ConSol Consulting & Solutions

D O C U M E N T

Release Notes ConSol*CM

Version 6.9.1

Author: Michael Siebenborn
Phone: +49 (0) 89 / 45841-100
Mail: michael.siebenborn@consol.de
Version: 1.3
Date: 10.01.2014
Status: Released

Table of contents

General update and installation instructions

For an update of ConSol*CM from one version to another two possible ways exist:

- Distribution installation
  The distribution is installed into the application server. For an update every local configuration, like the data source configuration, has to be saved before and reconfigured afterwards.
  This type of update ensures that really every change between the versions is installed.
  This type of update is recommended for updates of the major or minor version, e.g. for an update from 6.6.3 to 6.7.5.
- EAR / WAR Update
  For this type of update of the ConSol*CM, the EAR (cm6.ear, cmrf.ear) and WAR (cm-track.war) files of the new version have to be installed into the application server.
  Additionally every installation related changes described in the chapters ‘Update and installation instructions’ have to be applied manually. The changes have to be applied for every version between your original CM version and the new CM version, e.g. for an update from 6.6.3 to 6.6.7 the instructions of the versions 6.6.5, 6.6.6 and 6.6.7 have to be checked.
  This type of update is only recommended for updates within a minor version.

Additionally for every type of update, the ‘Update and installations instructions’ chapter has to be checked for further important notes.

If available, the solution specific ReleaseNotes have to be checked too.

1 Version 6.9.1.0 (12.11.2013)

Version 6.9.1.0 includes 6.8 versions up to 6.8.5.5 and 6.7 versions up to 6.7.13

1.1 Update and installation instructions

1.1.1 New License is needed

To use the ConSol*CM 6.9 a valid license for version 6.9 or higher is needed. Please check the license in the CM6 Admin Tool (*General configuration*, tab *Licence*).

1.1.2 Database schema update procedure to 6.9 may take a longer time (#623344)

Because of database schema changes the procedure when updating to 6.9 may take a longer time. The duration depends on the number of rows of the table `CMAS_CNT_ENTRY` and the size of the ticket history entries that are stored within this table.
The following are update times measured for test installations that use different databases:

<u>Oracle 11g:</u>
`CMAS_CNT_ENTRY`: ~5.4 million rows with ~1 TB of content data (entries with pictures)
Update time: ~45 min

<u>SQLServer 2008:</u>
`CMAS_CNT_ENTRY`: ~5 million rows with text entries of small size (~10 characters)
Update time: 7 min 30 s

<u>MySQL 5.6:</u>
`CMAS_CNT_ENTRY`: ~2.7 million rows with text entries of medium size (~4000 characters)
Update time: 5 h 15 min

`CMAS_CNT_ENTRY`: ~1 million rows with text entries of small size (~10 characters)
Update time: 7 min

1.1.3 DWH database schema must be modified (#622829)

Due to changes in 6.9.0.0 and 6.9.1.0 it is necessary to modify the database schema of the DWH database.
Update scripts have been provided which can be executed via JMX bean on the server where CMRF is deployed. Please request those update scripts by contacting the ConSol*CM Support Team.
The upgrade steps depend on the DWH mode that is currently in use.

- For DWH Admin mode:
  Please execute the update scripts of 6.9.0.0 and 6.9.1.0 after the upgrade to the version 6.9.1.0 **but before** the next start of a DWH transfer/update.
  The scripts can be executed directly on the DWH database or via JMX console.

- For DWH Live mode:
  1. Stop the Live mode by switching it to *OFF* in the CM6 Admin Tool.
  2. Before the start of the CM6 upgrade to 6.9.1.0 please ensure that that CMRF has finished all pending jobs by checking the following things:
     a. That there are no messages in JMS queues like *transfer, live, log, control*

   b. That there are no messages in buffered tables like *int_live_buffer, int_transfer_buffer, cmas_dwh_ser_sync_object*
3. Make the CM6 upgrade to 6.9.1.0.
4. Execute the update scripts of 6.9.0.0 and 6.9.1.0. The scripts can be executed directly on the DWH database or via JMX console.
5. Start the Live mode by switching it to *LIVE* in the CM6 Admin Tool.

The execution of the update scripts can be done via JMX console:
1. Open the JMX console (*http://<host>:<port>/jmx-console*).
2. In the left frame *Object Name Filter* select *consol.cmrf*.
3. On the right side select *name=cmrf.sqlUpdater,topic=cmrf.update,type=update*.

Please follow the instructions on this page.

1.1.4 Deprecated methods were removed (#622686)

Methods that were declared as deprecated in the releases 6.7 or in older releases were removed for 6.9.0.0.

IMPORTANT: Before updating to 6.9 you must ensure that you are not using such deprecated methods in your scripts (Workflow scripts, CM6 Admin Tool scripts). Otherwise these scripts will stop working after the update to 6.9.

This is the list of removed or changed methods:

`AbstractField` (removed custom value accessor for each custom field type)

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

`ActivityFormFieldsSet` (removed accessors with plain `FieldDefinition`, use `ActivityFormElement` instead)

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

`ContentFile` (added size parameter to input stream methods)

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

`ContentResource` (same changes as in `ContentFile`)

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

`FieldLogEntry` (removed modification accessors)

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

`Ticket` (removed renamed custom fields accessors + other changes)

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

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

`TimerTrigger` (removed `setDuedate` method)

Removed / changed method	Replacement
`TimerTrigger.setDuedate`	`TimerTrigger.setDueTime`

`Unit` (removed renamed custom fields accessors)

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

1.1.5 MySQL 5.6 support (#622820)

ConSol*CM6 now supports MySQL 5.6 (5.6.13 or newer). Therefore MySQL 5.1 is no longer supported. If you want to update to version 6.9 or higher you must also upgrade your MySQL installation to the supported version 5.6.
Please note that there might be an issue related to stuck threads in combination with a WebLogic application server. It can be fixed by changing the default configuration of the MySQL DB.
Please set:

```
query_cache_size = 0
```

in */etc/mysql/my.cnf* or execute the following commands:

```
SET GLOBAL query_cache_type = OFF;
SET GLOBAL query_cache_size = 0;
```

1.1.6 MySQL driver 5.1.25 support (#617775)

The MySQL driver 5.1.25 is now supported. Please use this driver when updating to the release 6.9.

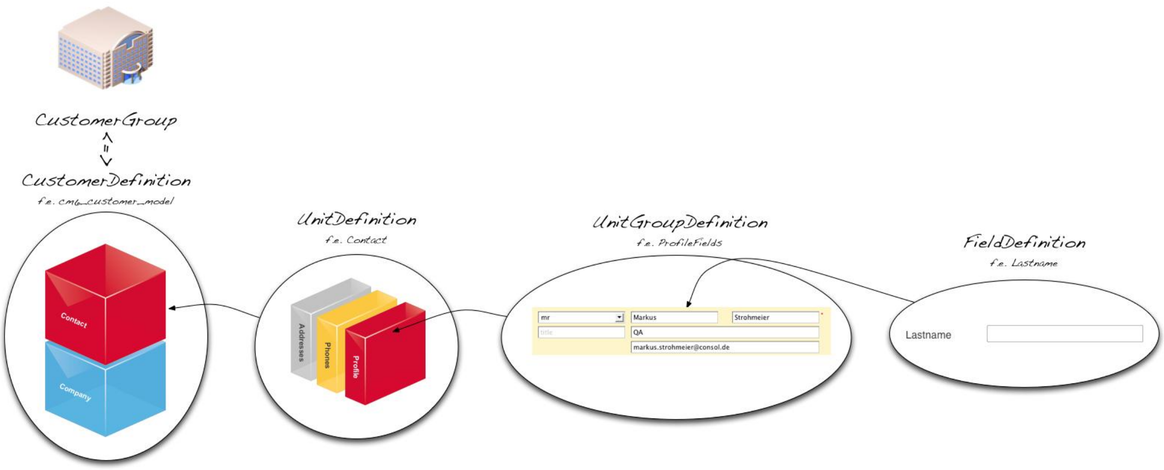
1.1.7 MS Active Directory 2012 support (#620562)

The SSO functionality of ConSol*CM6 has been checked and confirmed to work against MS Active Directory 2012.

1.2 New Features

1.2.1 Flexible Customer Data Management

The feature makes it possible to have several different customer models (i.e. contacts having different sets of custom fields) coexisting in one CM6 system.
Idea of the new structure:

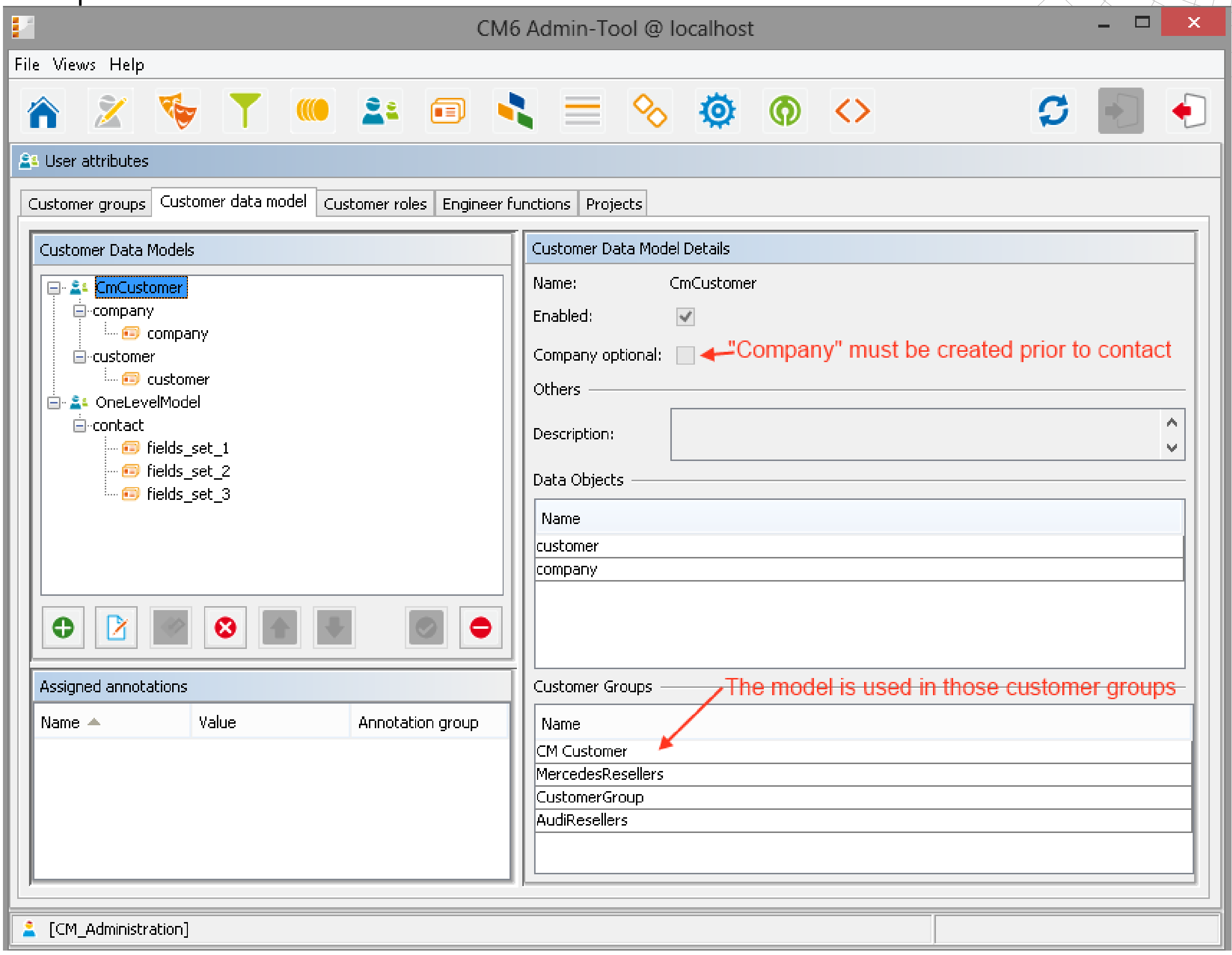


Each customer group now contains a customer definition (the same customer definition might be used in many customer groups), which may have one or two levels (i.e. contact and company) and each level (called data object) contains many data object groups – sets of custom fields.
In order to implement this feature some general modifications had to be applied to ConSol*CM6:

CM6 Admin Tool changes (#621984)

The *User Attributes* view and the *Customer Data Model* tab is now used to configure the customer definitions instead of the *Custom Fields Administration* view.

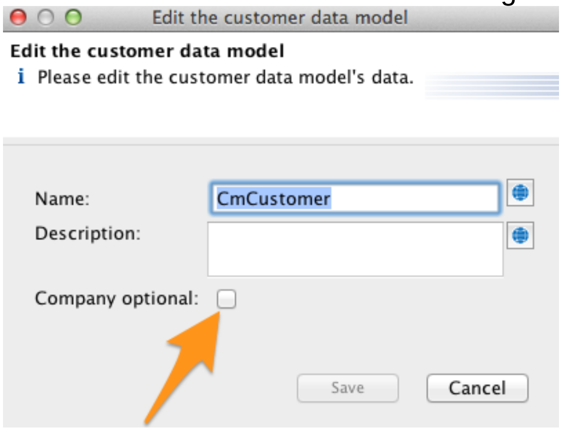
Example:



In the above picture it can be seen that there are two customer models defined in the system. One having two levels: *company* and *customer*, each having one group of custom fields having the same name. The other has only one level *contact* that in turn contains three groups of fields. To use the *customer* level of the first model one needs to create a company data object for it – this is indicated by the unchecked *Company optional* option. The view also shows which customer groups are using the selected model.

It is possible to define customer models that consist of two levels (i.e. *Contact* and *Company*). By default this second level is mandatory for such models.
A checkbox was added that makes it possible to define that the second level is optional so such a customer model may contain customers that consist only of a contact and customers that are contacts assigned to companies (#622975).

