.4.0 and up

```
List<Ticket> mytickets = workflowApi.getSlaveTickets()
```

Code example 67: *Finding all slave tickets of the current ticket*

Syntax: Finding the Master Ticket

ConSol CM versions 6.10.4.0 and up

```
def mticket = workflowApi.getMasterTicket()
```

Code example 68: *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 Child of 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 69: *Creating a child ticket*

Example 2: Finding the Parent Ticket of a Ticket

```
def my_parent = workflowAPI.getParentTicket()
```

Code example 70: *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 71: *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 72: *Finding all brother tickets of a (child) ticket*

D.12.1.5 Important Methods for the Work 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, priority, and category 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 <i>null</i> 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 <i>pTicketId</i> 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.

Method of workflowApi (workflowContextService)	Explanation
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.
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 > pTargetTicketIds)	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 <i>sourceTicketId</i> and <i>targetTicketId</i> .
Ticket getMasterTicket()	Available starting with CM version 6.10.4.0 Return the master ticket of the current ticket.
List <Ticket> getSlaveTickets()	Available starting with CM version 6.10.4.0 Returns a list of all slave tickets of the current ticket

Method of workflowApi (workflowContextService)	Explanation
List<Ticket> getReferencedTickets()	Available starting with CM version 6.10.4.0 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 type and optionally the end at which the ticket should be attached.
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 type and optionally the end at which the tickets should be attached.

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 ->
    println ' Slave Ticket is now ' + st.getId() + ' -- ' + st.getSubject()
}
```

Code example 73: 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 ->
    println 'Source ticket is now ' + tr.sourceTicket.id + ' -- ' +
        tr.sourceTicket.subject
    println 'Target ticket is now ' + tr.targetTicket.id + ' -- ' +
        tr.targetTicket.subject
}
```

Code example 74: *Example Script, display IDs and names of slave tickets Admin Tool script version*

```
scriptExecutionService.execute("DisplaySlaveTickets.groovy")
```

Code example 75: *Calling previous AT script from workflow activity*

D.12.2 Working with Customer Relations (Data Object Relations)

This chapter discusses the following:

- [Introduction](#)
- [Creating Unit Relations Using the Programming Interface](#)
- [Important Java Classes for the Work with Unit Relations](#)

D.12.2.1 Introduction

Since version 6.9.0, ConSol CM offers *customer relations*. In older versions, this feature is not available!

To be able to work with customer relations, you have to have a profound knowledge of the *FlexCDM*, the ConSol CM Flexible Customer Model. Please refer to the *ConSol CM Administrator Manual (Version 6.9)* for a detailed introduction.

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 or company who is registered in the CM database
Company	Unit	Data Object of type <i>company</i>	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 <i>company level</i> can be the <i>parent level</i> for an object on the <i>contact level</i> . From a logical point of view, a company can have several contacts.
Contact	Unit	Data Object of type <i>contact</i>	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 <i>contact level</i> can be a stand-alone object (in a one-level customer model) or can belong to a <i>company level</i> object. From a logical point of view, a contact can belong to none or exactly one company.



Keep in mind that, starting with CM version 6.9, 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*.

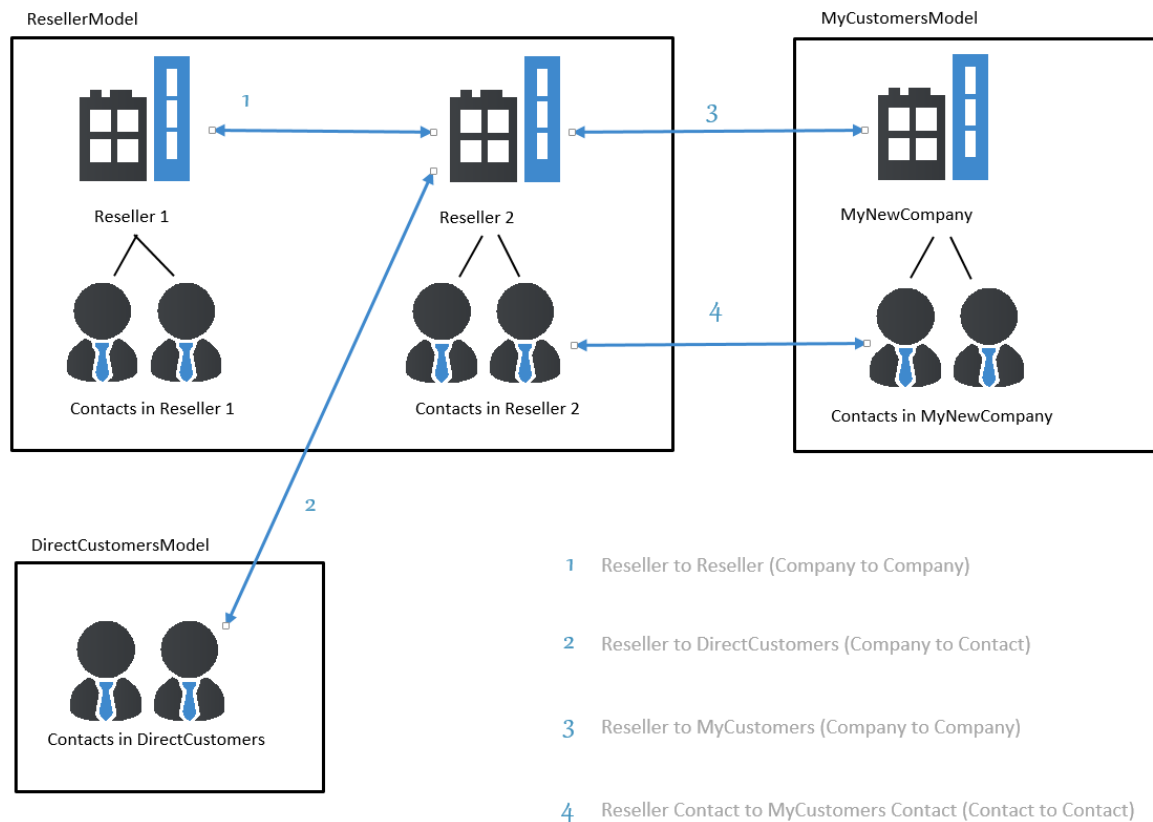


Figure 145: Customer relations in ConSol CM

! To work with *unit* 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 Unit Relations Using the Programming Interface

! In this book we sometimes use the new terms *data object* and *data object definition* which are part of the new customer model of ConSol CM version 6.9 and higher (*FlexCDM*). However, the names of the corresponding Java classes are still *Unit* and *UnitDefinition*. All other Java classes which deal with customer data objects are also still named *Unit...*. Please keep that in mind when you work on the administrator level as well as on the programmer's level with a 6.9.x version. Please refer to the *ConSol CM Java API* documentation for details.

Example: 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*.

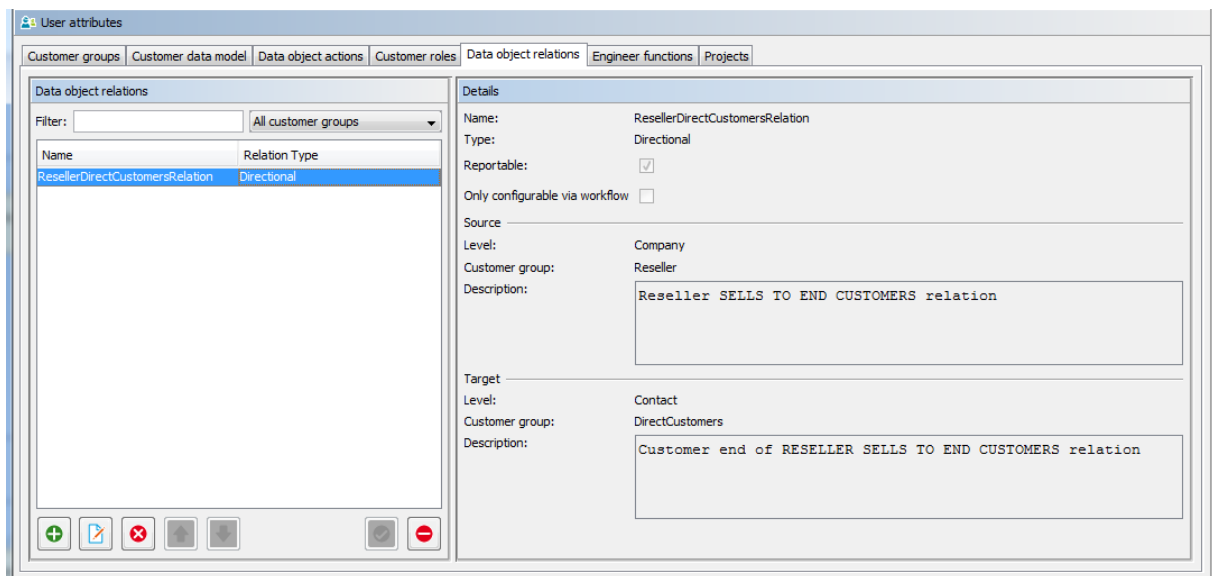


Figure 146: ConSol CM Admin Tool - Definition of reseller - end customer relation

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.

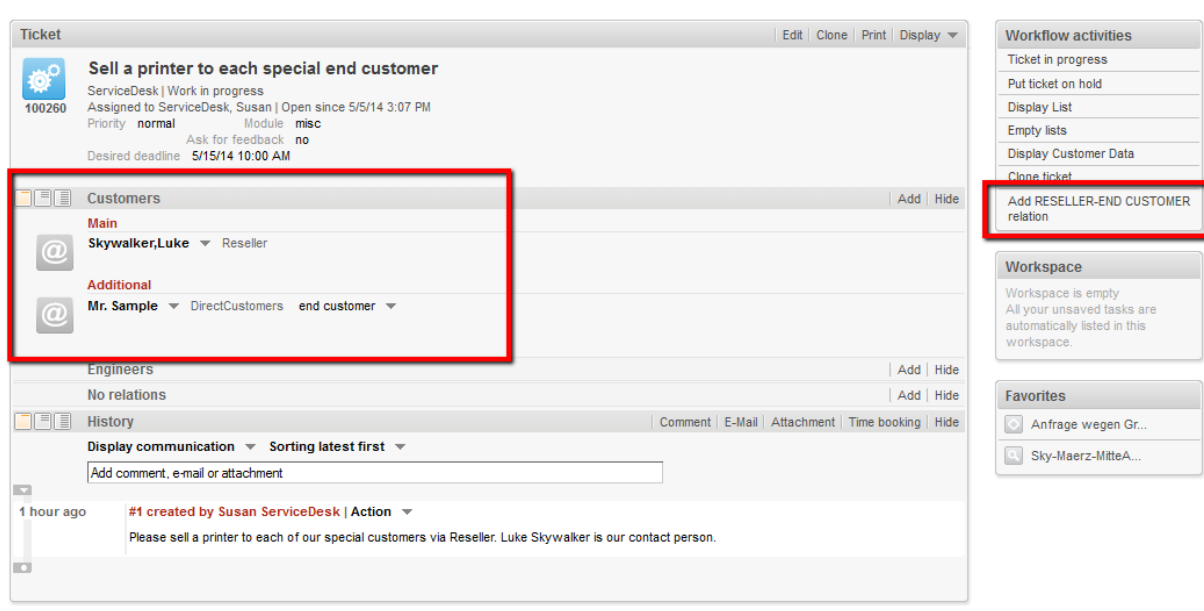


Figure 147: ConSol CM Web Client - Example ticket with main customer and one additional customer

In the *Service Desk* workflow, there is a workflow activity *Add RESELLER-END CUSTOMER relation* (see next figure).

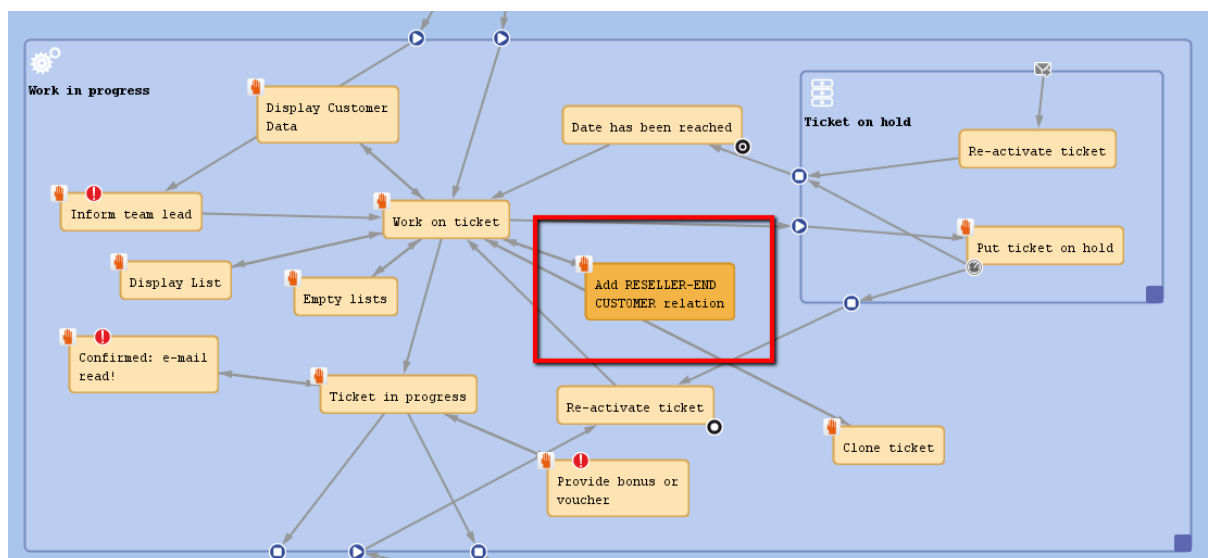


Figure 148: ConSol CM Process Designer - Workflow activity for adding a unit relation

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:
// 1. get the main contact of the ticket. Here, this is a person = contact:
def cont = ticket.getMainContact()
// 2. get the company of the contact, this is the reseller company
def comp = cont.getCompany()

// get all additional contacts of the ticket in the customer role „end customer“
//and start the loop for all those additional customers:
def end_custs = ticket.getContacts("end customer").each() { e_cust ->

    //build all components for new unit relation:
    // 1.get the UnitDefinition by name (this is the name used in the Admin Tool):
    def unitrel_def = unitRelationDefinitionService.getByname
        ("ResellerDirectCustomersRelation")
    // create a new unit relation object with the unit definition and source
    // (the reseller company) and target (the end customer person)
    def new_rel = new UnitRelation(unitrel_def, comp, e_cust, "This Reseller sells
        to the end customer")

    // create the new unit relation in the system
    def new_rel2 = unitRelationService.create(new_rel)
}
```

Code example 76: 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.

Company Display

MyNewSpaceCompany 999 ▼ Reseller

Groups Edit Hide

ResellerCompanyData **Service Contract Data** **Internal responsibilities**

MyNewSpaceCompany 999
Milkyway 77 Alderaan 7777
Unknown
123

Tickets (0) Hide

All tickets ▼

No search results

Contacts (1) Hide Add

Add/Remove column 'email', 'forename', ... OK Number per page 10 ▼

Contact	email	forename	customer_name	phone	vip_person
Skywalker, Luke	katja@consol.de	Luke	MyNewSpaceCompany 999 Skywalker	123	no

Additional details Hide

Comments **Attachments**

New

List of comments

This company does not have any comments.

Relations Add Hide

Reseller SELLS TO END CUSTOMERS relation (DirectCustomers) (Contact)

Add/Remove column 'Customer name' OK Number per page 10 ▼

Date	Customer name	Note	Actions
5/5/14 15:51	Mr. Sample	This Reseller sells to the end customer	Edit

Figure 149: ConSol CM Web Client - New unit relation (created by workflow script)

D.12.2.3 Important Java Classes for the Work with Unit Relations

Java class	Explanation
Unit	A data object (unit): a contact or a company.
UnitRelation	A relation between two data objects (units). Visible in the Web Client on the contact or company page under <i>Relations</i> .

Java class	Explanation
UnitRelationDefinition	The definition of a unit relation as configured in the Admin Tool under <i>User attributes - Data object relations</i> (CM version 6.9) or <i>Customers - Relations</i> (CM version 6.10). A <code>UnitRelation</code> always has a certain <code>UnitRelationDefinition</code> .
UnitRelationDefinitionService	Singleton. Available as object <code>unitRelationDefinitionService</code> . Service which provides helpful methods for the work with data object (unit) relations. See the <i>ConSol CM Java API</i> documentation for details.
UnitRelationService	Singleton. Available as object <code>unitRelationService</code> . Service which provides helpful methods for the work with data object (unit) relations. See the <i>ConSol CM Java API</i> 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

Since version 6.10.0, ConSol CM offers the module CM.Resource Pool and with it the possibility to establish relations between resource objects and other objects. A resource relation can connect a resource to

- a ticket
- a customer (contact or company, i.e. a unit)
- another resource

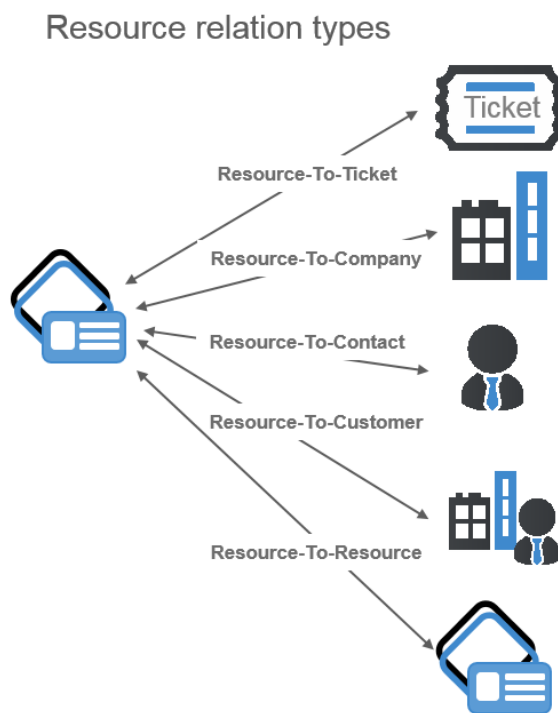


Figure 150: *Resource relation types*

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 (versions 6.10 and up)*, section *CM.Resource Pool - Resource Relations* 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 description	Explanation
resource	Resource	Resource	An object of the defined Resource Type.
resourceRelationService	Interface ResourceRelationService	-	the service which offers a lot of helpful methods for dealing with resources
ResourceRelationxy	ResourceRelation	Resource (type) relation	Relation between a resource and another object
	ResourceRelationWithTargetResourceCriteria ResourceRelationWithTargetTicketCriteria ResourceRelationWithTargetUnitCriteria	-	Stores the criteria which are the basis for a search, performed by the resourceRelationService

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.

```
///Take SLA (resource) from company of main customer and calculate deadline from
it
def maincust = ticket.mainContact
def unit_type = maincust.definition.type
println 'TYPE is now : ' + unit_type
def comp
if(unit_type.equals('CONTACT')) {
    comp = maincust.get("company()")
} else if (unit_type.equals('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)
println '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")
    // calculate reaction time
    def now = new Date()
    def deadline = now + 1
    ticket.set("serviceDesk_fields.desiredDeadline",deadline)
}
```

Code example 77: *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	224
D.13.2 Example: Checking if a Solution Exists Before the Ticket can be Closed	225
D.13.3 Example: Adding a Text as Ticket Comment and Setting a Class of Text	227

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 e-mail that was sent from the ticket
- an e-mail that was received in the ticket
- an attachment

Assigning a class of text can 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.

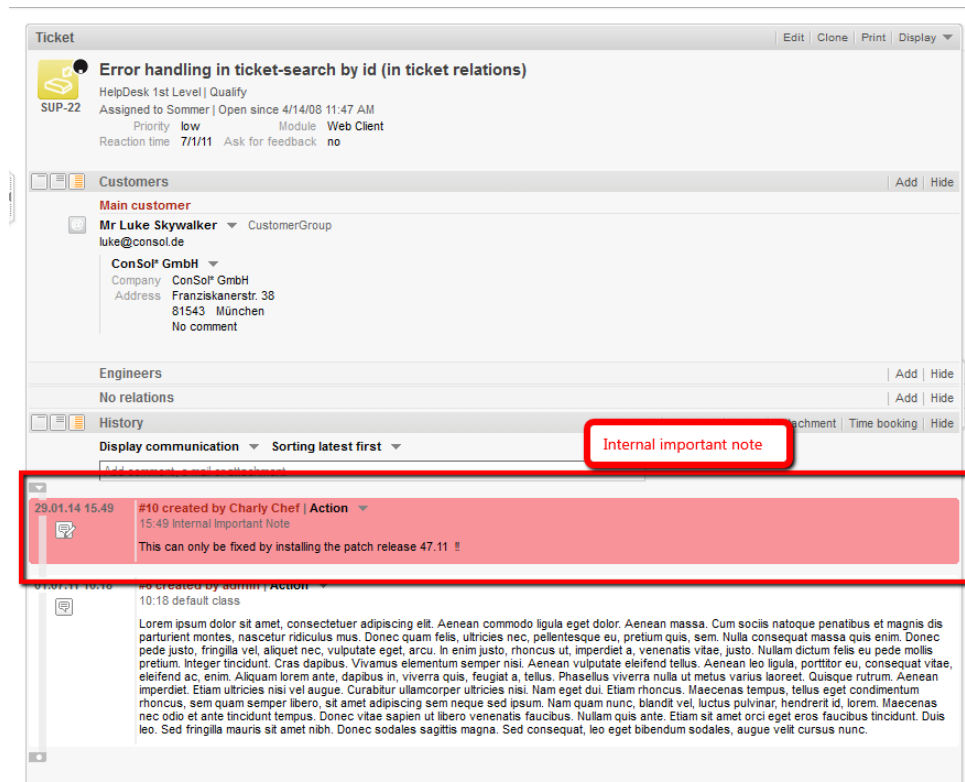


Figure 151: ConSol CM Web Client - Using a class of text for an internal important note

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 everyday consulting life.

D.13.2 Example: Checking if a Solution Exists Before the Ticket can be Closed

In the following example, all text entries of a ticket (i.e. comments, e-mails) 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.

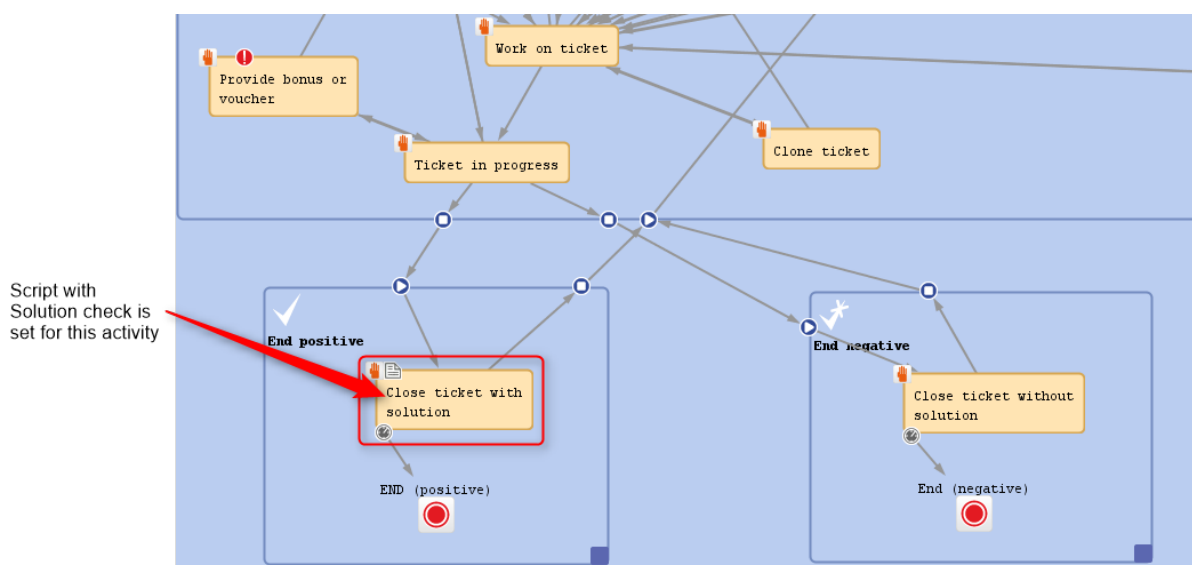


Figure 152: Workflow activity where the script is used

```
// 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 78: Workflow script to check if a solution has been defined in the ticket

The screenshot shows a web client interface for a ticket management system. At the top, a red error message reads: "Please enter solution first by setting a comment and marking it with the class of text Solution." Below this, the "Ticket" section is titled "Close ticket with solution" and prompts the user to "Enter the feedback of the customer". The form includes several dropdown menus: "Queue:" (set to "ServiceDesk"), "Assigned to:" (set to "ServiceDesk, Susan"), "Was the processing fast enough?" (set to "1: Yes"), "Were you treated kindly?" (set to "1: Yes"), and "Could the problem be solved?" (set to "Yes"). At the bottom of the form are "Save and continue" and "Cancel" buttons. On the right side, there are two panels: "Workflow activities" with buttons for "Close ticket with solution", "Close ticket without solution", and "Display List"; and "Workspace" which states "Workspace is empty. All your unsaved tasks are automatically listed in this workspace." Below these is a "Favorites" section.

Figure 153: Web Client: Error message if no solution has been defined in the ticket


D.13.3 Example: 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.

The screenshot displays a 'Ticket' window for 'testmail 1' (ID 100557). The ticket details include 'ServiceDesk | End positive', 'Assigned to ServiceDesk, Susan', 'Created: 8/10/16 3:59 PM', 'Priority: normal', and 'Desired deadline: 8/11/16 3:59 PM'. Below the details are sections for 'Groups' (Customers (1)), 'No additional engineers', 'No relations', and 'Calendar'. The 'Related Resources' section shows 'PCs' and 'Printers' tabs, with 'PC Desktop Relation (0)' listed. The 'History' section at the bottom shows a list of actions, with the most recent being '18 minutes ago #10 created by Susan ServiceDesk | Action'. A blue bar at the bottom of the history section contains a checkmark icon and the text 'This is the solution - where is the problem?'.


Figure 154: Service Desk ticket with solution

Ticket | Accept | Edit | Clone | Print

 **New FAQ ticket from ticket: testmail 1**
FAQs_active | defaultScope
100573 | Unassigned | Created: 8/12/16 11:23 AM

Customers (1) | Add | ^

Main

 **Skywalker Luke** ▼ Reseller
E-mail: katja@consol.de | Phone: 0211/1235678 ☎
VIP? no
CM.Track Login (LDAP): luke | CM.Track Password: *****

No additional engineers | Add | ^


No relations | Add | ^

Calendar | Add appointment | Refresh ▼

History | Comment | E-Mail | Attachment | Time booking | ^

Display communication ▼ | Sorting latest first ▼

Add comment, e-mail or attachment

1 minute ago |  **#1 created by Susan ServiceDesk | Action ▼**
Solution
This is the solution - where is the problem?

No attachments | ^

Code example 79: 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 80: Excerpt from the *postActivityExecutionScript*

i 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	231
D.14.2 Example 1: Attaching all attachments of a ServiceDesk ticket to the child ticket	231

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 e-mail with an attachment is sent to ConSol CM.
 - The e-mail 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. E-Mail 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 e-mail attachments in incoming e-mails. Other possible values are `OUTGOING_MAIL` or `DEFAULT`.

D.14.2 Example 1: Attaching all attachments of a ServiceDesk 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.

