OTRS Administration Manual

Release 2024.6.1

OTRS AG

Sep 06, 2024

CONTENTS

CHAPTER

ONE

INTRODUCTION

This manual is intended for use by OTRS administrators. The chapters describe the administration of the OTRS software. The administrator interface is available in the *Admin* menu item of the main menu, when you logged in as an administrator. Administrators are users, who are member of the *admin* group.

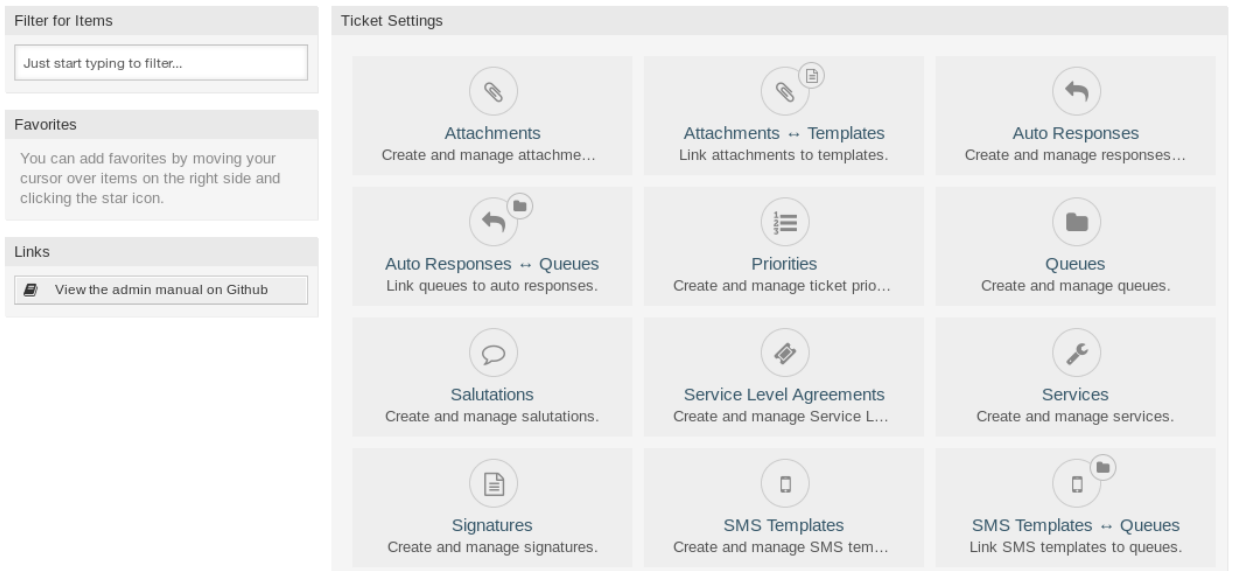

Filter for Items
Just start typing to filter...
Favorites
You can add favorites by moving your cursor over items on the right side and clicking the star icon.
Links
View the admin manual on Github
Ticket Settings
Attachments
Create and manage attachme...
Attachments ↔ Templates
Link attachments to templates.
Auto Responses
Create and manage responses...
Auto Responses ↔ Queues
Link queues to auto responses.
Priorities
Create and manage ticket prio...
Queues
Create and manage queues.
Salutations
Create and manage salutations.
Service Level Agreements
Create and manage Service L...
Services
Create and manage services.
Signatures
Create and manage signatures.
SMS Templates
Create and manage SMS tem...
SMS Templates ↔ Queues
Link SMS templates to queues.

Fig. 1: Administrator Interface

The administrator interface contains several modules collected into groups. Use the filter box in the left sidebar to find a particular module by just typing the name to filter.

This manual shows you the configuration possibilities needed to solve common problems. The chapters:

1. Identify a typical use-case for the administrator, to aid in orientation, and explain **what** OTRS does to provide a solution (**warranty**).
2. Direct you **how** to configure OTRS to fit your use-case (**utility**).

The chapters are the same as the modules in the administrator interface. The order of the chapters are also the same as they are displayed alphabetically in the (English) administrator interface. However, the steps to do to configure a fresh new OTRS installation is different from this order. We recommend to configure OTRS to do the steps as explained below.

1.1 Getting Started

Note: **OTRS** is installed by the *Customer Solutions Team*. Please contact us via support@otrs.com or in the OTRS Portal.

A fresh new OTRS installation contains only the default settings. You need to customize your system to work properly and meet your needs.

First, you need to check some *System Configuration* and modify the values, if needed. FQDN, SystemID and SendmailModule::Host are the most important. Defining the TimeWorkingHours setting for working hours and the TimeVacationDays setting for public holidays are also needed to calculate the escalation times correctly in OTRS.