This checkbox is available when editing the customer data model:

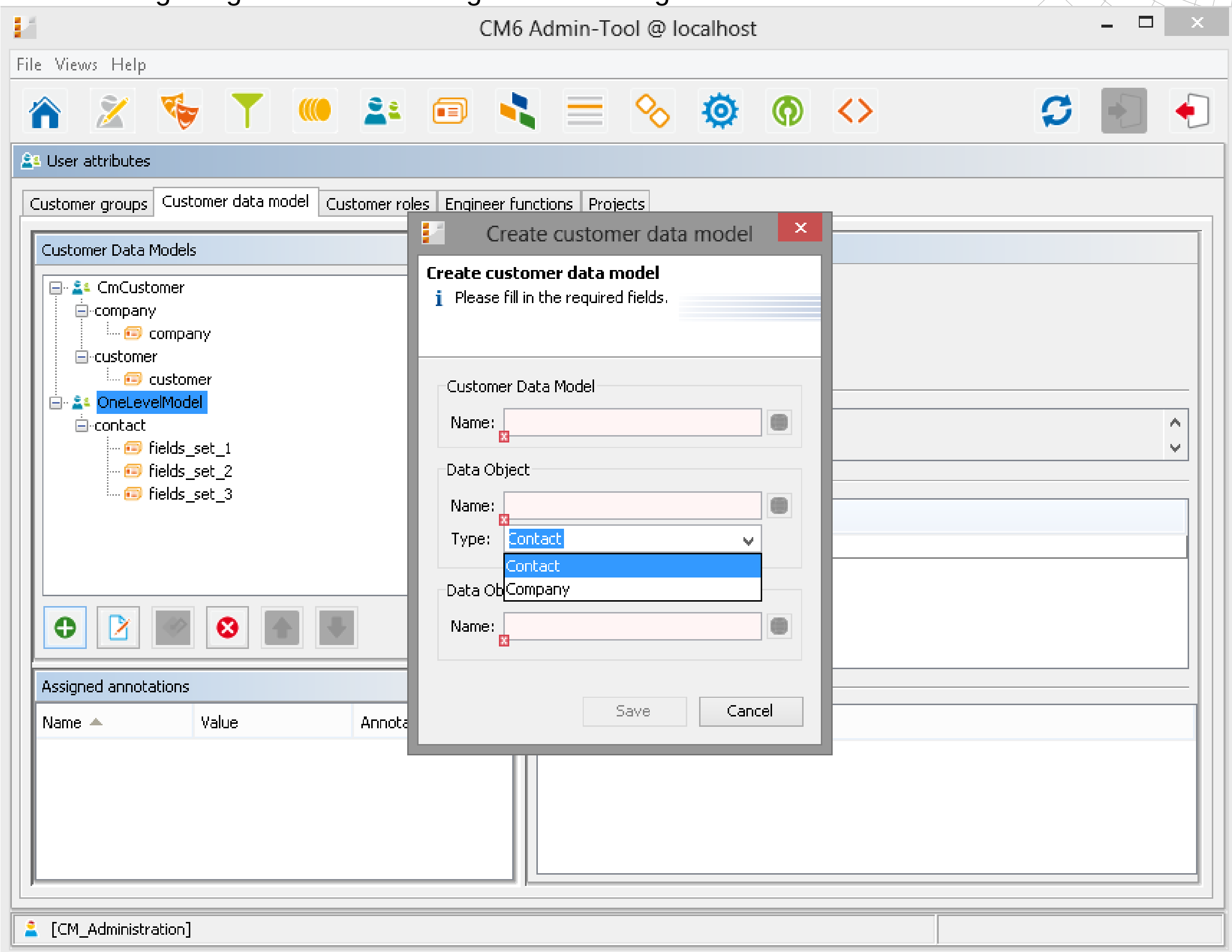


When there are several *data object groups* they can be shown in form of tabs in the CM6 Web Client by setting the annotation *shown-in-group-section*.

Adding new elements to the models depends on the currently selected tree node:

- When the most upper node is selected a new model can be added.
- When the node below (in the middle) is selected *company* or *contact* data objects can be created
- When the lowest node is selected then a new *data object group* (set of custom fields) can be created.

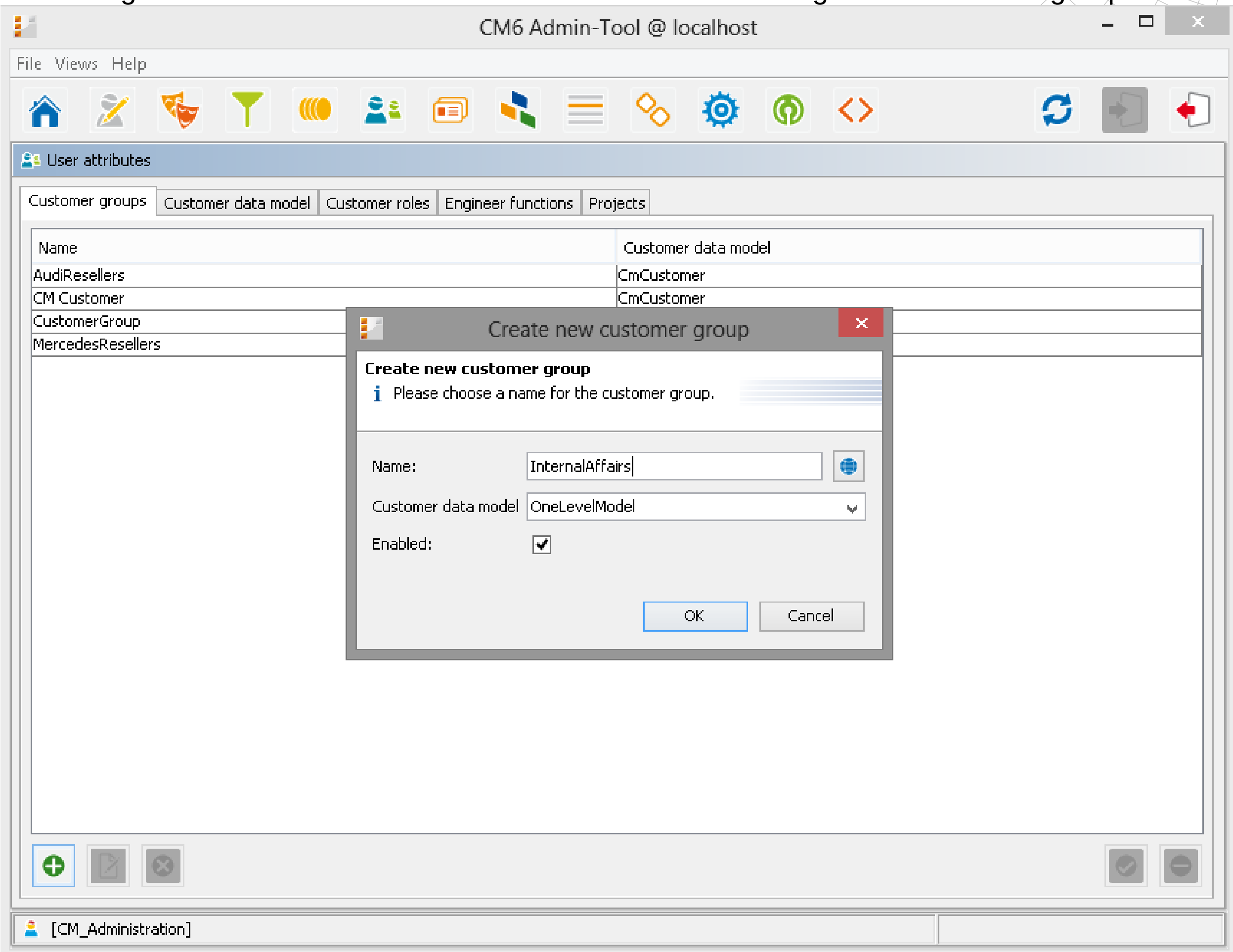
The following image shows the dialog when creating a new customer data model:



Using this dialog the administrator also creates the first *data object* (*contact* or *company* level) and the first *data object group* for it. When in the next steps the administrator decides to create the second *data object* the reference between those objects is created automatically (contact data reference field for the *contact* data object) – formerly this field had to be configured manually to bind the company and contact levels together. Also the reference field is now no longer shown on the custom fields list since it is a field that is only used internally by the system.

Another change is that now the annotation *unit is a contact* is obsolete and was removed from the system.

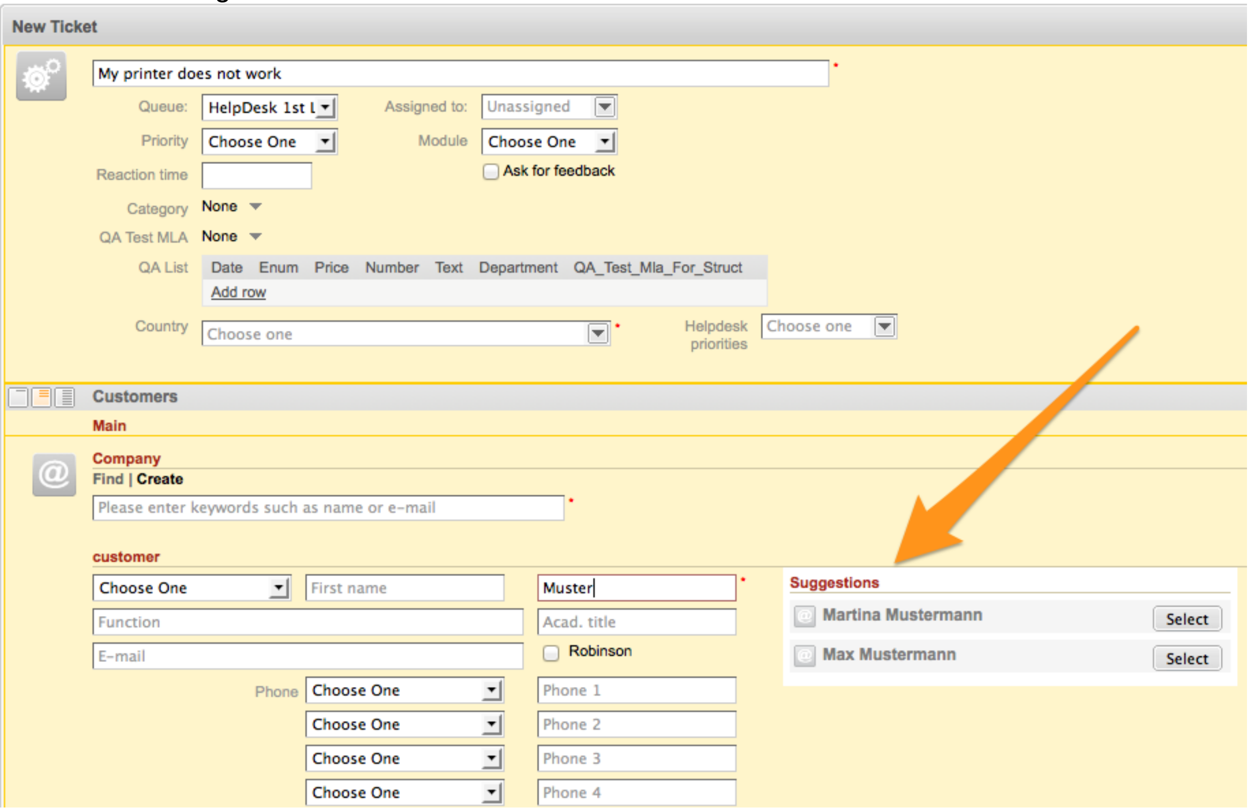
The configured customer model can then be used when creating a new customer group:



Which then in turn is processed as usual (bound to a queue, made available via role permissions).

CM6 Web Client changes

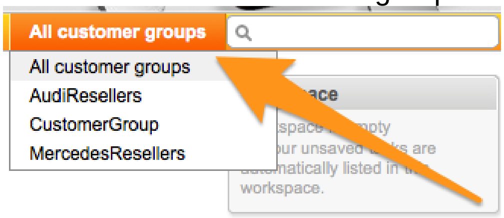
When having access to only one customer group the engineer gets the same user interface as in the previous CM versions except that on the *Customer* or *Create ticket* pages a new mechanism for reducing doublets is used:
The engineer gets a form to create or search for a contact or company. When typing into specific fields search queries are executed presenting possible matches as a so called *proactive duplicate prevention* – so when a result is found the engineer can see that there is already a data object with this information and she does not have to create a new data object but can select the existing one:



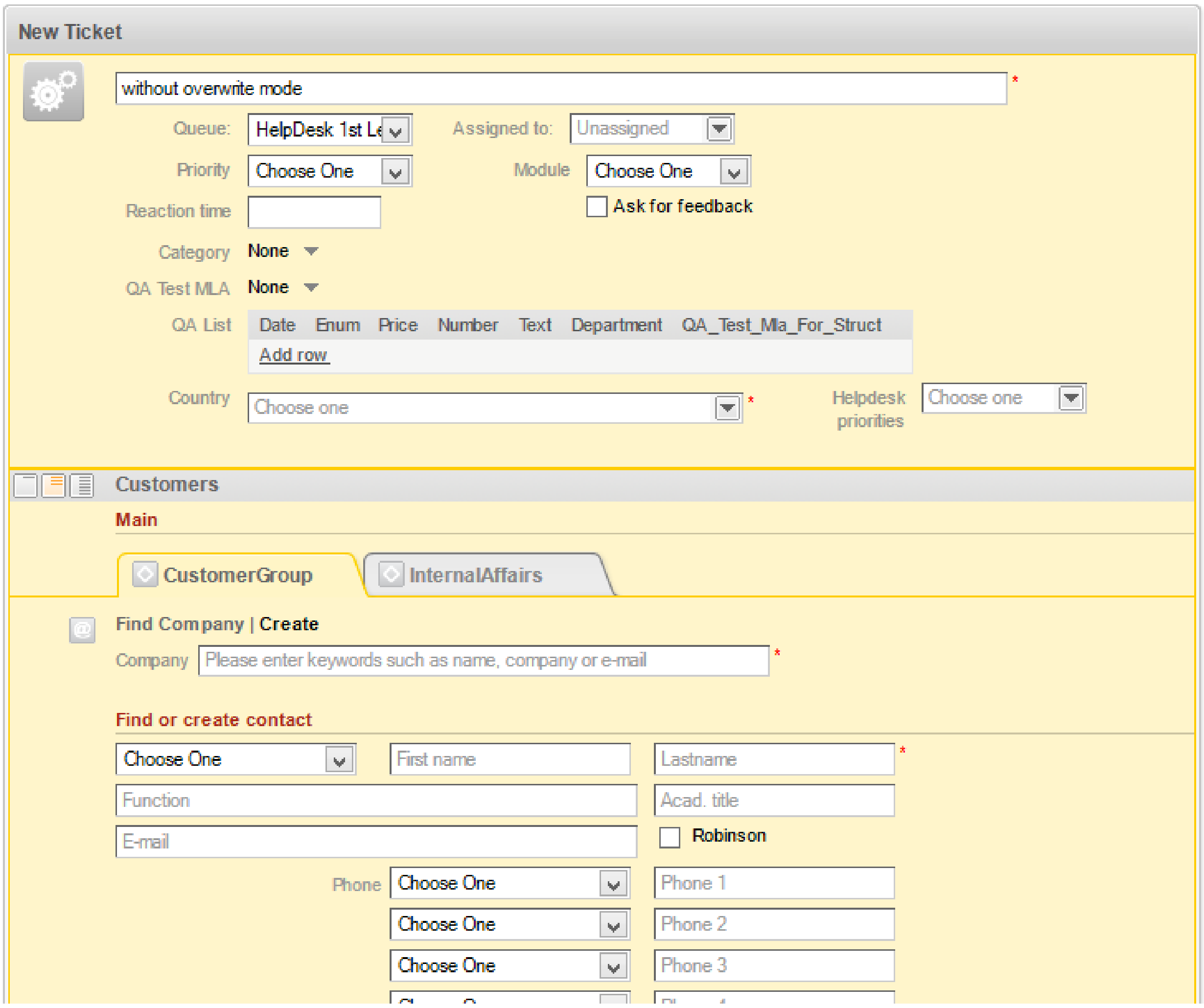
By default the *proactive duplicate prevention* list shows five results. This search result value can be configured using the *page customization* feature. To change the configuration, login to the web client as administrator, open i.e. the *Create ticket* page and click the button *Enable page customization* in the navigation bar. In the lower right frame click the link *contactSection* (*unitFormPanel* → *ticketCreatePage* → *contactSection*) and follow the instructions in the lower left frame. The feature is configured by the attribute *maxSuggestions*. If you want to change this search result value also on other pages please open the page and configure it in the same way.

When there are several customer groups available there are two ways of handling them: first is to use the already known customer group selector located next to the Quick&Easy search which will reduce the number of available groups to one and allows further operations as if the user had access to only that single customer group.

Screenshot of this customer group selector:

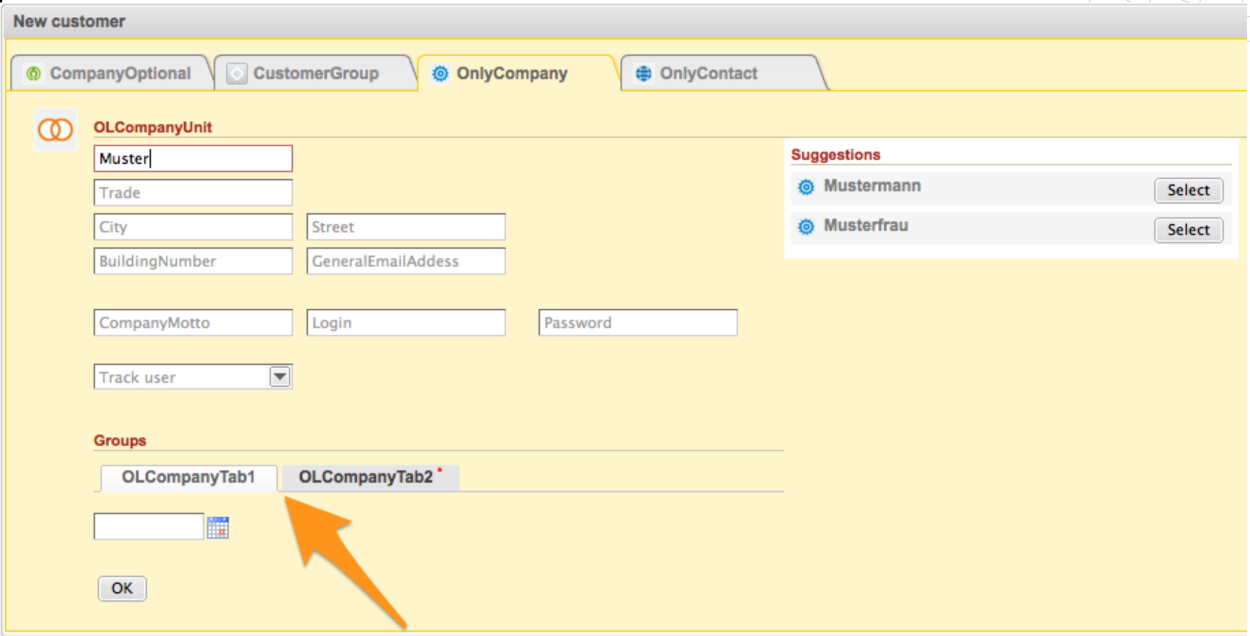


The second way is to use the new tabs that will show up in all places where contacts are to be used. For example on the *Create ticket* page:



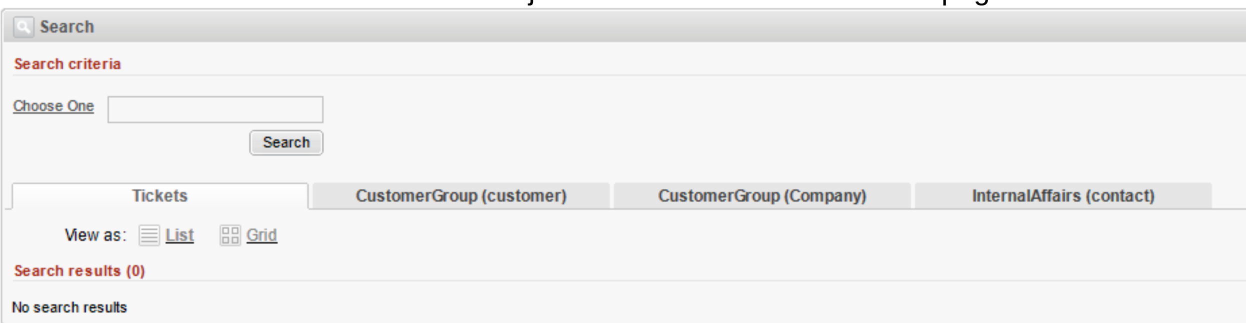
There the engineer may select from which customer group (*CustomerGroup* or *InternalAffairs*) the customer should be selected or created.

On the *Create customer* page it works in a similar way. There also the *proactive duplicate prevention* mechanism works:



Please note that the data object groups *OLCompanyTab1* and *OLCompanyTab2* are visualized in form of tabs. Setting the annotation *show-in-group-section* for these groups configured this.

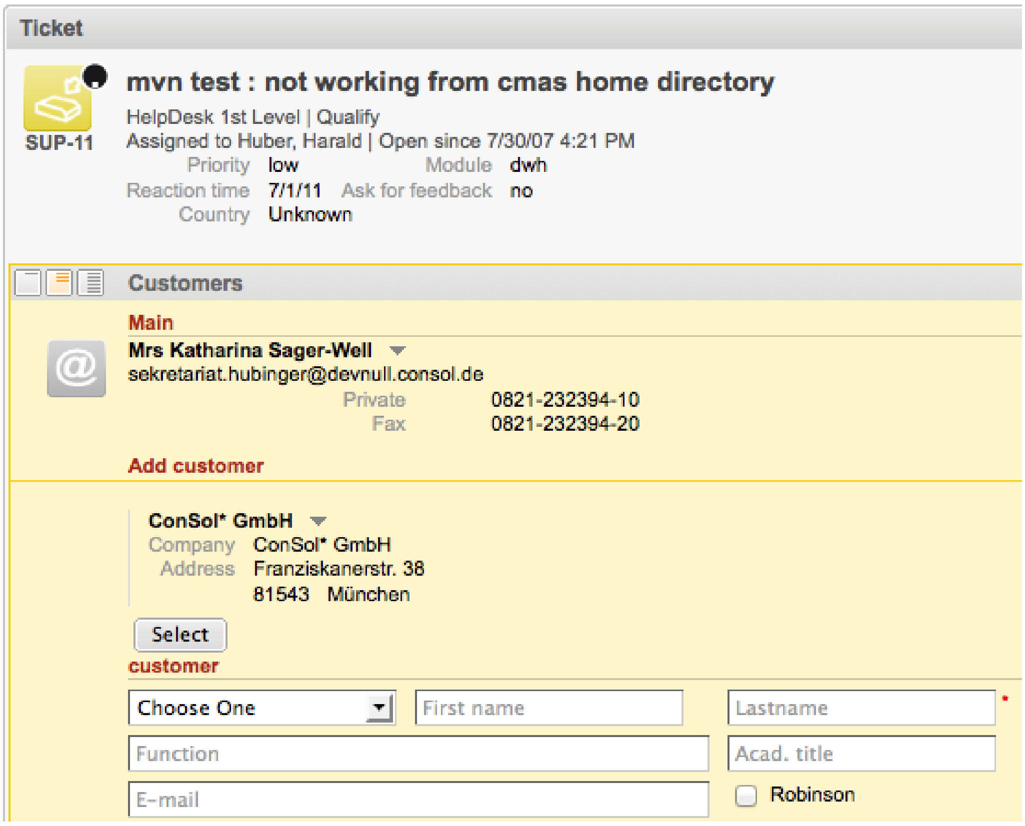
Also there are now tabs for each data object on the detail search results page:



Also each data object has its own section on the Quick&Easy search result list:

HelpDesk 1st Level	SUP-90	AT: Reordering of Attribute Values
	SUP-99	Test
	SUP-39	Ticket attribute relation not deleted when ticket c...
	SUP-72	Out of synchronization messages on single Admin...
	SUP-64	CM/Help and CM/API: Priority-Id of ticket doesn't...
	SUP-19	AT: After deploying new workflow, states list is no...
	SUP-63	Exception during Status Change in "IT" queue
	SUP-20	AT: double click on "Save" during new queue crea...
	SUP-2	No permission granted
	SUP-40	Change of Status/Queue results in logout
Company (CustomerGroup)	Ingenieurbro Glaser Frankfurt a. M.	
	Siebenmaier Gartenbau Oldenburg	
	Granulat GmbH Leutenbach	
	Atlantis - Commercial Real Estate Nürnberg	
	Finanzdienstleistungen Koch Karlsruhe	
	Medizintechnik Grabowski München	
	Achter Chemie AG Ludwigshafen	
	Stern EDV GmbH Bamberg	
	Bio-Genomics AG Martinsried	
	ConSol* GmbH München	
customer (CustomerGroup)	Max Test1	

Another change in context of this feature is the possibility to add a company as the customer to a ticket:



All what needs to be done after searching for a company is to click the *Select* button. Companies may be the main or additional customers of tickets as well as mixed together with contacts and objects from different customer groups.

REST API changes (#623436)

Get customer definition by name

```
GET /definitions/customers/{name}
```

Returns customer definition with given name.

Path param
name - customer definition name

Query param
v - version of element [optional]

Header param
X-Template indicates name of template used to render the response
default: rest_response_customer_definition

Format
JSON
XML

Cache

Variant - based on locale
ETAG - customer definition version

Returns
200 - customer definition was loaded successfully
303 - see other if given version is stale
404 - customer definition doesn't exist with given name

Example:

```
curl -u Huber:consol
http://localhost:8888/restapi/definitions/customers/CmCustomer

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<customerDefinition
uri="http://localhost:8888/restapi/definitions/customers/CmCustomer?v=c-
P1MTQSpiZx-v-wfV2Yag%3D%3D" name="CmCustomer">
  <enabled>true</enabled>
  <company uri="http://localhost:8888/restapi/definitions/model/company?v=c-
P1MTQSpiZx-v-wfV2Yag%3D%3D" />
  <companyOptional>false</companyOptional>
  <contact uri="http://localhost:8888/restapi/definitions/model/customer?v=c-
P1MTQSpiZx-v-wfV2Yag%3D%3D" />
</customerDefinition>
```

Get all customer definitions

```
GET /definitions/customers
```

Returns all customer definitions from system.