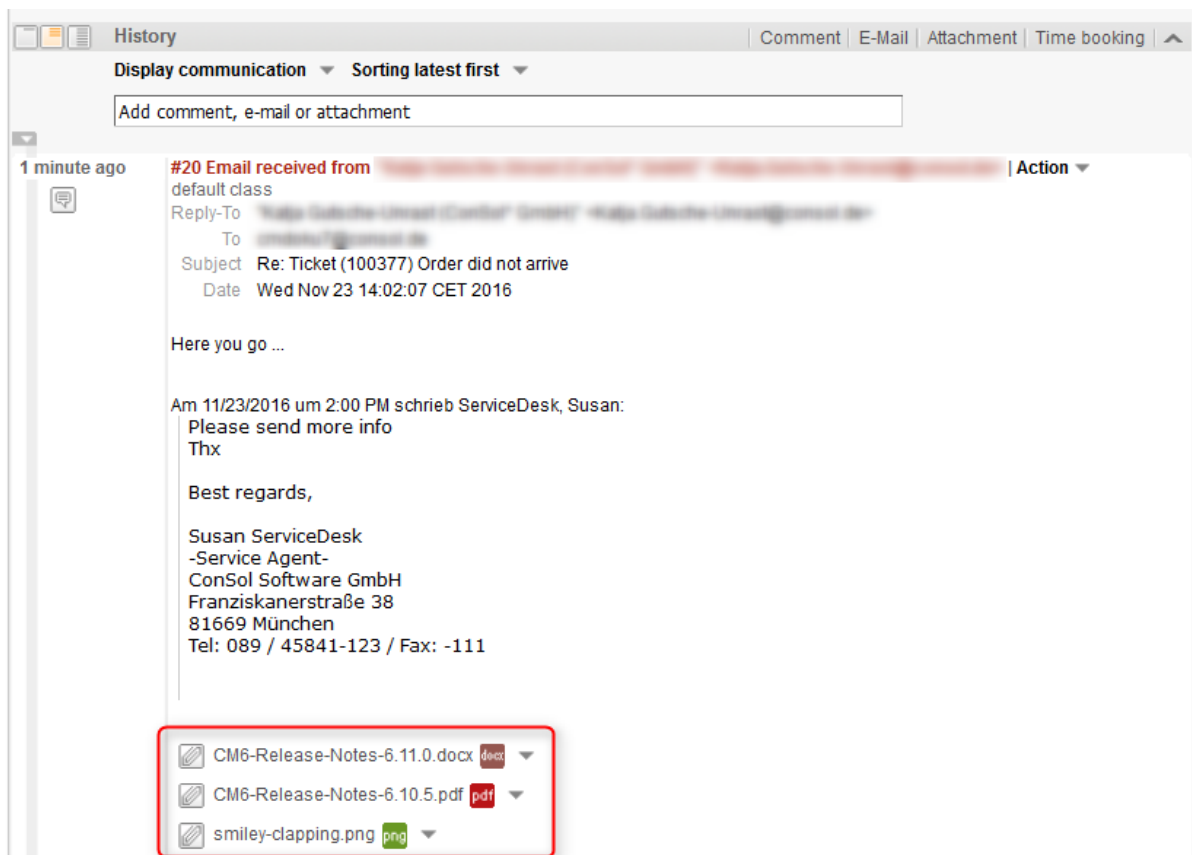


Figure 155: Web Client: ServiceDesk ticket with three attachments

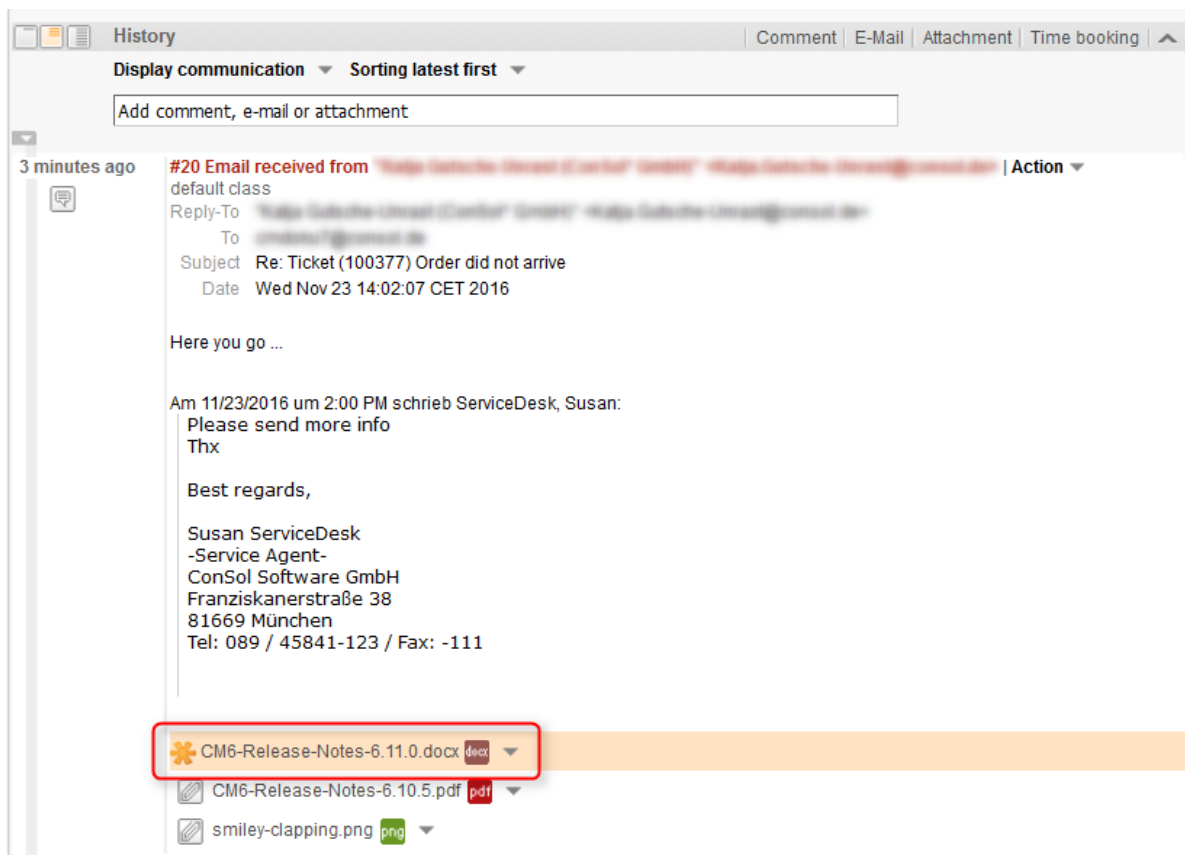


Figure 156: Web Client: ServiceDesk tickets with three attachments, one marked as Important attachment using the respective class of text

Ticket | Accept | Edit | Clone | Print

Task - Child of Ticket 100536 : Order did not arrive
 ServiceDeskSpecialTasks | defaultScope
 100612 | Unassigned | Created: 11/23/16 2:17 PM

Groups | Edit | ^
 Orders

Customers (1) | Add | ^
Main
 RetailCustomers
 Salutation: Mrs | Name: Marilyn | Last name: Monroe
 Street: Sunset Blvd. | House number: 111
 ZIP: 1234 | City: Los Angeles
 Email: marylin@hoolywwod.com

No additional engineers | Add | ^

Relations (1) | Add | ^

Calendar | Add appointment | Refresh | ^

Related Resources | ^

History | Comment | E-Mail | Attachment | Time booking | ^
 Display communication | Sorting latest first | ^
 Add comment, e-mail or attachment

11/23/16
 #1 created by Susan ServiceDesk | Action | ^
 14:17 default class
 I am a Child of ticket 100536

14:17 Attachment smiley-clapping.png png added
 14:17 Attachment CM6-Release-Notes-6.10.5.pdf pdf added
 14:17 Attachment CM6-Release-Notes-6.11.0.docx docx added

Attachments (3) | ^

Class	File type	Name	Description	Date	Added by	Action
Default class for attachments	docx	CM6-Release-Notes-6.11.0.docx		11/23/16 2:17 PM	Susan ServiceDesk	Show in history
Default class for attachments	pdf	CM6-Release-Notes-6.10.5.pdf		11/23/16 2:17 PM	Susan ServiceDesk	Show in history
Default class for attachments	png	smiley-clapping.png		11/23/16 2:17 PM	Susan ServiceDesk	Show in history

Figure 157: Web Client: Child ticket with the three attachments

```

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 81: *Admin Tool script called from a workflow activity: creating a child ticket and handing over the products list and all ticket attachments*

Since CM version 6.9.2.0, 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 82: *Alternative (additional) solution: handing over only the important attachments*

D.15 Searching for Tickets, Customers, and Resources Using the ConSol CM Workflow API

This chapter discusses the following:

D.15.1 Introduction	238
D.15.2 Searching for Tickets	239
D.15.3 Searching for Units (Contacts and Companies)	244
D.15.4 Searching for Resources	246

D.15.1 Introduction

In ConSol CM you can search the database for tickets or for units (contacts and companies). If your CM system contains the module CM.ResourcePool, 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.15.2 Searching for Tickets

To search for tickets you have to create the *TicketCriteria* object. For example, the following fields can be set (see also the respective *setter* methods in the following picture):

- Date of ticket creation
- Engineer
- System-specific Custom 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

void	<code>setCreationDateRange(DateRange pCreationDateRange)</code>
void	<code>setEngineerCriteria(TicketCriteria.EngineerCriteria pEngineerCriteria)</code>
void	<code>setExcludeIds(boolean pExcludeIds)</code>
void	<code>setFields(Set<AbstractField> pFields)</code>
void	<code>setHistoryCriteria(TicketCriteria.HistoryCriteria pHistoryCriteria)</code>
void	<code>setIdRange(org.apache.commons.lang.math.LongRange pIdRange)</code>
void	<code>setIds(Set<Long> pIds)</code>
void	<code>setModificationDateRange(DateRange modificationDateRange)</code>
void	<code>setName(String pName)</code>
void	<code>setPattern(String pPattern)</code>
void	<code>setPermission(QueuePermissionType pPermission)</code> Set permission to filter out the unwanted results (along with READ).
void	<code>setQueueIds(Set<Long> pQueueIds)</code>
void	<code>setResourceRelations(Set<ResourceRelationRelatedElementCriteria<?>> pResourceRelations)</code>
void	<code>setScopeIds(Set<Long> pScopeIds)</code>
void	<code>setStatus(TicketCriteria.Status pStatus)</code>
void	<code>setSubject(String pSubject)</code>
void	<code>setUserCriteria(Set<TicketUserCriteria> pUserCriteria)</code>

Figure 158: Setter methods of class *TicketCriteria*, Java API Doc, CM version 6.10.5.2

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.15.2.1 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 83: *Search for tickets (pseudocode)*

D.15.2.2 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 159: *Web Client: Selecting the module for a Service Desk ticket*

```
def mod = ticket.getField("helpdesk_standard", "module")
def crit = new TicketCriteria()
crit.setStatus(TicketCriteria.Status.OPEN)
Set<AbstractField> myfields = [mod]
crit.setFields(myfields)
List<Ticket> tics = ticketService.getByCriteria(crit)
tics.each() { tic ->
    println 'Next Ticket subject is now ' + tic.subject
    // do whatever is required with the tickets
}
```

Code example 84: *Find tickets with the same module as the current ticket*

D.15.2.3 Example 3: Search for Tickets by Unit

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 85: *Search for tickets by unit*

D.15.2.4 Example 4: Search for Tickets by Engineer to Avoid Work Overload for Engineers

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.

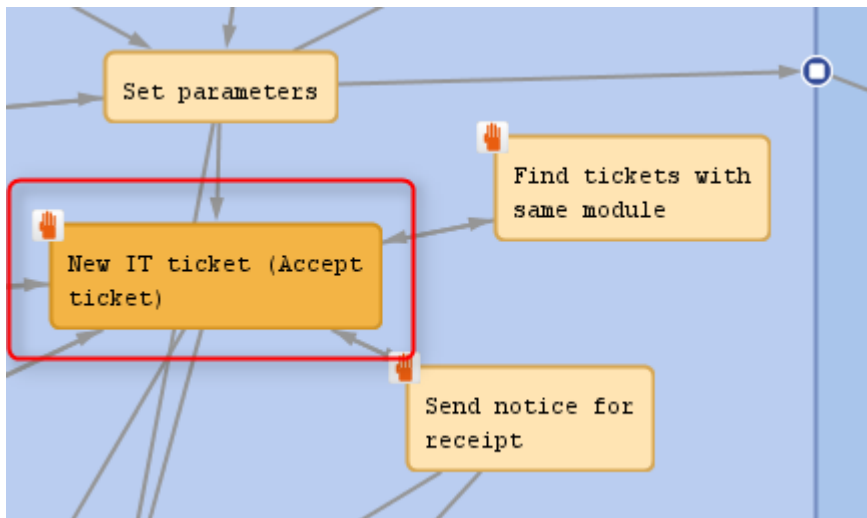


Figure 160: Workflow activity which contains the ticket number control script

```
// 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
if (tic_number > max_tics) {

    workflowApi.addValidationError("INFO","You have too many tickets (" + tic_number
+ ") already, so you cannot accept another ticket. Maximum allowed number is "
+ max_tics)

} else {
    ticket.setEngineer(curr_eng)
}
```

Code example 86: Script of activity "New IT ticket (Accept ticket)"

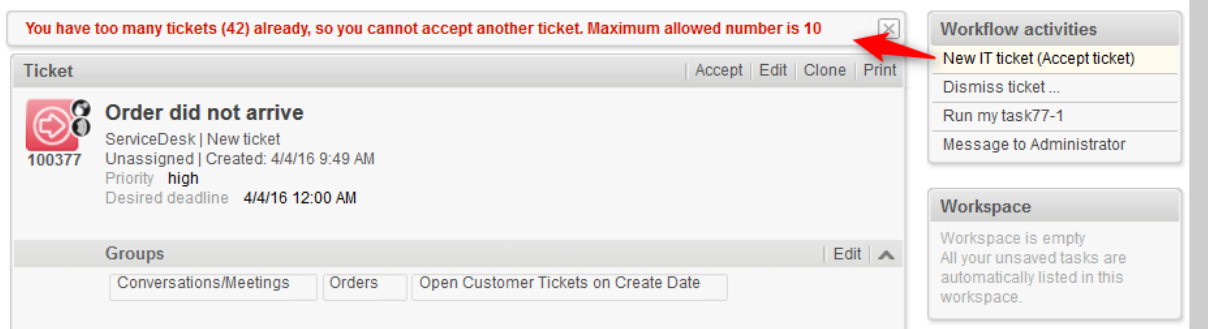


Figure 161: Web Client: Control of the number of tickets which can be assigned to one engineer

D.15.3 Searching for Units (Contacts and Companies)

To search for units (i.e. for contacts and/or companies) you have to create the *UnitCriteria* object. For example, the following fields can be set (see also the respective *setter* methods in the following picture):

- Customer group
- System-specific Data Object Group Fields
- Unit IDs
- Patterns for units
- Phone number (new in CM version 6.9.3, used for CM.Phone)
- TicketCriteria
- UnitDefinition name
- Boolean UseInCriterion

Then you use the *unitService* to get the search result.

void	setActive(Boolean pActive)
void	setCustomerGroupIds(Set<Long> pCustomerGroupIds)
void	setExcludeIds(boolean pExcludeIds)
void	setFields(Set<AbstractField> pFields)
void	setGroupNames(Set<String> pGroupNames)
	Deprecated.
void	setIdRange(org.apache.commons.lang.math.LongRange pIdRange)
void	setIds(Set<Long> pIds)
void	setPattern(String pPattern)
void	setPermission(CustomerGroupPermissionType pPermission)
	Set permission to filter out the unwanted results (along with READ).
void	setPhoneNumber(String pPhoneNumber)
void	setResourceRelations(Set<ResourceRelationRelatedElementCriteria<?>> pResourceRelations)
void	setTicketCriteria(Set<AbstractField> pCallUnitReferences, TicketCriteria pTicketCriteria)
void	setUnitDefinitionNames(Set<String> pUnitDefinitionNames)
void	setUseInCriterion(boolean pUseInCriterion)

Figure 162: Setter methods of class *UnitCriteria*, Java API Doc, CM version 6.10.5.2

D.15.3.1 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 87: Search for contacts by first name and last name

D.15.3.2 General Syntax for Unit 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 88: *Search for units by enum value (general syntax)*

D.15.3.3 Example 2: Search for Units 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 89: *Search for units by enum value (example)*

D.15.4 Searching for Resources

To search for resources, you have to create the *ResourceCriteria* object. For example, the following fields can be set (see also the respective *setter* methods in the following picture):

- Date range
- Excluded resource ids
- Resource group ids of the desired resources

Then you use the *resourceService* to get the search result.

void	<code>setAccessModeDateRange(DateRange pAccessModeDateRange)</code>
void	<code>setActive(Boolean pActive)</code>
void	<code>setExcludeIds(boolean pExcludeIds)</code>
void	<code>setFields(Set<AbstractField> pFields)</code>
void	<code>setIdRange(org.apache.commons.lang.math.LongRange pIdRange)</code>
void	<code>setIds(Set<Long> pIds)</code>
void	<code>setModificationDateRange(DateRange pModificationDateRange)</code>
void	<code>setPattern(String pPattern)</code>
void	<code>setPermission(ResourceTypePermissionType pPermission)</code> Set permission to filter out the unwanted results (along with READ).
void	<code>setResourceGroupsIds(Set<Long> pResourceGroupsIds)</code>
void	<code>setResourceGroupsTechnicalNames(Set<String> pResourceGroupsTechnicalNames)</code>
void	<code>setResourceRelations(Set<ResourceRelationRelatedElementCriteria<?>> pResourceRelations)</code>
void	<code>setResourceTypesIds(Set<Long> pResourceTypesIds)</code>
void	<code>setResourceTypesTechnicalNames(Set<String> pResourceTypesTechnicalNames)</code>

Figure 163: Setter methods of class *ResourceCriteria*, Java API Doc, CM version 6.10.5.2

D.15.4.1 Creating an IT Inventory List and Writing it as 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.

Ticket

[Edit](#) | [Clone](#) | [Print](#)

100174

Inventory (IT), 4711

ServiceDeskSpecialTasks | defaultScope/TaskInProgress
Assigned to Testen, Zum | Created: 12/16/14 3:02 PM
Deadline 8/18/16

Groups

[Edit](#) ▾

Customers (1)

[Add](#) ▾

No additional engineers

[Add](#) ▲

No relations

[Add](#) ▲

Calendar

[Add appointment](#) | [Refresh](#) ▾

Related Resources

[Add](#) ▾

History

[Comment](#) | [Email](#) | [Attachment](#) | [Time booking](#) ▾

Display communication ▾

Sorting latest first ▾

Add comment, e-mail or attachment

8/12/16

#9 created by Susan ServiceDesk | Action ▾
15:27 default class

1234567890(MS_Word2013)
Best Printer ever(HP_Printer)
Mein Lieblingsdrucker(HP_Printer)
My cool PC(PC_Desktops)
My new HP printer(HP_Printer)
My new HP printer(HP_Printer)
My new PC Number 1(PC_Desktops)
MyNewPC desktop(PC_Desktops)

Workflow activities

Finish task

Check Asset Base -> List in Ticket

Workspace

Workspace is empty
All your unsaved tasks are automatically listed in this workspace.

Favorites

Inventory (IT), 4711

MyCustomerGroup

Question about Order #4711

myOpenTickets

Mia Skydriever

Figure 164: 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()
    // println '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 90: *Print an IT asset (resource) list into the ticket*

D.16 Debug Information

This chapter discusses the following:

D.16.1 Introduction	250
D.16.2 Using Statements for Debug Output	251

D.16.1 Introduction

Sometimes you might want to check the output of a workflow or Admin Tool script by using debug output into log files. In ConSol CM, the debug output usually is written to *server.log* which (in a standard configuration) is located in the following path:

- **In JBoss 5:**
<JBoss_HOME>\log\server.log
- **In JBoss 7 (single instance):**
<JBoss_HOME>/standalone/server.log
- **In Oracle WebLogic:**
<DOMAIN_HOME>\cm-logs and
<DOMAIN_HOME>\cmrf-logs\server.log

The logging configuration can be changed by editing the respective configuration file:

- In JBoss 5:
<JBoss_HOME>/conf/jboss-log4j.xml
- In JBoss 7:
<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 Set-Up Manual.

As an alternative, you can write information into the ticket as text.

D.16.2 Using Statements for Debug Output

D.16.2.1 Debug Output to 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.

D.16.2.2 Debug Output as Text Entry in 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.16.2.3 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 91: *Debug entry in ConSol CM standard e-mail 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: e-mail) 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 - Best Practices

This chapter discusses the following:

E.1 The Basic Organization of a Workflow: Using Scopes	253
E.1.1 Variant A: Use of a Global Scope	254
Variant B: Use of Three or More Main Scopes	256
E.2 The Position of the START Node	258
E.3 Store Some Workflow Scripts in the Admin Tool	259
E.3.1 When to Use Admin Tool Workflow Scripts	261
E.3.2 How to Use Admin Tool Workflow Scripts	262
E.4 Consider the Use of Trigger Combinations Well	263
E.5 Do Not Trigger Ticket Update Events If Not Really Required	266
E.6 How to Use the Disable Auto Update Parameter	267
E.7 Avoid Self-Triggering Business Event Triggers	269

E.1 The Basic Organization of a Workflow: Using Scopes

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.

E.1.1 Variant A: Use of 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 e-mail 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 e-mail should only be supervised 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!

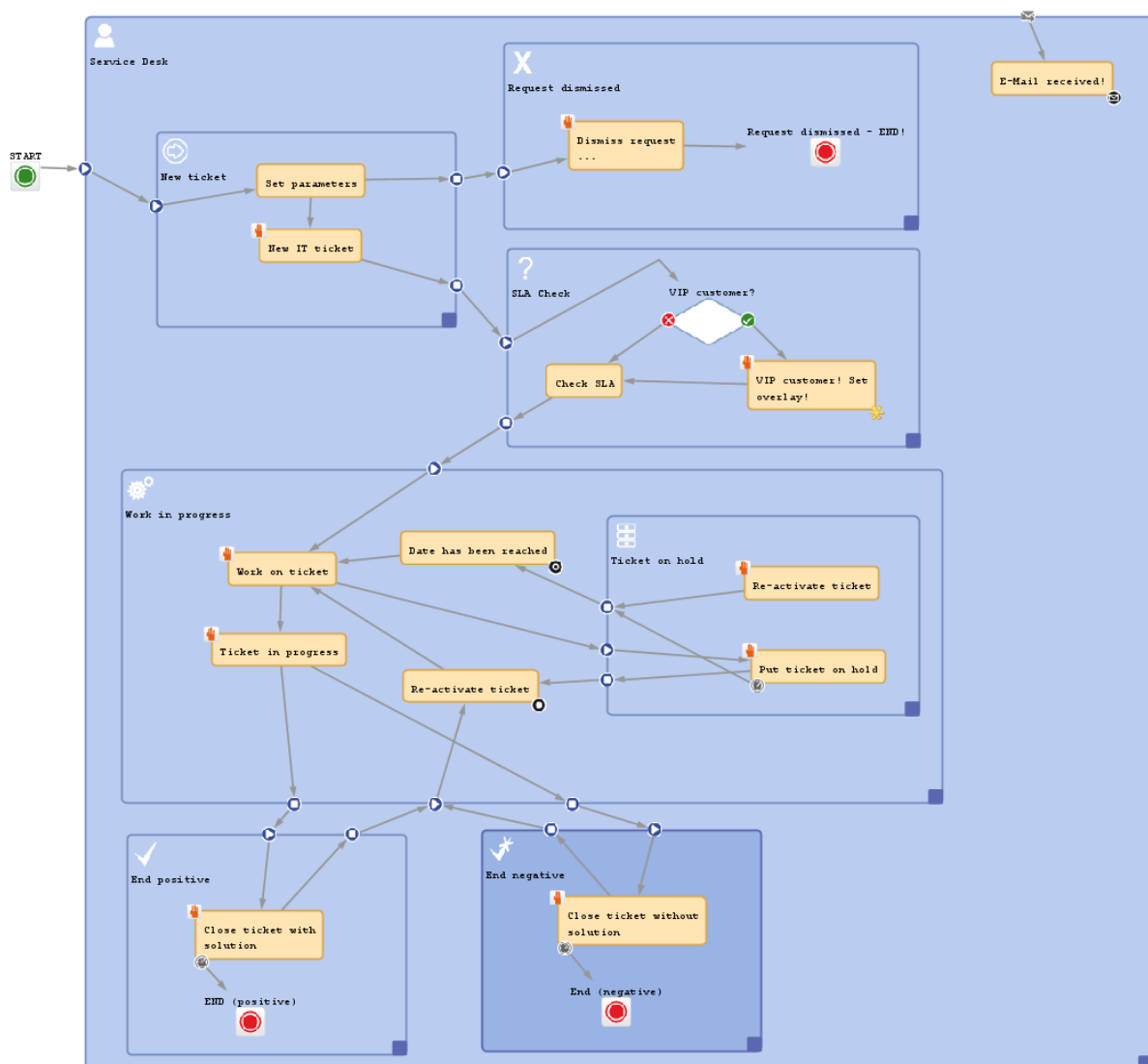


Figure 165: ConSol CM Process Designer - Workflow with global scope

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 business 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.

Variant B: Use of 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.

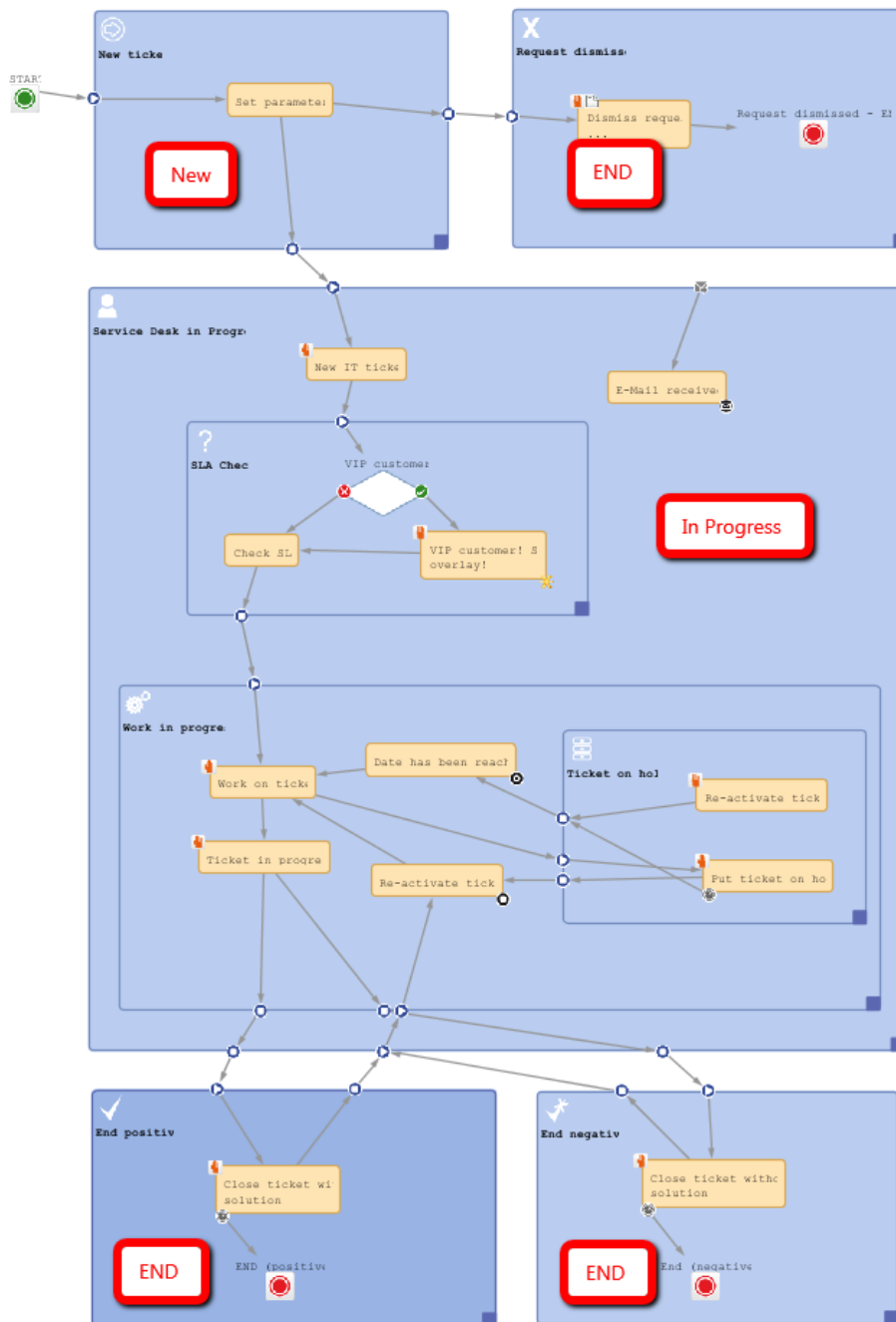


Figure 166: ConSol CM Process Designer - Workflow with three types of main scopes

E.2 The Position of the START Node

The best position of the START Node (see section [Workflow Components: START Node](#)) 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.

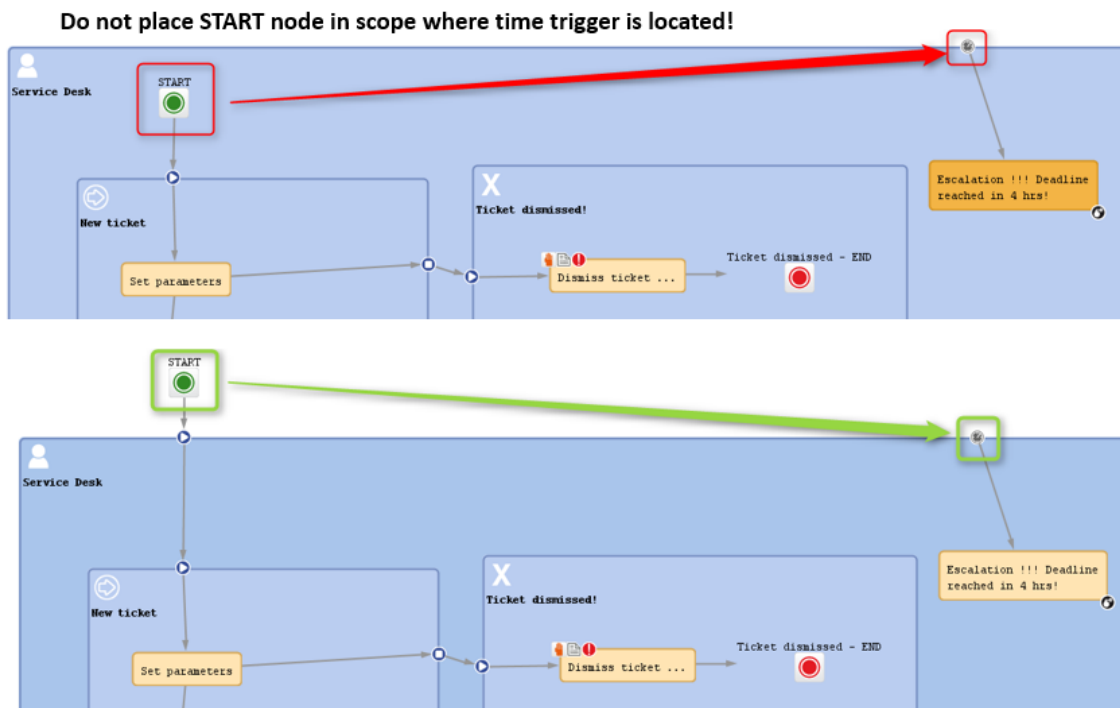


Figure 167: ConSol CM Process Designer - Position of START node

E.3 Store Some Workflow Scripts in the Admin Tool

For scripts, which are used over and over again in workflow activity and/or precondition scripts, it might be better to store them in the *Script* section of the Admin Tool and call them from the workflow script.

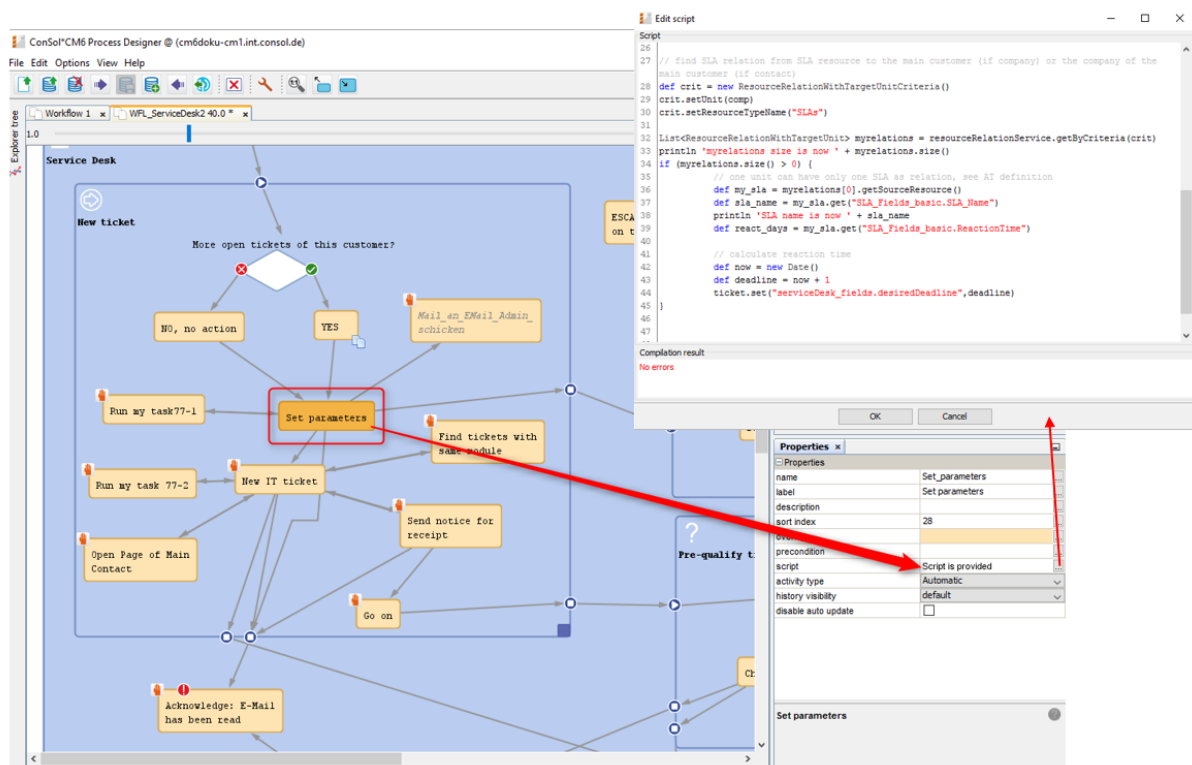


Figure 168: Calling a script directly in the workflow

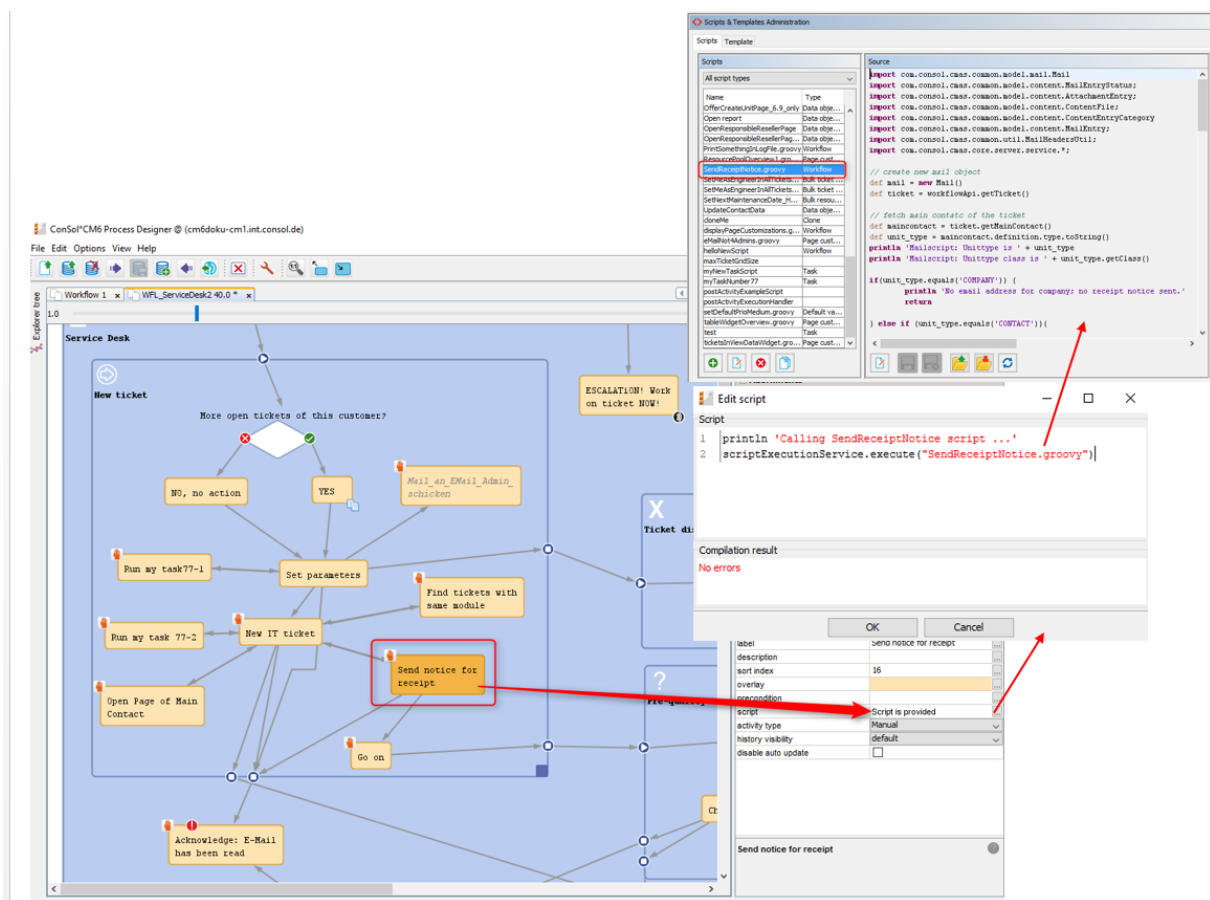


Figure 169: Calling an Admin Tool script from a workflow activity

E.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.
- The Script Editor in the Admin Tool is not as comfortable as the Workflow Script Editor.
- 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.

E.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 is always called from the workflow using the interface *ScriptProvider*.