Then, open the *PostMaster Mail Accounts* module to add email addresses used by the system. For sending email, you can add more *Email Addresses*.

To improve the security, *PGP Keys* or *S/MIME Certificates* can be used for encryption of emails.

Let’ s continue with users, but you might need to add some *Groups* and/or *Roles* to the system first. It is recommended to create new groups for each main queues. There are some groups in your OTRS, but no roles are defined by default. You can add roles, if needed, and you can set the *Roles   Groups* relations.

The next step is to add *Agents* to the system and set the *Agents   Groups* and/or *Agents   Roles* relations.

Now you can add *Customers* and *Customer Users*. Customers are companies and customer users are the employees of the company.

Note: Both agents and customer users can log in using Active Directory or LDAP for authentication. In these cases doesn’ t need to add users manually.

Like for agents, customers users can also assign to groups using the *Customer Users   Groups* management screen. Similarly, *Customer Users   Customers* relations can also be set.

Your OTRS installation already contains a standard salutation and a standard signature, but you might need to edit them or create new *Salutations* and *Signatures*. Queues can not be created without salutations and signatures, and only one salutation and signature can be assigned to a queue.

After system addresses, salutations, signatures, groups are set, you can continue the setup with *Queues*. Each queue has to assign to a group, and only the group members can see the tickets in the assigned queue.

Now you can add *Auto Responses* and assign them to queues using the *Queues   Auto Responses* management screen. Your OTRS installation already contains some automatic responses, you can use or edit them instead of create new ones.

To reduce the time needed for answering the tickets, *Templates* or *SMS Templates* can be created.

Normal templates can contain *Attachments*, and you can assign the uploaded attachments to templates using the *Templates   Attachments* management screen.

When templates are created, you can set the templates to use in queues in the *Templates   Queues* or *SMS Templates   Queues* management screens.

You need to review the default *Priorities*, *States* and *Types*, and add new elements, if needed.

The customer requests can be categorize into services. If you would like to use this possibility, then create some *Services* and set the *Customer Users   Services* relations. Furthermore, *Service Level Agreements* can be assign to the services.

Now you can add some notifications to be received by agents, if particular events occur. You can do this in the *Ticket Notifications* screen.

To help agents to organize appointments, you can setup the *Calendars* and the *Appointment Notifications*.

Tickets, articles and other objects in OTRS can be extended with *Dynamic Fields* or can be reduced with *Access Control Lists (ACL)*.

Without doing everything manually, the number of failure can be reduced. Automatize some process in OTRS using *Generic Agent* jobs or creating processes with *Process Management*. The incoming emails can be pre-processed and dispatched automatically by defining some *Postmaster Filters*.

If external systems need to integrate with OTRS, *Web Services* will be very useful for this.

However OTRS has many features by default, you can extend the functionality by installing packages with the *Package Manager*.

If your system is ready for productive work, don' t forget to register it by using the *System Registration* procedure.

Finally, you can set the *Home Page*, the *Custom Pages* and the *Layout* of the external interface, as well as you can define a *Customer Service Catalogue* displayed in the external interface.

1.2 Become OTRS Expert

The next chapters of this manual describe the features and configuration settings of OTRS more detailed. There is a separated manual for Configuration Options References, that gives you a good overview of *System Configuration*, that can be modify the behavior of OTRS.

CHAPTER

TWO

TICKET SETTINGS

A *ticket* is similar to a medical report created for a hospital patient. When a patient first visits the hospital, a medical report is created to hold all necessary personal and medical information on him. Over multiple visits, as he is attended to by the same or additional doctors, the attending doctor updates the report by adding new information on the patient's health and the ongoing treatment. This allows any other doctors or the nursing staff to get a complete picture on the case at hand. When the patient recovers and leaves the hospital, all information from the medical report is archived and the report is closed.

Ticket systems such as OTRS handle tickets like normal emails. The messages are saved in the system. When a customer sends a request, a new ticket is generated by the system which is comparable to a new medical report being created. The response to this new ticket is comparable to a doctor's entry in the medical report. A ticket is closed if an answer is sent back to the customer, or if the ticket is separately closed by the system. If a customer responds again on an already closed ticket, the ticket is reopened with the new information added.