Header param
X-Template indicates name of template used to render the response
default: rest_response_customer_group

Format
JSON
XML
Cache

Variant - based on locale
ETAG - customer definition version

Returns
200 - customer definitions were loaded successfully
404 - customer definitions doesn't exist

Example:

```
curl -u Huber:consol http://localhost:8888/restapi/definitions/customers
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<customerDefinitions>
  <customerDefinition
uri="http://localhost:8888/restapi/definitions/customers/OnlyCompany?v=c-
P1MTQSpiZx-v-wfV2Yag%3D%3D" name="OnlyCompany">
    <enabled>true</enabled>
    <company
uri="http://localhost:8888/restapi/definitions/model/OLCompanyUnit?v=c-
P1MTQSpiZx-v-wfV2Yag%3D%3D" />
    <companyOptional>false</companyOptional>
  </customerDefinition>
  <customerDefinition
uri="http://localhost:8888/restapi/definitions/customers/CompanyOptional?v=c-
P1MTQSpiZx-v-wfV2Yag%3D%3D" name="CompanyOptional">
    <enabled>true</enabled>
    <company
uri="http://localhost:8888/restapi/definitions/model/TLCompanyUnit?v=c-
P1MTQSpiZx-v-wfV2Yag%3D%3D" />
    <companyOptional>true</companyOptional>
    <contact
uri="http://localhost:8888/restapi/definitions/model/TLContactUnit?v=c-
P1MTQSpiZx-v-wfV2Yag%3D%3D" />
  </customerDefinition>
  <customerDefinition
uri="http://localhost:8888/restapi/definitions/customers/OnlyContact?v=c-
P1MTQSpiZx-v-wfV2Yag%3D%3D" name="OnlyContact">
    <enabled>true</enabled>
    <companyOptional>false</companyOptional>
    <contact
uri="http://localhost:8888/restapi/definitions/model/OLContactUnit?v=c-
P1MTQSpiZx-v-wfV2Yag%3D%3D" />
  </customerDefinition>
  <customerDefinition
uri="http://localhost:8888/restapi/definitions/customers/CmCustomer?v=c-
P1MTQSpiZx-v-wfV2Yag%3D%3D" name="CmCustomer">
    <enabled>true</enabled>
    <company uri="http://localhost:8888/restapi/definitions/model/company?v=c-
P1MTQSpiZx-v-wfV2Yag%3D%3D" />
    <companyOptional>false</companyOptional>
    <contact
uri="http://localhost:8888/restapi/definitions/model/customer?v=c-P1MTQSpiZx-
v-wfV2Yag%3D%3D" />
  </customerDefinition>
</customerDefinitions>
```


Get unit definition by name

```
GET /definitions/model/{name}
```

Returns unit definition with given name.

<u>Path param</u>
name - unit definition name

Query param
v - version of element [optional]

Header param
X-Template indicates name of template used to render the response
default: rest_response_unit_definition
Format
JSON
XML
Cache

Variant - based on locale
ETAG - unit definition version

Returns
200 - unit definition was loaded successfully
303 - See other if given version is stale
404 - unit definition doesn't exist with given name

Example:

```
curl -u Huber:consol http://localhost:8888/restapi/definitions/model/customer

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<unitDefinition
uri="http://localhost:8888/restapi/definitions/model/customer?v=c-P1MTQSpiZx-
v-wfV2Yag%3D%3D" name="customer">
  <enabled>true</enabled>
  <groups>
    <group
uri="http://localhost:8888/restapi/definitions/groups/customer?v=xO4QbG8oa8CH2
pyKtKWyCg%3D%3D" name="customer">
      <enabled>true</enabled>
      <sortIndex>0</sortIndex>
    </group>
  </groups>
  <type>CONTACT</type>
</unitDefinition>
```

Custom Field Expression Language (CFEL) changes

Before a unit (data object) could have only one custom field group. The expression `unit.get("name")` was always valid since we knew that it can only have one custom field e.g. `contact.name`. Currently a unit may have many custom fields with field `name` e.g. `contact.name`, `business.name`. In such a case the expression `unit.get("name")` is not valid and throws an exception.
In order to be backward compatible `unit.get("name")` will work as long as the custom field `name` is unique. When the admin decides to add another custom field with the same name, `unit.get("name")` will stop working. To solve this issue, following notation was introduced:

```
unit.get("customer:name")
```

This is the only way we recommend, using ":" enforces the group name `customer` and will not cause any troubles.

Another notation is still possible:

```
unit.get("customer.name")
```

It is not recommended since it will stop working when the admin will introduce a custom field with name `customer.customer`.

To sum up please always use `group1:field1.group2:field2` in your CFEL expressions. They will become longer, but a lot safer.

API Improvements

Groovy level enhancements:

On groovy scripts level it is possible to call some additional methods added dynamically to support the feature related restrictions. These methods are aware of the fact of existing ticket-contact-company relations and are named according to them.
Below are examples of such calls:

```
Ticket ticket = getTicket()
Unit mainContact = ticket.getMainContact()

Unit company = mainContact.getCompany()
List contacts = company.getContacts()

List tickets = company.getTickets()
tickets = mainContact.getTickets()
```

Search criteria builder:

A new service *mdcmCriteriaBuilder* was added. Its main goal is to create patterns for searching for contacts and hiding the complex custom fields structure from the developer.

To search for tickets of a given contact (or company):

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

To search for tickets of a given contact of a company:

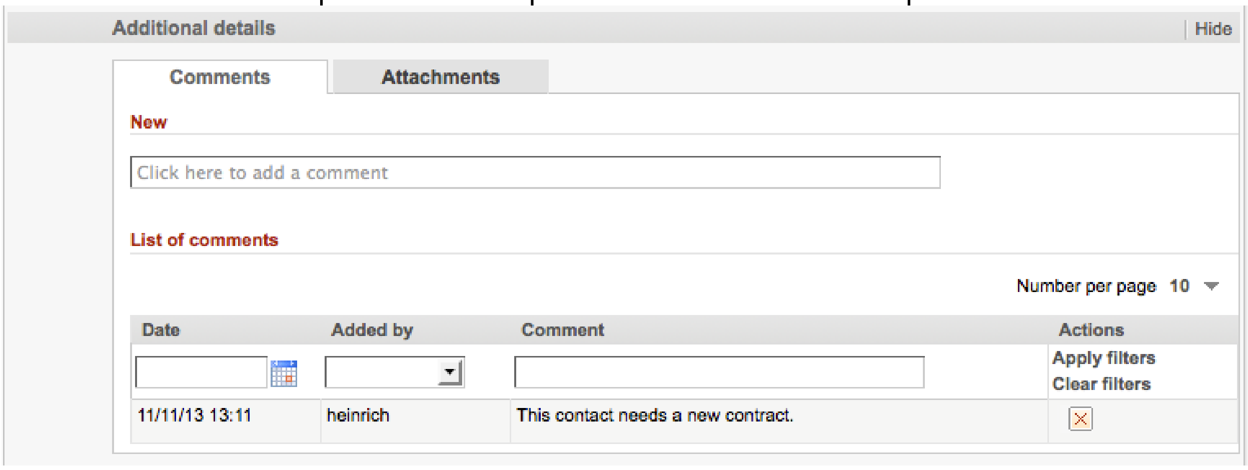
```
TicketCriteria ticketCriteria = new TicketCriteria();
Unit contactPattern = new Unit("contact", customerGroup);
mdcmCriteriaBuilder.setReferencedContactCriteria(ticketCriteria,
contactPattern);
Unit companyPattern = new Unit("company", customerGroup);
companyPattern.setFieldValue("name", "aaa");
mdcmCriteriaBuilder.setReferencedCompanyCriteria(contactPattern,
companyPattern);
```

To search for contacts of a given company:

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

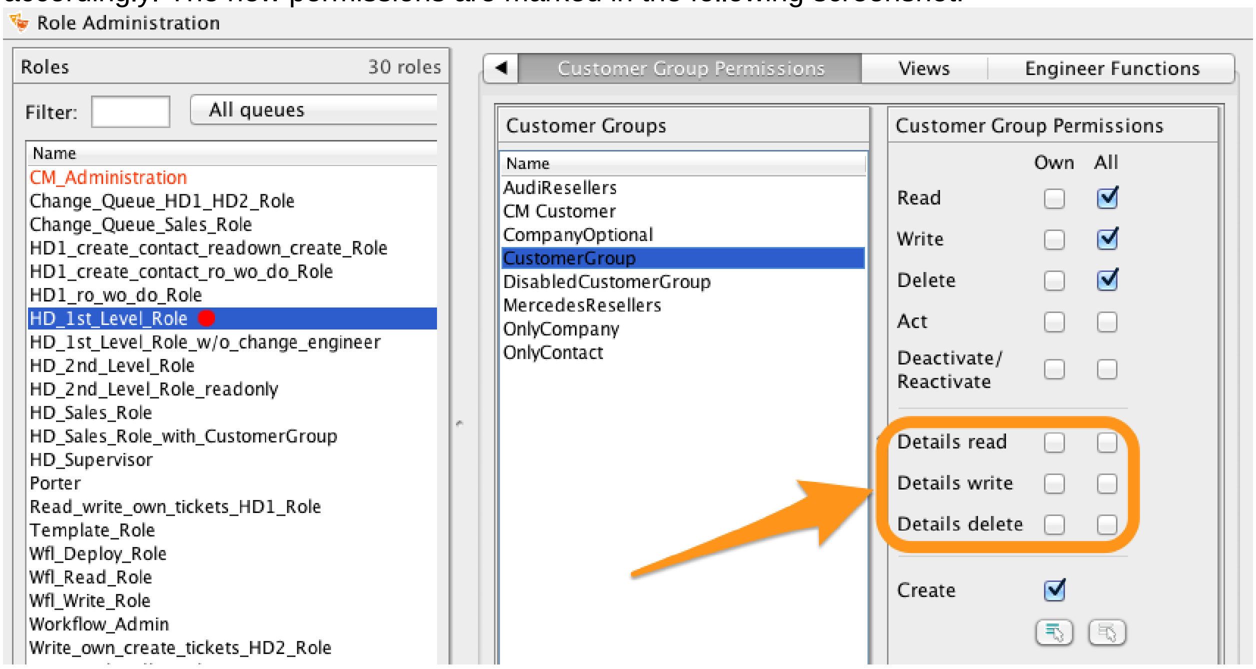
1.2.2 Comments and attachments for customers (#622426)

A new section *Additional details* has been added to the data object pages (like contact and company).
The section may contain short comments and attachments connected to the customer. Those two kinds of content are presented on separate tabs as shown on the picture below:



The tabs use a similar filtering and sorting component as the one that can be found in the ticket's attachment section.

The administrator can restrict access to this section by setting new customer group permissions accordingly. The new permissions are marked in the following screenshot:



Workflow API

For manipulating the content from workflow scripts new methods have been added to the workflow API:

```
/**
 * Creates new unit comment
 * @param pUnit Target unit
 * @param pComment Comment text
 * @return Newly created comment entry
 */
UnitCommentEntry addUnitComment(Unit pUnit, String pComment);

/**
 * Deactivates comment for given unit
 * @param pEntry entry to deactivate
 */
void deleteUnitComment(UnitCommentEntry pEntry);

/**
 * Returns all comments created for given unit
 * @param pUnit Target unit
 * @return List of comments entries
 */
List<UnitCommentEntry> getUnitComments(Unit pUnit);

/**
 * Creates new unit attachment. Data input stream should be set using
pAttachment.setInputStream(stream,size)
 * Description fields should be set using:
 *
 * pEntry.setDescription(new UnitAttachmentEntry.Description(filename,
mimetype, description))
 *
 * @param pAttachment Attachment entry, description fields (mimeType,
filename) and inputstream must be set
 * @return Newly created comment entry
 */
UnitAttachmentEntry addUnitAttachment(UnitAttachmentEntry pAttachment);

/**
 * Deactivates attachment entry of given unit
 * @param pAttachment Attachment to deactivate
 */
void deleteUnitAttachment(UnitAttachmentEntry pAttachment);

/**
 * Returns all attachments of given unit
 * @param pUnit Target unit
 * @return Attachments entries list
 */
List<UnitAttachmentEntry> getUnitAttachments(Unit pUnit);
```

REST

For manipulating the content via REST API new methods are available:

Get comments:

```
curl -u Huber:consol http://localhost:8888/restapi/units/123/comments
```

Will return:

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<comments>
  <comment id="213-123-12-12" creationDate="02-03-2013">
    <text>something</text>
    <active>true</active>
    <engineer uri="http://localhost:8888/restapi/engineers/Huber?v=0" />
 </comment>
 <comment id="231-123-aaf-123" creationDate="03-03-2013">
   <text>something important</text>
   <active>true</active>
   <engineer name="Huber" firstname="Helard" lastname="Huber"
email="hhuber@devnull.consol.de" />
 </comment>
</comments>
```

The REST API has to check whether the engineer still exists in the system. If not, the engineer tag does not generate the *uri* argument. Instead all required information is retrieved from the *HistoryData* object and placed in the tag arguments.

Add comment:

```
curl -u Huber:consol -X PUT -d "text=123" http://localhost:8888/restapi/units/123/comments
```

Delete comment:

```
curl -u Huber:consol -X DELETE http://localhost:8888/restapi/units/123/comments?text=123
```

Get attachments:

```
curl -u Huber:consol http://localhost:8888/restapi/units/123/attachments
Will return:

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<attachments>
  <attachment id="aa46dd51-e53e-11e2-a2d0-d3e5339f95aa" creationDate="2013-02-
02">
    <description>This is description</description>
    <active>true</active>
    <enginner uri="http://localhost:8888/restapi/engineers/Huber?v=0" name="Huber">
    <file uri="http://localhost:8888/restapi/units/123/attachments/file/aa46dd52-e53e-11e2-a2d0-
d3e5339f95aa/mypicture.jpg">
      <mimetype>image/jpeg</mimetype>
      <name>mypicture.jpg</name>
      <size>123123</size>
    </file>
  </attachment>
```

```
</attachments>
```

Get the attachment file:

```
curl -u Huber:consol http://localhost:8888/restapi/units/123/attachments/file/aa46dd52-e53e-11e2-a2d0-d3e5339f95aa/mypicture.jpg
```

Delete attachment:

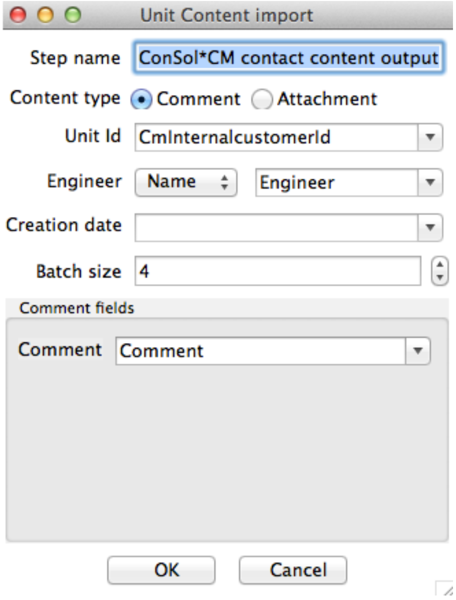
```
curl -u Huber:consol -X DELETE http://localhost:8888/restapi/units/123/attachment/aeec-123-123-123-123
```

Add attachment:

```
curl -u Huber:consol -X POST -F "file=@/home/huber/Desktop/file.png" -F "description=the description" http://localhost:8888/restapi/units/123/attachment
```

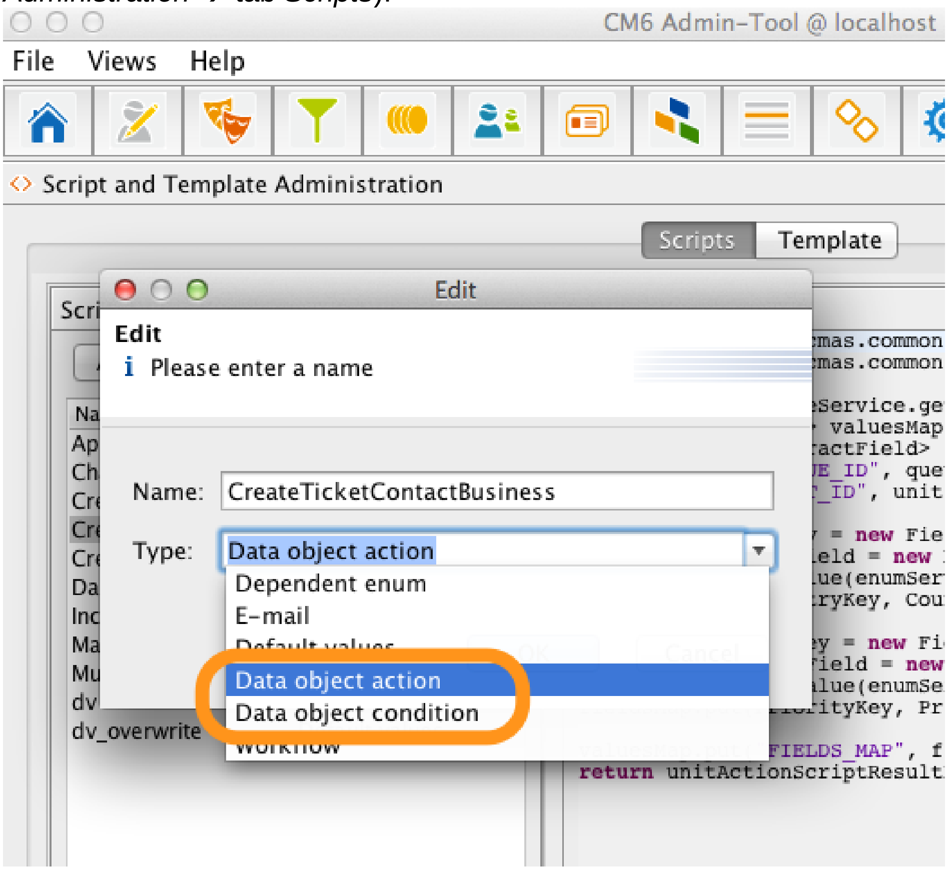
ETL (#623492)

There is a new plugin *ConSol*CM contact content output* that can be used to import customer comments and attachments:

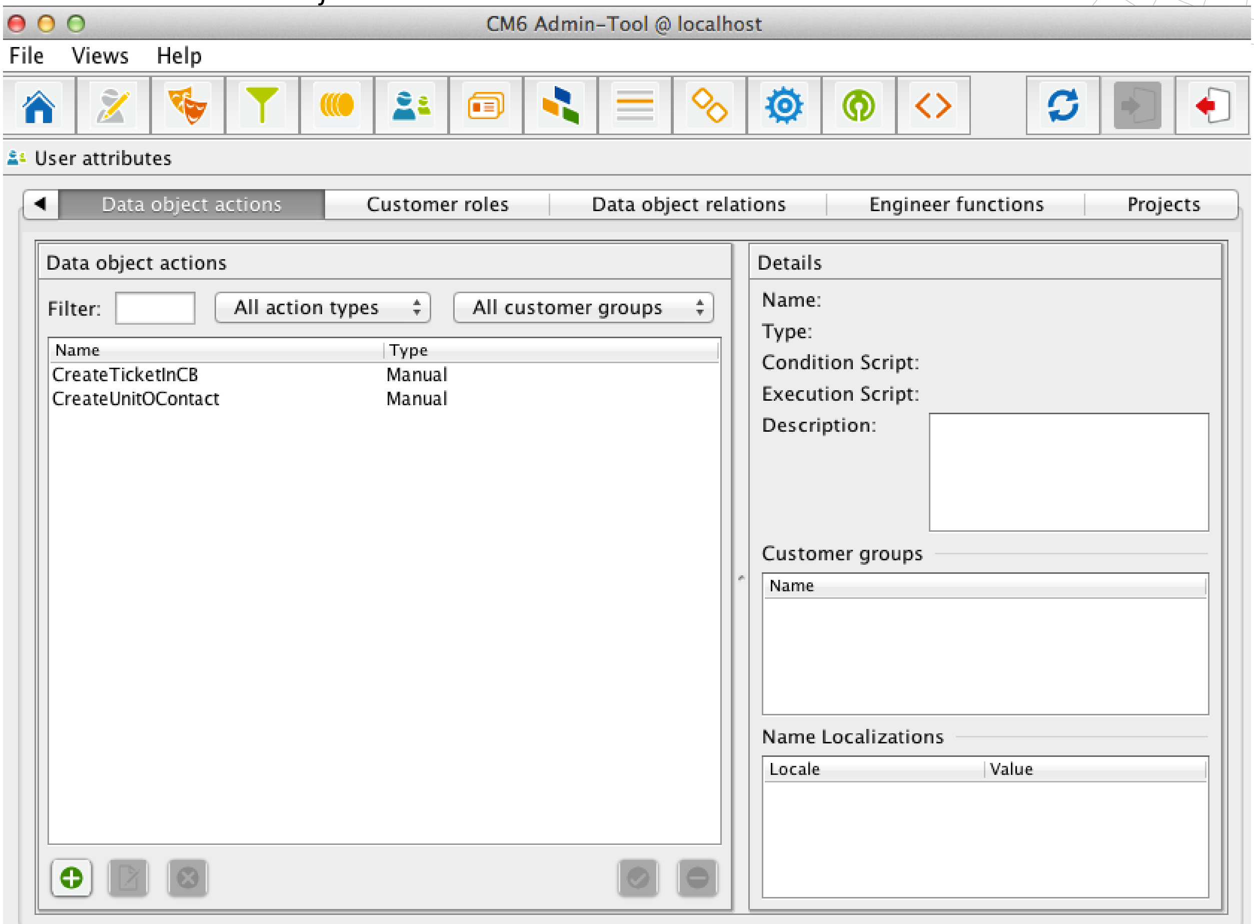


1.2.3 Data Object Actions

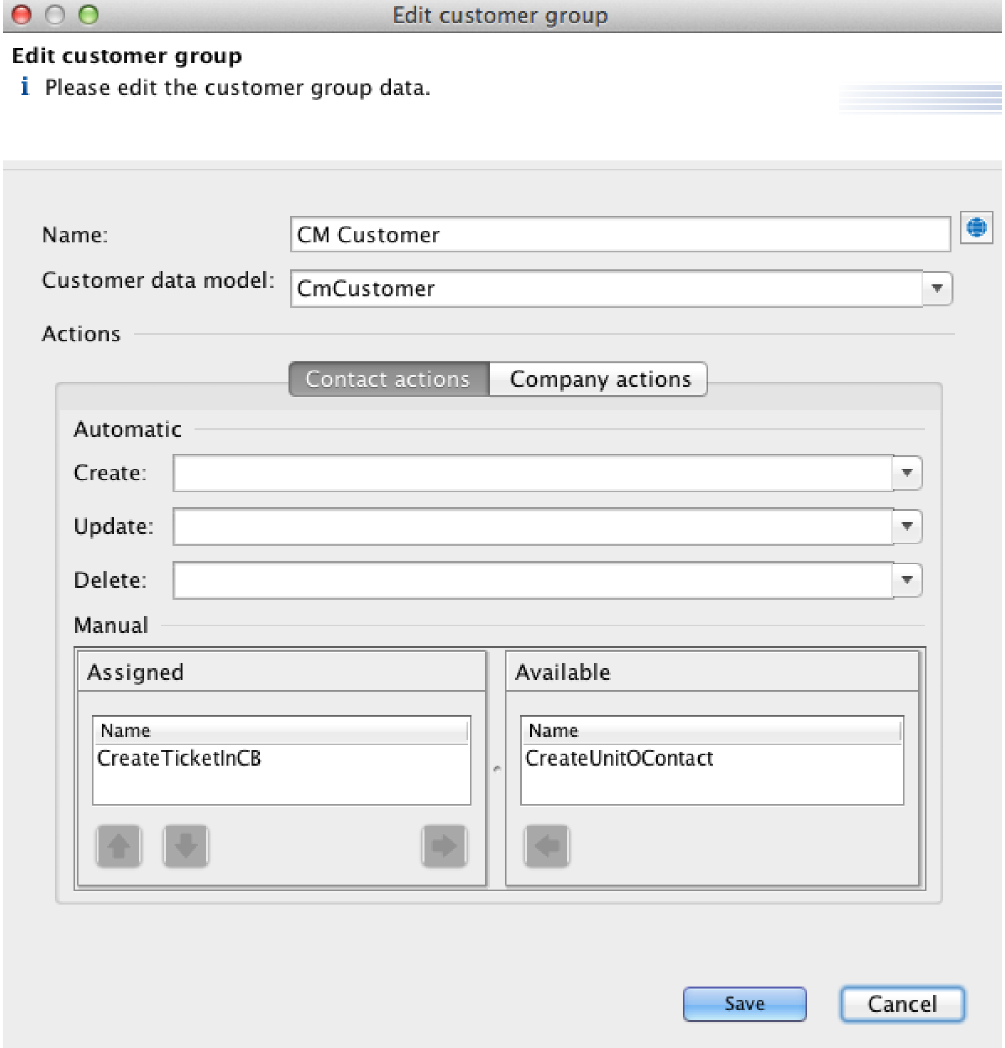
It is now possible to define actions that will either be executed automatically when a data object is created, updated or deleted, or manual actions that can be triggered by the engineer on the data object page in the CM6 Web Client.
These actions are executed in form of Groovy scripts. There are two types of action scripts: *Data object action* and *data object condition*.
Data object condition scripts are used to determine if the activity should be available in the web client or not. The *data object action* script is used to perform the actual activity on the data object, i.e. open the *Create ticket* page and directly set the data object as main customer of the ticket.
Scripts of these two types can be composed in the CM6 Admin Tool (*Script and Template Administration* → tab *Scripts*):



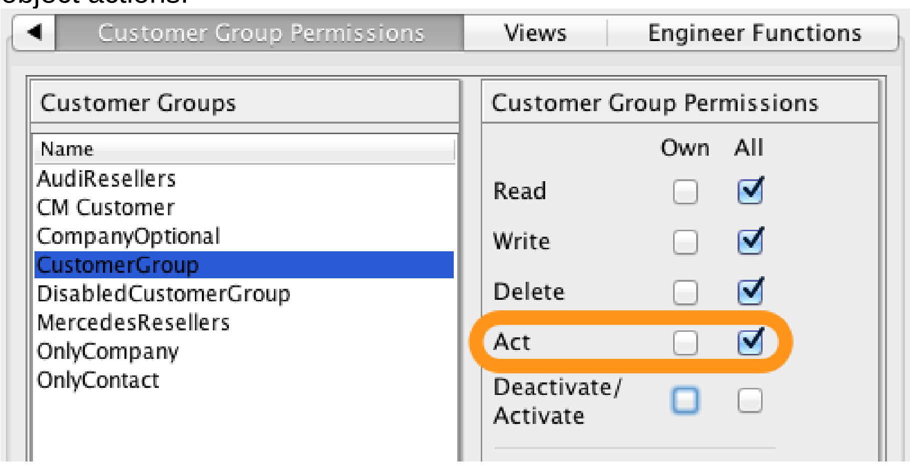
The data object actions can be created and administered in the CM6 Admin Tool → *User attributes* → tab *Data object actions*:



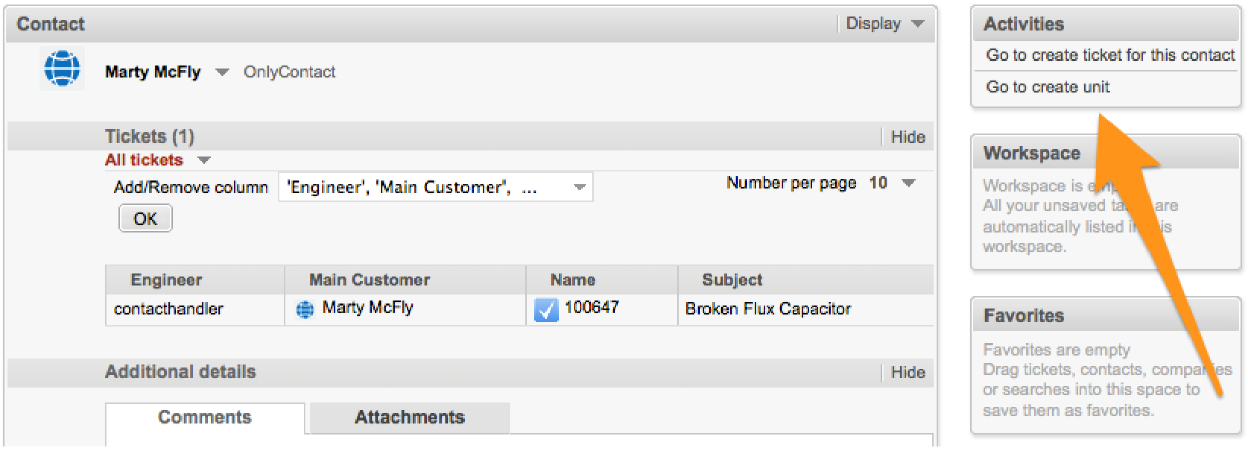
The actions are assigned to customer groups then (*User attributes* → tab *Customer groups*):



There is a new permission *Act* for customer groups that controls the access to manual data object actions:

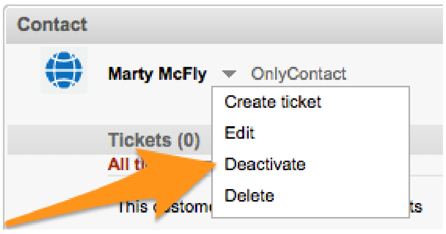


Manual actions are shown in a new box *Activities* on a data object page as can be seen in this screenshot:



1.2.4 Deactivation of Customers

CM6 now supports the deactivation of customers. They are still present in the system (i.e. assigned to closed tickets), but they can no longer be found by the Quick&Easy search, and new tickets can no longer be created for them.
A new option *Deactivate* (or *Activate* for already deactivated customers) was added to the context menu:



A customer can only be deactivated if she has no open tickets assigned (neither as main nor as an additional customer).

For a deactivated customer the following options are available:

- Edit customer data, e.g. update name, address fields, custom fields groups.
- Delete customer
- Transfer her closed tickets

It is no longer allowed to:

- Create a new ticket for a deactivated customer
- Assign an existing ticket to a deactivated customer
- Assign the deactivated contact to another company
- Assign new contacts to a deactivated company

A deactivated customer is visualized by an italic font as can be seen in the following screenshot:

Search for deactivated customers:

The Quick&Easy search does not find deactivated customers and they also won't be shown at the *proactive duplicate prevention* list.

Deactivation for a two level model:

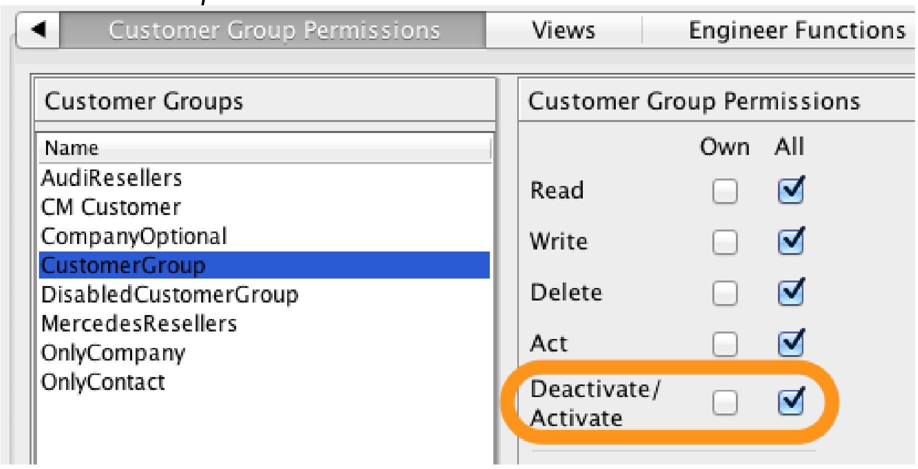
All contacts of the company can be deactivated (if they have no open tickets assigned).
In this case the company and all assigned contacts are deactivated.

Customer reactivation:

It should be possible to reactivate a customer.
For two level model: in case of a company reactivation the assigned contacts (which are deactivated) will not be automatically reactivated. The engineers can reactivate such contacts afterwards manually.

Permissions:

A new permission to deactivate/activate a customer per customer group has been added to the *Customer Group Permissions* section in the CM6 Admin Tool:



REST:

The REST API should provide a method to deactivate/reactivate a customer.