```
def scriptProvider = scriptProviderService.createDatabaseProvider
("scriptName.groovy")
def r = scriptExecutionService.execute(scriptProvider)
```

Code example 92: *Calling an Admin Tool script from the workflow (only way in CM versions 6.10.4 and older, in CM 6.10.5 and up still available)*

```
// Create the scriptProvider for the required Admin Tool script, here
"scriptName.groovy"
def scriptProvider = scriptProviderService.createDatabaseProvider
("scriptName.groovy")
// 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(scriptProvider, params)
```

Code example 93: *Calling an Admin Tool script from the workflow with use of parameters (only way in CM versions 6.10.4 and older, in CM 6.10.5 and up still available)*

```
scriptExecutionService.execute("MyScript")
```

Code example 94: *Calling an Admin Tool script from the workflow (only CM versions 6.10.5 and up)*



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`.

E.4 Consider the Use of Trigger Combinations Well



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.

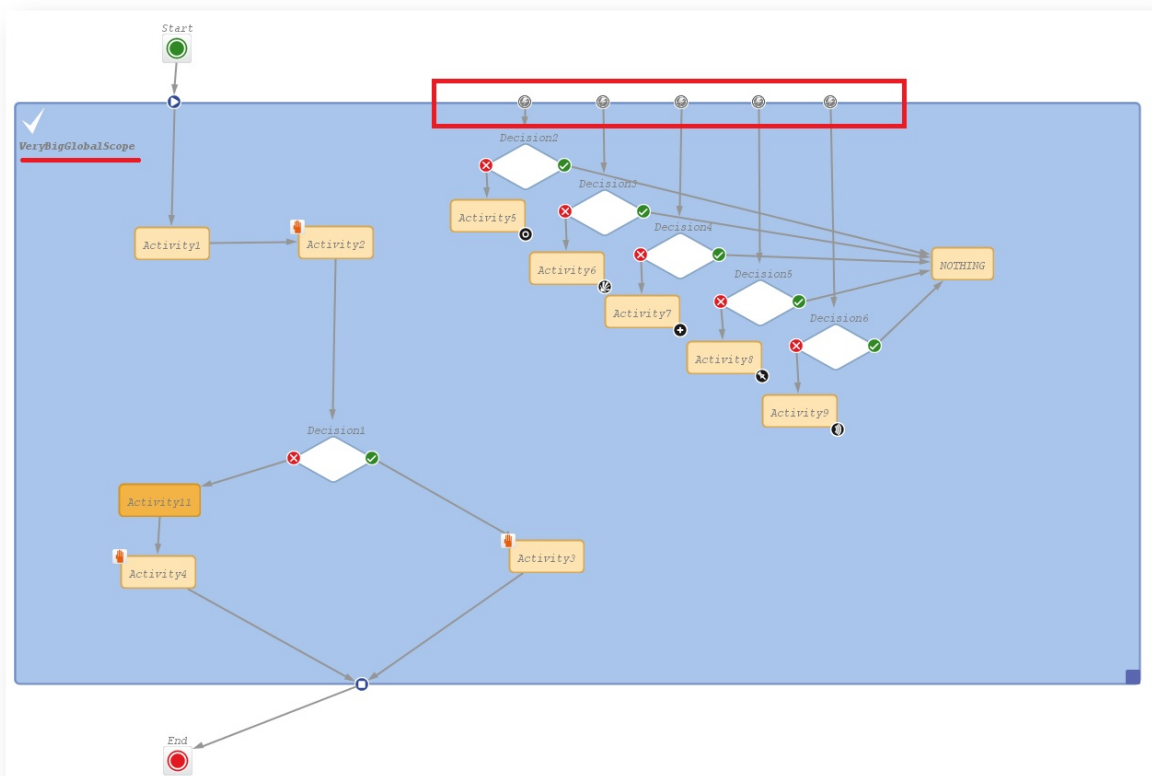


Figure 170: ConSol CM Process Designer - Scope with triggers

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).

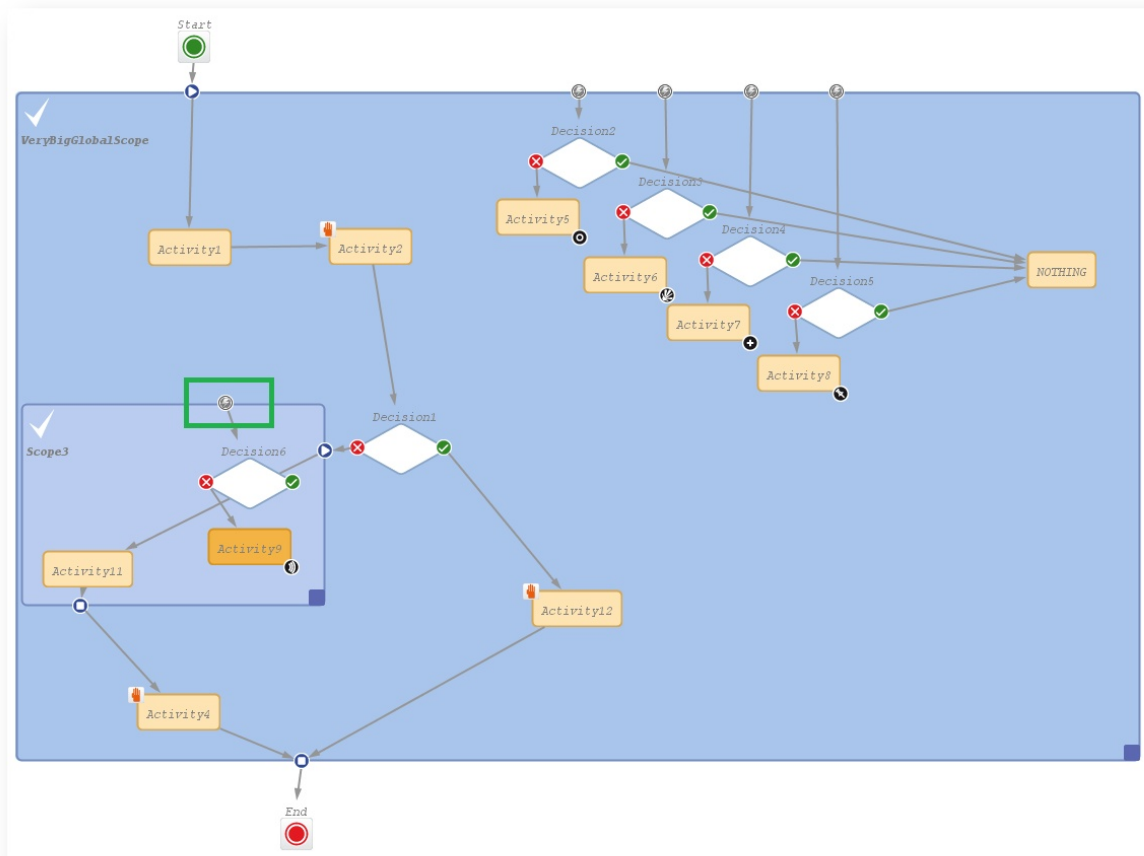


Figure 171: ConSol CM Process Designer - Move trigger to smaller scope

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.

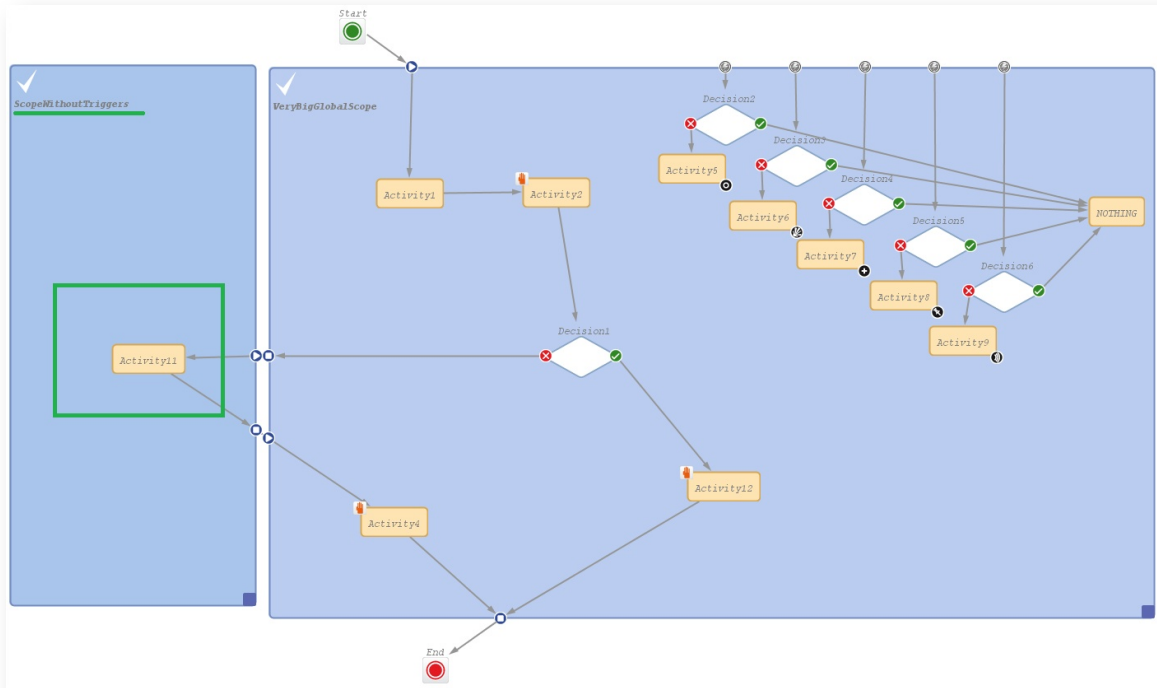


Figure 172: ConSol CM Process Designer - Separate scopes with and without triggers

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!

E.5 Do Not Trigger Ticket Update Events If Not Really Required



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 95: *Code which triggers TicketUpdateEvent*