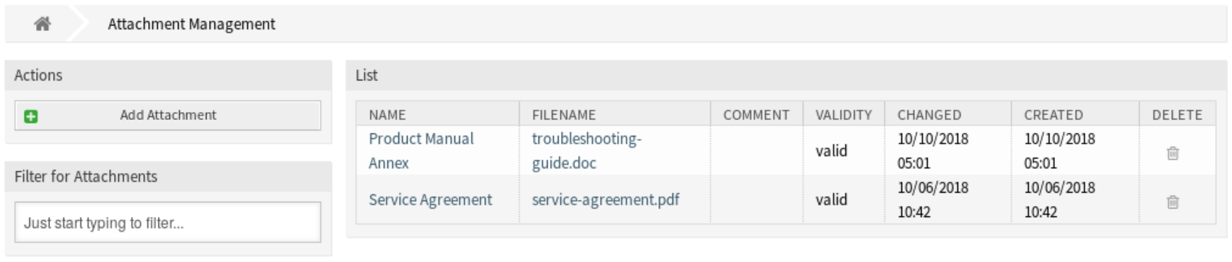
Every ticket is stored and archived with complete information. Since tickets are handled like normal emails, attachments and contextual annotations will also be stored with each email. In addition, information on relevant dates, employees involved, working time needed for ticket resolution, etc. are also saved. At any later stage, tickets can be sorted, and it is possible to search through and analyze all information using different filtering mechanisms.

2.1 Attachments

For any size of organization it is often required to send a service agreement, the terms of service or a privacy statement out when a customer signs a contract.

OTRS can handle an infinite number of attachments (PDF, image, etc.) and can bundle them into templates. Your agents don't need to maintain the attachments on their own, nor don't they need to upload them again and again - they can just use the predefined templates.

Use this screen to add attachments for use in templates. A fresh OTRS installation doesn't contain any attachments by default. The attachment management screen is available in the *Attachments* module of the *Ticket Settings* group.



NAME	FILENAME	COMMENT	VALIDITY	CHANGED	CREATED	DELETE
Product Manual Annex	troubleshooting-guide.doc		valid	10/10/2018 05:01	10/10/2018 05:01	
Service Agreement	service-agreement.pdf		valid	10/06/2018 10:42	10/06/2018 10:42	

Fig. 1: Attachment Management Screen

2.1.1 Manage Attachments

To add an attachment:

1. Click on the *Add Attachments* button in the left sidebar.
2. Fill in the required fields.
3. Click on the *Save* button.