```
curl -u webadmin:consol -X PUT
http://localhost:8888/restapi/units/1242.html?active=false
```

DWH:

The DWH customer table DIM_CONTACT was extended for storing the customer's deactivated flag and the deactivation time stamp.

Workflow:

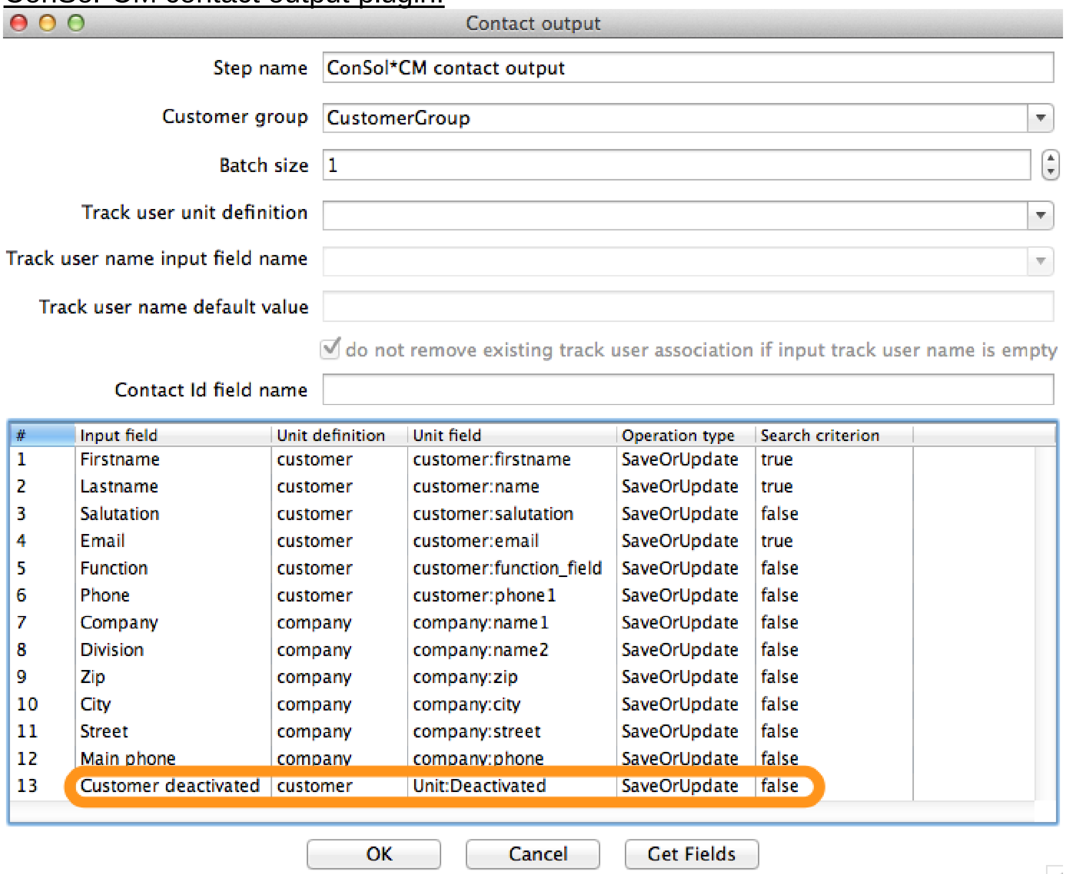
The Workflow API provides methods for customer deactivation/reactivation:

```
/**
 * Deactivates given unit, If unit is a parent unit (e.g company) of other
units
 * all children units will be deactivated as well.
 *
 * @param pUnit Unit to deactivate
 * @throws UnitDeactivationValidationException if there are open tickets
associated with any of the deactivated units
 */
void deactivateUnit(Unit pUnit);