```
//this method does NOT throw a TicketUpdateEvent!  
ticket.setEngineer(workflowApi.currentEngineer)
```

Code example 96: *Code which does not trigger TicketUpdateEvent*

E.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!)

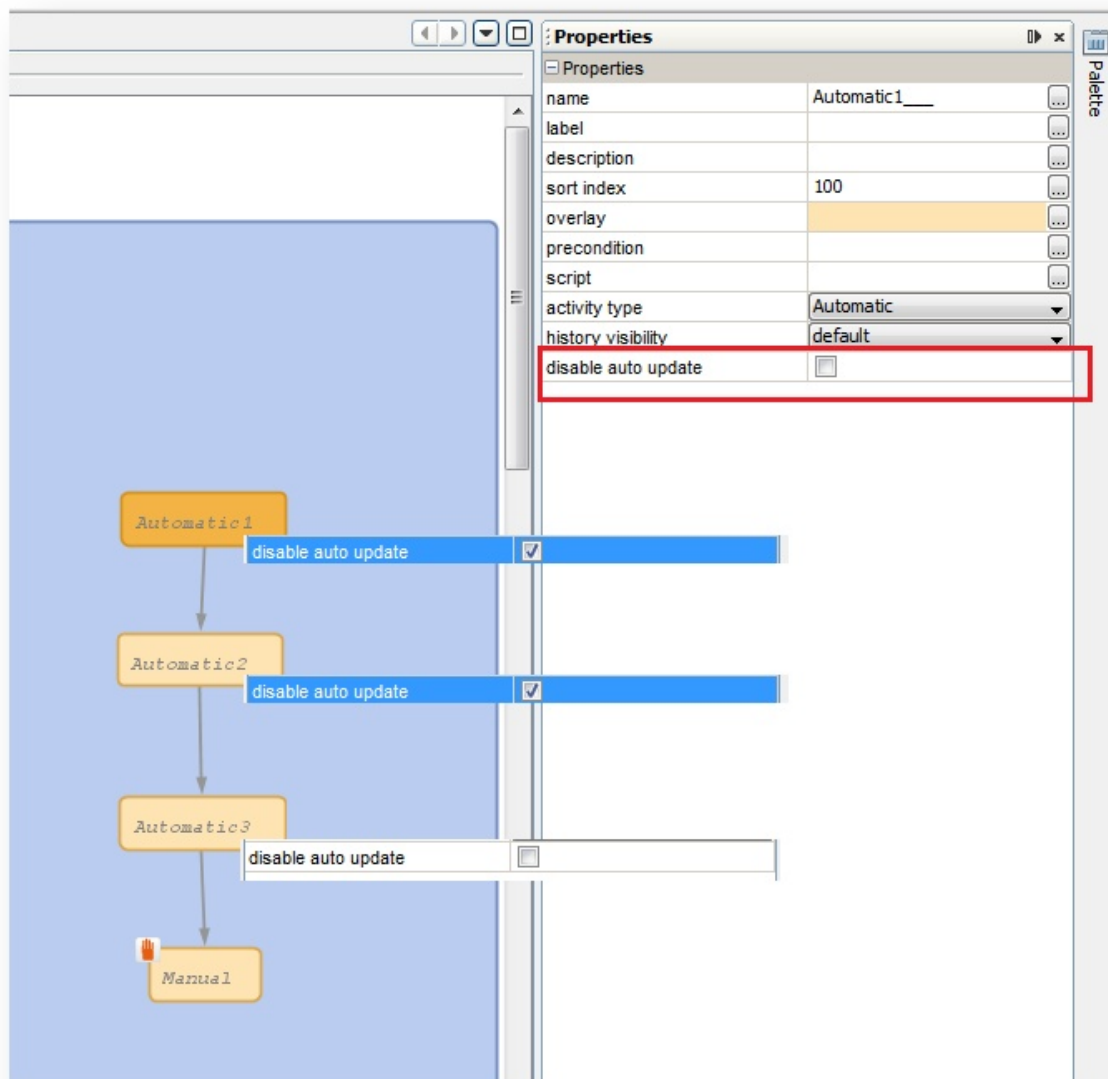


Figure 173: ConSol CM Process Designer - Activities with "disable auto update" option

E.7 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.

Business Event Trigger reacts to change of parameter XY

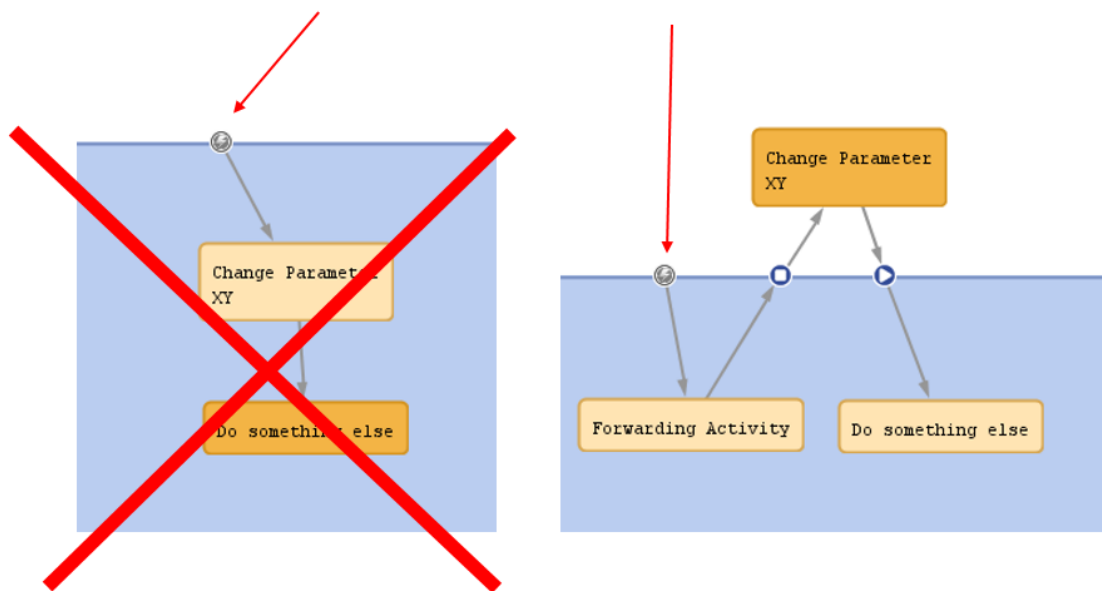


Figure 174: ConSol CM Process Designer - Avoiding self-triggering business event triggers

F - Deploying Workflows

This chapter discusses the following:

F.1 Introduction and Workflow Life Cycle	271
F.2 Engineer Permissions Required for Workflow Deployment	272
F.3 Actions During Workflow Deployment	273

F.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](#).

F.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
- **Workflow Permissions:**
Deploy workflow

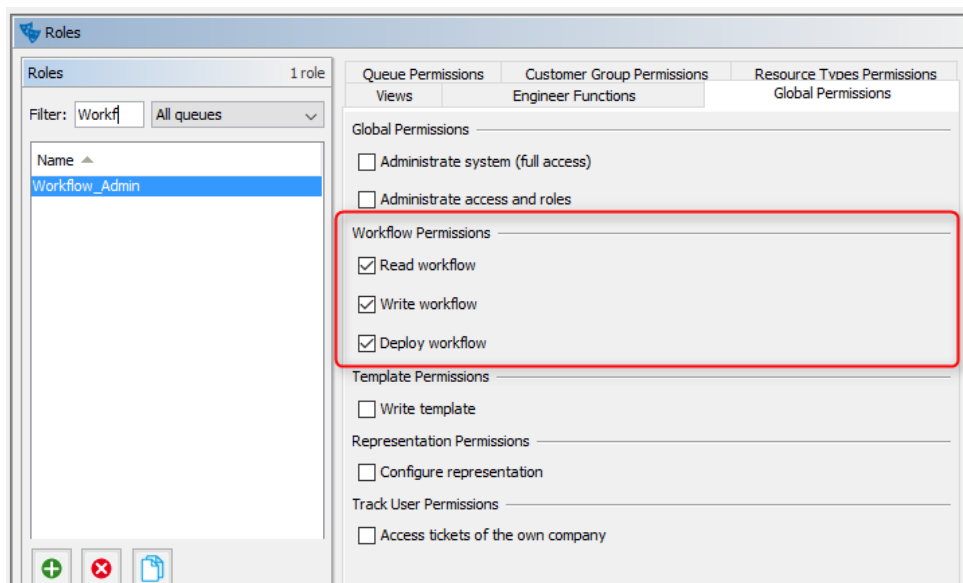


Figure 175: ConSol CM Admin Tool - Engineer permissions for deploying workflows

F.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:

1. Stay as close as possible to the previous position (default).
2. 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.

G - Appendix

This section contains several appendices:

- [Annotations](#)
- [System Properties](#)
- [Administrator and Notification E-Mail Addresses](#)
- [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:

groupable	279
sortable	279
leave-trailing-zeros	279
readonly	279
visibility	279
boolean-type	280
enum-in-search-type	280
enum-type	280
list-type	280
text-type	280
ldapid	281
password	282
username	282
dwh-no-history-field	282
reportable	282
field indexed	282
colspan	283
field-group	283
fieldsize	283
label-group	284
label-in-view	284
order-in-result	284
position	284
rowspan	284
show-label-in-edit	285
show-label-in-view	285
show-tooltip	285
show-watermark	285
ticket-list-colspan	285
ticket-list-position	285
ticket-list-rowspan	286

no-history-field	286
dialable	286
resource-color	286
contact search result column	287
contains contacts	287
enum field with ticket color	287
accuracy	287
email	288
format	288
matches	288
maxLength	288
maxValue	288
minLength	288
minValue	289
required	289
visibility configuration	289

G.1.1.1 cmweb-common (type)

groupable

- **type:** cmweb-common
- **description:** Enables grouping in the ticket list.
- **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.
- **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.1.1.2 common (type)

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 Custom Field cannot be modified.
- **values:**
 - *true / false*: Field is read-only if value is set to *true*. Lack of value, or any value except *false*, is treated as *true*.

visibility

- **type:** common
- **description:** Defines when the field is visible.
- **values:**
 - *edit*: Field will be displayed in edit mode.
 - *view*: Field will be displayed in view mode.
 - *none*: Field is not visible.

- If any other, or no value, is set then the field will always be visible.

G.1.1.3 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.
- *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+)(?:\:(?:\| |)?(\.*)?)?$"`
 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.1.1.4 contact authentication (type)

ldapid

- **type**: contact authentication
- **description**: Used in a Data Object Group of type *contact*, for the Data Object Group 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 authentication
- **description:** Indicates that this field will be used as a login name in the authentication process.
- **values:**
 - *true / false*: Used for CM.Track.

G.1.1.5 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.1.1.6 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.

- *not indexed*: 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.1.1.7 layout (type)

colspan

- **type**: layout
- **description**: Defines how many columns are reserved for the field in the layout.
- **values**:
 - *<number>*: Number of columns.

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.

fieldsize

- **type**: layout
- **description**: Displayed field size within ticket layout.
- **values**:
 - *<rows>;<cols>*: Displayed field size.
 Format for *string* fields and *number* fields: n indicates the number of characters, for string fields this is the number of monospaced capital M characters.
 Format for *textarea*: rows;cols (corresponds to *<textarea rows="" cols="">*).
 Enums are displayed as a choice box with n elements, instead of a drop-down. Format for enums: n.
Note: this is only a layout configuration, for validation use *maxlength* of type *group validation*.
 - *<number>*: For *enum* data fields. Defines how many values are directly visible in the list box. Used only for layout purposes.

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 and Indexer 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.

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.

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.

G.1.1.8 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.1.1.9 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.1.1.10 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.1.1.11 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!
- **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.1.1.12 ticket contact relation type (type)

contains contacts

- **type:** ticket contact relation type
- **description:** Used only for list field definition, indicates that it can hold unit references to units annotated as contacts.
- **values:**
 - *true/false:* Value type is boolean. Specifies whether the list is shown with the contact (true) or with the ticket (false).

G.1.1.13 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.1.1.14 validation (type)

accuracy

- **type:** validation
- **description:** For data fields, 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 e-mail 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.

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.1.1.15 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	291
dwh-no-history	291
reportable group	291
auto-open-group	291
show-contact-in-ticket-list	291
show-in-group-section	292
show-labels-in-edit	292
show-labels-in-view	292
show-tooltips	292
show-watermarks	292
no-history	292
resource-fields-group-mode	293
resource-custom-fields-group-mode	293
unit is a contact	293

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.
 - *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 Groups 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 historized. 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 Group 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 Group Field is not editable in the Web Client if value is *external*.
Since 6.10.5.0

G.1.2.6 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](#)

G.2.1 Alphabetical List of System Properties

This chapter describes the following properties:

admin.email	303
admin.login	303
admin.tool.session.check.interval	303
attachment.allowed.types	303
attachment.max.size	304
attachment.upload.timeout	304
authentication.method	304
autocommit.cf.changes	305
autocomplete.enabled	305
automatic.booking.enabled	305
batch-commit-interval	306
big.task.minimum.size	306
cache-cluster-name	306
checkUserOnlineIntervalInSeconds	306
cluster.mode	307
cmas.dropSchemaBeforeSetup	307
cmoffice.enabled	307
commentRequiredForTicketCreation	308
communication.channel	308
config.data.version	308
contact.authentication.method	308
contact.inherit.permissions.only.to.own.customer.group	309
customizationVersion	309
data.directory	309
data.optimization	310
database.notification.enabled	310
database.notification.redelivery.delay.seconds	310
database.notification.redelivery.max.attempts	311
defaultCommentClassName	311
defaultContentEntryClassName	311
defaultIncommingMailClassName	311

defaultNumberOfCustomFieldsColumns	312
defaultOutgoingMailClassName	312
delete.ticket.enabled	312
diffTrackingEnabled	313
disable.admin.task.auto.commit	313
dwh.mode	313
esb.directory	313
eviction.event.queue.size	314
eviction.max.nodes	314
eviction.wakeup.interval	314
favoritesSizeLimit	315
fetchLock.interval	315
fetchSize.strategy	315
fetchSize.strategy.FetchSizeFixedStrategy.value	315
fetchSize.strategy.FetchSizePageBasedStrategy.limit	316
fetchSize.strategy.FetchSizeThresholdStrategy.value	316
filesystem.polling.threads.number	316
filesystem.polling.threads.shutdown.timeout.seconds	317
filesystem.polling.threads.watchdog.interval.seconds	317
filesystem.task.enabled	317
filesystem.task.interval.seconds	317
filesystem.task.polling.folder	318
filesystem.task.timeout.seconds	318
filesystem.task.transaction.timeout.seconds	318
globalSearchResultSizeLimit	319
helpFilePath	319
hibernate.dialect	319
hideTicketSubject	319
ignore-queues	320
index.attachment	320
index.history	320
index.status	321
index.task.worker.threads	321

index.version.current	321
index.version.newest	321
indexed.assets.per.thread.in.memory	322
indexed.engineers.per.thread.in.memory	322
indexed.resources.per.thread.in.memory	322
indexed.tickets.per.thread.in.memory	323
indexed.units.per.thread.in.memory	323
initialized	323
is.cmrf.alive	324
java.naming.factory.initial	324
java.naming.factory.url.pkgs	324
java.naming.provider.url	325
jobExecutor.adminMail	325
jobExecutor.idleInterval	325
jobExecutor.idleInterval.seconds	325
jobExecutor.jobExecuteRetryNumber	326
jobExecutor.jobMaxRetries	326
jobExecutor.jobMaxRetriesReachedSubject	326
jobExecutor.lockingLimit	327
jobExecutor.lockTimeout.seconds	327
jobExecutor.mailFrom	327
jobExecutor.maxInactivityInterval.minutes	327
jobExecutor.threads	328
jobExecutor.timerRetryInterval	328
jobExecutor.timerRetryInterval.seconds	328
jobExecutor.txTimeout.seconds	329
kerberos.v5.enabled	329
kerberos.v5.username.regex	329
last.config.change	329
ldap.authentication	330
ldap.basedn	330
ldap.certificate.basedn	330
ldap.certificate.content.attribute	331

ldap.certificate.password	331
ldap.certificate.providerurl	331
ldap.certificate.searchattr	331
ldap.certificate.userdn	332
ldap.contact.name.basedn	332
ldap.contact.name.password	332
ldap.contact.name.providerurl	333
ldap.contact.name.searchattr	333
ldap.contact.name.userdn	333
ldap.initialcontextfactory	333
ldap.password	334
ldap.providerurl	334
ldap.searchattr	334
ldap.userdn	334
mail.attachments.validation.info.sender	335
mail.attachments.validation.info.sender	335
mail.attachments.validation.info.subject	335
mail.attachments.validation.info.subject	336
mail.callname.pattern	336
mail.cluster.node.id	336
mail.db.archive	337
mail.db.archive	337
mail.delete.read	337
mail.encryption	338
mail.error.from.address	338
mail.error.to.address	338
mail.from	338
mail.incoming.uri	338
mail.max.restarts	339
mail.mime.strict	339
mail.mule.service	339
mail.notification.engineerChange	340
mail.notification.sender	340

mail.on.error	340
mail.polling.interval	341
mail.process.error	341
mail.process.error	341
mail.process.retry.attempts	341
mail.process.timeout	342
mail.redelivery.retry.count	342
mail.reply.to	342
mail.sender.address	343
mail.smtp.email	343
mail.smtp.envelopesender	343
mail.ticketname.pattern	344
mailbox.1.connection.host	344
mailbox.1.connection.password	344
mailbox.1.connection.port	344
mailbox.1.connection.protocol	344
mailbox.1.connection.username	344
mailbox.2.connection.host	344
mailbox.2.connection.password	345
mailbox.2.connection.port	345
mailbox.2.connection.protocol	345
mailbox.2.connection.username	345
mailbox.default.connection.host	345
mailbox.default.connection.password	345
mailbox.default.connection.port	346
mailbox.default.connection.protocol	346
mailbox.default.connection.username	346
mailbox.default.session.mail.debug	347
mailbox.default.session.mail.imap.timeout	347
mailbox.default.session.mail.mime.address.strict	347
mailbox.default.session.mail.pop3.timeout	347
mailbox.default.session.mail.<protocol>.partialfetch	348
mailbox.default.task.delete.read.messages	348

mailbox.default.task.enabled	348
mailbox.default.task.interval.seconds	349
mailbox.default.task.max.message.size	349
mailbox.default.task.max.messages.per.run	349
mailbox.default.task.timeout.seconds	350
mailbox.default.task.transaction.timeout.seconds	350
mailbox.polling.threads.mail.log.enabled	350
mailbox.polling.threads.number	350
mailTemplateAboveQuotedText	351
max.licences.perUser	351
maxSizePerPagemapInMegaBytes	351
monitoring.engineer.login	352
monitoring.unit.login	352
nimh.enabled	352
notification.error.description	352
notification.error.from	353
notification.error.subject	353
notification.error.to	353
notification.finished_successfully.description	353
notification.finished_successfully.from	354
notification.finished_successfully.subject	354
notification.finished_successfully.to	354
notification.finished_unsuccessfully.description	355
notification.finished_unsuccessfully.from	355
notification.finished_unsuccessfully.subject	355
notification.finished_unsuccessfully.to	355
notification.host	356
notification.password	356
notification.port	356
notification.protocol	356
notification.username	357
outdated.lock.age	357
pagemapLockDurationInSeconds	357

policy.password.age	358
policy.password.pattern	358
policy.rotation.ratio	358
policy.username.case.sensitive	358
postActivityExecutionScriptName	359
queue.polling.threads.number	359
queue.polling.threads.shutdown.timeout.seconds	359
queue.polling.threads.watchdog.interval.seconds	360
queue.task.error.pause.seconds	360
queue.task.interval.seconds	360
queue.task.max.retries	360
queue.task.timeout.seconds	361
queue.task.transaction.timeout.seconds	361
queuesExcludedFromGS	361
refreshTimeInCaseOfConcurrentRememberMeRequests	362
rememberMeLifetimeInMinutes	362
request.scope.transaction	362
resource.replace.batchSize	362
resource.replace.timeout	363
scene	363
script.logging.threshold.seconds	363
searchPageSize	364
searchPageSizeOptions	364
server.session.archive.reaper.interval	364
server.session.archive.timeout	364
server.session.reaper.interval	365
server.session.timeout	365
serverPoolingInterval	366
skip-ticket	366
skip-ticket-history	366
skip-unit	367
skip-unit-history	367
skip.wfl.transfer.cleanup	367

split.history	368
start.groovy.task.enabled	368
supportEmail	368
synchronize.master.address	368
synchronize.master.security.token	369
synchronize.master.security.user	369
synchronize.master.timeout.minutes	369
synchronize.megabits.per.second	370
synchronize.sleep.millis	370
task.panel.refresh.interval.seconds	370
themeOverlay	371
ticket.delete.timeout	371
ticketListRefreshIntervalInSeconds	371
ticketListSizeLimit	372
tickets.delete.size	372
transaction.timeout.minutes	372
unit.replace.batchSize	372
unit.replace.timeout	373
unit.transfer.order	373
unitIndexSearchResultSizeLimit	373
unused.content.remover.cluster.node.id	374
unused.content.remover.enabled	374
unused.content.remover.polling.minutes	374
unused.content.remover.ttl.minutes	374
urlLogoutPath	375
warmup.executor.enabled	375
webSessionTimeoutInMinutes	375
wicketAjaxRequestHeaderFilterEnabled	376

admin.email

- **Module:** cmas-core-security
- **Description:** The e-mail 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.session.check.interval

- **Module:** cmas-app-admin-tool
- **Description:** Admin Tool inactive (ended) sessions check time interval (in seconds)
- **Type:** integer
- **Restart required:** yes
- **System:** yes
- **Optional:** no
- **Example value:** 30
- **Since:** 6.7.5

attachment.allowed.types

- **Module:** cmas-core-server
- **Description:** Comma-separated list of allowed filename extensions (if no value defined, all file extensions are allowed).
- **Type:** string

- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** txt,zip,doc
- **Since:** 6.5.0

attachment.max.size

- **Module:** cmas-core-server
- **Description:** Maximum attachment size, in MB. This is a validation property of the CM API. It controls the size of attachments at tickets, at units, and at resources. It also controls the size of incoming (not outgoing!) e-mail attachments in NIMH as well as in Mule/ESB mode.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 100
- **Since:** 6.4.0

attachment.upload.timeout

- **Module:** cmweb-server-adapter
- **Description:** Defines the transaction timeout in minutes for adding attachments to a ticket, a resource or a customer. Counts the time for the upload of all attachments of one transaction. When the timeout occurs, all files which have been temporarily stored on the server are deleted. No file is uploaded.
- **Type:** Integer
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** 3
- **Since:** 6.10.5.3

authentication.method

- **Module:** cmas-core-security
- **Description:** User authentication method (internal CM database or LDAP authentication). Allowed values are LDAP or DATABASE.
- **Type:** string
- **Restart required:** no
- **System:** yes

- **Optional:** no
- **Example value:** DATABASE
- **Since:** 6.0

autocommit.cf.changes

- **Module:** cmas-dwh-server
- **Description:**
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** false
- **Since:** 6.7.0

autocomplete.enabled

- **Module:** cmas-app-admin-tool
- **Description:** If the flag is missing or its value is *false*, then the *Autocomplete address* navigation item is hidden in Admin Tool.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** true
- **Since:** 6.9.2.0

automatic.booking.enabled

- **Module:** cmweb-server-adapter
- **Description:** If enabled, time spend on creating comment/e-mail will be measured and automatic time booking will be added.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** true
- **Since:** 6.9.4.2

batch-commit-interval

- **Module:** cmas-dwh-server
- **Description:** Number of objects in a JMS message. Larger values mean better transfer performance at the cost of higher memory usage.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** 100
- **Since:** 6.0.0

big.task.minimum.size

- **Module:** cmas-core-index-common
- **Description:** Indicates the minimum size of index task (in parts, each part has 100 entities) to qualify this task as a big one. Big tasks have lower priority than normal tasks.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 15 (default)
- **Since:** 6.8.3

cache-cluster-name

- **Module:** cmas-core-cache
- **Description:** JBoss cache cluster name.
- **Type:** string
- **Restart required:** yes
- **System:** yes
- **Optional:** no
- **Example value:** 635a6de1-629a-4129-8299-2d98633310f0
- **Since:** 6.4.0

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

cluster.mode

- **Module:** cmas-core-shared
- **Description:** Specifies whether CMAS is running in cluster.
- **Type:** boolean
- **Restart required:** yes
- **System:** yes
- **Optional:** no
- **Example value:** false
- **Since:** 6.1.0

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

commentRequiredForTicketCreation

- **Module:** cmweb-server-adapter
- **Description:** Flag if comment is a required field for ticket creation.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** true (default)
- **Since:** 6.2.0

communication.channel

- **Module:** cmas-dwh-server
- **Description:** Communication channel. Possible values are DIRECT (database communication channel, default value since 6.9.4.1), JMS (default value before 6.9.4.1). Before 6.9.4.1 it has to be manually added.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** DIRECT
- **Since:** 6.8.5.0

config.data.version

- **Module:** cmas-core-server
- **Description:**
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 11
- **Since:** 6.0

contact.authentication.method

- **Module:** cmas-core-security
- **Description:** Indicates contact authentication method, where possible values are DATABASE or LDAP or LDAP,DATABASE or DATABASE,LDAP.
- **Type:** string

- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Since:** 6.9.3.0

`contact.inherit.permissions.only.to.own.customer.group`

- **Module:** cmas-core-security
- **Description:** Indicates whether authenticated contact inherits all customer group permissions from the representing engineer (false) or only has permissions to his own customer group (true).
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Since:** 6.9.2.3

`customizationVersion`

- **Module:** cmweb-server-adapter
- **Description:**
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** cd58453e-f3cc-4538-8030-d15e8796a4a7
- **Since:** 6.5.0

`data.directory`

- **Module:** cmas-core-shared
- **Description:** Directory for CMAS data (e.g., index)
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** C:\Users\user\cmas
- **Since:** 6.0

data.optimization

- **Module:** cmweb-server-adapter
- **Description:** Defines optimization to be applied on response data. So far, the following values are supported (for setting more than one value, separate values by '|'): MINIFICATION and COMPRESSION. MINIFICATION minifies HTML data by e.g. stripping whitespaces and comments. COMPRESSION applies gzip compression to HTTP response. (Note: If you are running in cluster mode and want to test different configurations in parallel, you can set different values for each cluster node by specifying property *data.optimization.nodeId* to override default property.)
- **Type:** string
- **Restart required:** COMPRESSION can be switched on/off without restart, MINIFICATION requires restart.
- **System:** yes
- **Optional:** yes
- **Example value:** MINIFICATION|COMPRESSION

database.notification.enabled

- **Module:** cmas-core-index-common
- **Description:** Indicates whether index update database notification channel should be used instead of JMS.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** false
- **Since:** 6.8.4.7

database.notification.redelivery.delay.seconds

- **Module:** cmas-core-index-common
- **Description:** In case of index update database notification channel, indicates notification redelivery delay when an exception occurs.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 60
- **Since:** 6.8.4.7

database.notification.redelivery.max.attempts

- **Module:** cmas-core-index-common
- **Description:** In case of index update database notification channel, indicates maximum redelivery attempts when an exception occurs.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 60
- **Since:** 6.8.4.7

defaultCommentClassName

- **Module:** cmas-core-server
- **Description:** Default text class name for comments.
- **Type:** string
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:**
- **Since:** 6.3.0

defaultContentEntryClassName

- **Module:** cmweb-server-adapter
- **Description:** Default text class for new ACIMs.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** default_class
- **Since:** 6.3.0

defaultIncommingMailClassName

- **Module:** cmas-core-server
- **Description:** Default text class name for incoming e-mails.
- **Type:** string
- **Restart required:** no

- **System:** no
- **Optional:** yes
- **Since:** 6.3.0

defaultNumberOfCustomFieldsColumns

- **Module:** cmweb-server-adapter
- **Description:** Default number of columns for Custom Fields.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 3
- **Since:** 6.2.0

defaultOutgoingMailClassName

- **Module:** cmas-core-server
- **Description:** Default text class name for outgoing e-mails.
- **Type:** string
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:**
- **Since:** 6.3.0

delete.ticket.enabled

- **Module:** cmas-app-admin-tool
- **Description:** Controls if the menu entry *Delete* is displayed in the context menu in the Admin Tool for the ticket list in ticket administration.
- **Type:** boolean
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** true
- **Since:** 6.9.4.0

diffTrackingEnabled

- **Module:** cmweb-server-adapter
- **Description:** Defines if parallel editing of a ticket by different engineers should be possible. Default is *true*.
- **Type:** boolean
- **Restart required:** no
- **System:**
- **Optional:**
- **Example value:** true
- **Since:** 6.10.1

disable.admin.task.auto.commit

- **Module:** cmas-core-index-common
- **Description:** All tasks created for index update will be automatically executed right after creation.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** false
- **Since:** 6.6.1

dwh.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

esb.directory

- **Module:** cmas-esb-core
- **Description:** Directory used by Mule/ESB.
- **Type:** string
- **Restart required:** no

- **System:** yes
- **Optional:** no
- **Example value:** C:\Users\user\cmas\mule
- **Since:** 6.0

eviction.event.queue.size

- **Module:** cmas-core-cache
- **Description:**
- **Type:** integer
- **Restart required:** yes
- **System:** yes
- **Optional:** no
- **Example value:** 200000
- **Since:** 6.4.0

eviction.max.nodes

- **Module:** cmas-core-cache
- **Description:**
- **Type:** integer
- **Restart required:** yes
- **System:** yes
- **Optional:** no
- **Example value:** 100000
- **Since:** 6.4.0

eviction.wakeup.interval

- **Module:** cmas-core-cache
- **Description:**
- **Type:** integer
- **Restart required:** yes
- **System:** yes
- **Optional:** no
- **Example value:** 3000
- **Since:** 6.4.0

favoritesSizeLimit

- **Module:** cmweb-server-adapter
- **Description:** Maximum number of items in Favorites list.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 10
- **Since:** 6.0

fetchLock.interval

- **Module:** cmas-workflow-jbpm
- **Description:**
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 5000
- **Removed in:** 6.8.0

fetchSize.strategy

- **Module:** cmas-core-server
- **Description:** Strategy for selecting the fetch size on JDBC result sets.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** FetchSizePageBasedStrategy, FetchSizeThresholdStrategy, FetchSizeFixedStrategy
- **Since:** 6.8.4.1

fetchSize.strategy.FetchSizeFixedStrategy.value

- **Module:** cmas-core-server
- **Description:** Sets fetch size value if the selected strategy to set the fetch size is *FetchSizeFixedStrategy*.
- **Type:** integer
- **Restart required:** no

- **System:** yes
- **Optional:** yes
- **Example value:** 150
- **Since:** 6.8.4.1

fetchSize.strategy.FetchSizePageBasedStrategy.limit

- **Module:** cmas-core-server
- **Description:** Sets maximum fetch size value if the selected strategy to set the fetch size is *FetchSizePageBasedStrategy*.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** 10000
- **Since:** 6.8.4.1

fetchSize.strategy.FetchSizeThresholdStrategy.value

- **Module:** cmas-core-server
- **Description:** Sets fetch size threshold border values if the selected strategy to set the fetch size is *FetchSizeThresholdStrategy*.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** 150,300,600,1000
- **Since:** 6.8.4.1

filesystem.polling.threads.number

- **Module:** cmas-nimh
- **Description:** Number of threads started for db e-mails' queue polling. Default: 1
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 10
- **Since:** 6.4.0

`filesystem.polling.threads.shutdown.timeout.seconds`

- **Module:** cmas-nimh
- **Description:** Waiting time after the shutdown signal. When the timeout reached, thread will be terminated. Default: 60
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 60
- **Since:** 6.4.0

`filesystem.polling.threads.watchdog.interval.seconds`

- **Module:** cmas-nimh
- **Description:** Watchdog thread interval. Default: 30
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 60
- **Since:** 6.4.0

`filesystem.task.enabled`

- **Module:** cmas-nimh
- **Description:** With this property service thread related to given poller can be disabled. Default: true
- **Type:** boolean
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** true
- **Since:** 6.4.0

`filesystem.task.interval.seconds`

- **Module:** cmas-nimh
- **Description:** Default interval for polling mailboxes. Default: 60 seconds
- **Type:** integer
- **Restart required:** no

- **System:** no
- **Optional:** yes
- **Example value:** 60
- **Since:** 6.4.0

`filesystem.task.polling.folder`

- **Module:** cmas-nimh
- **Description:** Polling folder location which will be scanned for e-mails in the format of eml files. Default: "mail" subdir of cmas data directory
- **Type:** string
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** c://cmas//mail
- **Since:** 6.4.0

`filesystem.task.timeout.seconds`

- **Module:** cmas-nimh
- **Description:** After this time (of inactivity) the service thread is considered as damaged and automatically restarted. Default: 120 seconds
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 60
- **Since:** 6.4.0

`filesystem.task.transaction.timeout.seconds`

- **Module:** cmas-nimh
- **Description:** Default transaction timeout for e-mail fetching transactions. Should be correlated with number of messages fetched at once. Default: 60 seconds
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 60
- **Since:** 6.4.0

globalSearchResultSizeLimit

- **Module:** cmweb-server-adapter
- **Description:** Maximum number of items in Quick Search result.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 10
- **Since:** 6.0

helpFilePath

- **Module:** cmweb-server-adapter
- **Description:** URL for online help. If not empty, Help button is displayed in Web Client.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** http://www.consol.de
- **Since:** 6.2.1

hibernate.dialect

- **Module:** cmas-setup-hibernate
- **Description:** The dialect used by hibernate. Usually set during initial set-up (depending on the database system).
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** org.hibernate.dialect.MySQL5InnoDBDialect
- **Since:** 6.0

hideTicketSubject

- **Module:** cmweb-server-adapter
- **Description:** If set to *true*, ticket subject is hidden.
- **Type:** boolean
- **Restart required:** no

- **System:** yes
- **Optional:** no
- **Example value:** false
- **Since:** 6.2.1

ignore-queues

- **Module:** cmas-dwh-server
- **Description:** A comma-separated list of queue names which are not transferred to the DWH.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** QueueName1,QueueName2,QueueName3
- **Since:** 6.6.19
- **Removed in:** 6.8.1

index.attachment

- **Module:** cmas-core-index-common
- **Description:** Specifies whether content of attachments is indexed.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** true
- **Since:** 6.4.3

index.history

- **Module:** cmas-core-index-common
- **Description:** Specifies whether unit and ticket history are indexed.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** false
- **Since:** 6.1.0

index.status

- **Module:** cmas-core-index-common
- **Description:** Status of the Indexer, possible values RED, YELLOW, GREEN, will be displayed in the Admin Tool.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** GREEN
- **Since:** 6.6.1

index.task.worker.threads

- **Module:** cmas-core-index-common
- **Description:** How many threads will be used to execute index tasks (synchronization, administrative, and repair tasks).
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 1 (default) (we recommend to use a value not larger than 2)
- **Since:** 6.6.14, 6.7.3. Since 6.8.0 and exclusively in 6.6.21 also normal (live) index updates are affected by this property.

index.version.current

- **Module:** cmas-core-index-common
- **Description:** Holds information about current (possibly old) index version.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 1 (default)
- **Since:** 6.7.0

index.version.newest

- **Module:** cmas-core-index-common
- **Description:** Holds information about which index version is considered newest.
- **Type:** integer

- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 1 (default)
- **Since:** 6.7.0

`indexed.assets.per.thread.in.memory`

- **Module:** cmas-core-index-common
- **Description:** How many assets should be loaded into memory at once, per thread, during indexing.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 200 (default)
- **Since:** 6.8.0

`indexed.engineers.per.thread.in.memory`

- **Module:** cmas-core-index-common
- **Description:** How many engineers should be loaded into memory at once, per thread, during indexing.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 300 (default)
- **Since:** 6.6.14, 6.7.3

`indexed.resources.per.thread.in.memory`

- **Module:** cmas-core-index-common
- **Description:** How many resources should be loaded into memory at once, per thread, during indexing.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no

- **Example value:** 200 (default)
- **Since:** 6.10.0.0

indexed.tickets.per.thread.in.memory

- **Module:** cmas-core-index-common
- **Description:** How many tickets should be loaded into memory at once, per thread, during indexing.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 100 (default)
- **Since:** 6.6.14, 6.7.3

indexed.units.per.thread.in.memory

- **Module:** cmas-core-index-common
- **Description:** How many units should be loaded into memory at once, per thread, during indexing.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 200 (default)
- **Since:** 6.6.14, 6.7.3

initialized

- **Module:** cmas-setup-manager
- **Description:** Flag if CMAS is initialized. If this value is missing or not *true*, set-up will be performed.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** true
- **Since:** 6.0



Be careful with using this property!!! When you set the value to *false*, the ConSol CM server will perform the system set-up at the next start, i.e. all data of the existing system is lost, including system properties!!!

is.cmrf.alive

- **Module:** cmas-dwh-server
- **Description:** As a starting point, the time the last message was sent to CMRF should be used. If a response from CMRF is not received after value (in seconds), it should create a DWH operation status with an error message indicating that CMRF is down.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 1200
- **Since:** 6.7.0

java.naming.factory.initial

- **Module:** cmas-dwh-server
- **Description:** Factory class for the DWH context factory.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** org.jnp.interfaces.NamingContextFactory
- **Since:** 6.0.1

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

`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

`jobExecutor.adminMail`

- **Module:** cmas-workflow-engine
- **Description:** E-mail address which will get notified about job execution problems (when retry counter is exceeded).
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** admin@consol.de
- **Since:** 6.8.0

`jobExecutor.idleInterval`

- **Module:** cmas-workflow-jbpm
- **Description:**
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 45000
- **Removed in:** 6.8.0
- **Replaced by:** `jobExecutor.idleInterval.seconds`

`jobExecutor.idleInterval.seconds`

- **Module:** cmas-workflow-engine
- **Description:** Determines how often job executor thread will look for new jobs to execute.
- **Type:** integer

- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** 45 (default up to CM version 6.10.5.2. Default CM versions 6.10.5.3 and up is 5)
- **Since:** 6.8.0

`jobExecutor.jobExecuteRetryNumber`

- **Module:** cmas-workflow-jbpm
- **Description:**
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 5
- **Removed in:** 6.8.0
- **Replaced by:** `jobExecutor.jobMaxRetries`

`jobExecutor.jobMaxRetries`

- **Module:** cmas-workflow-engine
- **Description:**
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** 5 (default)
- **Since:** 6.8.0

`jobExecutor.jobMaxRetriesReachedSubject`

- **Module:** cmas-workflow-engine
- **Description:**
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** Job maximum retries reached. Job was removed!!! (default)
- **Since:** 6.8.0

`jobExecutor.lockingLimit`

- **Module:** cmas-workflow-engine
- **Description:** Number of jobs locked at once (marked for execution) by job executor thread.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** 5 (default since CM version 6.10.5.3)
- **Since:** 6.8.0

`jobExecutor.lockTimeout.seconds`

- **Module:** cmas-workflow-engine
- **Description:** How long the job can be locked (marked for execution) by job executor.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** 360 (default)
- **Since:** 6.8.0

`jobExecutor.mailFrom`

- **Module:** cmas-workflow-engine
- **Description:** E-mail which will be set as FROM-header during admin notifications.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** jobexecutor@consol.de
- **Since:** 6.8.0

`jobExecutor.maxInactivityInterval.minutes`

- **Module:** cmas-workflow-engine
- **Description:** Number of minutes of allowed job executor inactivity (e.g. when it is blocked by long timer execution). After this time executors threads are restarted.
- **Type:** integer
- **Restart required:** no

- **System:** yes
- **Optional:** yes. Default value is set to 30 minutes
- **Example value:** 15 (default)
- **Since:** 6.9.2.0

`jobExecutor.threads`

- **Module:** cmas-workflow-engine
- **Description:** Number of job execution threads.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** 1 (default)
- **Since:** 6.8.0

`jobExecutor.timerRetryInterval`

- **Module:** cmas-workflow-jbpm
- **Description:**
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 10000
- **Removed in:** 6.8.0
- **Replaced by:** `jobExecutor.timerRetryInterval.seconds`

`jobExecutor.timerRetryInterval.seconds`

- **Module:** cmas-workflow-engine
- **Description:** Determines how long job executor thread will wait after job execution error.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** 10 (default up to CM version 6.10.5.2. Default CM versions 6.10.5.3 and up is 30)
- **Since:** 6.8.0

`jobExecutor.txTimeout.seconds`

- **Module:** cmas-workflow-engine
- **Description:** Transaction timeout used for job execution.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** 60 (default)
- **Since:** 6.8.0

`kerberos.v5.enabled`

- **Module:** cmas-core-security
- **Description:** Indicates whether SSO via Kerberos is enabled.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** false (default if Kerberos was not enabled during system set-up)
- **Since:** 6.2.0

`kerberos.v5.username.regex`

- **Module:** cmas-core-security
- **Description:** Regular expression used for mapping Kerberos principals to CM user login names.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** (.*)@.*
- **Since:** 6.2.0

`last.config.change`

- **Module:** cmas-core-server
- **Description:** Random UUID created during the last configuration change.
- **Type:** string
- **Restart required:** no
- **System:** yes

- **Optional:** no
- **Example value:** 2573c7b7-2bf5-47ff-b5a2-bad31951a266
- **Since:** 6.1.0, 6.2.1

ldap.authentication

- **Module:** cmas-core-security
- **Description:** Authentication method used when using LDAP authentication.
- **Type:** string
- **Restart required:** yes