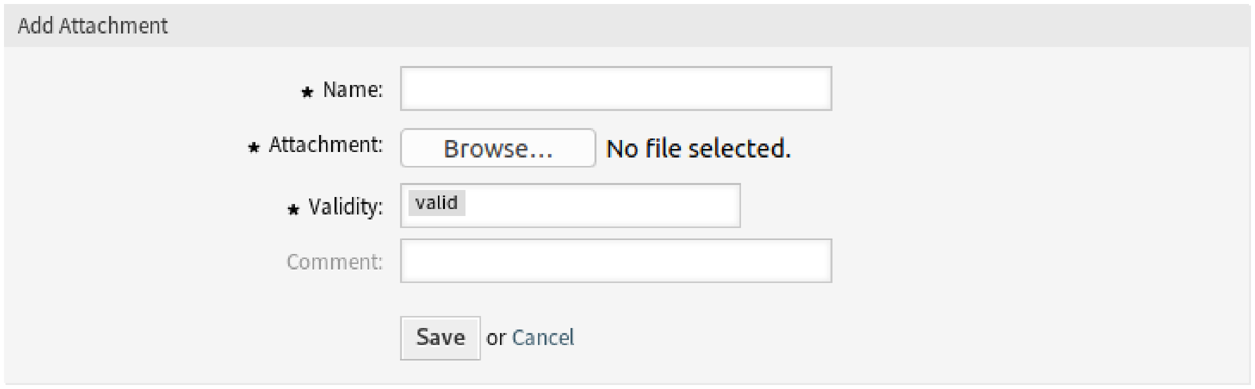


Fig. 2: Add Attachment Screen

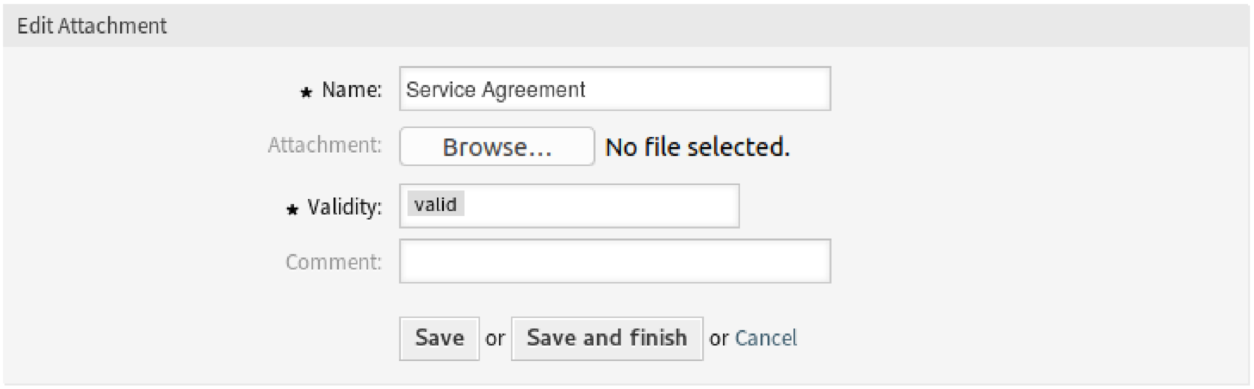
To edit an attachment:

1. Click on an attachment in the list of attachments.
2. Modify the fields.
3. Click on the *Save* or *Save and finish* button.

To delete an attachment:

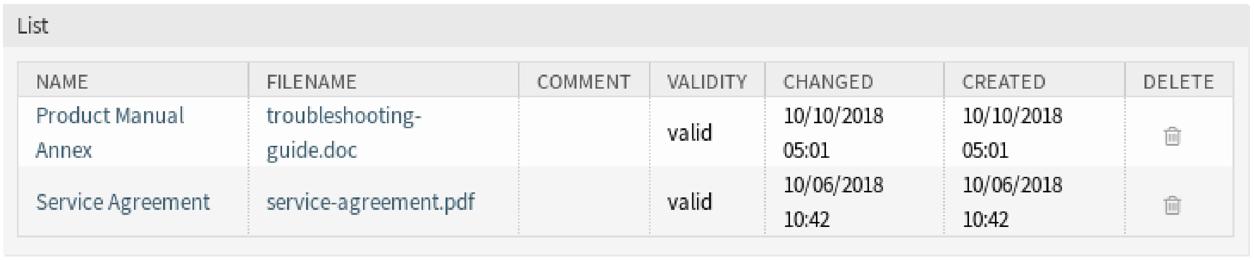
1. Click on the trash icon in the last column of the overview table.
2. Click on the *Confirm* button.

Note: If several attachments are added to the system, use the filter box to find a particular attachment by just typing the name to filter.



⋆ Name:	Service Agreement
Attachment:	No file selected.
⋆ Validity:	valid
Comment:	

Fig. 3: Edit Attachment Screen



NAME	FILENAME	COMMENT	VALIDITY	CHANGED	CREATED	DELETE
Product Manual Annex	troubleshooting-guide.doc		valid	10/10/2018 05:01	10/10/2018 05:01	🗑
Service Agreement	service-agreement.pdf		valid	10/06/2018 10:42	10/06/2018 10:42	🗑

Fig. 4: Delete Attachment Screen

2.1.2 Attachment Settings

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

Name *
: The name of this resource. Any type of characters can be entered to this field including uppercase letters and spaces. The name will be displayed in the overview table.

Attachment *
: Open the file dialog to add a file from the file system. This field is mandatory in the attachment add screen, but optional in the attachment edit screen. Adding a new file in the edit screen will overwrite the existing attachment.

Validity *
: Set the validity of this resource. Each resource can be used in OTRS only, if this field is set to *valid*. Setting this field to *invalid* or *invalid-temporarily* will disable the use of the resource.

Comment
: Add additional information to this resource. It is recommended to always fill this field as a description of the resource with a full sentence for better clarity, because the comment will be also displayed in the overview table.