/**
 * Activated given unit and all of its child units which were deactivated
together with parent
 * @param pUnit Unit to activate. By default all child units (previously
deactivated by parent) will be activated as well
 */
public void activateUnit(Unit pUnit);
```

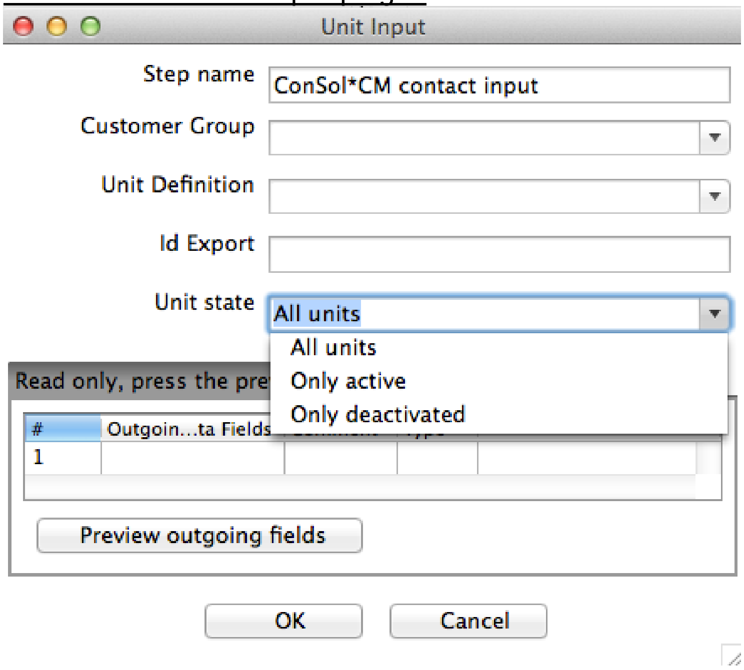
ETL:

ConSol*CM contact output plugin:



When importing data objects (units) there is a new unit field *Unit:Deactivated* which can be set during import in order to set the activation status for units.

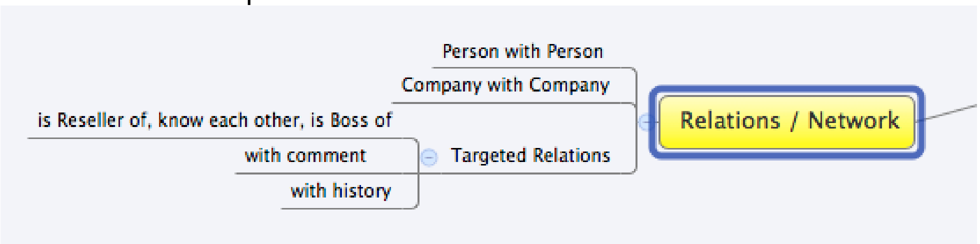
ConSol*CM contact input plugin:



When exporting data objects (units) one can choose whether to export all units or only active or deactivated ones.

1.2.5 Customer Relations

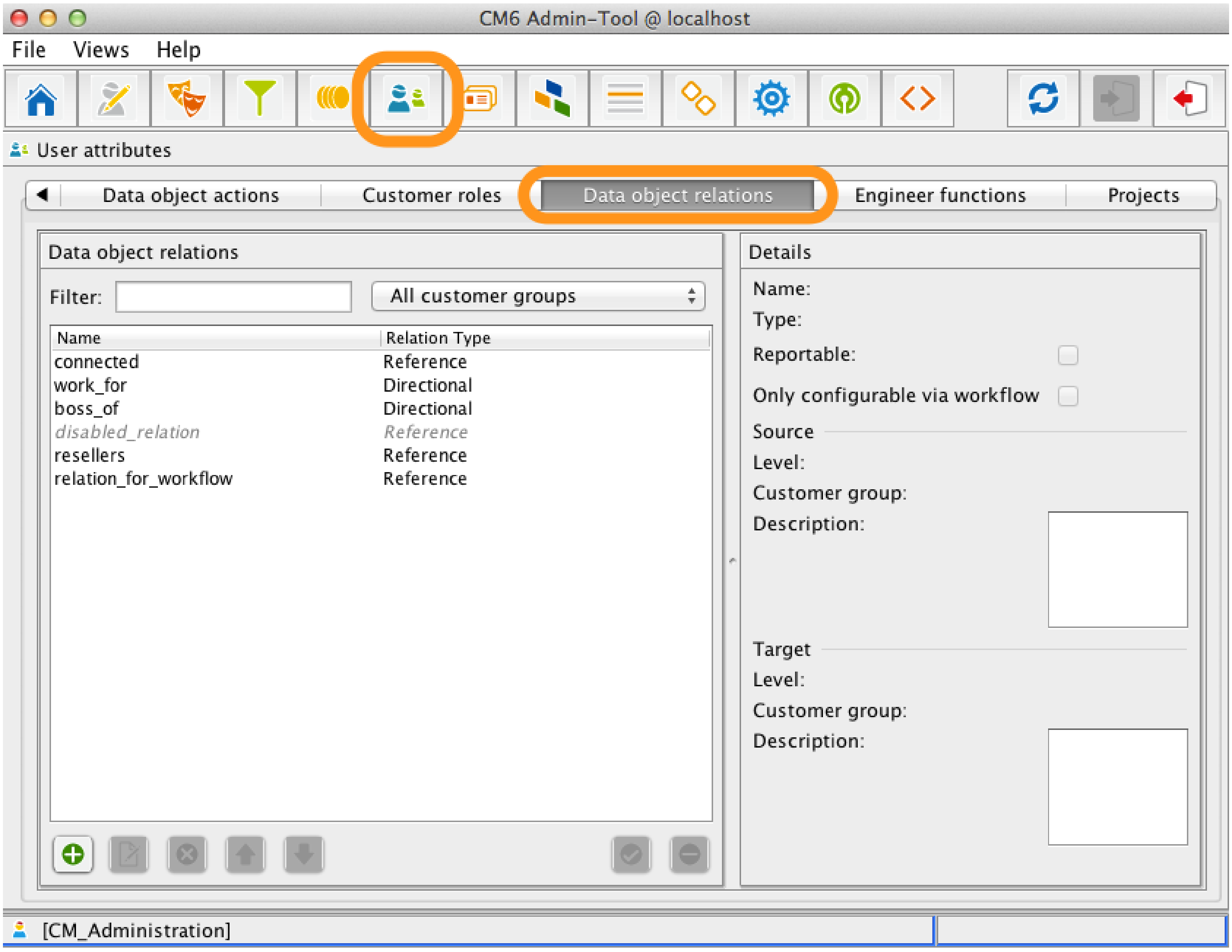
Since 6.9.1.0 a new feature is available to define relations between customers. The main idea behind the relations between *Customers* on all its levels is the possibility to create a **Customer Network** which will allow a user to define and see what the connections between contacts and companies in all possible combinations are.
Like shown on the picture below:



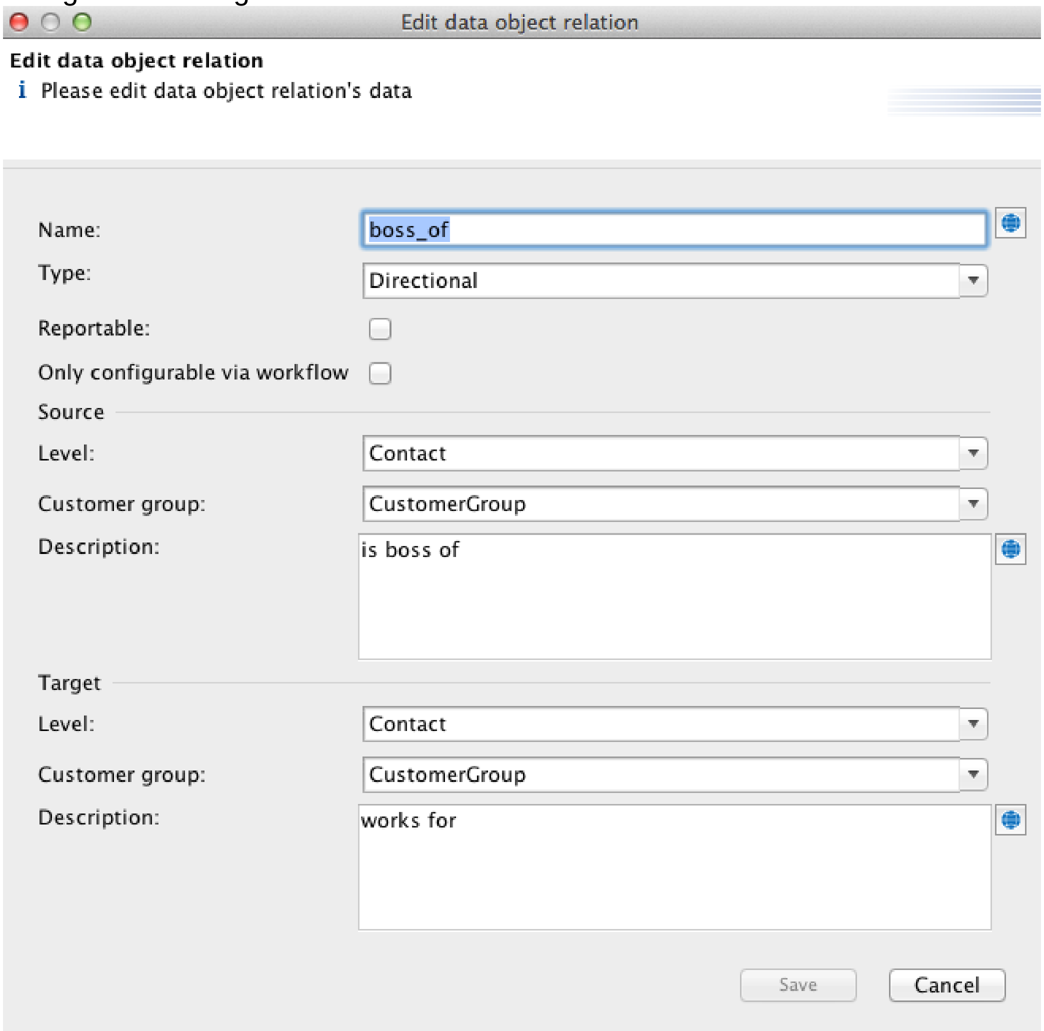
The relations will enable to define business connections with the help of appropriate comments and relation types.

CM6 Admin Tool: Configuration of Customer Relations

Customer Relations can be configured via the CM6 Admin Tool (page *User attributes* → tab *Data object relations*):



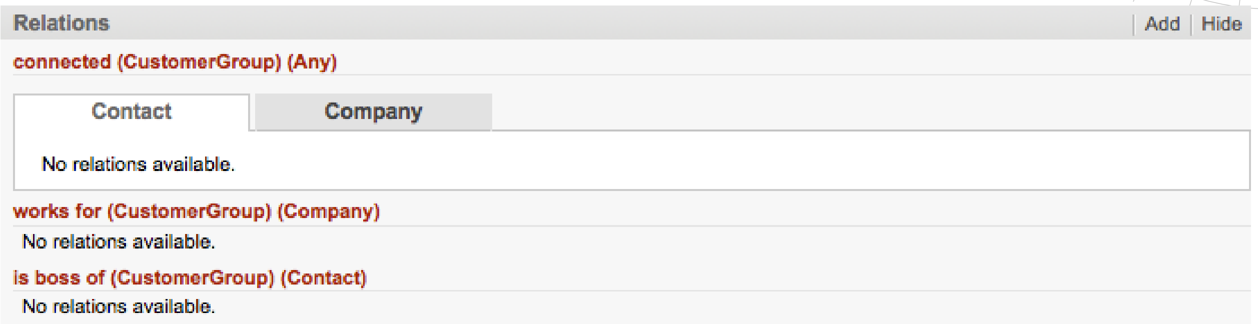
Using the following form the administrator can create/edit relations:



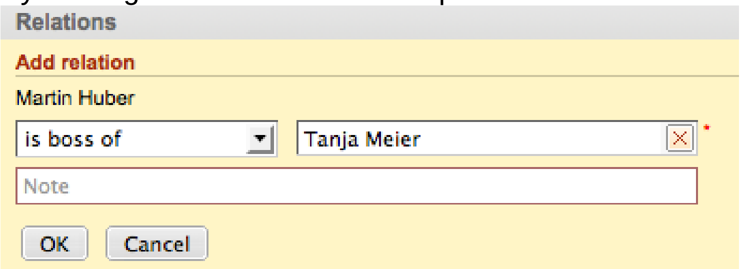
There are two types of relations: *Directional* and *Reference*.
For directional relations different descriptions can be defined for the source and target of the relation in order to define relations like i.e. “Mrs. Jane Doe is the boss of Mr. John Doe” and “Mr. John Doe works for Mrs. Jane Doe”.
By checking the checkbox *Reportable* the relation is made available in the DWH (tables `DIM_UNIT_RELATION` and `DIM_UNIT_RELATION_DEFINITION`).
Only configurable via workflow configures that the relation cannot be edited in the CM6 Web Client. Creating, modifying or deleting such a relation is only possible via workflow scripts.
For the source and target of a relation it can be defined for which levels of the customer model it should be used (any level, or contact or company level for two level customer models). And the administrator has to choose from which customer groups the source and target data object will be used.

Customer Relations in the CM6 Web Client

A new Section *Relations* was added to the contact and company pages:

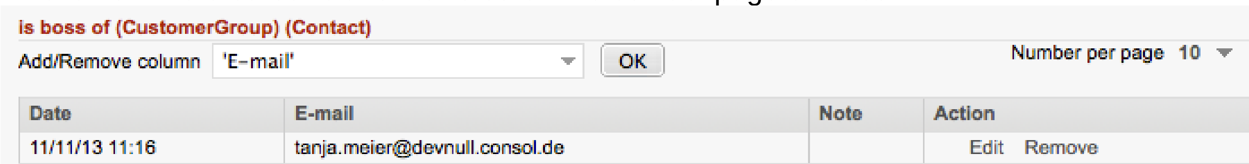


By clicking the link *Add* one can open the form to create a relation:

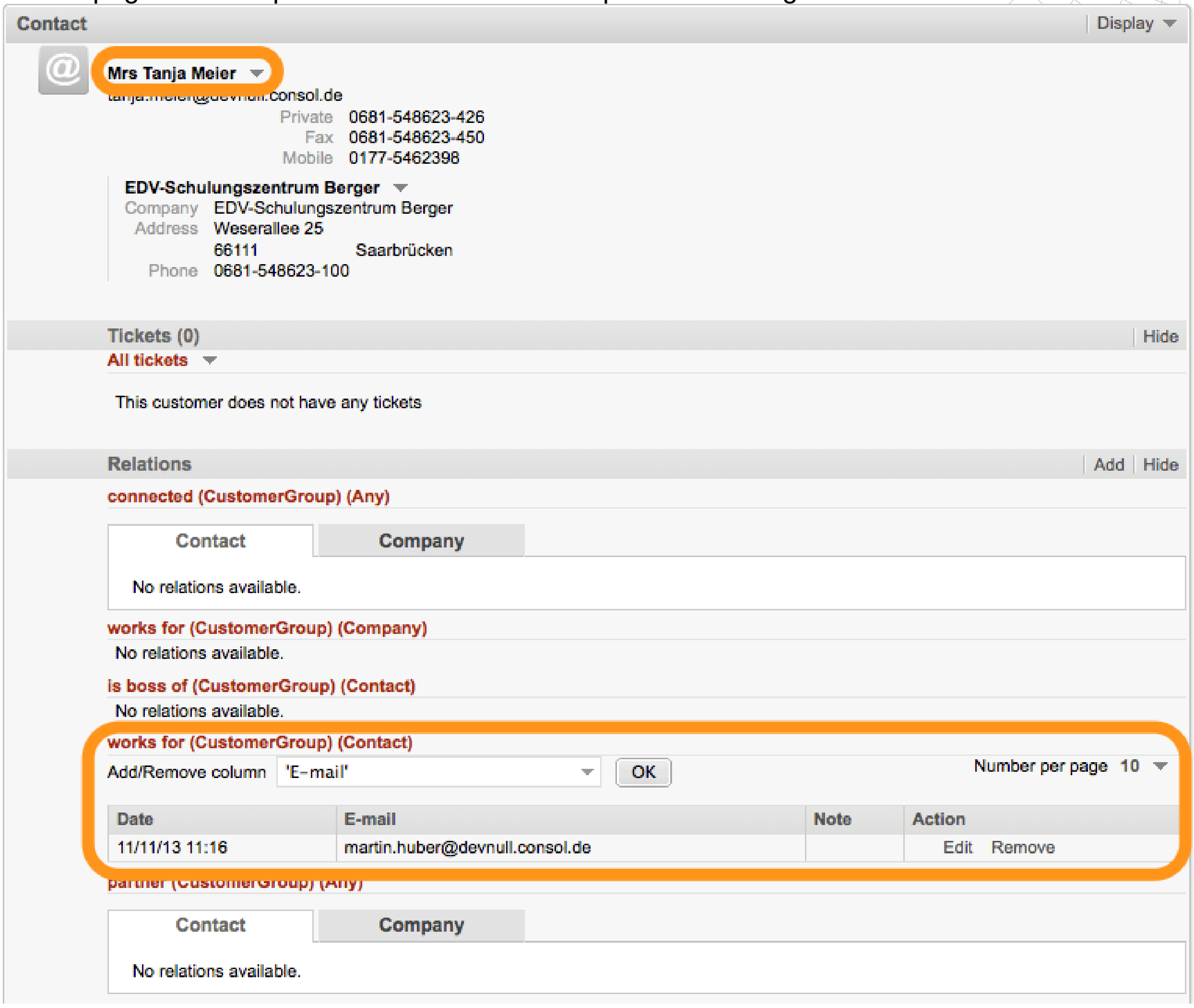


The engineer has to choose the relation and can search for the target data object of the relation. Also it is possible to optionally add a note to the relation.

This is how the created relation is listed on the contact page of *Huber*:



On the page of *Meier* please note that the description of the target relation *works for* is used:



Workflow

In order to create, modify and delete Customer Relations via Workflow please use the methods of the class `UnitRelationService`. Please refer to the official Javadoc documentation for detailed information about the service methods.

REST

Creating, modifying or deleting Customer Relations via REST API is currently not supported. The REST API support will be added in an upcoming release.

1.2.6 Background service to remove deleted content from the database (#622688)

Until now if one deleted an attachment of a ticket, or comments or attachments of a data object, it still remained in the database. It was only marked as deleted.
With 6.9.0.1 a background service was added that removes such entries from the database that are marked as deleted.
The name of the service is *unused content remover* and it can be stopped and started in the CM6 Admin Tool in the same way as other services there. It is also possible to completely disable this service by setting the property *unused.content.remover.enabled* (module *cmas-core-server*) to *false*.
The service is started automatically for single server environments and for cluster environments only if it is configured on which cluster node this service should run. This is configured by the property *unused.content.remover.cluster.node.id* (module *cmas-core-server*).
It can be configured how often it runs in the background (property *unused.content.remover.polling.minutes*; default is every 15 minutes) and how for how long content is kept in the database until it is removed (property *unused.content.remover.ttl.minutes*; default time span is 24 hours).
The service can also be invoked via JMX. When using JBoss as application server you can access the bean in the following way:

1. Open the JMX console (http://<host>:<port>/jmx-console).
2. In the left frame *Object Name Filter* select *consol.cmas*.
3. On the right side select *name=unusedContentRemover, topic=global,type=config*
4. Invoke the operation *removeUnusedContent* (parameter is for how many minutes the content to be removed was already marked as deleted).

After the update to 6.9 this background service is active. In order not to consume too many resources it will only remove at most 1000 entries during each run.

1.2.7 REST: Track User can be assigned upon contact creation/update (#623342)

It is now possible to directly assign a *Track user* to a contact during its creation or update. A new parameter *engineer* was added for this.

Example:

```
curl -u Huber:consol -d
"customerGroup=CustomerGroup&customer.companyRef=62301&customer.name=Musterman
n&engineer=track_faq" http://localhost:8888/restapi/units
```

returns

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<location>
  <uri>http://localhost:8888/restapi/units/73437</uri>
</location>
```

1.2.8 CM6 Process Designer: Precondition script for Activity Control Forms (#620953)

Activity Control Forms now have a second script field: *Precondition script*. This script is used to determine if an Activity Control Form will be displayed or not.
The author of the process can then decide if the form should be shown if its required fields are set or if another condition is fulfilled. By default, if no script is there, it will behave as in former versions.
Please note also that the name of the first script field was changed from *Script* to *Initializing script*.