- **System:** yes
- **Optional:** no
- **Example value:** simple
- **Since:** 6.0

ldap.basedn

- **Module:** cmas-core-security
- **Description:** Base DN used for looking up LDAP user accounts when using LDAP authentication.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** ou=accounts,dc=consol,dc=de
- **Since:** 6.0

ldap.certificate.basedn

- **Module:** cmas-core-server
- **Description:** Base DN for certificates location in the LDAP tree. If not provided, *cmas-core-security*, *ldap.basedn* is used.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** ou=accounts,dc=consol,dc=de
- **Since:** 6.8.4

ldap.certificate.content.attribute

- **Module:** cmas-core-server
- **Description:** LDAP attribute name used where certificate data is stored in the LDAP tree. Default value: usercertificate
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** usercertificate
- **Since:** 6.8.4

ldap.certificate.password

- **Module:** cmas-core-server
- **Description:** LDAP Certificates manager password. If not set, *cmas-core-security, ldap.-password* is used.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Since:** 6.8.4

ldap.certificate.providerurl

- **Module:** cmas-core-server
- **Description:** LDAP Certificates provider URL. If not set, *cmas-core-security, ldap.providerurl* is used.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** ldap://ldap.consol.de:389
- **Since:** 6.8.4

ldap.certificate.searchattr

- **Module:** cmas-core-server
- **Description:** LDAP attribute name used to search for certificate in the LDAP tree. Default value: mail
- **Type:** string

- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** mail
- **Since:** 6.8.4

ldap.certificate.userdn

- **Module:** cmas-core-server
- **Description:** LDAP Certificates manager DN. If not set, *cmas-core-security*, *ldap.userdn* is used.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Since:** 6.8.4

ldap.contact.name.basedn

- **Module:** cmas-core-security
- **Description:** Base path to search for contact DN by LDAP ID (e.g. ou=a-ccounts,dc=consol,dc=de).
- **Type:** string
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Since:** 6.9.3.0

ldap.contact.name.password

- **Module:** cmas-core-security
- **Description:** Password to look up contact DN by LDAP ID. If not set, the anonymous account is used.
- **Type:** string
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Since:** 6.9.3.0

ldap.contact.name.providerurl

- **Module:** cmas-core-security
- **Description:** Address of the LDAP server (ldap[s]://host:port).
- **Type:** string
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Since:** 6.9.3.0

ldap.contact.name.searchattr

- **Module:** cmas-core-security
- **Description:** Attribute to search for contact DN by LDAP ID (e.g. uid).
- **Type:** string
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Since:** 6.9.3.0

ldap.contact.name.userdn

- **Module:** cmas-core-security
- **Description:** User DN to look up contact DN by LDAP ID. If not set, the anonymous account is used.
- **Type:** string
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Since:** 6.9.3.0

ldap.initialcontextfactory

- **Module:** cmas-core-security
- **Description:** Class name for the initial context factory of the LDAP implementation when using LDAP authentication. If it is not set, *com.sun.jndi.ldap.LdapCtxFactory* is used.
- **Type:** string
- **Restart required:** yes
- **System:** yes
- **Optional:** no

- **Example value:** com.sun.jndi.ldap.LdapCtxFactory
- **Since:** 6.0

ldap.password

- **Module:** cmas-core-security
- **Description:** Password for connecting to LDAP to look up users when using LDAP authentication. Only needed if look-up cannot be performed anonymously.
- **Type:** password
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Since:** 6.1.2

ldap.providerurl

- **Module:** cmas-core-security
- **Description:** LDAP provider when using LDAP authentication.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** ldap://myserver.consol.de:389
- **Since:** 6.0

ldap.searchattr

- **Module:** cmas-core-security
- **Description:** Search attribute for looking up LDAP entry associated with a CM login.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** uid
- **Since:** 6.0

ldap.userdn

- **Module:** cmas-core-security
- **Description:** LDAP user for connecting to LDAP to look up users when using LDAP authentication. Only needed if look-up cannot be performed anonymously.

- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Since:** 6.1.2

mail.attachments.validation.info.sender

- **Module:** cmas-esb-mail
- **Description:** Sets FROM-header of attachments type *error notification e-mail*. As a default the e-mail address of the administrator which you have entered during system set-up is used.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** admin@consolcm.com
- **Since:** 6.7.5

mail.attachments.validation.info.sender

- **Module:** cmas-nimh-extension
- **Description:** Sets FROM-header of attachments type *error notification mail*
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** admin@mail.com
- **Since:** 6.7.5



This is an equivalent to the old *cmas-esb-mail*, *mail.attachments.validation.info.sender*

mail.attachments.validation.info.subject

- **Module:** cmas-esb-mail
- **Description:** Sets subject of attachments type *error notification e-mail*.
- **Type:** string
- **Restart required:** no
- **System:** yes

- **Optional:** no
- **Example value:** E-mail was not processed because its attachments were rejected!
- **Since:** 6.7.5

mail.attachments.validation.info.subject

- **Module:** cmas-nimh-extension
- **Description:** Sets subject of attachments type error notification mail.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** E-mail was not processed because its attachments were rejected!
- **Since:** 6.7.5



This is an equivalent to the old *cmas-esb-mail*, *mail.attachments.validation.info.subject*

mail.callname.pattern

- **Module:** cmas-esb-mail
- **Description:** Regular expression for subject of incoming e-mails. Available as TICKET_NAME_PATTERN_FORMAT in incoming e-mail scripts.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** .*?Ticket\s+\\((\\S+)\\).*
- **Since:** 6.0

mail.cluster.node.id

- **Module:** cmas-esb-mail
- **Description:** Only the node whose *mail.cluster.node.id* equals *cmas.clusternode.id* will start the Mule/ESB e-mail services.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no

- **Example value:** unspecified
- **Since:** 6.6.5

mail.db.archive

- **Module:** cmas-esb-mail
- **Description:** If property is set to *true*, incoming e-mails are archived in the database.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** false (default)
- **Since:** 6.8.5.5

Obsolete! In Mule/ESB mode, no e-mails are saved in the database. E-mails which could not be processed are stored in the file system, see section *E-Mail Backups* in the *ConSol CM Administrator Manual*.

mail.db.archive

- **Module:** cmas-nimh-extension
- **Description:** If property is set to *true*, incoming e-mails are archived in the database.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** false (default)
- **Since:** 6.8.5.5


mail.delete.read

- **Module:** cmas-esb-mail
- **Description:** Determines whether CM deletes messages fetched via IMAP(S). Setting value to *true* will cause deletion of messages after fetching. Default is to not delete messages fetched via IMAP(S). Note: Messages fetched via POP3(S) will always be deleted.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** true
- **Since:** 6.7.3


mail.encryption

- **Module:** cmas-core-server
- **Description:** If property is set to *true*, the encrypt checkbox in the Ticket E-Mail Editor is checked by default.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** true (default = false)
- **Since:** 6.8.4.0

mail.error.from.address

 This is an equivalent to the old *cmas-esb-mail, mail.mule.service*

mail.error.to.address

 This is an equivalent to the old *cmas-esb-mail, mail.process.error*


mail.from

- **Module:** cmweb-server-adapter
- **Description:** Use this address if set instead of engineer e-mail address during e-mail conversation.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Since:** 6.1.2

mail.incoming.uri

- **Module:** cmas-esb-mail
- **Description:** URL for incoming e-mails.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no

- **Example value:** pop3://cm-incoming-user:password@localhost:10110
- **Since:** 6.0

 This value should not be edited here using the system properties pop-up window, but the mailboxes should be configured using the navigation item *E-mail*. Using this standard feature all entries are controlled - i.e., for each mailbox which is added, CM establishes a test connection during mailbox set-up. That way it is not possible to enter wrong values.

mail.max.restarts

- **Module:** cmas-esb-mail
- **Description:** Maximum number of e-mail service restarts before giving up.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 3
- **Since:** 6.0

mail.mime.strict

- **Module:** cmas-esb-mail
- **Description:** If set to *false*, e-mail addresses are not parsed for strict MIME compliance. Default is *true*, which means check for strict MIME compliance.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** false
- **Since:** 6.6.17, 6.7.3

mail.mule.service

- **Module:** cmas-esb-mail
- **Description:** FROM-address for e-mails sent by Mule service
- **Type:** email
- **Restart required:** no
- **System:** yes
- **Optional:** no

- **Example value:** myuser@consol.de
- **Since:** 6.0

mail.notification.engineerChange

- **Module:** cmas-core-server
- **Description:** Whether notification e-mails 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 e-mails 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 e-mail is sent to the above configured address in case the e-mail message could not be processed. Default: true
- **Type:** boolean
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** false
- **Since:** 6.4.0

mail.polling.interval

- **Module:** cmas-esb-mail
- **Description:** E-mail polling interval in ms.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 60000
- **Since:** 6.0

mail.process.error

- **Module:** cmas-esb-mail
- **Description:** TO-address for error e-mails from Mule. As a default the e-mail address of the administrator which you have entered during system set-up is used.
- **Type:** email
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** myuser@consol.de
- **Since:** 6.0

mail.process.error

- **Module:** cmas-nimh-extension
- **Description:** TO-address for error e-mails from Mule.
- **Type:** email
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** myuser@consol.de
- **Since:** 6.4.0

mail.process.retry.attempts

- **Module:** cmas-esb-mail
- **Description:** Number of retries when processing e-mail
- **Type:** integer
- **Restart required:** no

- **System:** yes
- **Optional:** no
- **Example value:** 3
- **Since:** 6.0.2

mail.process.timeout

- **Module:** cmas-esb-mail
- **Description:** E-mail processing timeout in seconds.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 60
- **Since:** 6.1.3

mail.redelivery.retry.count

- **Module:** cmas-esb-mail
- **Description:** Indicates the number of retries of re-delivering an e-mail from the CM system.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 3
- **Since:** 6.1.0

mail.reply.to

- **Module:** cmweb-server-adapter
- **Description:** When set, Web Client will display REPLY-TO-field on e-mail send, prefilled with this value.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Since:** 6.0.1



Please read the detailed information about ConSol CM REPLY-TO-addresses in section *Scripts of Type E-Mail* in the *ConSol CM Administrator Manual*.

mail.sender.address

- **Module:** cmas-workflow-jbpm
- **Description:** FROM-address for e-mails from the workflow engine.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** myuser@consol.de
- **Removed in:** 6.8.0
- **Replaced by:** jobExecutor.mailFrom

mail.smtp.email

- **Module:** cmas-core-server
- **Description:** SMTP e-mail URL for outgoing e-mails
- **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:** E-mail address used as sender in SMTP envelope. If not set, the FROM-address of the e-mail is used.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** mysender@mydomain.com
- **Since:** 6.5.7

mail.ticketname.pattern

- **Module:** cmas-nimh-extension
- **Description:**
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** .*?Ticket\s+\\((\S+)\).*
- **Since:** 6.4.0

mailbox.1.connection.host

- **Module:** cmas-nimh
- **Description:** Host (server) for first configured mailbox. Will overwrite the default parameter *mailbox.default.connection.host*.

mailbox.1.connection.password

- **Module:** cmas-nimh
- **Description:** Password for first configured mailbox. Will overwrite the default parameter *mailbox.default.connection.password*.

mailbox.1.connection.port

- **Module:** cmas-nimh
- **Description:** Port for first configured mailbox. Will overwrite the default parameter *mailbox.default.connection.port*.

mailbox.1.connection.protocol

- **Module:** cmas-nimh
- **Description:** Protocol (e.g., IMAP or POP3) for first configured mailbox. Will overwrite the default parameter *mailbox.default.connection.protocol*.

mailbox.1.connection.username

- **Module:** cmas-nimh
- **Description:** User name for first configured mailbox. Will overwrite the default parameter *mailbox.default.connection.username*.

mailbox.2.connection.host

- **Module:** cmas-nimh
- **Description:** Host (server) for second configured mailbox. Will overwrite the default parameter *mailbox.default.connection.host*.

mailbox.2.connection.password

- **Module:** cmas-nimh
- **Description:** Password for second configured mailbox. Will overwrite the default parameter *mailbox.default.connection.password*.

mailbox.2.connection.port


- **Module:** cmas-nimh
- **Description:** Port for second configured mailbox. Will overwrite the default parameter *mailbox.default.connection.port*.

mailbox.2.connection.protocol

- **Module:** cmas-nimh
- **Description:** Protocol (e.g., IMAP or POP3) for second configured mailbox. Will overwrite the default parameter *mailbox.default.connection.protocol*.

mailbox.2.connection.username

- **Module:** cmas-nimh
- **Description:** User name for second configured mailbox. Will overwrite the default parameter *mailbox.default.connection.username*.

 For all NIMH-related mailbox properties, the following principle is used: a default property is defined (e.g. *mailbox.default.connection.port*). If no mailbox-specific value is configured, this default value will be used.

mailbox.default.connection.host

- **Module:** cmas-nimh
- **Description:** Host (server name) of a given mailbox from which the poller reads e-mails.
- **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 e-mails.
- **Type:** string
- **Restart required:** no

- **System:** no
- **Optional:** yes
- **Example value:** consol
- **Since:** 6.4.0

mailbox.default.connection.port

- **Module:** cmas-nimh
- **Description:** Port for a given mailbox from which the poller reads e-mails.
- **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 e-mails.
- **Type:** string
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** username
- **Since:** 6.4.0

mailbox.default.session.mail.debug

- **Module:** cmas-nimh
- **Description:** Example javax.mail property - allows for more detailed javax.mail session debugging
- **Type:** boolean
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** true
- **Since:** 6.4.0

mailbox.default.session.mail.imap.timeout

- **Module:** cmas-nimh
- **Description:** Example javax.mail property
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 120
- **Since:** 6.4.0

mailbox.default.session.mail.mime.address.strict

- **Module:** cmas-nimh
- **Description:** Example javax.mail property - counterpart of the old *mule mail.mime.strict*, allows to set not so strict e-mail header parsing
- **Type:** boolean
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** true
- **Since:** 6.4.0

mailbox.default.session.mail.pop3.timeout

- **Module:** cmas-nimh
- **Description:** Example javax.mail property.
- **Type:**
- **Restart required:**

- **System:**
- **Optional:**
- **Example value:**
- **Since:** 6.4.0

mailbox.default.session.mail.<protocol>.partialfetch

- **Module:** cmas-nimh
- **Description:** e.g. mailbox.default.session.mail.imaps.partialfetch
- **Type:** boolean
- **Restart required:**
- **System:**
- **Optional:** yes
- **Example value:** false
- **Since:**

mailbox.default.task.delete.read.messages

- **Module:** cmas-nimh
- **Description:** This defines whether messages should be removed from the mailbox after processing. For IMAP protocol messages are marked as SEEN by default. For POP3 protocol, when flag is set to true the message is removed, otherwise remains on server and will result in infinite reads. Default: false.
- **Type:** boolean
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** false
- **Since:** 6.4.0

mailbox.default.task.enabled

- **Module:** cmas-nimh
- **Description:** With this property service thread related to given poller can be disabled. Default: true
- **Type:** boolean
- **Restart required:** no
- **System:** no
- **Optional:** yes

- **Example value:** false
- **Since:** 6.4.0

mailbox.default.task.interval.seconds

- **Module:** cmas-nimh
- **Description:** Default interval for polling mailboxes. Default: 60 seconds
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 60
- **Since:** 6.4.0

mailbox.default.task.max.message.size

- **Module:** cmas-nimh
- **Description:** Maximum size of e-mail messages (i.e., e-mail plus attachment). E-mails 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 e-mail backups in the Admin Tool (see section *E-Mail 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. Default is set to 10MB: 10485760
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 10485760
- **Since:** 6.4.0

mailbox.default.task.max.messages.per.run

- **Module:** cmas-nimh
- **Description:** Number of messages fetched at once from mailbox. Must be correlated with transaction timeout. Default set to: 20
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 60
- **Since:** 6.4.0

mailbox.default.task.timeout.seconds

- **Module:** cmas-nimh
- **Description:** After this time (of inactivity) the service thread is considered as damaged and automatically restarted. Default: 120 seconds
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 60
- **Since:** 6.4.0

mailbox.default.task.transaction.timeout.seconds

- **Module:** cmas-nimh
- **Description:** Default transaction timeout for e-mail fetching transactions. Should be correlated with number of messages fetched at once. Default: 60 seconds
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 60
- **Since:** 6.4.0

mailbox.polling.threads.mail.log.enabled

- **Module:** cmas-nimh
- **Description:** Enables e-mail logging which is especially crucial in cluster environment (used as semaphore there)
- **Type:** boolean
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** true (default)
- **Since:** 6.9.4.1

mailbox.polling.threads.number

- **Module:** cmas-nimh
- **Description:** Number of threads for accessing mailboxes. Default: 1
- **Type:** integer

- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 1
- **Since:** 6.4.0

mailTemplateAboveQuotedText

- **Module:** cmweb-server-adapter
- **Description:** Indicates behavior of e-mail template in the Ticket E-Mail Editor when another e-mail is quoted, i.e. forwarded or replied to. Often used to place the signature correctly.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** false
- **Since:** 6.2.4

max.licences.perUser

- **Module:** cmas-core-server
- **Description:** Sets maximum licenses single user can use (e.g., logging in from different browsers). By default this value is not restricted.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** 10
- **Since:** 6.8.4.5

maxSizePerPagemapInMegaBytes

- **Module:** cmweb-server-adapter
- **Description:** Maximum size (in MB) for each Wicket pagemap.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 15
- **Since:** 6.3.5

monitoring.engineer.login

- **Module:** cmas-core-server
- **Description:** Login of monitoring engineer.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** nagios
- **Since:** 6.9.3.0

monitoring.unit.login

- **Module:** cmas-core-server
- **Description:** Login of monitoring unit.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** nagios
- **Since:** 6.9.3.0

nimh.enabled

- **Module:** cmas-core-server
- **Description:** Enables NIMH service. Must be suffixed with the cluster node ID, e.g., *nimh.enabled.NODEID = true*.
- **Type:** boolean
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** false
- **Since:** 6.9.4.0

notification.error.description

- **Module:** cmas-dwh-server
- **Description:** Text for error e-mails 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 e-mails from the DWH
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Since:** 6.0.1

notification.error.subject

- **Module:** cmas-dwh-server
- **Description:** Subject for error e-mails 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 e-mails 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 e-mails 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 e-mails from the DWH when a transfer finishes successfully.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Since:** 6.0.1

notification.finished_successfully.subject

- **Module:** cmas-dwh-server
- **Description:** Subject for e-mails 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 e-mails 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 e-mails 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 e-mails from the DWH when a transfer finishes unsuccessfully.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Since:** 6.0.1

`notification.finished_unsuccessfully.subject`

- **Module:** cmas-dwh-server
- **Description:** Subject for e-mails 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 e-mails 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:** E-mail (SMTP) server hostname for sending DWH e-mails.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** myserver.consol.de
- **Since:** 6.0.1

notification.password

- **Module:** cmas-dwh-server
- **Description:** Password for sending DWH e-mails (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 e-mails.
- **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 e-mails from the DWH.
- **Type:** string
- **Restart required:** no

- **System:** yes
- **Optional:** yes
- **Example value:** pop3\

notification.username

- **Module:** cmas-dwh-server
- **Description:** (SMTP) User name for sending DWH e-mails.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** myuser
- **Since:** 6.0.1

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

policy.password.age

- **Module:** cmas-core-security
- **Description:** Defines (in days) how old the password may be.
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 5500 (15 years, default)
- **Since:** 6.10.1.0

policy.password.pattern

- **Module:** cmas-core-security
- **Description:** Defines password pattern.
- **Type:** string
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** "^.{3,}\$" (default)
- **Since:** 6.10.1.0

policy.rotation.ratio

- **Module:** cmas-core-security
- **Description:** Defines how often password may repeat. E.g., setting the value to X means that the new password cannot be present among the user's X previous passwords.
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 1 (default)
- **Since:** 6.10.1.0

policy.username.case.sensitive

- **Module:** cmas-core-security
- **Description:** Defines whether user names are case-sensitive.
- **Type:** boolean
- **Restart required:** no

- **System:** no
- **Optional:** yes
- **Example value:** true (default)
- **Since:** 6.10.1.0

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 e-mails' queue polling. Default: 1
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 1
- **Since:** 6.4.0

queue.polling.threads.shutdown.timeout.seconds

- **Module:** cmas-nimh
- **Description:** Waiting time after the shutdown signal. When the timeout is reached, the thread will be terminated. Default: 60
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 60
- **Since:** 6.4.0

queue.polling.threads.watchdog.interval.seconds

- **Module:** cmas-nimh
- **Description:** Watchdog thread interval. Default: 30
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 30
- **Since:** 6.4.0

queue.task.error.pause.seconds

- **Module:** cmas-nimh
- **Description:** Maximum number of seconds, the queue poller waits after infrastructure (e.g. database) error. Default 180 seconds
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 180
- **Since:** 6.4.0

queue.task.interval.seconds

- **Module:** cmas-nimh
- **Description:** Main e-mails' queue polling thread interval. Default: 15
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 15
- **Since:** 6.4.0

queue.task.max.retries

- **Module:** cmas-nimh
- **Description:** Maximum number of e-mail processing retries after an exception. When reached, the e-mail is moved to the e-mail archive. This e-mail can be rescheduled again using NIMH API (or the Admin Tool).
- **Type:** integer

- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 10
- **Since:** 6.4.0

queue.task.timeout.seconds

- **Module:** cmas-nimh
- **Description:** After this time (of inactivity) the service thread is considered as damaged and automatically restarted. Default: 600 seconds
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 600
- **Since:** 6.4.0

queue.task.transaction.timeout.seconds

- **Module:** cmas-nimh
- **Description:** Transaction timeout for e-mail processing in the pipe. Default: 60
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 60
- **Since:** 6.4.0

queuesExcludedFromGS

- **Module:** cmweb-server-adapter
- **Description:** Comma-separated list of queue names which are excluded from Quick Search.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Since:** 6.0

refreshTimeInCaseOfConcurrentRememberMeRequests

- **Module:** cmas-workflow-jbpm
- **Description:** It sets the refresh time (in seconds) after which page will be reloaded in case of concurrent remember me requests. This feature prevents one user from occupying many licenses. Please increase that time if sessions are still occupying.
- **Type:** integer
- **Restart required:** yes
- **System:** yes
- **Optional:** yes
- **Example value:** 5
- **Since:** 6.8.2

rememberMeLifetimeInMinutes

- **Module:** cmweb-server-adapter
- **Description:** Lifetime for *remember me* in minutes.
- **Type:** integer
- **Restart required:** yes
- **System:** yes
- **Optional:** no
- **Example value:** 1440
- **Since:** 6.0

request.scope.transaction

- **Module:** cmweb-server-adapter
- **Description:** It allows to disable request scope transaction. By default one transaction is used per request. Setting this property to *false* there will cause one transaction per service method invocation.
- **Type:** boolean
- **Restart required:** yes
- **System:** yes
- **Optional:** yes
- **Example value:** true
- **Since:** 6.8.1

resource.replace.batchSize

- **Module:** cmas-core-server
- **Description:** Defines the number of objects to be processed in a resource replace action.

- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 5
- **Since:** 6.10.0.0

resource.replace.timeout

- **Module:** cmas-core-server
- **Description:** Transaction timeout (in seconds) of a resource replacement action step.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 120
- **Since:** 6.10.0.0

scene

- **Module:** cmas-setup-scene
- **Description:** Scene file which was imported during set-up (can be empty).
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** vfszip:/P:/dist/target/jboss/server/cmas/deploy/cm-dist-6.5.1-SNAPSHOT.ear/APP-INF/lib/dist-scene-6.5.1-SNAPSHOT.jar/META-INF/cmas/scenes/helpdesk-sales_scene.jar/
- **Since:** 6.0

script.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

searchPageSize

- **Module:** cmweb-server-adapter
- **Description:** Default page size for search results.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 20
- **Since:** 6.0

searchPageSizeOptions

- **Module:** cmweb-server-adapter
- **Description:** Options for page size for search results.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 10|20|30|40|50|75|100
- **Since:** 6.0

server.session.archive.reaper.interval

- **Module:** cmas-core-server
- **Description:** Server archived sessions reaper interval (in seconds).
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** 60
- **Since:** 6.7.1

server.session.archive.timeout

- **Module:** cmas-core-server
- **Description:** Server sessions archive validity timeout (in days). After this time session info is removed from the DB.

- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 31
- **Since:** 6.7.1

`server.session.reaper.interval`

- **Module:** cmas-core-server
- **Description:** Server inactive (ended) sessions reaper interval (in seconds).
- **Type:** integer
- **Restart required:** only Session Service
- **System:** yes
- **Optional:** no
- **Example value:** 60
- **Since:** 6.6.1, 6.7.1

`server.session.timeout`

- **Module:** cmas-core-server
- **Description:** Server session timeout (in seconds) for connected clients. Each client can over-write 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:**
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 5
- **Since:** 6.1.0

skip-ticket

- **Module:** cmas-dwh-server
- **Description:** Tickets are not transferred during transfer/update.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** false
- **Since:** 6.6.19
- **Removed in:** 6.8.1

skip-ticket-history

- **Module:** cmas-dwh-server
- **Description:** History of ticket is not transferred during transfer/update.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** false

- **Since:** 6.6.19
- **Removed in:** 6.8.1

skip-unit

- **Module:** cmas-dwh-server
- **Description:** Units are not transferred during transfer/update.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** false
- **Since:** 6.6.19
- **Removed in:** 6.8.1

skip-unit-history

- **Module:** cmas-dwh-server
- **Description:** History of unit is not transferred during transfer/update.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** false
- **Since:** 6.6.19
- **Removed in:** 6.8.1

skip.wfl.transfer.cleanup

- **Module:** cmas-core-server
- **Description:** If set to *true*, skips workflow cleanup after transfer.
- **Type:** boolean
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** false (default)
- **Since:** 6.9.4.1

split.history

- **Module:** cmas-dwh-server
- **Description:** Changes the SQL that fetches the history for the tickets during DWH transfer not to all tickets at once but only for one ticket per SQL.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** false
- **Since:** 6.8.0

start.groovy.task.enabled

- **Module:** cmas-app-admin-tool
- **Description:** For being able to run Admin Tool scripts of type *Task* in the Admin Tool (navigation group *Services*, navigation item *Task Execution*). It is required to enable the *Start task* button, which is hidden by default. This is done by setting this system property to *true*.
- **Type:** boolean
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** true
- **Since:** 6.9.4.0

supportEmail

- **Module:** cmweb-server-adapter
- **Description:**
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Since:** 6.0

synchronize.master.address

- **Module:** cmas-core-index-common

- **Description:** Value of *-Dcmas.http.host.port* specifying how to connect to the indexing master server. Default null. Since 6.6.17 this value is configurable in set-up to designate the initial indexing master server. Please note that changing this value is only allowed when all cluster nodes' index change receivers are stopped.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** 127.0.0.1:80
- **Since:** 6.6.0

[synchronize.master.security.token](#)

- **Module:** cmas-core-index-common
- **Description:** The password for accessing the index snapshot via URL, e.g., for index synchronization or for backups.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** token
- **Since:** 6.6.0

[synchronize.master.security.user](#)

- **Module:** cmas-core-index-common
- **Description:** The user name for accessing the index snapshot via URL, e.g., for index synchronization or for backups.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** user
- **Since:** 6.6.0

[synchronize.master.timeout.minutes](#)

- **Module:** cmas-core-index-common
- **Description:** How long the master server may continually fail until a new master gets elected. Default 5. Since 6.6.17 this value is configurable in set-up, where zero means that master server will never change (failover is disabled).

- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 5
- **Since:** 6.6.0

`synchronize.megabits.per.second`

- **Module:** cmas-core-index-common
- **Description:** How much bandwidth the master server may consume when transferring index changes to all slave servers. Default 85. Please do not use all available bandwidth to transfer index changes between hosts, as doing so will most probably partition the cluster due to some subsystems being unable to communicate.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 85
- **Since:** 6.6.0

`synchronize.sleep.millis`

- **Module:** cmas-core-index-common
- **Description:** How often each slave server polls the master server for index changes. Default 1000.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 1000
- **Since:** 6.6.0

`task.panel.refresh.interval.seconds`

- **Module:** cmas-app-admin-tool
- **Description:** Time in seconds after which the task list (in the Admin Tool) of the Task Execution Framework is refreshed.
- **Type:** Integer
- **Restart required:** no

- **System:** no
- **Optional:** no
- **Example value:** 10
- **Since:** 6.10.5.3 (not added automatically during update from versions prior to 6.10.5.3!)

themeOverlay

- **Module:** cmweb-server-adapter
- **Description:** Name of used theme overlay
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** consoliNT
- **Since:** 6.0

ticket.delete.timeout

- **Module:** cmas-core-server
- **Description:** Transaction timeout (in seconds) for deleting tickets.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 60
- **Since:** 6.1.3

ticketListRefreshIntervalInSeconds

- **Module:** cmweb-server-adapter
- **Description:** Refresh interval for ticket list (in seconds).
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 180
- **Since:** 6.0

ticketListSizeLimit

- **Module:** cmweb-server-adapter
- **Description:** Maximum number of tickets in ticket list.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 100
- **Since:** 6.0

tickets.delete.size

- **Module:** cmas-core-server
- **Description:** Defines a number of tickets deleted per transaction. By default it is set to 10.
- **Type:** integer
- **Restart required:** only Session Service
- **System:** yes
- **Optional:** no
- **Example value:** 10
- **Since:** 6.8.1

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:**
- **Optional:** no
- **Example value:** 60
- **Since:**

unit.replace.batchSize

- **Module:** cmas-core-server
- **Description:** Defines the number of objects to be processed in a unit replace action.
- **Type:** integer
- **Restart required:** no

- **System:** yes
- **Optional:** no
- **Example value:** 5
- **Since:** 6.8.2

`unit.replace.timeout`

- **Module:** cmas-core-server
- **Description:** Transaction timeout (seconds) of a unit replacement action step.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 120
- **Since:** 6.8.2

`unit.transfer.order`

- **Module:** cmas-dwh-server
- **Description:** Define in which order Data Object Groups should be transferred to the DWH.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** company;customer
- **Since:** 6.6.19
- **Removed in:** 6.8.1

`unitIndexSearchResultSizeLimit`

- **Module:** cmweb-server-adapter
- **Description:** Maximum number of units in unit search result (e.g. when searching for contact).
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 5
- **Since:** 6.0

unused.content.remover.cluster.node.id

- **Module:** cmas-core-server
- **Description:** Value of a *cmas.clusternode.id* designating which node will remove unused ticket attachments and unit content entries.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** 1 (assuming cluster node started with the parameter *-Dcmas.clusternode.id=1*)
- **Since:** 6.9.0.0

unused.content.remover.enabled

- **Module:** cmas-core-server
- **Description:** Specifies whether removal of unused ticket attachments and unit content entries should take place.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** true
- **Since:** 6.9.0.0

unused.content.remover.polling.minutes

- **Module:** cmas-core-server
- **Description:** How often unused ticket attachments and unit content entries should be checked for removal.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 15
- **Since:** 6.9.0.0

unused.content.remover.ttl.minutes

- **Module:** cmas-core-server
- **Description:** Minimum interval, in minutes, after which unused ticket attachments and unit content entries can be removed.

- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 1440
- **Since:** 6.9.0.0

urlLogoutPath

- **Module:** cmweb-server-adapter
- **Description:** URL which is used when user logs out. (If no value is set, logout leads to login-mask.)
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** http://intranet.consol.de
- **Since:** 6.3.1

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

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.2.2 List of System Properties by Module

This chapter lists the system properties included in the following modules:

- [cmas-app-admin-tool \(module\)](#)
- [cmas-core-cache \(module\)](#)
- [cmas-core-index-common \(module\)](#)
- [cmas-core-security \(module\)](#)
- [cmas-core-server \(module\)](#)
- [cmas-core-shared \(module\)](#)
- [cmas-dwh-server \(module\)](#)
- [cmas-esb-core \(module\)](#)
- [cmas-esb-mail \(module\)](#)
- [cmas-nimh \(module\)](#)
- [cmas-nimh-extension \(module\)](#)
- [cmas-setup-hibernate \(module\)](#)
- [cmas-setup-manager \(module\)](#)
- [cmas-setup-scene \(module\)](#)
- [cmas-workflow-engine \(module\)](#)
- [cmas-workflow-jbpm \(module\)](#)
- [cmweb-server-adapter \(module\)](#)

G.2.2.1 cmas-app-admin-tool (module)

admin.tool.session.check.interval

- **Module:** cmas-app-admin-tool
- **Description:** Admin Tool inactive (ended) sessions check time interval (in seconds)