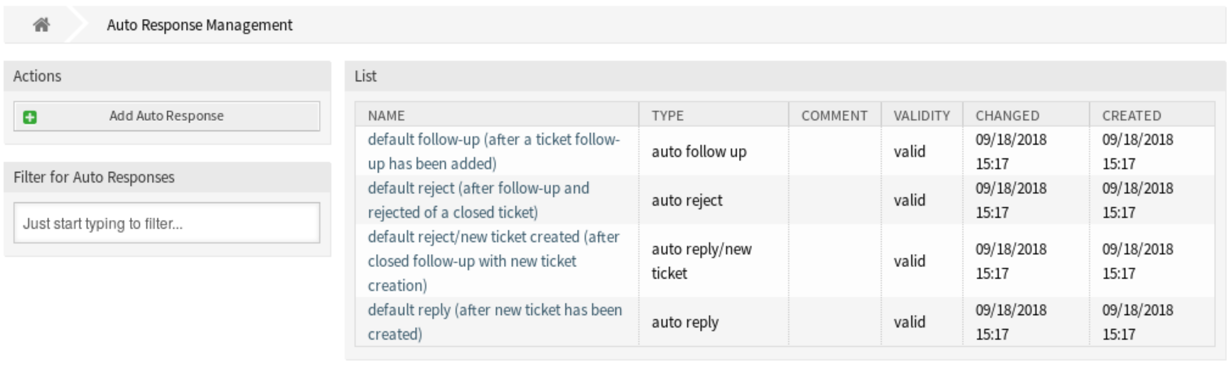
2.2 Auto Responses

Quick and transparent service is vital to maintaining a good working relationship with your customer. Email, fax, social media and other non-real-time communication are patient, but you want to engage your customer immediately upon receipt of a request.

OTRS allows you to respond to a customer immediately upon receipt of a request giving the customers instantaneous feedback assuring them that their request is in processing, establishing expectation.

Automatic responses can be sent to customers based on the occurrence of certain events, such as the creation of a ticket in a specific queue, the receipt of a follow-up message in regards to a closed or rejected ticket, etc.

Use this screen to add automatic responses for use in queues. A fresh OTRS installation contains some automatic responses by default. The automatic response management screen is available in the *Auto Responses* module of the *Ticket Settings* group.



NAME	TYPE	COMMENT	VALIDITY	CHANGED	CREATED
default follow-up (after a ticket follow-up has been added)	auto follow up		valid	09/18/2018 15:17	09/18/2018 15:17
default reject (after follow-up and rejected of a closed ticket)	auto reject		valid	09/18/2018 15:17	09/18/2018 15:17
default reject/new ticket created (after closed follow-up with new ticket creation)	auto reply/new ticket		valid	09/18/2018 15:17	09/18/2018 15:17
default reply (after new ticket has been created)	auto reply		valid	09/18/2018 15:17	09/18/2018 15:17

Fig. 5: Auto Response Management Screen

2.2.1 Manage Auto Responses

Note: Adding automatic responses requires a valid system address. Create system addresses in the *Email Addresses* screen.

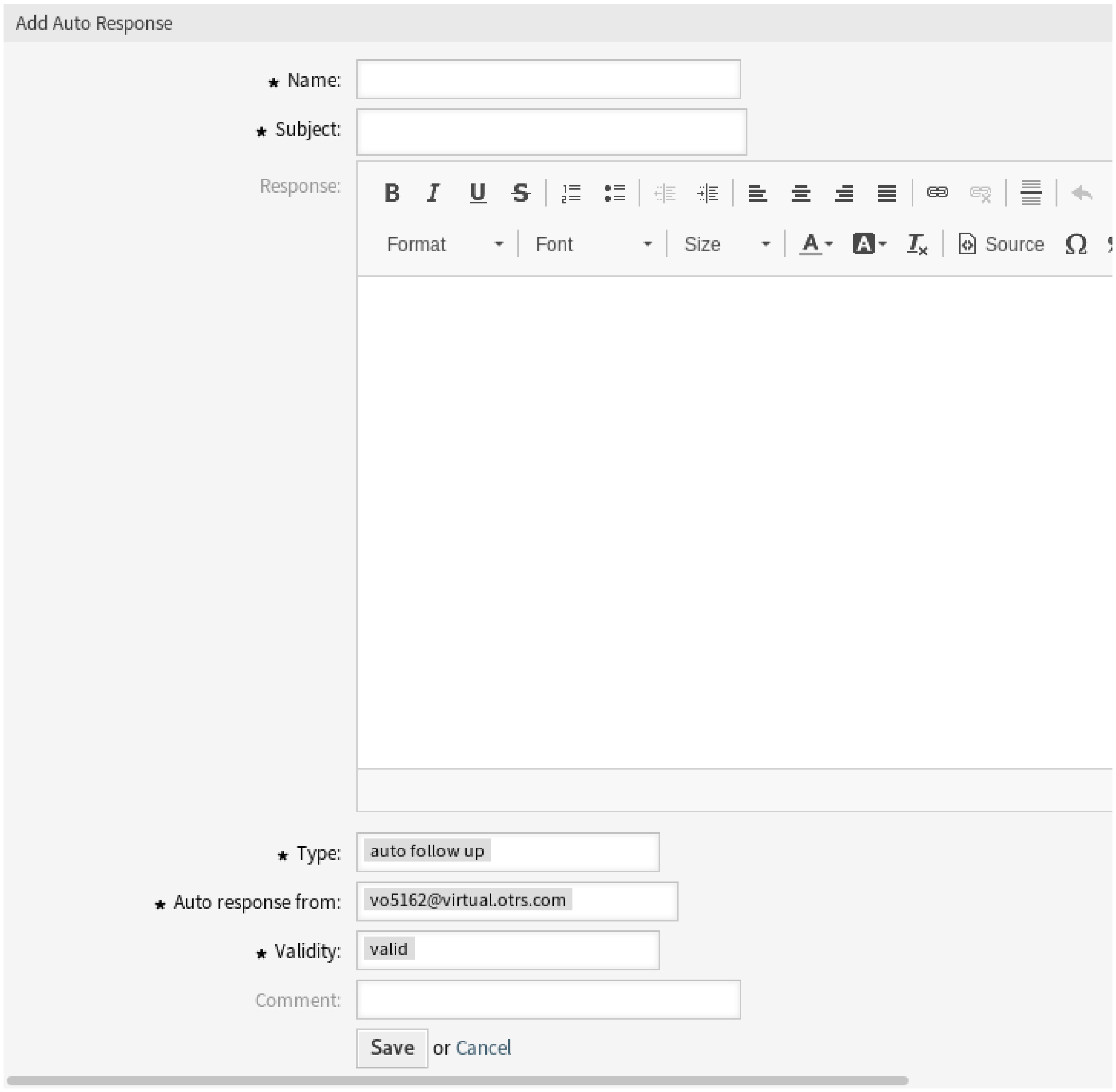
To add an automatic response:

1. Click on the *Add Auto Response* button in the left sidebar.
2. Fill in the required fields.
3. Click on the *Save* button.

Warning: Auto responses can not be deleted from the system. They can only be deactivated by setting the *Validity* option to *invalid* or *invalid-temporarily*.

To edit an automatic response:

1. Click on an automatic response in the list of automatic responses.



★ Name:

★ Subject:

Response:

★ Type: auto follow up

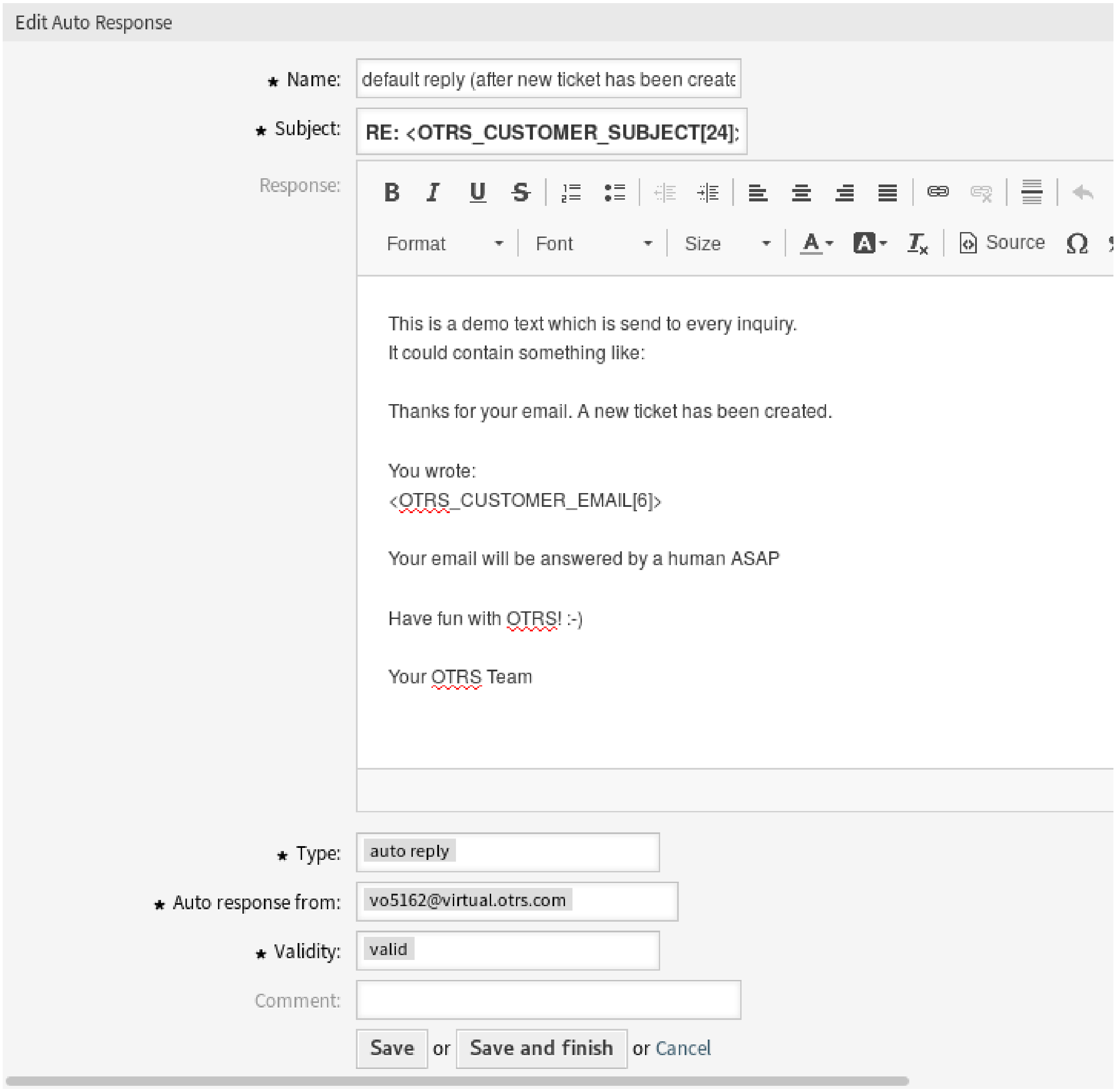
★ Auto response from: vo5162@virtual.otrs.com