1.2.9 Possibility to gather mailbox information in the mail processing scripts (#621956)

The mailbox information can now be fetched in mail processing scripts like i.e. *CreateTicket.groovy* by using this new method:

```
String mailboxInfo = MailContextServiceImpl.extractMailIncomingMailboxURI(msg)
```

The method will return the connection URI in the form as it is shown here:
imaps://mailboxUser@imap.mailserver.com:993

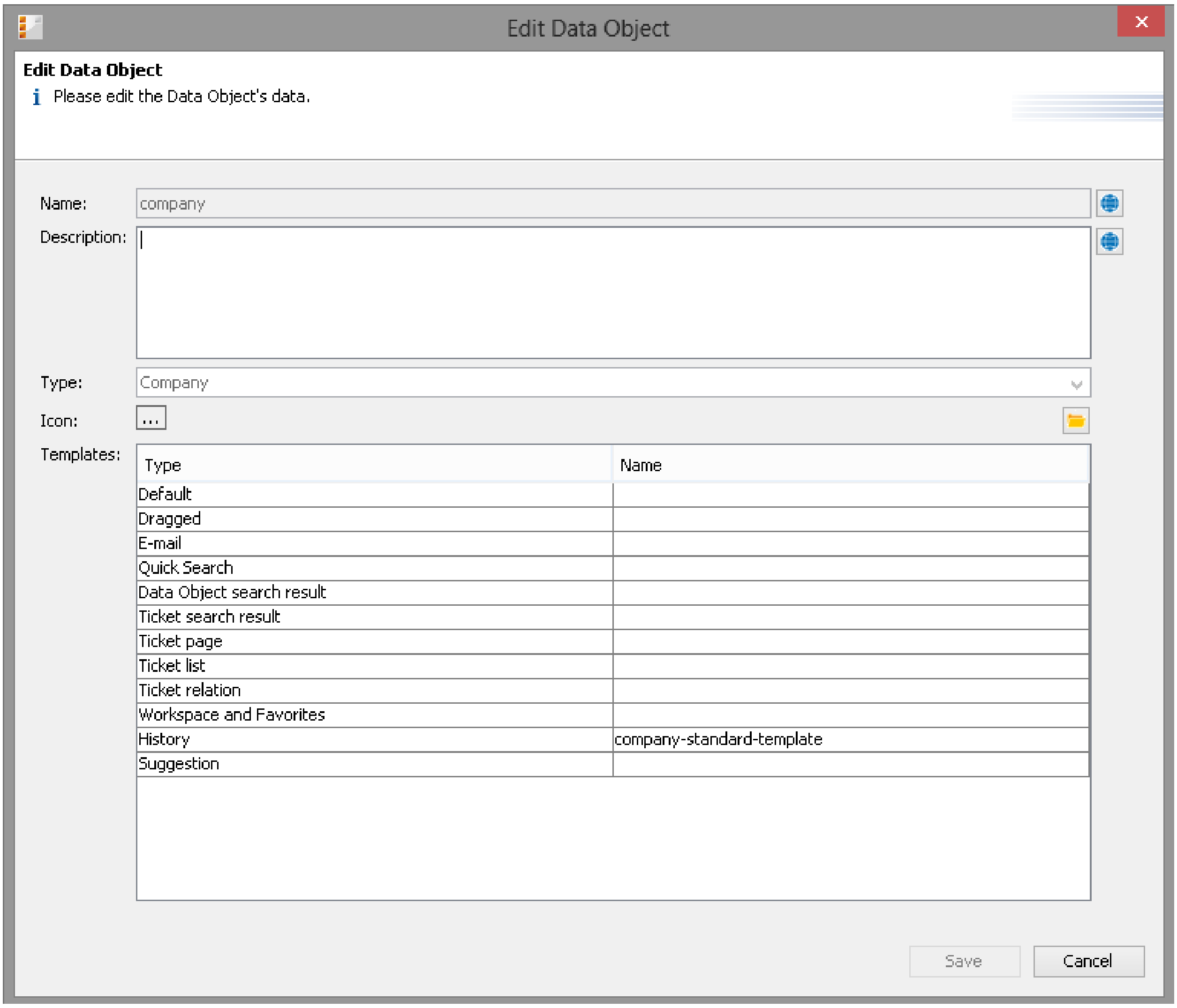
(password omitted).

1.3 Changes

1.3.1 Obsolete annotations were removed

Due to the changes related to the feature *Flexible Customer Data Management* the annotations *unit is a contact* and *show-contact-in-ticket-list* became obsolete. They are automatically removed during the update to 6.9 or during the import of older export files.
The first annotation is now no longer needed since currently data objects of type *contact* are explicitly defined in the dedicated dialog of the CM6 Admin Tool.
Showing/hiding contacts on the ticket list can now be configured using the *page customization* feature. To change the configuration, login to the web client as administrator and click the button *Enable page customization* in the navigation bar. In the lower right frame click the link *accordionTicketList* and follow the instructions in the lower left frame. The feature is configured by the attribute *mainCustomerDescriptionVisible*.

Also customer templates are now configured in a different way as the related annotations were removed. Instead the templates can be set at the customer data object administration view in the CM6 Admin Tool:



1.3.2 CM6 Setup without scenario import no longer creates a default customer group (#622970)

In former versions if one performed a CM6 Setup without importing a scenario then a default customer group was created. This customer group would now be invalid with 6.9 since it is not assigned to any customer data model. Therefore the creation of such a default customer group is now omitted.

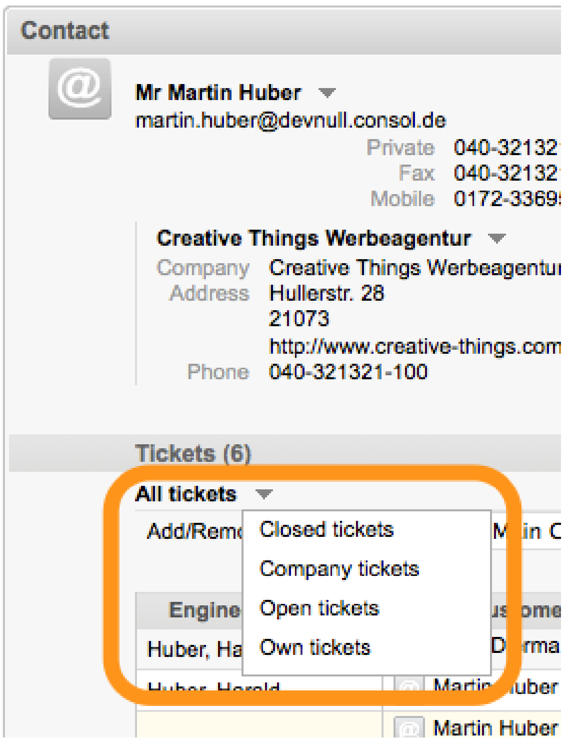
1.3.3 CM6 Admin Tool: Customer related Custom Fields are configured on a different page now

Due to the changes related to the feature *Flexible Customer Data Management* the *Custom Fields Administration* view no longer contains the customer related configuration tab. The functionality was moved to the *User Attributes* view, on the *Customer Data Model* tab.

1.3.4 Ticket filter options on data object pages (#623225)

The ticket filter options on data object pages like i.e. *Contact* page or *Company* page now have the following entries:

- *All tickets* (default setting that shows all tickets of the customer where it is assigned either with the main role or as an additional customer)
- *Open tickets*
- *Closed tickets*
- *Own tickets* (tickets where the customer is only assigned with the main role)
- *Company tickets* (all tickets of the contacts of a company plus those tickets that were just assigned to the company (in a two-level model))



1.3.5 *Mandant* was renamed to *Kundengruppe* (#623431)

Please note that the German word *Mandant* was renamed to *Kundengruppe* in order to match its English translation *customer group*.

1.3.6 CM/Office support for the *Flexible Customer Data Management* (#623547)

The new feature *Flexible Customer Data Management* makes it possible to assign different customer models to a queue. Those models itself can consist of multiple groups of fields.
Since CM/Office was built to support only queues with one customer model and one group of fields for a data object, changes had to be applied in order to support the new structure.
The new structure is now reflected by a new syntax for the *Mail Merge* fields:

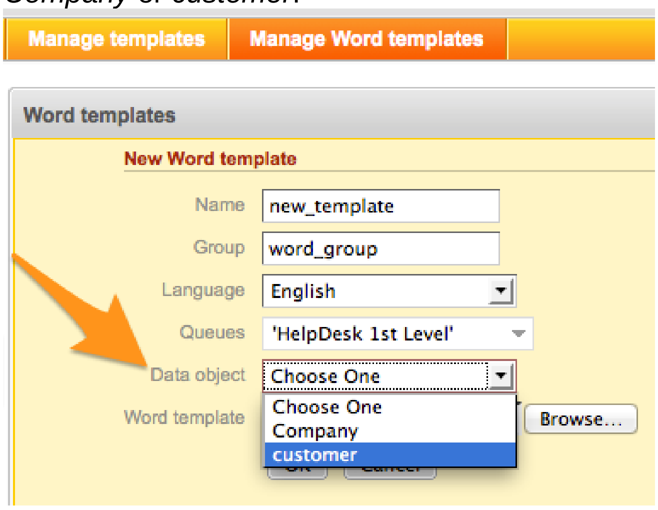
```
ticket_customer_data object name]_[group of fields]_[field name]
```

Example:

```
ticket_customer_contact_profile_firstname
ticket_customer_contact_addresses_city

ticket_customer_company_general_name
ticket_customer_company_addresses_city
```

This new syntax makes it possible to define the group of this field and the data object.
Please note that the old syntax is still supported in order to provide backward compatibility. If by using the old syntax the *Mail Merge* field would point to two fields of the same name (i.e. located in two different groups of the data object) then the field of the first group (based on the sort index) will be used.
Additionally a *Word* template is now not only restricted by queue but also bound to one data object definition. A new field *Data object* has been introduced on the page *Manage Word templates*. In the following example one has to choose one of the available data objects *Company* or *customer*:



1.4 Known Issues

1.4.1 E-Mail encryption: Problem during import of certificates (#622697)

This issue occurs for a packaged EAR only for JBoss because of a bug in the VFS. The issue is described here: https://issues.jboss.org/browse/JBAS-7882. It was not fixed and probably will be not as there is no source code for the JCE – the part of Java, which is responsible for security providers.
At least a workaround exists: expand the CM6 EAR structure in the server deploy directory.
It is only necessary when encrypted e-mails functionality must be used.
To avoid any problems under WebLogic please use the JRockit JVM (with JCE Unlimited Strength Policy Files installed).

1.5 Bugs fixed

Number	Description
622306	(CM #157417) `mlaService.getAssignedMla(ticket, mlaFieldName)` cannot differ identical CustomField names from different CustomField groups
622429	(CM #157731) Deleting an Enum value no longer causes an NPE when it is used as a static criterion
621348	Logging of incoming mails is now written to mail.log instead of server.log
622496	CM6 Admin Tool: Exception when mapping Enum values no longer occurs
622678	CM6 Admin Tool: Enum works now for bigger number of tickets
622650	DWH – problems with Live mode solved
622401	WebClient in IE9 – random issue with showing css code instead of login page solved
622338	ClientAbortException no longer results in "Header already written exception"
611957	Search in the attachment sections (of tickets and units) should be case insensitive
622705	DWH Update now works again after a database shutdown
622021	Missing FK constraints after update
622023	Primary Key inconsistency after update
622026	Foreign key constraint names differ after update
623244	Rich Text Editor: *Heading 4* style overwritten in history
623260	Workflow: NonUniqueResultException when two scopes with different capitalization
623274	Session handling improvements
623252	(CM #159163) CM-Admin role causes NPE DefaultViewVoService.getViewsForCurrentUser
622943	*DetailSearch* → tab *Companies*: Column order is not saved after the first change
623031	Large fonts are not displayed correctly in ticket history
621908	Old, inactive workflow engine threads are now removed on engine restart
623456	Autocomplete enums show options after clicking on it
623467	Validation on ACF does not work for custom field in struct
623480	CM6 Admin Tool: User not moved to home page after session timeout
623596	Rest API - total of elements needed for ticket search
623652	*CustomFieldsHelper* fix
623675	History entry of autocomplete enum is shown on visibility level *Basic*
623689	Cannot open CM6 Admin Tool with Java 1.7.0_45
623700	Security warning with Java 1.7.0_45

2 Version 6.9.1.1 (20.11.2013)

Version 6.9.1.1 includes 6.8 versions up to 6.8.5.5 and 6.7 versions up to 6.7.13

2.1 Update and installation instructions

No further instructions available.

2.2 Bugs fixed

Number	Description
623883	`NoViableAltException` when trying to submit an ACF
623867	NPE at `TicketServiceImpl.createWithUnit(TicketServiceImpl.java:776)`
623890	[RestAPI] Sorting tickets/unit's via custom field type *String* and *ShortString* does not work

3 Version 6.9.1.2 (19.12.2013)

Version 6.9.1.2 includes 6.8 versions up to 6.8.5.6 (6.8.4 versions up to 6.8.4.5) and 6.7 versions up to 6.7.13

3.1 Update and installation instructions

No further instructions available.

3.2 Bugs fixed

Number	Description
623996	Update from 6.8.5.4 to 6.9.1.1 fails: NPE when executing script: templates_1 for version: 6.9.0.0
623981	Web client templates: references to data object custom fields are now correct after an update to 6.9
624021	`ContactAuthenticationToken` - `pUnitType` never evaluated
624036	Manual unit action with `CREATE_TICKET` post action does not work
624071	Request time measurement extensions: Wrong operation and/or network time
624084	REST API: sorting by scope when getting ticket lists (backport of 623592)
624084	REST API: Customer data model templates (backport of 623919)
624121	Export of data object condition scripts does not work: `UnitCondition` enum value is missing in `ScriptTypeTo`