- **Type:** integer
- **Restart required:** yes
- **System:** yes
- **Optional:** no
- **Example value:** 30
- **Since:** 6.7.5

autocomplete.enabled

- **Module:** cmas-app-admin-tool
- **Description:** If the flag is missing or its value is *false*, then the *Autocomplete address* navigation item is hidden in Admin Tool.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** true
- **Since:** 6.9.2.0

delete.ticket.enabled

- **Module:** cmas-app-admin-tool
- **Description:** Controls if the menu entry *Delete* is displayed in the context menu in the Admin Tool for the ticket list in ticket administration.
- **Type:** boolean
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** true
- **Since:** 6.9.4.0

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-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:**
- **Type:** integer
- **Restart required:** yes
- **System:** yes
- **Optional:** no
- **Example value:** 200000
- **Since:** 6.4.0

eviction.max.nodes

- **Module:** cmas-core-cache
- **Description:**
- **Type:** integer
- **Restart required:** yes
- **System:** yes
- **Optional:** no
- **Example value:** 100000
- **Since:** 6.4.0

eviction.wakeup.interval

- **Module:** cmas-core-cache
- **Description:**
- **Type:** integer
- **Restart required:** yes
- **System:** yes
- **Optional:** no
- **Example value:** 3000
- **Since:** 6.4.0

G.2.2.3 cmas-core-index-common (module)

big.task.minimum.size

- **Module:** cmas-core-index-common
- **Description:** Indicates the minimum size of index task (in parts, each part has 100 entities) to qualify this task as a big one. Big tasks have lower priority than normal tasks.
- **Type:** integer

- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 15 (default)
- **Since:** 6.8.3

database.notification.enabled

- **Module:** cmas-core-index-common
- **Description:** Indicates whether index update database notification channel should be used instead of JMS.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** false
- **Since:** 6.8.4.7

database.notification.redelivery.delay.seconds

- **Module:** cmas-core-index-common
- **Description:** In case of index update database notification channel, indicates notification redelivery delay when an exception occurs.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 60
- **Since:** 6.8.4.7

database.notification.redelivery.max.attempts

- **Module:** cmas-core-index-common
- **Description:** In case of index update database notification channel, indicates maximum redelivery attempts when an exception occurs.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no

- **Example value:** 60
- **Since:** 6.8.4.7

`disable.admin.task.auto.commit`

- **Module:** cmas-core-index-common
- **Description:** All tasks created for index update will be automatically executed right after creation.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** false
- **Since:** 6.6.1

`index.attachment`

- **Module:** cmas-core-index-common
- **Description:** Specifies whether content of attachments is indexed.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** true
- **Since:** 6.4.3

`index.history`

- **Module:** cmas-core-index-common
- **Description:** Specifies whether unit and ticket history are indexed.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** false
- **Since:** 6.1.0

index.status

- **Module:** cmas-core-index-common
- **Description:** Status of the Indexer, possible values RED, YELLOW, GREEN, will be displayed in the Admin Tool.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** GREEN
- **Since:** 6.6.1

index.task.worker.threads

- **Module:** cmas-core-index-common
- **Description:** How many threads will be used to execute index tasks (synchronization, administrative, and repair tasks).
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 1 (default) (we recommend to use a value not larger than 2)
- **Since:** 6.6.14, 6.7.3. Since 6.8.0 and exclusively in 6.6.21 also normal (live) index updates are affected by this property.

index.version.current

- **Module:** cmas-core-index-common
- **Description:** Holds information about current (possibly old) index version.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 1 (default)
- **Since:** 6.7.0

index.version.newest

- **Module:** cmas-core-index-common
- **Description:** Holds information about which index version is considered newest.
- **Type:** integer

- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 1 (default)
- **Since:** 6.7.0

`indexed.assets.per.thread.in.memory`

- **Module:** cmas-core-index-common
- **Description:** How many assets should be loaded into memory at once, per thread, during indexing.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 200 (default)
- **Since:** 6.8.0

`indexed.engineers.per.thread.in.memory`

- **Module:** cmas-core-index-common
- **Description:** How many engineers should be loaded into memory at once, per thread, during indexing.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 300 (default)
- **Since:** 6.6.14, 6.7.3

`indexed.resources.per.thread.in.memory`

- **Module:** cmas-core-index-common
- **Description:** How many resources should be loaded into memory at once, per thread, during indexing.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no

- **Example value:** 200 (default)
- **Since:** 6.10.0.0

`indexed.tickets.per.thread.in.memory`

- **Module:** cmas-core-index-common
- **Description:** How many tickets should be loaded into memory at once, per thread, during indexing.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 100 (default)
- **Since:** 6.6.14, 6.7.3

`indexed.units.per.thread.in.memory`

- **Module:** cmas-core-index-common
- **Description:** How many units should be loaded into memory at once, per thread, during indexing.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 200 (default)
- **Since:** 6.6.14, 6.7.3

`synchronize.master.address`

- **Module:** cmas-core-index-common
- **Description:** Value of *-Dcmas.http.host.port* specifying how to connect to the indexing master server. Default null. Since 6.6.17 this value is configurable in set-up to designate the initial indexing master server. Please note that changing this value is only allowed when all cluster nodes' index change receivers are stopped.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** 127.0.0.1:80
- **Since:** 6.6.0

`synchronize.master.security.token`

- **Module:** cmas-core-index-common
- **Description:** The password for accessing the index snapshot via URL, e.g., for index synchronization or for backups.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** token
- **Since:** 6.6.0

`synchronize.master.security.user`

- **Module:** cmas-core-index-common
- **Description:** The user name for accessing the index snapshot via URL, e.g., for index synchronization or for backups.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** user
- **Since:** 6.6.0

`synchronize.master.timeout.minutes`

- **Module:** cmas-core-index-common
- **Description:** How long the master server may continually fail until a new master gets elected. Default 5. Since 6.6.17 this value is configurable in set-up, where zero means that master server will never change (failover is disabled).
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 5
- **Since:** 6.6.0

[synchronize.megabits.per.second](#)

- **Module:** cmas-core-index-common
- **Description:** How much bandwidth the master server may consume when transferring index changes to all slave servers. Default 85. Please do not use all available bandwidth to transfer index changes between hosts, as doing so will most probably partition the cluster due to some subsystems being unable to communicate.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 85
- **Since:** 6.6.0

[synchronize.sleep.millis](#)

- **Module:** cmas-core-index-common
- **Description:** How often each slave server polls the master server for index changes. Default 1000.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 1000
- **Since:** 6.6.0

G.2.2.4 cmas-core-security (module)

[admin.email](#)

- **Module:** cmas-core-security
- **Description:** The e-mail address of the ConSol CM administrator. The value which you entered during system set-up is used initially.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** myuser@consol.de
- **Since:** 6.0

admin.login

- **Module:** cmas-core-security
- **Description:** The name of the ConSol CM administrator. The value which you entered during system set-up is used initially.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** admin
- **Since:** 6.0

authentication.method

- **Module:** cmas-core-security
- **Description:** User authentication method (internal CM database or LDAP authentication). Allowed values are LDAP or DATABASE.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** DATABASE
- **Since:** 6.0

contact.authentication.method

- **Module:** cmas-core-security
- **Description:** Indicates contact authentication method, where possible values are DATABASE or LDAP or LDAP,DATABASE or DATABASE,LDAP.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Since:** 6.9.3.0

contact.inherit.permissions.only.to.own.customer.group

- **Module:** cmas-core-security
- **Description:** Indicates whether authenticated contact inherits all customer group permissions from the representing engineer (false) or only has permissions to his own customer group (true).

- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Since:** 6.9.2.3

kerberos.v5.enabled

- **Module:** cmas-core-security
- **Description:** Indicates whether SSO via Kerberos is enabled.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** false (default if Kerberos was not enabled during system set-up)
- **Since:** 6.2.0

kerberos.v5.username.regex

- **Module:** cmas-core-security
- **Description:** Regular expression used for mapping Kerberos principals to CM user login names.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** (.*)@.*
- **Since:** 6.2.0

ldap.authentication

- **Module:** cmas-core-security
- **Description:** Authentication method used when using LDAP authentication.
- **Type:** string
- **Restart required:** yes
- **System:** yes
- **Optional:** no
- **Example value:** simple
- **Since:** 6.0

ldap.basedn

- **Module:** cmas-core-security
- **Description:** Base DN used for looking up LDAP user accounts when using LDAP authentication.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** ou=accounts,dc=consol,dc=de
- **Since:** 6.0

ldap.contact.name.basedn

- **Module:** cmas-core-security
- **Description:** Base path to search for contact DN by LDAP ID (e.g. ou=accounts,dc=consol,dc=de).
- **Type:** string
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Since:** 6.9.3.0

ldap.contact.name.password

- **Module:** cmas-core-security
- **Description:** Password to look up contact DN by LDAP ID. If not set, the anonymous account is used.
- **Type:** string
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Since:** 6.9.3.0

ldap.contact.name.providerurl

- **Module:** cmas-core-security
- **Description:** Address of the LDAP server (ldap[s]://host:port).
- **Type:** string
- **Restart required:** no
- **System:** no

- **Optional:** yes
- **Since:** 6.9.3.0

`ldap.contact.name.searchattr`

- **Module:** cmas-core-security
- **Description:** Attribute to search for contact DN by LDAP ID (e.g. uid).
- **Type:** string
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Since:** 6.9.3.0

`ldap.contact.name.userdn`

- **Module:** cmas-core-security
- **Description:** User DN to look up contact DN by LDAP ID. If not set, the anonymous account is used.
- **Type:** string
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Since:** 6.9.3.0

`ldap.initialcontextfactory`

- **Module:** cmas-core-security
- **Description:** Class name for the initial context factory of the LDAP implementation when using LDAP authentication. If it is not set, *com.sun.jndi.ldap.LdapCtxFactory* is used.
- **Type:** string
- **Restart required:** yes
- **System:** yes
- **Optional:** no
- **Example value:** *com.sun.jndi.ldap.LdapCtxFactory*
- **Since:** 6.0

`ldap.password`

- **Module:** cmas-core-security
- **Description:** Password for connecting to LDAP to look up users when using LDAP authentication. Only needed if look-up cannot be performed anonymously.
- **Type:** password

- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Since:** 6.1.2

ldap.providerurl

- **Module:** cmas-core-security
- **Description:** LDAP provider when using LDAP authentication.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** ldap://myserver.consol.de:389
- **Since:** 6.0

ldap.searchattr

- **Module:** cmas-core-security
- **Description:** Search attribute for looking up LDAP entry associated with a CM login.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** uid
- **Since:** 6.0

ldap.userdn

- **Module:** cmas-core-security
- **Description:** LDAP user for connecting to LDAP to look up users when using LDAP authentication. Only needed if look-up cannot be performed anonymously.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Since:** 6.1.2

policy.password.age

- **Module:** cmas-core-security
- **Description:** Defines (in days) how old the password may be.
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 5500 (15 years, default)
- **Since:** 6.10.1.0

policy.password.pattern

- **Module:** cmas-core-security
- **Description:** Defines password pattern.
- **Type:** string
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** "^.{3,}\$" (default)
- **Since:** 6.10.1.0

policy.rotation.ratio

- **Module:** cmas-core-security
- **Description:** Defines how often password may repeat. E.g., setting the value to X means that the new password cannot be present among the user's X previous passwords.
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 1 (default)
- **Since:** 6.10.1.0

policy.username.case.sensitive

- **Module:** cmas-core-security
- **Description:** Defines whether user names are case-sensitive.
- **Type:** boolean
- **Restart required:** no

- **System:** no
- **Optional:** yes
- **Example value:** true (default)
- **Since:** 6.10.1.0

G.2.2.5 cmas-core-server (module)

attachment.allowed.types

- **Module:** cmas-core-server
- **Description:** Comma-separated list of allowed filename extensions (if no value defined, all file extensions are allowed).
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** txt,zip,doc
- **Since:** 6.5.0

attachment.max.size

- **Module:** cmas-core-server
- **Description:** Maximum attachment size, in MB. This is a validation property of the CM API. It controls the size of attachments at tickets, at units, and at resources. It also controls the size of incoming (not outgoing!) e-mail attachments in NIMH as well as in Mule/ESB mode.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 100
- **Since:** 6.4.0

config.data.version

- **Module:** cmas-core-server
- **Description:**
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no

- **Example value:** 11
- **Since:** 6.0

defaultCommentClassName

- **Module:** cmas-core-server
- **Description:** Default text class name for comments.
- **Type:** string
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:**
- **Since:** 6.3.0

defaultIncommingMailClassName

- **Module:** cmas-core-server
- **Description:** Default text class name for incoming e-mails.
- **Type:** string
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Since:** 6.3.0

defaultOutgoingMailClassName

- **Module:** cmas-core-server
- **Description:** Default text class name for outgoing e-mails.
- **Type:** string
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:**
- **Since:** 6.3.0

fetchSize.strategy

- **Module:** cmas-core-server
- **Description:** Strategy for selecting the fetch size on JDBC result sets.
- **Type:** string
- **Restart required:** no

- **System:** yes
- **Optional:** yes
- **Example value:** FetchSizePageBasedStrategy, FetchSizeThresholdStrategy, FetchSizeFixedStrategy
- **Since:** 6.8.4.1

fetchSize.strategy.FetchSizeFixedStrategy.value

- **Module:** cmas-core-server
- **Description:** Sets fetch size value if the selected strategy to set the fetch size is *FetchSizeFixedStrategy*.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** 150
- **Since:** 6.8.4.1

fetchSize.strategy.FetchSizePageBasedStrategy.limit

- **Module:** cmas-core-server
- **Description:** Sets maximum fetch size value if the selected strategy to set the fetch size is *FetchSizePageBasedStrategy*.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** 10000
- **Since:** 6.8.4.1

fetchSize.strategy.FetchSizeThresholdStrategy.value

- **Module:** cmas-core-server
- **Description:** Sets fetch size threshold border values if the selected strategy to set the fetch size is *FetchSizeThresholdStrategy*.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** yes

- **Example value:** 150,300,600,1000
- **Since:** 6.8.4.1

last.config.change

- **Module:** cmas-core-server
- **Description:** Random UUID created during the last configuration change.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 2573c7b7-2bf5-47ff-b5a2-bad31951a266
- **Since:** 6.1.0, 6.2.1

ldap.certificate.basedn

- **Module:** cmas-core-server
- **Description:** Base DN for certificates location in the LDAP tree. If not provided, *cmas-core-security*, *ldap.basedn* is used.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** ou=accounts,dc=consol,dc=de
- **Since:** 6.8.4

ldap.certificate.content.attribute

- **Module:** cmas-core-server
- **Description:** LDAP attribute name used where certificate data is stored in the LDAP tree. Default value: usercertificate
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** usercertificate
- **Since:** 6.8.4

ldap.certificate.password

- **Module:** cmas-core-server
- **Description:** LDAP Certificates manager password. If not set, *cmas-core-security, ldap.-password* is used.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Since:** 6.8.4

ldap.certificate.providerurl

- **Module:** cmas-core-server
- **Description:** LDAP Certificates provider URL. If not set, *cmas-core-security, ldap.providerurl* is used.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** ldap://ldap.consol.de:389
- **Since:** 6.8.4

ldap.certificate.searchattr

- **Module:** cmas-core-server
- **Description:** LDAP attribute name used to search for certificate in the LDAP tree. Default value: mail
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** mail
- **Since:** 6.8.4

ldap.certificate.userdn

- **Module:** cmas-core-server
- **Description:** LDAP Certificates manager DN. If not set, *cmas-core-security, ldap.userdn* is used.
- **Type:** string
- **Restart required:** no

- **System:** yes
- **Optional:** yes
- **Since:** 6.8.4

mail.encryption

- **Module:** cmas-core-server
- **Description:** If property is set to *true*, the encrypt checkbox in the Ticket E-Mail Editor is checked by default.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** true (default = false)
- **Since:** 6.8.4.0

mail.notification.engineerChange

- **Module:** cmas-core-server
- **Description:** Whether notification e-mails 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 e-mails 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.smtp.email

- **Module:** cmas-core-server
- **Description:** SMTP e-mail URL for outgoing e-mails
- **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:** E-mail address used as sender in SMTP envelope. If not set, the FROM-address of the e-mail is used.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** mysender@mydomain.com
- **Since:** 6.5.7

max.licences.perUser

- **Module:** cmas-core-server
- **Description:** Sets maximum licenses single user can use (e.g., logging in from different browsers). By default this value is not restricted.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** 10
- **Since:** 6.8.4.5

monitoring.engineer.login

- **Module:** cmas-core-server
- **Description:** Login of monitoring engineer.
- **Type:** string
- **Restart required:** no

- **System:** yes
- **Optional:** yes
- **Example value:** nagios
- **Since:** 6.9.3.0

monitoring.unit.login

- **Module:** cmas-core-server
- **Description:** Login of monitoring unit.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** nagios
- **Since:** 6.9.3.0

nimh.enabled

- **Module:** cmas-core-server
- **Description:** Enables NIMH service. Must be suffixed with the cluster node ID, e.g., *nimh.enabled.NODEID = true*.
- **Type:** boolean
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** false
- **Since:** 6.9.4.0

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.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

server.session.archive.reaper.interval

- **Module:** cmas-core-server
- **Description:** Server archived sessions reaper interval (in seconds).
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** 60
- **Since:** 6.7.1

server.session.archive.timeout

- **Module:** cmas-core-server
- **Description:** Server sessions archive validity timeout (in days). After this time session info is removed from the DB.
- **Type:** integer
- **Restart required:** no

- **System:** yes
- **Optional:** no
- **Example value:** 31
- **Since:** 6.7.1

`server.session.reaper.interval`

- **Module:** cmas-core-server
- **Description:** Server inactive (ended) sessions reaper interval (in seconds).
- **Type:** integer
- **Restart required:** only Session Service
- **System:** yes
- **Optional:** no
- **Example value:** 60
- **Since:** 6.6.1, 6.7.1

`server.session.timeout`

- **Module:** cmas-core-server
- **Description:** Server session timeout (in seconds) for connected clients. Each client can over-write this timeout with custom value using its ID (ADMIN_TOOL, WEB_CLIENT, WORKFLOW_EDITOR, TRACK (before 6.8, please use PORTER), ETL, REST) appended to property name, e.g., *server.session.timeout.ADMIN_TOOL*.
Please see also the Page Customization attributes *updateTimeServerSessionActivityEnabled* and *updateTimeServerSessionActivity*, both of type *cmApplicationCustomization*.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 1800
- **Since:** 6.6.1, 6.7.1

Detailed explanation for the Admin Tool:

- `server.session.timeout.ADMIN_TOOL`
Defines the time interval how long the server considers a session valid while there is no activity from the Admin Tool holding the session. The Admin Tool is not aware of this value, it only suffers having an invalid session, if the last activity has been longer in the past.
- `admin.tool.session.check.interval`
Defines the time between two checks done by the Admin Tool, if the server still considers its session valid.

For example, if `admin.tool.session.check.interval = 60` the Admin Tool queries the server every minute if its session is still active/valid. In case `server.session.timeout.ADMIN_TOOL = 600` the Admin Tool will get the response that the session is now invalid after ten minutes of inactivity.

`skip.wfl.transfer.cleanup`

- **Module:** cmas-core-server
- **Description:** If set to *true*, skips workflow cleanup after transfer.
- **Type:** boolean
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** false (default)
- **Since:** 6.9.4.1

`tickets.delete.size`

- **Module:** cmas-core-server
- **Description:** Defines a number of tickets deleted per transaction. By default it is set to 10.
- **Type:** integer
- **Restart required:** only Session Service
- **System:** yes
- **Optional:** no
- **Example value:** 10
- **Since:** 6.8.1

`ticket.delete.timeout`

- **Module:** cmas-core-server
- **Description:** Transaction timeout (in seconds) for deleting tickets.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 60
- **Since:** 6.1.3

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:**
- **Optional:** no
- **Example value:** 60
- **Since:**

unit.replace.batchSize

- **Module:** cmas-core-server
- **Description:** Defines the number of objects to be processed in a unit replace action.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 5
- **Since:** 6.8.2

unit.replace.timeout

- **Module:** cmas-core-server
- **Description:** Transaction timeout (seconds) of a unit replacement action step.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 120
- **Since:** 6.8.2

unused.content.remover.cluster.node.id

- **Module:** cmas-core-server
- **Description:** Value of a *cmas.clusternode.id* designating which node will remove unused ticket attachments and unit content entries.
- **Type:** string

- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** 1 (assuming cluster node started with the parameter *-Dcmas.clusternode.id=1*)
- **Since:** 6.9.0.0

unused.content.remover.enabled

- **Module:** cmas-core-server
- **Description:** Specifies whether removal of unused ticket attachments and unit content entries should take place.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** true
- **Since:** 6.9.0.0

unused.content.remover.polling.minutes

- **Module:** cmas-core-server
- **Description:** How often unused ticket attachments and unit content entries should be checked for removal.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 15
- **Since:** 6.9.0.0

unused.content.remover.ttl.minutes

- **Module:** cmas-core-server
- **Description:** Minimum interval, in minutes, after which unused ticket attachments and unit content entries can be removed.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no

- **Example value:** 1440
- **Since:** 6.9.0.0

warmup.executor.enabled

- **Module:** cmas-core-server
- **Description:** Specifies whether the server should asynchronously warm up during startup (e.g., fill some of the internal caches).
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** true
- **Since:** 6.9.4.2

G.2.2.6 cmas-core-shared (module)

cluster.mode

- **Module:** cmas-core-shared
- **Description:** Specifies whether CMAS is running in cluster.
- **Type:** boolean
- **Restart required:** yes
- **System:** yes
- **Optional:** no
- **Example value:** false
- **Since:** 6.1.0

data.directory

- **Module:** cmas-core-shared
- **Description:** Directory for CMAS data (e.g., index)
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** C:\Users\user\cmas
- **Since:** 6.0

G.2.2.7 cmas-dwh-server (module)

autocommit.cf.changes

- **Module:** cmas-dwh-server
- **Description:**
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** false
- **Since:** 6.7.0

batch-commit-interval

- **Module:** cmas-dwh-server
- **Description:** Number of objects in a JMS message. Larger values mean better transfer performance at the cost of higher memory usage.
- **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. Possible values are DIRECT (database communication channel, default value since 6.9.4.1), JMS (default value before 6.9.4.1). Before 6.9.4.1 it has to be manually added.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** DIRECT
- **Since:** 6.8.5.0

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 transferred to the DWH.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** QueueName1,QueueName2,QueueName3
- **Since:** 6.6.19
- **Removed in:** 6.8.1

is.cmrf.alive

- **Module:** cmas-dwh-server
- **Description:** As a starting point, the time the last message was sent to CMRF should be used. If a response from CMRF is not received after value (in seconds), it should create a DWH operation status with an error message indicating that CMRF is down.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 1200
- **Since:** 6.7.0

java.naming.factory.initial

- **Module:** cmas-dwh-server
- **Description:** Factory class for the DWH context factory.

- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** org.jnp.interfaces.NamingContextFactory
- **Since:** 6.0.1

[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

[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

[notification.error.description](#)

- **Module:** cmas-dwh-server
- **Description:** Text for error e-mails 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 e-mails from the DWH
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Since:** 6.0.1

[notification.error.subject](#)

- **Module:** cmas-dwh-server
- **Description:** Subject for error e-mails 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 e-mails 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 e-mails 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 e-mails from the DWH when a transfer finishes successfully.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Since:** 6.0.1

notification.finished_successfully.subject

- **Module:** cmas-dwh-server
- **Description:** Subject for e-mails 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 e-mails 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 e-mails 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 e-mails from the DWH when a transfer finishes unsuccessfully.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Since:** 6.0.1

notification.finished_unsuccessfully.subject

- **Module:** cmas-dwh-server
- **Description:** Subject for e-mails 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 e-mails 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:** E-mail (SMTP) server hostname for sending DWH e-mails.

- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** myserver.consol.de
- **Since:** 6.0.1

notification.password

- **Module:** cmas-dwh-server
- **Description:** Password for sending DWH e-mails (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 e-mails.
- **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 e-mails from the DWH.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** pop3\

notification.username

- **Module:** cmas-dwh-server
- **Description:** (SMTP) User name for sending DWH e-mails.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** myuser
- **Since:** 6.0.1

skip-ticket

- **Module:** cmas-dwh-server
- **Description:** Tickets are not transferred during transfer/update.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** false
- **Since:** 6.6.19
- **Removed in:** 6.8.1

skip-ticket-history

- **Module:** cmas-dwh-server
- **Description:** History of ticket is not transferred during transfer/update.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** false
- **Since:** 6.6.19
- **Removed in:** 6.8.1

skip-unit

- **Module:** cmas-dwh-server
- **Description:** Units are not transferred during transfer/update.
- **Type:** boolean

- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** false
- **Since:** 6.6.19
- **Removed in:** 6.8.1

skip-unit-history

- **Module:** cmas-dwh-server
- **Description:** History of unit is not transferred during transfer/update.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** false
- **Since:** 6.6.19
- **Removed in:** 6.8.1

split.history

- **Module:** cmas-dwh-server
- **Description:** Changes the SQL that fetches the history for the tickets during DWH transfer not to all tickets at once but only for one ticket per SQL.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** false
- **Since:** 6.8.0

unit.transfer.order

- **Module:** cmas-dwh-server
- **Description:** Define in which order Data Object Groups should be transferred to the DWH.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes

- **Example value:** company;customer
- **Since:** 6.6.19
- **Removed in:** 6.8.1

G.2.2.8 cmas-esb-core (module)

esb.directory

- **Module:** cmas-esb-core
- **Description:** Directory used by Mule/ESB.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** C:\Users\user\cmas\mule
- **Since:** 6.0

G.2.2.9 cmas-esb-mail (module)

mail.attachments.validation.info.sender

- **Module:** cmas-esb-mail
- **Description:** Sets FROM-header of attachments type *error notification e-mail*. As a default the e-mail address of the administrator which you have entered during system set-up is used.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** admin@consolcm.com
- **Since:** 6.7.5

mail.attachments.validation.info.subject

- **Module:** cmas-esb-mail
- **Description:** Sets subject of attachments type *error notification e-mail*.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no

- **Example value:** E-mail was not processed because its attachments were rejected!
- **Since:** 6.7.5

mail.callname.pattern

- **Module:** cmas-esb-mail
- **Description:** Regular expression for subject of incoming e-mails. Available as TICKET_NAME_PATTERN_FORMAT in incoming e-mail scripts.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** .*?Ticket\s+\\((\\S+)\\).*
- **Since:** 6.0

mail.cluster.node.id

- **Module:** cmas-esb-mail
- **Description:** Only the node whose *mail.cluster.node.id* equals *cmas.clusternode.id* will start the Mule/ESB e-mail services.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** unspecified
- **Since:** 6.6.5

mail.db.archive

- **Module:** cmas-esb-mail
- **Description:** If property is set to *true*, incoming e-mails are archived in the database.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** false (default)
- **Since:** 6.8.5.5

Obsolete! In Mule/ESB mode, no e-mails are saved in the database. E-mails which could not be processed are stored in the file system, see section *E-Mail Backups* in the *ConSol CM Administrator Manual*.

mail.delete.read

- **Module:** cmas-esb-mail
- **Description:** Determines whether CM deletes messages fetched via IMAP(S). Setting value to *true* will cause deletion of messages after fetching. Default is to not delete messages fetched via IMAP(S). Note: Messages fetched via POP3(S) will always be deleted.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** true
- **Since:** 6.7.3

mail.incoming.uri

- **Module:** cmas-esb-mail
- **Description:** URL for incoming e-mails.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** pop3://cm-incoming-user:password@localhost:10110
- **Since:** 6.0



This value should not be edited here using the system properties pop-up window, but the mailboxes should be configured using the navigation item *E-mail*. Using this standard feature all entries are controlled - i.e., for each mailbox which is added, CM establishes a test connection during mailbox set-up. That way it is not possible to enter wrong values.

mail.max.restarts

- **Module:** cmas-esb-mail
- **Description:** Maximum number of e-mail service restarts before giving up.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 3
- **Since:** 6.0

mail.mime.strict

- **Module:** cmas-esb-mail
- **Description:** If set to *false*, e-mail addresses are not parsed for strict MIME compliance. Default is *true*, which means check for strict MIME compliance.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** false
- **Since:** 6.6.17, 6.7.3

mail.mule.service

- **Module:** cmas-esb-mail
- **Description:** FROM-address for e-mails sent by Mule service
- **Type:** email
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** myuser@consol.de
- **Since:** 6.0

mail.polling.interval

- **Module:** cmas-esb-mail
- **Description:** E-mail polling interval in ms.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 60000
- **Since:** 6.0

mail.process.error

- **Module:** cmas-esb-mail
- **Description:** TO-address for error e-mails from Mule. As a default the e-mail address of the administrator which you have entered during system set-up is used.
- **Type:** email
- **Restart required:** no

- **System:** yes
- **Optional:** no
- **Example value:** myuser@consol.de
- **Since:** 6.0

mail.process.retry.attempts

- **Module:** cmas-esb-mail
- **Description:** Number of retries when processing e-mail
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 3
- **Since:** 6.0.2

mail.process.timeout

- **Module:** cmas-esb-mail
- **Description:** E-mail processing timeout in seconds.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 60
- **Since:** 6.1.3

mail.redelivery.retry.count

- **Module:** cmas-esb-mail
- **Description:** Indicates the number of retries of re-delivering an e-mail from the CM system.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 3
- **Since:** 6.1.0

G.2.2.10 cmas-nimh (module)

filesystem.polling.threads.number

- **Module:** cmas-nimh
- **Description:** Number of threads started for db e-mails' queue polling. Default: 1
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 10
- **Since:** 6.4.0

filesystem.polling.threads.shutdown.timeout.seconds

- **Module:** cmas-nimh
- **Description:** Waiting time after the shutdown signal. When the timeout reached, thread will be terminated. Default: 60
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 60
- **Since:** 6.4.0

filesystem.polling.threads.watchdog.interval.seconds

- **Module:** cmas-nimh
- **Description:** Watchdog thread interval. Default: 30
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 60
- **Since:** 6.4.0

filesystem.task.enabled

- **Module:** cmas-nimh
- **Description:** With this property service thread related to given poller can be disabled. Default: true

- **Type:** boolean
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** true
- **Since:** 6.4.0

`filesystem.task.interval.seconds`

- **Module:** cmas-nimh
- **Description:** Default interval for polling mailboxes. Default: 60 seconds
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 60
- **Since:** 6.4.0

`filesystem.task.polling.folder`

- **Module:** cmas-nimh
- **Description:** Polling folder location which will be scanned for e-mails in the format of eml files. Default: "mail" subdir of cmas data directory
- **Type:** string
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** c://cmas//mail
- **Since:** 6.4.0

`filesystem.task.timeout.seconds`

- **Module:** cmas-nimh
- **Description:** After this time (of inactivity) the service thread is considered as damaged and automatically restarted. Default: 120 seconds
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes

- **Example value:** 60
- **Since:** 6.4.0

filesystem.task.transaction.timeout.seconds

- **Module:** cmas-nimh
- **Description:** Default transaction timeout for e-mail fetching transactions. Should be correlated with number of messages fetched at once. Default: 60 seconds
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 60
- **Since:** 6.4.0

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 e-mails.
- **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 e-mails.
- **Type:** string
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** consol
- **Since:** 6.4.0

mailbox.default.connection.port

- **Module:** cmas-nimh
- **Description:** Port for a given mailbox from which the poller reads e-mails.
- **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 e-mails.
- **Type:** string
- **Restart required:** no
- **System:** no

- **Optional:** yes
- **Example value:** username
- **Since:** 6.4.0

mailbox.default.session.mail.debug

- **Module:** cmas-nimh
- **Description:** Example javax.mail property - allows for more detailed javax.mail session debugging