★ Validity: valid

Comment:

Fig. 6: Add Auto Response Screen

2. Modify the fields.
3. Click on the *Save* or *Save and finish* button.


Edit Auto Response

⋆ Name:	default reply (after new ticket has been create
⋆ Subject:	RE: <OTRS_CUSTOMER_SUBJECT[24]:
Response:	This is a demo text which is send to every inquiry. It could contain something like: Thanks for your email. A new ticket has been created. You wrote: <OTRS_CUSTOMER_EMAIL[6]> Your email will be answered by a human ASAP Have fun with OTRS! :-) Your OTRS Team
⋆ Type:	auto reply
⋆ Auto response from:	vo5162@virtual.otrs.com
⋆ Validity:	valid
Comment:	

Fig. 7: Edit Auto Response Screen

Note: If several automatic responses are added to the system, use the filter box to find a particular automatic response by just typing the name to filter.

2.2.2 Auto Response Settings

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

Name *
The name of this resource. Any type of characters can be entered to this field including uppercase letters and spaces. The name will be displayed in the overview table.

Subject *
The subject of the email sent to the users.

Response
The body of the email sent to the users.

Type *
The event type that triggers the sending of this automatic response. Only one automatic response can be sent automatically. The following event types are available:

auto follow up
Confirms receipt of the follow-up.

auto reject
The message rejects a customer follow-up.

auto remove
Deletion of a ticket, done by the system.

auto reply
A newly raised ticket will trigger this auto response.

auto reply/new ticket
This message informs the customer of the new ticket number.

Auto response from *
The sender email address, from which the automatic response will be sent.

Validity *
Set the validity of this resource. Each resource can be used in OTRS only, if this field is set to *valid*. Setting this field to *invalid* or *invalid-temporarily* will disable the use of the resource.

Comment
Add additional information to this resource. It is recommended to always fill this field as a description of the resource with a full sentence for better clarity, because the comment will be also displayed in the overview table.