- **Type:** boolean
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** true
- **Since:** 6.4.0

mailbox.default.session.mail.imap.timeout

- **Module:** cmas-nimh
- **Description:** Example javax.mail property
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 120
- **Since:** 6.4.0

mailbox.default.session.mail.mime.address.strict

- **Module:** cmas-nimh
- **Description:** Example javax.mail property - counterpart of the old *mule mail.mime.strict*, allows to set not so strict e-mail header parsing
- **Type:** boolean
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** true
- **Since:** 6.4.0

mailbox.default.session.mail.pop3.timeout

- **Module:** cmas-nimh
- **Description:** Example javax.mail property.
- **Type:**
- **Restart required:**
- **System:**
- **Optional:**
- **Example value:**
- **Since:** 6.4.0

mailbox.default.session.mail.<protocol>.partialfetch

- **Module:** cmas-nimh
- **Description:** e.g. mailbox.default.session.mail.imaps.partialfetch
- **Type:** boolean
- **Restart required:**
- **System:**
- **Optional:** yes
- **Example value:** false
- **Since:**

mailbox.default.task.delete.read.messages

- **Module:** cmas-nimh
- **Description:** This defines whether messages should be removed from the mailbox after processing. For IMAP protocol messages are marked as SEEN by default. For POP3 protocol, when flag is set to true the message is removed, otherwise remains on server and will result in infinite reads. Default: false.
- **Type:** boolean
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** false
- **Since:** 6.4.0

mailbox.default.task.enabled

- **Module:** cmas-nimh
- **Description:** With this property service thread related to given poller can be disabled. Default: true

- **Type:** boolean
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** false
- **Since:** 6.4.0

mailbox.default.task.interval.seconds

- **Module:** cmas-nimh
- **Description:** Default interval for polling mailboxes. Default: 60 seconds
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 60
- **Since:** 6.4.0

mailbox.default.task.max.message.size

- **Module:** cmas-nimh
- **Description:** Maximum size of e-mail messages (i.e., e-mail plus attachment). E-mails 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 e-mail backups in the Admin Tool (see section *E-Mail 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. Default is set to 10MB: 10485760
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 10485760
- **Since:** 6.4.0

mailbox.default.task.max.messages.per.run

- **Module:** cmas-nimh
- **Description:** Number of messages fetched at once from mailbox. Must be correlated with transaction timeout. Default set to: 20
- **Type:** integer
- **Restart required:** no

- **System:** no
- **Optional:** yes
- **Example value:** 60
- **Since:** 6.4.0

mailbox.default.task.timeout.seconds

- **Module:** cmas-nimh
- **Description:** After this time (of inactivity) the service thread is considered as damaged and automatically restarted. Default: 120 seconds
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 60
- **Since:** 6.4.0

mailbox.default.task.transaction.timeout.seconds

- **Module:** cmas-nimh
- **Description:** Default transaction timeout for e-mail fetching transactions. Should be correlated with number of messages fetched at once. Default: 60 seconds
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 60
- **Since:** 6.4.0

mailbox.polling.threads.mail.log.enabled

- **Module:** cmas-nimh
- **Description:** Enables e-mail logging which is especially crucial in cluster environment (used as semaphore there)
- **Type:** boolean
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** true (default)
- **Since:** 6.9.4.1

mailbox.polling.threads.number

- **Module:** cmas-nimh
- **Description:** Number of threads for accessing mailboxes. Default: 1
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 1
- **Since:** 6.4.0

queue.polling.threads.number

- **Module:** cmas-nimh
- **Description:** Number of threads started for e-mails' queue polling. Default: 1
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 1
- **Since:** 6.4.0

queue.polling.threads.shutdown.timeout.seconds

- **Module:** cmas-nimh
- **Description:** Waiting time after the shutdown signal. When the timeout is reached, the thread will be terminated. Default: 60
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 60
- **Since:** 6.4.0

queue.polling.threads.watchdog.interval.seconds

- **Module:** cmas-nimh
- **Description:** Watchdog thread interval. Default: 30
- **Type:** integer
- **Restart required:** no

- **System:** no
- **Optional:** yes
- **Example value:** 30
- **Since:** 6.4.0

`queue.task.error.pause.seconds`

- **Module:** cmas-nimh
- **Description:** Maximum number of seconds, the queue poller waits after infrastructure (e.g. database) error. Default 180 seconds
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 180
- **Since:** 6.4.0

`queue.task.interval.seconds`

- **Module:** cmas-nimh
- **Description:** Main e-mails' queue polling thread interval. Default: 15
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 15
- **Since:** 6.4.0

`queue.task.max.retries`

- **Module:** cmas-nimh
- **Description:** Maximum number of e-mail processing retries after an exception. When reached, the e-mail is moved to the e-mail archive. This e-mail can be rescheduled again using NIMH API (or the Admin Tool).
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 10
- **Since:** 6.4.0

queue.task.timeout.seconds

- **Module:** cmas-nimh
- **Description:** After this time (of inactivity) the service thread is considered as damaged and automatically restarted. Default: 600 seconds
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 600
- **Since:** 6.4.0


queue.task.transaction.timeout.seconds

- **Module:** cmas-nimh
- **Description:** Transaction timeout for e-mail processing in the pipe. Default: 60
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 60
- **Since:** 6.4.0

G.2.2.11 cmas-nimh-extension (module)


mail.attachments.validation.info.sender

- **Module:** cmas-nimh-extension
- **Description:** Sets FROM-header of attachments type *error notification mail*
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** admin@mail.com
- **Since:** 6.7.5

 This is an equivalent to the old *cmas-esb-mail*, *mail.attachments.validation.info.sender*

mail.attachments.validation.info.subject


- **Module:** cmas-nimh-extension
- **Description:** Sets subject of attachments type error notification mail.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** E-mail was not processed because its attachments were rejected!
- **Since:** 6.7.5

 This is an equivalent to the old *cmas-esb-mail, mail.attachments.validation.info.subject*


mail.db.archive

- **Module:** cmas-nimh-extension
- **Description:** If property is set to *true*, incoming e-mails are archived in the database.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** false (default)
- **Since:** 6.8.5.5

mail.error.from.address

 This is an equivalent to the old *cmas-esb-mail, mail.mule.service*

mail.error.to.address

 This is an equivalent to the old *cmas-esb-mail, mail.process.error*

mail.on.error

- **Module:** cmas-nimh-extension
- **Description:** If set to *true* an error e-mail is sent to the above configured address in case the e-mail message could not be processed. Default: true
- **Type:** boolean
- **Restart required:** no

- **System:** no
- **Optional:** yes
- **Example value:** false
- **Since:** 6.4.0

mail.process.error

- **Module:** cmas-nimh-extension
- **Description:** TO-address for error e-mails from Mule.
- **Type:** email
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** myuser@consol.de
- **Since:** 6.4.0

mail.ticketname.pattern

- **Module:** cmas-nimh-extension
- **Description:**
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** .*?Ticket\s+\\((\\S+)\\).*
- **Since:** 6.4.0

G.2.2.12 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

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

G.2.2.13 cmas-setup-manager (module)

initialized

- **Module:** cmas-setup-manager
- **Description:** Flag if CMAS is initialized. If this value is missing or not *true*, set-up will be performed.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** true
- **Since:** 6.0



Be careful with using this property!!! When you set the value to *false*, the ConSol CM server will perform the system set-up at the next start, i.e. all data of the existing system is lost, including system properties!!!

G.2.2.14 cmas-setup-scene (module)

scene

- **Module:** cmas-setup-scene
- **Description:** Scene file which was imported during set-up (can be empty).
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no

- **Example value:** vfszip:/P:/dist/target/jboss/server/emas/deploy/cm-dist-6.5.1-SNAPSHOT.ear/APP-INF/lib/dist-scene-6.5.1-SNAPSHOT.jar/META-INF/emas/scenes/helpdesk-sales_scene.jar/
- **Since:** 6.0

G.2.2.15 emas-workflow-engine (module)

`jobExecutor.adminMail`

- **Module:** emas-workflow-engine
- **Description:** E-mail address which will get notified about job execution problems (when retry counter is exceeded).
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** admin@consol.de
- **Since:** 6.8.0

`jobExecutor.idleInterval.seconds`

- **Module:** emas-workflow-engine
- **Description:** Determines how often job executor thread will look for new jobs to execute.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** 45 (default up to CM version 6.10.5.2. Default CM versions 6.10.5.3 and up is 5)
- **Since:** 6.8.0

`jobExecutor.jobMaxRetries`

- **Module:** emas-workflow-engine
- **Description:**
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** 5 (default)
- **Since:** 6.8.0

`jobExecutor.jobMaxRetriesReachedSubject`

- **Module:** cmas-workflow-engine
- **Description:**
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** Job maximum retries reached. Job was removed!!! (default)
- **Since:** 6.8.0

`jobExecutor.lockingLimit`

- **Module:** cmas-workflow-engine
- **Description:** Number of jobs locked at once (marked for execution) by job executor thread.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** 5 (default since CM version 6.10.5.3)
- **Since:** 6.8.0

`jobExecutor.lockTimeout.seconds`

- **Module:** cmas-workflow-engine
- **Description:** How long the job can be locked (marked for execution) by job executor.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** 360 (default)
- **Since:** 6.8.0

`jobExecutor.mailFrom`

- **Module:** cmas-workflow-engine
- **Description:** E-mail which will be set as FROM-header during admin notifications.
- **Type:** string
- **Restart required:** no
- **System:** yes

- **Optional:** yes
- **Example value:** jobexecutor@consol.de
- **Since:** 6.8.0

`jobExecutor.maxInactivityInterval.minutes`

- **Module:** cmas-workflow-engine
- **Description:** Number of minutes of allowed job executor inactivity (e.g. when it is blocked by long timer execution). After this time executors threads are restarted.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** yes. Default value is set to 30 minutes
- **Example value:** 15 (default)
- **Since:** 6.9.2.0

`jobExecutor.threads`

- **Module:** cmas-workflow-engine
- **Description:** Number of job execution threads.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** 1 (default)
- **Since:** 6.8.0

`jobExecutor.timerRetryInterval.seconds`

- **Module:** cmas-workflow-engine
- **Description:** Determines how long job executor thread will wait after job execution error.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** 10 (default up to CM version 6.10.5.2. Default CM versions 6.10.5.3 and up is 30)
- **Since:** 6.8.0

`jobExecutor.txTimeout.seconds`

- **Module:** cmas-workflow-engine
- **Description:** Transaction timeout used for job execution.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** 60 (default)
- **Since:** 6.8.0

G.2.2.16 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 e-mails from the workflow engine.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** myuser@consol.de
- **Removed in:** 6.8.0
- **Replaced by:** jobExecutor.mailFrom

outdated.lock.age

- **Module:** cmas-workflow-jbpm
- **Description:**
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no

- **Example value:** 60000
- **Removed in:** 6.8.0
- **Replaced by:** cmas-workflow-engine, jobExecutor.lockTimeout.seconds

refreshTimeInCaseOfConcurrentRememberMeRequests

- **Module:** cmas-workflow-jbpm
- **Description:** It sets the refresh time (in seconds) after which page will be reloaded in case of concurrent remember me requests. This feature prevents one user from occupying many licenses. Please increase that time if sessions are still occupying.
- **Type:** integer
- **Restart required:** yes
- **System:** yes
- **Optional:** yes
- **Example value:** 5
- **Since:** 6.8.2

G.2.2.17 cmweb-server-adapter (module)

attachment.upload.timeout

- **Module:** cmweb-server-adapter
- **Description:** Defines the transaction timeout in minutes for adding attachments to a ticket, a resource or a customer. Counts the time for the upload of all attachments of one transaction. When the timeout occurs, all files which have been temporarily stored on the server are deleted. No file is uploaded.
- **Type:** Integer
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** 3
- **Since:** 6.10.5.3

automatic.booking.enabled

- **Module:** cmweb-server-adapter
- **Description:** If enabled, time spend on creating comment/e-mail 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

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

commentRequiredForTicketCreation

- **Module:** cmweb-server-adapter
- **Description:** Flag if comment is a required field for ticket creation.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** true (default)
- **Since:** 6.2.0

customizationVersion

- **Module:** cmweb-server-adapter
- **Description:**

- **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

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 Custom Fields.
- **Type:** integer
- **Restart required:** no

- **System:** yes
- **Optional:** no
- **Example value:** 3
- **Since:** 6.2.0

diffTrackingEnabled

- **Module:** cmweb-server-adapter
- **Description:** Defines if parallel editing of a ticket by different engineers should be possible. Default is *true*.
- **Type:** boolean
- **Restart required:** no
- **System:**
- **Optional:**
- **Example value:** true
- **Since:** 6.10.1

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

globalSearchResultSizeLimit

- **Module:** cmweb-server-adapter
- **Description:** Maximum number of items in Quick Search result.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 10
- **Since:** 6.0

helpFilePath

- **Module:** cmweb-server-adapter
- **Description:** URL for online help. If not empty, Help button is displayed in Web Client.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** http://www.consol.de
- **Since:** 6.2.1

hideTicketSubject

- **Module:** cmweb-server-adapter
- **Description:** If set to *true*, ticket subject is hidden.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** false
- **Since:** 6.2.1

mail.from

- **Module:** cmweb-server-adapter
- **Description:** Use this address if set instead of engineer e-mail address during e-mail conversation.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Since:** 6.1.2

mail.reply.to

- **Module:** cmweb-server-adapter
- **Description:** When set, Web Client will display REPLY-TO-field on e-mail send, prefilled with this value.
- **Type:** string
- **Restart required:** no
- **System:** yes

- **Optional:** yes
- **Since:** 6.0.1



Please read the detailed information about ConSol CM REPLY-TO-addresses in section *Scripts of Type E-Mail* in the *ConSol CM Administrator Manual*.

mailTemplateAboveQuotedText

- **Module:** cmweb-server-adapter
- **Description:** Indicates behavior of e-mail template in the Ticket E-Mail Editor when another e-mail is quoted, i.e. forwarded or replied to. 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:** Maximum size (in MB) for each Wicket pagemap.
- **Type:** integer
- **Restart required:** no
- **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
- **Since:** 6.0

rememberMeLifetimeInMinutes

- **Module:** cmweb-server-adapter
- **Description:** Lifetime for *remember me* in minutes.
- **Type:** integer
- **Restart required:** yes
- **System:** yes
- **Optional:** no
- **Example value:** 1440
- **Since:** 6.0

request.scope.transaction

- **Module:** cmweb-server-adapter
- **Description:** It allows to disable request scope transaction. By default one transaction is used per request. Setting this property to *false* there will cause one transaction per service method invocation.
- **Type:** boolean

- **Restart required:** yes
- **System:** yes
- **Optional:** yes
- **Example value:** true
- **Since:** 6.8.1

searchPageSize

- **Module:** cmweb-server-adapter
- **Description:** Default page size for search results.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 20
- **Since:** 6.0

searchPageSizeOptions

- **Module:** cmweb-server-adapter
- **Description:** Options for page size for search results.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 10|20|30|40|50|75|100
- **Since:** 6.0

serverPoolingInterval

- **Module:** cmweb-server-adapter
- **Description:**
- **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

themeOverlay

- **Module:** cmweb-server-adapter
- **Description:** Name of used theme overlay
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** consoliNT
- **Since:** 6.0

ticketListRefreshIntervalInSeconds

- **Module:** cmweb-server-adapter
- **Description:** Refresh interval for ticket list (in seconds).
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 180
- **Since:** 6.0

ticketListSizeLimit

- **Module:** cmweb-server-adapter
- **Description:** Maximum number of tickets in ticket list.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no

- **Example value:** 100
- **Since:** 6.0

unitIndexSearchResultSizeLimit

- **Module:** cmweb-server-adapter
- **Description:** Maximum number of units in unit search result (e.g. when searching for contact).
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 5
- **Since:** 6.0

urlLogoutPath

- **Module:** cmweb-server-adapter
- **Description:** URL which is used when user logs out. (If no value is set, logout leads to login-mask.)
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** http://intranet.consol.de
- **Since:** 6.3.1

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](#)
- [E-Mail Configuration](#)
- [Activity Interval Configuration](#)
- [Administrator E-Mail Addresses](#)

G.2.3.1 CMRF & DWH Configuration

autocommit.cf.changes

- **Module:** cmas-dwh-server
- **Description:**
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** false
- **Since:** 6.7.0

batch-commit-interval

- **Module:** cmas-dwh-server
- **Description:** Number of objects in a JMS message. Larger values mean better transfer performance at the cost of higher memory usage.
- **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. Possible values are DIRECT (database communication channel, default value since 6.9.4.1), JMS (default value before 6.9.4.1). Before 6.9.4.1 it has to be manually added.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** DIRECT
- **Since:** 6.8.5.0

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 transferred to the DWH.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** QueueName1,QueueName2,QueueName3
- **Since:** 6.6.19
- **Removed in:** 6.8.1

is.cmrf.alive

- **Module:** cmas-dwh-server
- **Description:** As a starting point, the time the last message was sent to CMRF should be used. If a response from CMRF is not received after value (in seconds), it should create a DWH operation status with an error message indicating that CMRF is down.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 1200
- **Since:** 6.7.0

java.naming.factory.initial

- **Module:** cmas-dwh-server
- **Description:** Factory class for the DWH context factory.

- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** org.jnp.interfaces.NamingContextFactory
- **Since:** 6.0.1

[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

[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

[notification.error.description](#)

- **Module:** cmas-dwh-server
- **Description:** Text for error e-mails 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 e-mails from the DWH
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Since:** 6.0.1

notification.error.subject

- **Module:** cmas-dwh-server
- **Description:** Subject for error e-mails 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 e-mails 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 e-mails 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 e-mails from the DWH when a transfer finishes successfully.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Since:** 6.0.1

notification.finished_successfully.subject

- **Module:** cmas-dwh-server
- **Description:** Subject for e-mails 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 e-mails 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 e-mails 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 e-mails from the DWH when a transfer finishes unsuccessfully.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Since:** 6.0.1

notification.finished_unsuccessfully.subject

- **Module:** cmas-dwh-server
- **Description:** Subject for e-mails 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 e-mails 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:** E-mail (SMTP) server hostname for sending DWH e-mails.

- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** myserver.consol.de
- **Since:** 6.0.1

notification.password

- **Module:** cmas-dwh-server
- **Description:** Password for sending DWH e-mails (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 e-mails.
- **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 e-mails from the DWH.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** pop3\

notification.username

- **Module:** cmas-dwh-server
- **Description:** (SMTP) User name for sending DWH e-mails.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** myuser
- **Since:** 6.0.1

skip-ticket

- **Module:** cmas-dwh-server
- **Description:** Tickets are not transferred during transfer/update.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** false
- **Since:** 6.6.19
- **Removed in:** 6.8.1

skip-ticket-history

- **Module:** cmas-dwh-server
- **Description:** History of ticket is not transferred during transfer/update.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** false
- **Since:** 6.6.19
- **Removed in:** 6.8.1

skip-unit

- **Module:** cmas-dwh-server
- **Description:** Units are not transferred during transfer/update.
- **Type:** boolean

- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** false
- **Since:** 6.6.19
- **Removed in:** 6.8.1

skip-unit-history

- **Module:** cmas-dwh-server
- **Description:** History of unit is not transferred during transfer/update.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** false
- **Since:** 6.6.19
- **Removed in:** 6.8.1

split.history

- **Module:** cmas-dwh-server
- **Description:** Changes the SQL that fetches the history for the tickets during DWH transfer not to all tickets at once but only for one ticket per SQL.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** false
- **Since:** 6.8.0

unit.transfer.order

- **Module:** cmas-dwh-server
- **Description:** Define in which order Data Object Groups should be transferred to the DWH.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes

- **Example value:** company;customer
- **Since:** 6.6.19
- **Removed in:** 6.8.1

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 lower priority than normal tasks.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 15 (default)
- **Since:** 6.8.3

database.notification.enabled

- **Module:** cmas-core-index-common
- **Description:** Indicates whether index update database notification channel should be used instead of JMS.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** false
- **Since:** 6.8.4.7

database.notification.redelivery.delay.seconds

- **Module:** cmas-core-index-common
- **Description:** In case of index update database notification channel, indicates notification redelivery delay when an exception occurs.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no

- **Example value:** 60
- **Since:** 6.8.4.7

database.notification.redelivery.max.attempts

- **Module:** cmas-core-index-common
- **Description:** In case of index update database notification channel, indicates maximum redelivery attempts when an exception occurs.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 60
- **Since:** 6.8.4.7

disable.admin.task.auto.commit

- **Module:** cmas-core-index-common
- **Description:** All tasks created for index update will be automatically executed right after creation.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** false
- **Since:** 6.6.1

index.attachment

- **Module:** cmas-core-index-common
- **Description:** Specifies whether content of attachments is indexed.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** true
- **Since:** 6.4.3

index.history

- **Module:** cmas-core-index-common
- **Description:** Specifies whether unit and ticket history are indexed.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** false
- **Since:** 6.1.0

index.status

- **Module:** cmas-core-index-common
- **Description:** Status of the Indexer, possible values RED, YELLOW, GREEN, will be displayed in the Admin Tool.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** GREEN
- **Since:** 6.6.1

index.task.worker.threads

- **Module:** cmas-core-index-common
- **Description:** How many threads will be used to execute index tasks (synchronization, administrative, and repair tasks).
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 1 (default) (we recommend to use a value not larger than 2)
- **Since:** 6.6.14, 6.7.3. Since 6.8.0 and exclusively in 6.6.21 also normal (live) index updates are affected by this property.

index.version.current

- **Module:** cmas-core-index-common
- **Description:** Holds information about current (possibly old) index version.
- **Type:** integer

- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 1 (default)
- **Since:** 6.7.0

`index.version.newest`

- **Module:** cmas-core-index-common
- **Description:** Holds information about which index version is considered newest.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 1 (default)
- **Since:** 6.7.0

`indexed.assets.per.thread.in.memory`

- **Module:** cmas-core-index-common
- **Description:** How many assets should be loaded into memory at once, per thread, during indexing.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 200 (default)
- **Since:** 6.8.0

`indexed.engineers.per.thread.in.memory`

- **Module:** cmas-core-index-common
- **Description:** How many engineers should be loaded into memory at once, per thread, during indexing.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 300 (default)
- **Since:** 6.6.14, 6.7.3

indexed.resources.per.thread.in.memory

- **Module:** cmas-core-index-common
- **Description:** How many resources should be loaded into memory at once, per thread, during indexing.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 200 (default)
- **Since:** 6.10.0.0

indexed.tickets.per.thread.in.memory

- **Module:** cmas-core-index-common
- **Description:** How many tickets should be loaded into memory at once, per thread, during indexing.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 100 (default)
- **Since:** 6.6.14, 6.7.3

indexed.units.per.thread.in.memory

- **Module:** cmas-core-index-common
- **Description:** How many units should be loaded into memory at once, per thread, during indexing.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 200 (default)
- **Since:** 6.6.14, 6.7.3

`synchronize.master.address`

- **Module:** cmas-core-index-common
- **Description:** Value of *-Dcmas.http.host.port* specifying how to connect to the indexing master server. Default null. Since 6.6.17 this value is configurable in set-up to designate the initial indexing master server. Please note that changing this value is only allowed when all cluster nodes' index change receivers are stopped.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** 127.0.0.1:80
- **Since:** 6.6.0

`synchronize.master.security.token`

- **Module:** cmas-core-index-common
- **Description:** The password for accessing the index snapshot via URL, e.g., for index synchronization or for backups.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** token
- **Since:** 6.6.0

`synchronize.master.security.user`

- **Module:** cmas-core-index-common
- **Description:** The user name for accessing the index snapshot via URL, e.g., for index synchronization or for backups.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** user
- **Since:** 6.6.0

`synchronize.master.timeout.minutes`

- **Module:** cmas-core-index-common
- **Description:** How long the master server may continually fail until a new master gets elected. Default 5. Since 6.6.17 this value is configurable in set-up, where zero means that master server will never change (failover is disabled).
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 5
- **Since:** 6.6.0

`synchronize.megabits.per.second`

- **Module:** cmas-core-index-common
- **Description:** How much bandwidth the master server may consume when transferring index changes to all slave servers. Default 85. Please do not use all available bandwidth to transfer index changes between hosts, as doing so will most probably partition the cluster due to some subsystems being unable to communicate.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 85
- **Since:** 6.6.0

`synchronize.sleep.millis`

- **Module:** cmas-core-index-common
- **Description:** How often each slave server polls the master server for index changes. Default 1000.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 1000
- **Since:** 6.6.0

Search Results

globalSearchResultSizeLimit

- **Module:** cmweb-server-adapter
- **Description:** Maximum number of items in Quick Search result.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 10
- **Since:** 6.0

searchPageSize

- **Module:** cmweb-server-adapter
- **Description:** Default page size for search results.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 20
- **Since:** 6.0

searchPageSizeOptions

- **Module:** cmweb-server-adapter
- **Description:** Options for page size for search results.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 10|20|30|40|50|75|100
- **Since:** 6.0

unitIndexSearchResultSizeLimit

- **Module:** cmweb-server-adapter
- **Description:** Maximum number of units in unit search result (e.g. when searching for contact).
- **Type:** integer
- **Restart required:** no

- **System:** yes
- **Optional:** no
- **Example value:** 5
- **Since:** 6.0

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.
- **Type:** string
- **Restart required:** yes
- **System:** yes
- **Optional:** no
- **Example value:** simple
- **Since:** 6.0

`ldap.basedn`

- **Module:** cmas-core-security
- **Description:** Base DN used for looking up LDAP user accounts when using LDAP authentication.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no

- **Example value:** ou=accounts,dc=consol,dc=de
- **Since:** 6.0

ldap.initialcontextfactory

- **Module:** cmas-core-security
- **Description:** Class name for the initial context factory of the LDAP implementation when using LDAP authentication. If it is not set, *com.sun.jndi.ldap.LdapCtxFactory* is used.
- **Type:** string
- **Restart required:** yes
- **System:** yes
- **Optional:** no
- **Example value:** com.sun.jndi.ldap.LdapCtxFactory
- **Since:** 6.0

ldap.password

- **Module:** cmas-core-security
- **Description:** Password for connecting to LDAP to look up users when using LDAP authentication. Only needed if look-up cannot be performed anonymously.
- **Type:** password
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Since:** 6.1.2

ldap.providerurl

- **Module:** cmas-core-security
- **Description:** LDAP provider when using LDAP authentication.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** ldap://myserver.consol.de:389
- **Since:** 6.0

ldap.searchattr

- **Module:** cmas-core-security
- **Description:** Search attribute for looking up LDAP entry associated with a CM login.

- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** uid
- **Since:** 6.0

ldap.userdn

- **Module:** cmas-core-security
- **Description:** LDAP user for connecting to LDAP to look up users when using LDAP authentication. Only needed if look-up cannot be performed anonymously.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Since:** 6.1.2

LDAP Configuration (if LDAP is Used as Authentication Mode in CM.Track)

LDAP parameters apply only if the authentication mode for CM.Track has been set to *LDAP*:

contact.authentication.method

- **Module:** cmas-core-security
- **Description:** Indicates contact authentication method, where possible values are DATABASE or LDAP or LDAP,DATABASE or DATABASE,LDAP.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Since:** 6.9.3.0

ldap.contact.name.basedn

- **Module:** cmas-core-security
- **Description:** Base path to search for contact DN by LDAP ID (e.g. ou=a-ccounts,dc=consol,dc=de).
- **Type:** string
- **Restart required:** no
- **System:** no

- **Optional:** yes
- **Since:** 6.9.3.0

ldap.contact.name.password

- **Module:** cmas-core-security
- **Description:** Password to look up contact DN by LDAP ID. If not set, the anonymous account is used.
- **Type:** string
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Since:** 6.9.3.0

ldap.contact.name.providerurl

- **Module:** cmas-core-security
- **Description:** Address of the LDAP server (ldap[s]://host:port).
- **Type:** string
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Since:** 6.9.3.0

ldap.contact.name.searchattr

- **Module:** cmas-core-security
- **Description:** Attribute to search for contact DN by LDAP ID (e.g. uid).
- **Type:** string
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Since:** 6.9.3.0

ldap.contact.name.userdn

- **Module:** cmas-core-security
- **Description:** User DN to look up contact DN by LDAP ID. If not set, the anonymous account is used.
- **Type:** string
- **Restart required:** no

- **System:** no
- **Optional:** yes
- **Since:** 6.9.3.0

ldap.initialcontextfactory

- **Module:** cmas-core-security
- **Description:** Class name for the initial context factory of the LDAP implementation when using LDAP authentication. If it is not set, *com.sun.jndi.ldap.LdapCtxFactory* is used.
- **Type:** string
- **Restart required:** yes
- **System:** yes
- **Optional:** no
- **Example value:** com.sun.jndi.ldap.LdapCtxFactory
- **Since:** 6.0

G.2.3.4 E-Mail Configuration

Outgoing E-Mail

Independent of incoming e-mail mode (Mule/ESB and NIMH).

mail.smtp.email

- **Module:** cmas-core-server
- **Description:** SMTP e-mail URL for outgoing e-mails
- **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:** E-mail address used as sender in SMTP envelope. If not set, the FROM-address of the e-mail is used.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no

- **Example value:** mysender@mydomain.com
- **Since:** 6.5.7

mail.from

- **Module:** cmweb-server-adapter
- **Description:** Use this address if set instead of engineer e-mail address during e-mail conversation.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Since:** 6.1.2

mail.reply.to

- **Module:** cmweb-server-adapter
- **Description:** When set, Web Client will display REPLY-TO-field on e-mail send, prefilled with this value.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Since:** 6.0.1



Please read the detailed information about ConSol CM REPLY-TO-addresses in section *Scripts of Type E-Mail* in the *ConSol CM Administrator Manual*.

mailTemplateAboveQuotedText

- **Module:** cmweb-server-adapter
- **Description:** Indicates behavior of e-mail template in the Ticket E-Mail Editor when another e-mail is quoted, i.e. forwarded or replied to. 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 e-mails from the workflow engine.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** myuser@consol.de
- **Removed in:** 6.8.0
- **Replaced by:** jobExecutor.mailFrom

Incoming E-Mail

Settings for Mule/ESB

esb.directory

- **Module:** cmas-esb-core
- **Description:** Directory used by Mule/ESB.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** C:\Users\user\cmas\mule
- **Since:** 6.0

mail.attachments.validation.info.sender

- **Module:** cmas-esb-mail
- **Description:** Sets FROM-header of attachments type error *notification e-mail*. As a default the e-mail address of the administrator which you have entered during system set-up is used.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** admin@consolcm.com
- **Since:** 6.7.5

mail.attachments.validation.info.subject

- **Module:** cmas-esb-mail
- **Description:** Sets subject of attachments type *error notification e-mail*.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** E-mail was not processed because its attachments were rejected!
- **Since:** 6.7.5

mail.callname.pattern

- **Module:** cmas-esb-mail
- **Description:** Regular expression for subject of incoming e-mails. Available as TICKET_NAME_PATTERN_FORMAT in incoming e-mail scripts.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** .*?Ticket\s+\\((\\S+)\\).*
- **Since:** 6.0

mail.cluster.node.id

- **Module:** cmas-esb-mail
- **Description:** Only the node whose *mail.cluster.node.id* equals *cmas.clusternode.id* will start the Mule/ESB e-mail services.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** unspecified
- **Since:** 6.6.5

mail.db.archive

- **Module:** cmas-esb-mail
- **Description:** If property is set to *true*, incoming e-mails are archived in the database.
- **Type:** boolean
- **Restart required:** no

- **System:** yes
- **Optional:** yes
- **Example value:** false (default)
- **Since:** 6.8.5.5


Obsolete! In Mule/ESB mode, no e-mails are saved in the database. E-mails which could not be processed are stored in the file system, see section *E-Mail Backups* in the *ConSol CM Administrator Manual*.

mail.delete.read

- **Module:** cmas-esb-mail
- **Description:** Determines whether CM deletes messages fetched via IMAP(S). Setting value to *true* will cause deletion of messages after fetching. Default is to not delete messages fetched via IMAP(S). Note: Messages fetched via POP3(S) will always be deleted.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** true
- **Since:** 6.7.3

mail.incoming.uri

- **Module:** cmas-esb-mail
- **Description:** URL for incoming e-mails.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** pop3://cm-incoming-user:password@localhost:10110
- **Since:** 6.0

 This value should not be edited here using the system properties pop-up window, but the mailboxes should be configured using the navigation item *E-mail*. Using this standard feature all entries are controlled - i.e., for each mailbox which is added, CM establishes a test connection during mailbox set-up. That way it is not possible to enter wrong values.

mail.max.restarts

- **Module:** cmas-esb-mail
- **Description:** Maximum number of e-mail service restarts before giving up.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 3
- **Since:** 6.0

mail.mime.strict

- **Module:** cmas-esb-mail
- **Description:** If set to *false*, e-mail addresses are not parsed for strict MIME compliance. Default is *true*, which means check for strict MIME compliance.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** false
- **Since:** 6.6.17, 6.7.3

mail.mule.service

- **Module:** cmas-esb-mail
- **Description:** FROM-address for e-mails sent by Mule service
- **Type:** email
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** myuser@consol.de
- **Since:** 6.0

mail.polling.interval

- **Module:** cmas-esb-mail
- **Description:** E-mail polling interval in ms.
- **Type:** integer
- **Restart required:** no

- **System:** yes
- **Optional:** no
- **Example value:** 60000
- **Since:** 6.0

mail.process.error

- **Module:** cmas-esb-mail
- **Description:** TO-address for error e-mails from Mule. As a default the e-mail address of the administrator which you have entered during system set-up is used.
- **Type:** email
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** myuser@consol.de
- **Since:** 6.0

mail.process.retry.attempts

- **Module:** cmas-esb-mail
- **Description:** Number of retries when processing e-mail
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 3
- **Since:** 6.0.2

mail.process.timeout

- **Module:** cmas-esb-mail
- **Description:** E-mail processing timeout in seconds.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 60
- **Since:** 6.1.3

mail.redelivery.retry.count

- **Module:** cmas-esb-mail
- **Description:** Indicates the number of retries of re-delivering an e-mail from the CM system.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 3
- **Since:** 6.1.0

Settings for NIMH

Those settings apply if NIMH is enabled (and therefore Mule/ESB is disabled):

nimh.enabled

- **Module:** cmas-core-server
- **Description:** Enables NIMH service. Must be suffixed with the cluster node ID, e.g., *nimh.enabled.NODEID = true*.
- **Type:** boolean
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** false
- **Since:** 6.9.4.0

filesystem.polling.threads.number

- **Module:** cmas-nimh
- **Description:** Number of threads started for db e-mails' queue polling. Default: 1
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 10
- **Since:** 6.4.0

`filesystem.polling.threads.shutdown.timeout.seconds`

- **Module:** cmas-nimh
- **Description:** Waiting time after the shutdown signal. When the timeout reached, thread will be terminated. Default: 60
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 60
- **Since:** 6.4.0

`filesystem.polling.threads.watchdog.interval.seconds`

- **Module:** cmas-nimh
- **Description:** Watchdog thread interval. Default: 30
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 60
- **Since:** 6.4.0

`filesystem.task.enabled`

- **Module:** cmas-nimh
- **Description:** With this property service thread related to given poller can be disabled. Default: true
- **Type:** boolean
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** true
- **Since:** 6.4.0

`filesystem.task.interval.seconds`

- **Module:** cmas-nimh
- **Description:** Default interval for polling mailboxes. Default: 60 seconds
- **Type:** integer
- **Restart required:** no

- **System:** no
- **Optional:** yes
- **Example value:** 60
- **Since:** 6.4.0

`filesystem.task.polling.folder`

- **Module:** cmas-nimh
- **Description:** Polling folder location which will be scanned for e-mails in the format of eml files. Default: "mail" subdir of cmas data directory
- **Type:** string
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** c://cmas//mail
- **Since:** 6.4.0

`filesystem.task.timeout.seconds`

- **Module:** cmas-nimh
- **Description:** After this time (of inactivity) the service thread is considered as damaged and automatically restarted. Default: 120 seconds
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 60
- **Since:** 6.4.0

`filesystem.task.transaction.timeout.seconds`

- **Module:** cmas-nimh
- **Description:** Default transaction timeout for e-mail fetching transactions. Should be correlated with number of messages fetched at once. Default: 60 seconds
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 60
- **Since:** 6.4.0

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 e-mails.
- **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 e-mails.
- **Type:** string
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** consol
- **Since:** 6.4.0

mailbox.default.connection.port

- **Module:** cmas-nimh
- **Description:** Port for a given mailbox from which the poller reads e-mails.
- **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 e-mails.
- **Type:** string
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** username
- **Since:** 6.4.0

mailbox.default.session.mail.debug

- **Module:** cmas-nimh
- **Description:** Example javax.mail property - allows for more detailed javax.mail session debugging
- **Type:** boolean
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** true
- **Since:** 6.4.0

mailbox.default.session.mail.imap.timeout

- **Module:** cmas-nimh
- **Description:** Example javax.mail property
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 120
- **Since:** 6.4.0

mailbox.default.session.mail.mime.address.strict

- **Module:** cmas-nimh
- **Description:** Example javax.mail property - counterpart of the old *mule mail.mime.strict*, allows to set not so strict e-mail header parsing
- **Type:** boolean
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** true
- **Since:** 6.4.0

mailbox.default.session.mail.pop3.timeout

- **Module:** cmas-nimh
- **Description:** Example javax.mail property.
- **Type:**
- **Restart required:**
- **System:**
- **Optional:**
- **Example value:**
- **Since:** 6.4.0

mailbox.default.session.mail.<protocol>.partialfetch

- **Module:** cmas-nimh
- **Description:** e.g. mailbox.default.session.mail.imaps.partialfetch
- **Type:** boolean
- **Restart required:**

- **System:**
- **Optional:** yes
- **Example value:** false
- **Since:**

`mailbox.default.task.delete.read.messages`

- **Module:** cmas-nimh
- **Description:** This defines whether messages should be removed from the mailbox after processing. For IMAP protocol messages are marked as SEEN by default. For POP3 protocol, when flag is set to true the message is removed, otherwise remains on server and will result in infinite reads. Default: false.
- **Type:** boolean
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** false
- **Since:** 6.4.0

`mailbox.default.task.enabled`

- **Module:** cmas-nimh
- **Description:** With this property service thread related to given poller can be disabled. Default: true
- **Type:** boolean
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** false
- **Since:** 6.4.0

`mailbox.default.task.interval.seconds`

- **Module:** cmas-nimh
- **Description:** Default interval for polling mailboxes. Default: 60 seconds
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes

- **Example value:** 60
- **Since:** 6.4.0

mailbox.default.task.max.message.size

- **Module:** cmas-nimh
- **Description:** Maximum size of e-mail messages (i.e., e-mail plus attachment). E-mails 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 e-mail backups in the Admin Tool (see section *E-Mail 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. Default is set to 10MB: 10485760
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 10485760
- **Since:** 6.4.0

mailbox.default.task.max.messages.per.run

- **Module:** cmas-nimh
- **Description:** Number of messages fetched at once from mailbox. Must be correlated with transaction timeout. Default set to: 20
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 60
- **Since:** 6.4.0

mailbox.default.task.timeout.seconds

- **Module:** cmas-nimh
- **Description:** After this time (of inactivity) the service thread is considered as damaged and automatically restarted. Default: 120 seconds
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes

- **Example value:** 60
- **Since:** 6.4.0

mailbox.default.task.transaction.timeout.seconds

- **Module:** cmas-nimh
- **Description:** Default transaction timeout for e-mail fetching transactions. Should be correlated with number of messages fetched at once. Default: 60 seconds
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 60
- **Since:** 6.4.0

mailbox.polling.threads.mail.log.enabled

- **Module:** cmas-nimh
- **Description:** Enables e-mail logging which is especially crucial in cluster environment (used as semaphore there)
- **Type:** boolean
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** true (default)
- **Since:** 6.9.4.1

mailbox.polling.threads.number

- **Module:** cmas-nimh
- **Description:** Number of threads for accessing mailboxes. Default: 1
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 1
- **Since:** 6.4.0

queue.polling.threads.number

- **Module:** cmas-nimh
- **Description:** Number of threads started for e-mails' queue polling. Default: 1
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 1
- **Since:** 6.4.0

queue.polling.threads.shutdown.timeout.seconds

- **Module:** cmas-nimh
- **Description:** Waiting time after the shutdown signal. When the timeout is reached, the thread will be terminated. Default: 60
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 60
- **Since:** 6.4.0

queue.polling.threads.watchdog.interval.seconds

- **Module:** cmas-nimh
- **Description:** Watchdog thread interval. Default: 30
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 30
- **Since:** 6.4.0

queue.task.error.pause.seconds

- **Module:** cmas-nimh
- **Description:** Maximum number of seconds, the queue poller waits after infrastructure (e.g. database) error. Default 180 seconds
- **Type:** integer
- **Restart required:** no

- **System:** no
- **Optional:** yes
- **Example value:** 180
- **Since:** 6.4.0

queue.task.interval.seconds

- **Module:** cmas-nimh
- **Description:** Main e-mails' queue polling thread interval. Default: 15
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 15
- **Since:** 6.4.0

queue.task.max.retries

- **Module:** cmas-nimh
- **Description:** Maximum number of e-mail processing retries after an exception. When reached, the e-mail is moved to the e-mail archive. This e-mail can be rescheduled again using NIMH API (or the Admin Tool).
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 10
- **Since:** 6.4.0

queue.task.timeout.seconds


- **Module:** cmas-nimh
- **Description:** After this time (of inactivity) the service thread is considered as damaged and automatically restarted. Default: 600 seconds
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 600
- **Since:** 6.4.0

queue.task.transaction.timeout.seconds

- **Module:** cmas-nimh
- **Description:** Transaction timeout for e-mail processing in the pipe. Default: 60
- **Type:** integer
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** 60
- **Since:** 6.4.0


mail.attachments.validation.info.sender

- **Module:** cmas-nimh-extension
- **Description:** Sets FROM-header of attachments type *error notification mail*
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** admin@mail.com
- **Since:** 6.7.5

 This is an equivalent to the old *cmas-esb-mail*, *mail.attachments.validation.info.sender*

mail.attachments.validation.info.subject


- **Module:** cmas-nimh-extension
- **Description:** Sets subject of attachments type error notification mail.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** E-mail was not processed because its attachments were rejected!
- **Since:** 6.7.5

 This is an equivalent to the old *cmas-esb-mail*, *mail.attachments.validation.info.subject*


mail.db.archive

- **Module:** cmas-nimh-extension
- **Description:** If property is set to *true*, incoming e-mails are archived in the database.
- **Type:** boolean
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** false (default)
- **Since:** 6.8.5.5

mail.error.from.address

 This is an equivalent to the old *cmas-esb-mail,mail.mule.service*

mail.error.to.address

 This is an equivalent to the old *cmas-esb-mail, mail.process.error*

mail.on.error

- **Module:** cmas-nimh-extension
- **Description:** If set to *true* an error e-mail is sent to the above configured address in case the e-mail message could not be processed. Default: true
- **Type:** boolean
- **Restart required:** no
- **System:** no
- **Optional:** yes
- **Example value:** false
- **Since:** 6.4.0

mail.process.error

- **Module:** cmas-nimh-extension
- **Description:** TO-address for error e-mails from Mule.
- **Type:** email
- **Restart required:** no
- **System:** yes
- **Optional:** no

- **Example value:** myuser@consol.de
- **Since:** 6.4.0

mail.ticketname.pattern

- **Module:** cmas-nimh-extension
- **Description:**
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** .*?Ticket\s+\((\S+)\).*
- **Since:** 6.4.0

Mapping of Former Mule and New NIMH Properties

Mule property	NIMH property
cmas-esb-mail, mail.delete.read	cmas-nimh, mailbox.default.task.delete.read.messages
cmas-esb-mail, mail.polling.interval	cmas-nimh, mailbox.default.task.interval.seconds
cmas-esb-mail, mail.process.retry.attempts	cmas-nimh, queue.task.max.retries
cmas-esb-mail, mail.mime.strict	cmas-nimh, mailbox.default.session.mail.mime.address.strict
cmas-esb-mail, mail.encryption	cmas-core-server, mail.encryption (moved to core server properties)
cmas-esb-mail, mail.callname.pattern	cmas-nimh-extension, mail.ticketname.pattern
cmas-esb-mail, mail.attachments.validation.info.sender	cmas-nimh-extension, mail.attachments.validation.info.sender
cmas-esb-mail, mail.attachments.validation.info.subject	cmas-nimh-extension, mail.attachments.validation.info.subject
(cmas-esb-mail, mail.db.archive)	cmas-nimh-extension, mail.db.archive
cmas-esb-mail, mail.mule.service	cmas-nimh-extension, mail.error.from.address

Mule property	NIMH property
cmas-esb-mail, mail.process.error	cmas-nimh-extension, mail.error.to.address

Attachments for Incoming E-Mails

These settings apply to Mule/ESB and NIMH.

attachment.allowed.types

- **Module:** cmas-core-server
- **Description:** Comma-separated list of allowed filename extensions (if no value defined, all file extensions are allowed).
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** txt,zip,doc
- **Since:** 6.5.0

attachment.max.size

- **Module:** cmas-core-server
- **Description:** Maximum attachment size, in MB. This is a validation property of the CM API. It controls the size of attachments at tickets, at units, and at resources. It also controls the size of incoming (not outgoing!) e-mail attachments in NIMH as well as in Mule/ESB mode.
- **Type:** integer
- **Restart required:** no
- **System:** yes
- **Optional:** no
- **Example value:** 100
- **Since:** 6.4.0

E-Mail Encryption (Outgoing and Incoming)

These settings only apply if e-mail encryption is active (true). This is valid for Mule/ESB Mail and NIMH.

mail.encryption

- **Module:** cmas-core-server
- **Description:** If property is set to *true*, the encrypt checkbox in the Ticket E-Mail Editor is checked by default.
- **Type:** boolean
- **Restart required:** no

- **System:** yes
- **Optional:** no
- **Example value:** true (default = false)
- **Since:** 6.8.4.0

In case certificates are stored in an LDAP directory, the following settings have to be made:

`ldap.certificate.basedn`

- **Module:** cmas-core-server
- **Description:** Base DN for certificates location in the LDAP tree. If not provided, *cmas-core-security*, *ldap.basedn* is used.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** ou=accounts,dc=consol,dc=de
- **Since:** 6.8.4

`ldap.certificate.content.attribute`

- **Module:** cmas-core-server
- **Description:** LDAP attribute name used where certificate data is stored in the LDAP tree. Default value: *usercertificate*
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** *usercertificate*
- **Since:** 6.8.4

`ldap.certificate.password`

- **Module:** cmas-core-server
- **Description:** LDAP Certificates manager password. If not set, *cmas-core-security*, *ldap.-password* is used.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Since:** 6.8.4

ldap.certificate.providerurl

- **Module:** cmas-core-server
- **Description:** LDAP Certificates provider URL. If not set, *cmas-core-security*, *ldap.providerurl* is used.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** ldap://ldap.consol.de:389
- **Since:** 6.8.4

ldap.certificate.searchattr

- **Module:** cmas-core-server
- **Description:** LDAP attribute name used to search for certificate in the LDAP tree. Default value: mail
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Example value:** mail
- **Since:** 6.8.4

ldap.certificate.userdn

- **Module:** cmas-core-server
- **Description:** LDAP Certificates manager DN. If not set, *cmas-core-security*, *ldap.userdn* is used.
- **Type:** string
- **Restart required:** no
- **System:** yes
- **Optional:** yes
- **Since:** 6.8.4

G.2.3.5 Activity Interval Configuration

admin.tool.session.check.interval

- **Module:** cmas-app-admin-tool
- **Description:** Admin Tool inactive (ended) sessions check time interval (in seconds)
- **Type:** integer

- **Restart required:** yes
- **System:** yes
- **Optional:** no
- **Example value:** 30
- **Since:** 6.7.5

server.session.timeout

- **Module:** cmas-core-server
- **Description:** Server session timeout (in seconds) for connected clients. Each client can over-write 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 E-Mail Addresses

ConSol CM can use different administrator e-mail addresses, depending on the subsystem. Please see [Administrator and Notification E-Mail Addresses](#) for detailed explanations concerning admin e-mail addresses. If no specific admin e-mail addresses are configured, the global admin e-mail address (that you have defined during system set-up) is used.

G.3 Administrator and Notification E-Mail Addresses

This chapter discusses the following:

G.3.1 Introduction	502
G.3.2 Default Configuration	502
G.3.3 Subsystem-Specific Notification E-Mail Addresses	503

G.3.1 Introduction

In ConSol CM, several administrator e-mail addresses (or notification e-mail addresses respectively) can be configured. Here, an overview of all those addresses is provided.

G.3.2 Default Configuration

When you set-up a ConSol CM system, you have to enter one global admin e-mail address.

The screenshot shows the 'CM6 Setup' web interface with the 'Administrator' tab selected. The 'Administrator' section contains the following fields and options:

- Login:** admin (highlighted with a red box and labeled 'Login for admin user')
- Password:** (masked with dots, highlighted with a red box)
- Confirm password:** (masked with dots, highlighted with a red box)
- E-mail:** admin@localhost (highlighted with a red box and labeled 'Global E-mail address of CM administrator')
- Select authentication mode:** Internal (dropdown menu)
- Kerberos v5 authentication:** ☐ (checkbox)

Navigation buttons 'Previous' and 'Next' are at the bottom.

Figure 176: Admin e-mail address configuration during system set-up

This global e-mail address (system property *cmas-core-security, admin.email*) will be used for all notifications and will be entered automatically for all subsystem-specific e-mail addresses. This means that this admin e-mail address will initially be set automatically for all system properties which contain admin or notification e-mail addresses. The properties can then be changed using the Admin Tool GUI to configure the specific subsystem, e.g., you can configure a specific notification address for DWH operations in the *DWH Configuration* section of the Admin Tool.

By configuring different admin/notification e-mail addresses for different subsystems, you can spread responsibilities according to responsibilities and roles within your company. For some notifications, even the FROM address, text, and the subject can be configured.

G.3.3 Subsystem-Specific Notification E-Mail Addresses

G.3.3.1 DWH (Data Warehouse) - Specific Notification E-Mail Addresses and E-Mail Configurations

System Properties

- **Error**

- **cmas-dwh-server, notification.error.to**
An e-mail will be sent to this address when a DWH operation has failed. If the property is not set, no e-mail will be sent.
- **cmas-dwh-server, notification.error.from**
An e-mail will be sent with this FROM address when a DWH operation has failed.
- **cmas-dwh-server, notification.error.subject**
Subject for error e-mails from the DWH
- **cmas-dwh-server, notification.error.description**
Text for error e-mails from the DWH.

- **Successful**

- **cmas-dwh-server, notification.finished_successfully.to**
An e-mail will be sent to this address when a DWH transfer has been completed successfully, e.g., when the transfer has been completed without errors. If the property is not set, no e-mail will be sent.
- **cmas-dwh-server, notification.finished_successfully.from**
FROM address for e-mails from the DWH when a transfer finishes successfully.
- **cmas-dwh-server, notification.finished_successfully.subject**
Subject for e-mails from the DWH when a transfer finishes successfully.
- **cmas-dwh-server, notification.finished_successfully.description**
Text for e-mails from the DWH when a transfer finishes successfully.

- **Unsuccessful**

- **cmas-dwh-server, notification.finished_unsuccessfully.to**
An e-mail will be sent to this address when a DWH transfer has been completed, but not successfully, e.g., when the transfer has been completed with errors. If the property is not set, no e-mail will be sent.
- **cmas-dwh-server, notification.finished_unsuccessfully.from**
FROM address for e-mails from the DWH when a transfer finishes unsuccessfully.
- **cmas-dwh-server, notification.finished_unsuccessfully.subject**
Subject for e-mails from the DWH when a transfer finishes unsuccessfully.
- **cmas-dwh-server, notification.finished_unsuccessfully.description**
Text for e-mails from the DWH when a transfer finishes unsuccessfully.
- **cmas-dwh-server, cmas-dwh-server, notification.error.description**
The text for error e-mails from the DWH

For an overview of all system properties which can be set for DWH notifications, please refer to section [CMRF & DWH Configuration](#) of the [List of System Properties by Area](#) chapter. All properties which are relevant in this context start with *notification*.

Graphical Configuration

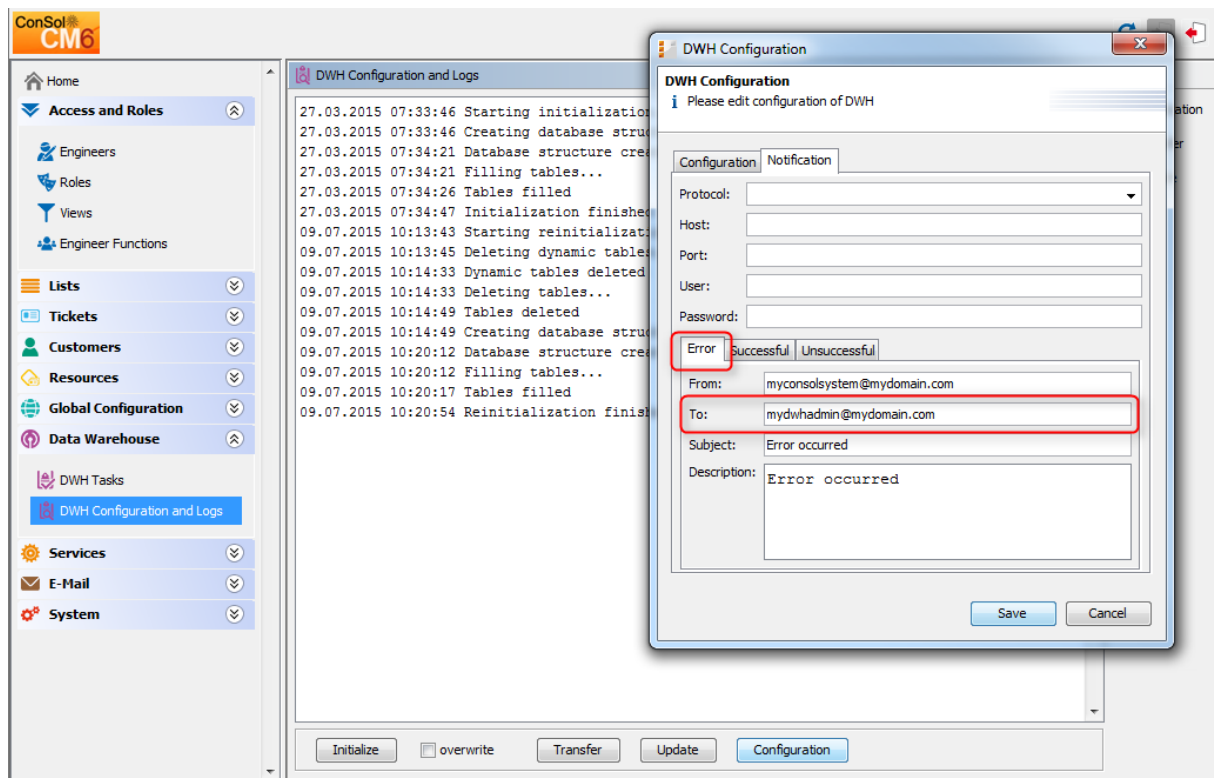


Figure 177: ConSol CM Admin Tool - Notification address for DWH errors

The e-mail addresses are checked when you click **Save**. If an e-mail address is not valid, a message is displayed. No mails will be sent to this address.

G.3.3.2 E-Mail - Specific Notification E-Mail Addresses

System Properties (NIMH Mode)

- **cmas-nimh-extension, mail.error.to.address**
An error e-mail is sent to the address in case an e-mail message could not be processed. Starting with CM version 6.10.5.1, an e-mail to this address is also sent when a mail timeout occurs. If the value for this property is not set, no e-mail will be sent and an exception will be written into the log file.
- **cmas-nimh-extension, mail.attachments.validation.info.sender**
Sets the FROM header of attachments type *error notification e-mail*.

Please note that the sending is controlled by the system property *cmas-nimh-extension, mail.on.error*. Only if this property is set to *true*, an error e-mail is sent to the above configured address in case an e-mail message could not be processed.

System Properties (Mule/ESB Mode)

- **cmas-esb-mail, mail.process.error**
TO address for error e-mails from Mule/ESB Mail Service. An error e-mail is sent to the address in case an e-mail message could not be processed, there are state problems, or any other problem with the Mule/ESB Mail Service. If the property is not set, the sending of the e-mail will fail and this fact will be logged.
- **cmas-esb-mail, mail.mule.service**
FROM address for e-mails sent by Mule/ESB Mail Service.

Graphical Configuration

The value which is entered here (*Error e-mail address*) in the graphical user interface is set for NIMH as well as for Mule/ESB.

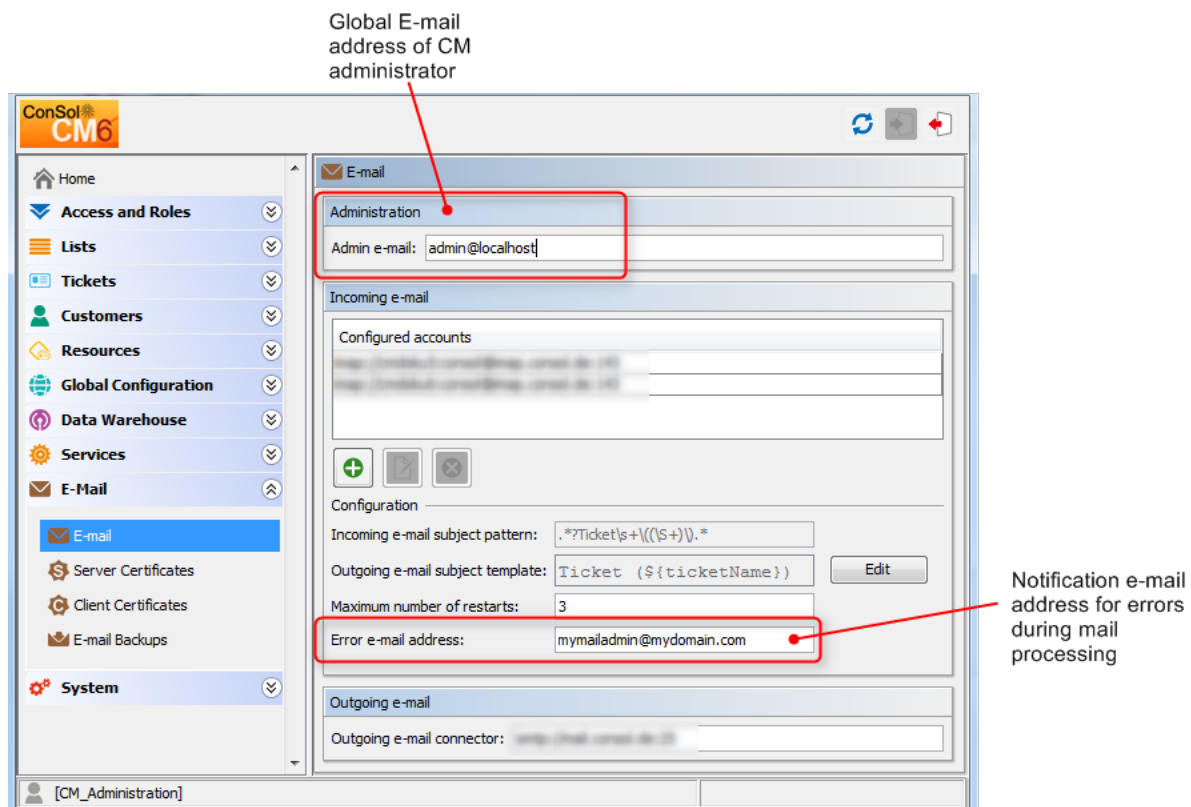



Figure 178: ConSol CM Admin Tool - Configuration of global admin e-mail address and of notification e-mail address for e-mail processing problems

 Do not mix up the two e-mail addresses which can be configured on the tab of the navigation item *E-mail*:

- **Admin e-mail**
is the global administrator e-mail (which has been set during system set-up).
- **Error e-mail address**
is the address of an e-mail administrator or another person who should receive the error messages if the processing of e-mails (incoming or outgoing) does not work correctly.

G.3.3.3 Workflow Engine - Specific Notification E-Mail Addresses

- **cmas-workflow-engine, jobExecutor.adminMail**
E-mail address which will get notified about job execution problems (when retry counter is exceeded). If the property is not set, no e-mail will be sent.
- **cmas-workflow-engine, jobExecutor.mailFrom**
E-mail address which will be set as *FROM* header during admin notifications.

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	28
Code example 2: Precondition script: activity should only be displayed for VIP customers	70
Code example 3: Script for automatic activity where receipt note is sent, variant 1	73
Code example 4: Script for automatic activity where receipt note is sent, variant 2	73
Code example 5: Script for assigning ticket to current engineer	74
Code example 6: Working with customer data. Example: using a Data Object Group Field of type boolean	80
Code example 7: Example for a script on timer start	93
Code example 8: Calculate and set time for TimerTrigger using BusinessCalendar	93
Code example 9: Code of decision node script	112
Code example 10: Code of automatic activity script Re-calculate priority	114
Code example 11: Process Designer: Initializing script for Create bid ACF	123
Code example 12: Using the ticket object	145
Code example 13: Using workflowApi to send an e-mail (older variant. Current variant would mean using an object of class Mail)	145
Code example 14: Using workflowApi to assign a ticket to current engineer	145
Code example 15: Using workflowApi to deactivate a trigger	146
Code example 16: Using workflowApi to display a GUI message for the engineer/user	146
Code example 17: Example for a script on timer start	146
Code example 18: Using the ConfigurationService to retrieve the number of the engineer management ticket	147
Code example 19: Using the ConfigurationService to retrieve base URL of the system	147
Code example 20: Use of EngineerService	148
Code example 21: Using EnumService to retrieve an enum value by name	148
Code example 22: Using TicketService to find ticket of a view	149
Code example 23: Using the EngineerRoleRelationService to send an e-mail to all engineers of a role	149
Code example 24: Admin Tool script used to display Custom Fields	159
Code example 25: Precondition script where boolean value is checked	160
Code example 26: Precondition script where boolean value is checked, short version	160
Code example 27: Retrieving an enum value for a CF	160
Code example 28: Code (Workflow or Admin Tool script) for displaying MLA data	162
Code example 29: Retrieving the entire path to the selected MLA value	163

Code example 30: Displaying the content of a list of date objects	165
Code example 31: Retrieve a certain value from a list of simple data types	165
Code example 32: Retrieve data from a list of structs	167
Code example 33: Set a CF value for a Date CF	167
Code example 34: Calculate with value of date CF	167
Code example 35: Setting a CF value to null	168
Code example 36: Setting a CF value to null via removing the value	168
Code example 37: Setting an enum value	168
Code example 38: Adding a new line in a list of strings	168
Code example 39: Setting a value in a list of strings	168
Code example 40: Adding a new line in a list of structs	169
Code example 41: Fade-in a CF group	169
Code example 42: Fade-out CF groups	169
Code example 43: Retrieving a field value for a company	171
Code example 44: Admin Tool script (called from workflow) for displaying customer data	173
Code example 45: Retrieving a value from a list of structs using index notation	174
Code example 46: Set and add values for a Data Object Group Field of type integer	175
Code example 47: Creating a new list of structs, version 2	175
Code example 48: Adding a new line in a list of structs for company data	175
Code example 49: Setting a value in a list of structs using index notation	175
Code example 50: Removing a struct (= line) from a list of structs (= table)	175
Code example 51: Convenience methods for access to customer data	176
Code example 52: Simple resource action script which displays information about the resource in the log file	180
Code example 53: Sending an automatic acknowledgment of receipt to the customer who has opened a ticket, using a Mail object	184
Code example 54: Sending an e-Mail to the engineer when a certain escalation level has been reached, using a Mail object	185
Code example 55: Sending an e-mail to a customer integrating the queue-specific mail script, using a Mail object	186
Code example 56: Sending an E-Mail to All Contacts of the Ticket	187
Code example 57: Admin Tool script for sending a notification of receipt and inserting the email as ticket history entry	190
Code example 58: Workflow or Admin Tool script used to send an e-mail to an engineer, using the representation feature by using the setTargetEngineer() method	194

Code example 59: Deactivate a time trigger	197
Code example 60: Re-initialize time trigger	197
Code example 61: Setting time for a time trigger	200
Code example 62: Script for time trigger for escalation 4 hours before deadline	201
Code example 63: Creating a ticket relation of type REFERENCE using workflowAPI	205
Code example 64: Creating a ticket relation of type MASTER_SLAVE using workflowAPI	206
Code example 65: Version A: Finding all target tickets (here: all slave tickets)	206
Code example 66: Version B: Finding all target tickets (here: all slave tickets)	206
Code example 67: Finding all slave tickets of the current ticket	206
Code example 68: Finding the master ticket of the current ticket	207
Code example 69: Creating a child ticket	208
Code example 70: Finding the parent ticket of a ticket	208
Code example 71: Finding all child tickets of a ticket	208
Code example 72: Finding all brother tickets of a (child) ticket	209
Code example 73: Example Script, display IDs and names of slave tickets workflow version	211
Code example 74: Example Script, display IDs and names of slave tickets Admin Tool script version	212
Code example 75: Calling previous AT script from workflow activity	212
Code example 76: Adding a data object relation using a workflow script	217
Code example 77: Calculate ticket deadline from SLA. SLA as resource which is linked to the ticket.	222
Code example 78: Workflow script to check if a solution has been defined in the ticket	226
Code example 79: New FAQ ticket with solution text	228
Code example 80: Excerpt from the postActivityExecutionScript	229
Code example 81: Admin Tool script called from a workflow activity: creating a child ticket and handing over the products list and all ticket attachments	235
Code example 82: Alternative (additional) solution: handing over only the important attachments	236
Code example 83: Search for tickets (pseudocode)	240
Code example 84: Find tickets with the same module as the current ticket	240
Code example 85: Search for tickets by unit	241
Code example 86: Script of activity "New IT ticket (Accept ticket)"	242
Code example 87: Search for contacts by first name and last name	244
Code example 88: Search for units by enum value (general syntax)	245
Code example 89: Search for units by enum value (example)	245
Code example 90: Print an IT asset (resource) list into the ticket	248
Code example 91: Debug entry in ConSol CM standard e-mail script	251

Code example 92: Calling an Admin Tool script from the workflow (only way in CM versions 6.10.4 and older, in CM 6.10.5 and up still available)	262
Code example 93: Calling an Admin Tool script from the workflow with use of parameters (only way in CM versions 6.10.4 and older, in CM 6.10.5 and up still available)	262
Code example 94: Calling an Admin Tool script from the workflow (only CM versions 6.10.5 and up)	262
Code example 95: Code which triggers TicketUpdateEvent	266
Code example 96: Code which does not trigger TicketUpdateEvent	266

G.5 Trademarks

- The Apache Commons Codec™ library is a trademark of the Apache Software Foundation. See [Apache Commons Codec web page](#).
- Apache OpenOffice™ – Apache and the Apache feather logos are trademarks of The Apache Software Foundation. [OpenOffice.org](#) and the seagull logo are registered trademarks of The Apache Software Foundation. See [Apache OpenOffice Trademarks web page](#).
- Google Maps™ – Google Maps is a trademark of Google Inc. See [Google trademark web page](#) for details.
- Microsoft® – Microsoft and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. See [Microsoft trademark web page](#).
- Microsoft® Active Directory® – Microsoft and Microsoft Active Directory are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. See [Microsoft trademark web page](#).
- Microsoft® Exchange Server – Microsoft and Microsoft Exchange Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. See [Microsoft trademark web page](#).
- Microsoft® Office – Microsoft and Microsoft Office are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. See [Microsoft trademark web page](#).
- Windows® operating system – Microsoft and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. See [Microsoft trademark web page](#).
- Microsoft® SQL Server® – Microsoft and Microsoft SQL Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. See [Microsoft trademark web page](#).
- Microsoft® Word® – Microsoft and Microsoft Word are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. See [Microsoft trademark web page](#).
- MuleSoft™ and Mule ESB™ are among the trademarks of MuleSoft, Inc. See [Mule Soft web page](#).
- Oracle® – Oracle is a registered trademark of Oracle Corporation and/or its affiliates. See [Oracle trademarks web page](#).
- Oracle® WebLogic – Oracle is a registered trademark of Oracle Corporation and/or its affiliates. See [Oracle trademarks web page](#).
- Pentaho® – Pentaho and the Pentaho logo are registered trademarks of Pentaho Inc. See [Pentaho trademark web page](#).

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, e-mail, 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.

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 Custom Fields, Data Object Group Fields and resource fields.

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 pre-filled with ConSol CM ticket or customer parameters.

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.

CTI

Computer Telephony Integration - a denomination for any technology that facilitates interaction between a telephone and a computer.

Custom Field

A field where ticket data can be stored.

Custom Field Group

A group of Custom Fields where ticket data can be stored.

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 group

The customer group determines which customer data model is used for its customers and which actions are available.

D

Data Object

A customer (a contact or a company). Formerly Unit.

Data Object Group

A group of fields where data for customers (contacts or companies) can be stored. Similar to Custom Field Group for ticket data.

Data Object Group Field

A field where data for customers (contacts or companies) can be stored.

Similar to Custom Field for ticket data.

group, a specific customer data model can be defined.

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.

G

GUI

Graphical User Interface

H

history

The history contains all changes which were carried out for the ticket, customer, or resource.

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.

I

IMAP

Internet Message Access Protocol - Internet standard protocol to access e-mail on a remote e-mail server. Can be used as plain IMAP or as secure IMAP (IMAPs). In the latter case, proper certificates are required.

ESB

Enterprise Service Bus - a software architecture used for communication between mutually interacting software applications in a service-oriented architecture (SOA).

J

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.

Java EE

Java Enterprise Edition

F

FlexCDM

Flexible Customer Data Model - the customer data model introduced in ConSol CM in version 6.9. For each customer

JMS

Java Message Service - Java EE component used to send messages between JMS clients.

JRE

Java Runtime Environment. Provides a Java Virtual Machine for Clients.

K

KPI

Key Performance Indicator - parameter used for performance measurement for companies, projects, etc.

L

LDAP

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 e-mail messages are delivered. Mailboxes are managed on an e-mail server. ConSol CM can access one or more mailboxes to retrieve e-mails.

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 e-mails, new in version 6.9.4.

P

PCDS

Page Customization Definition Section

Pentaho

Pentaho™ 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 e-mails 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 Custom 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 e-mails.

T

TAPI

Telephony Application Programming Interface - a Microsoft Windows API which provides computer/telephony integration and enables PCs running Microsoft Windows to use telephone services.

TEF

Task Execution Framework - a ConSol CM module which can execute tasks asynchronously. A new feature as of version 6.9.4.

template

Templates contain predefined and pre-formatted text. They can be used for comments, e-mails, 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.

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.

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.