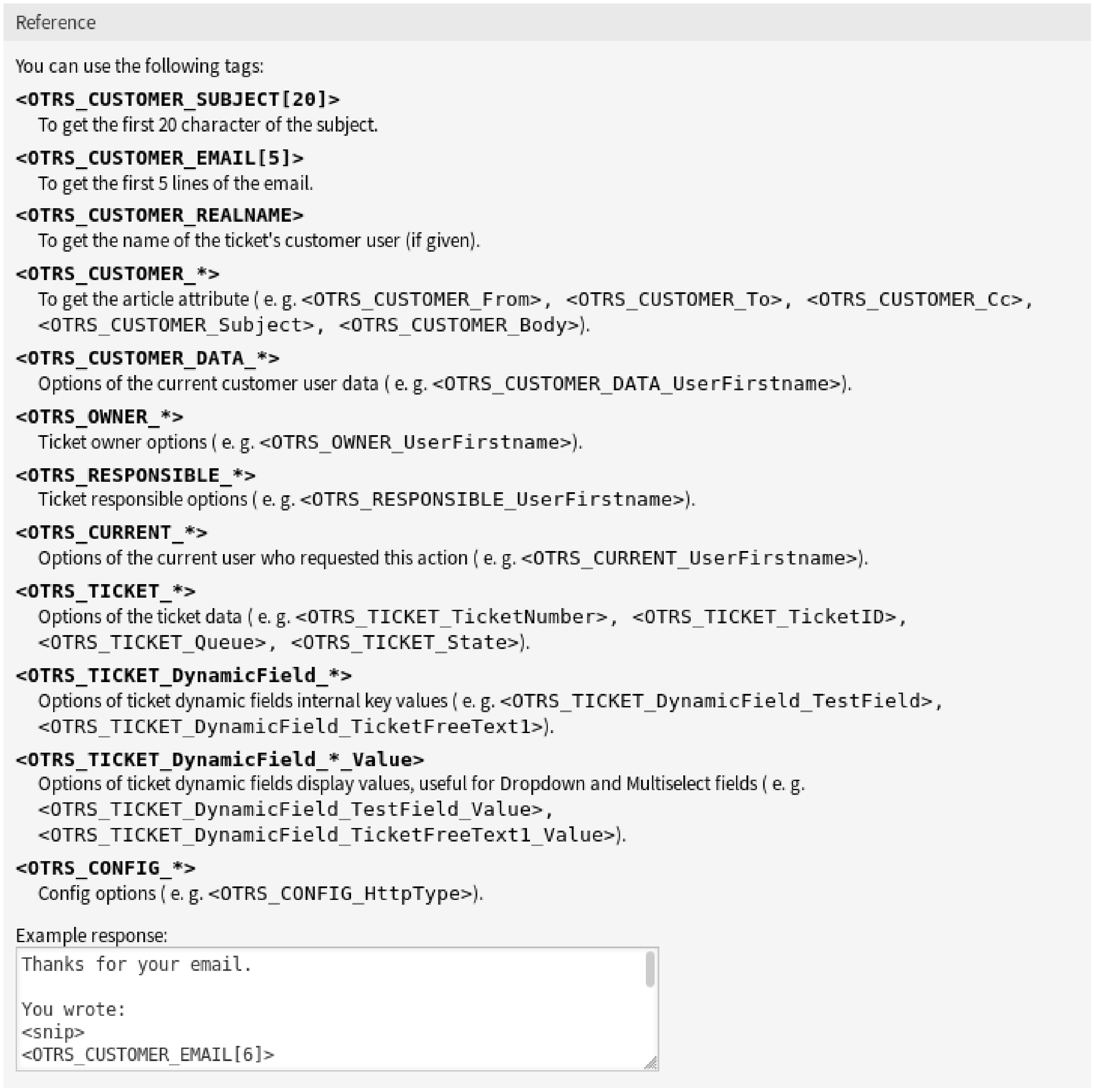
2.2.3 Auto Response Variables

Using variables in the text makes it possible to personalize messages. Variables, known as OTRS tags, are replaced by OTRS when generating the message. Find a list of available tags stems for this resource at the bottom of both add and edit screens.

For example, the variable `<OTRS_TICKET_TicketNumber>` expands to the ticket number allowing a template to include something like the following.

```
Ticket#<OTRS_TICKET_TicketNumber>
```

This tag expands, for example to:



Reference

You can use the following tags:

<OTRS_CUSTOMER_SUBJECT[20]>
To get the first 20 character of the subject.

<OTRS_CUSTOMER_EMAIL[5]>
To get the first 5 lines of the email.

<OTRS_CUSTOMER_REALNAME>
To get the name of the ticket's customer user (if given).

<OTRS_CUSTOMER_*>
To get the article attribute (e. g. `<OTRS_CUSTOMER_From>, <OTRS_CUSTOMER_To>, <OTRS_CUSTOMER_Cc>, <OTRS_CUSTOMER_Subject>, <OTRS_CUSTOMER_Body>`).

<OTRS_CUSTOMER_DATA_*>
Options of the current customer user data (e. g. `<OTRS_CUSTOMER_DATA_UserFirstname>`).

<OTRS_OWNER_*>
Ticket owner options (e. g. `<OTRS_OWNER_UserFirstname>`).

<OTRS_RESPONSIBLE_*>
Ticket responsible options (e. g. `<OTRS_RESPONSIBLE_UserFirstname>`).

<OTRS_CURRENT_*>
Options of the current user who requested this action (e. g. `<OTRS_CURRENT_UserFirstname>`).

<OTRS_TICKET_*>
Options of the ticket data (e. g. `<OTRS_TICKET_TicketNumber>, <OTRS_TICKET_TicketID>, <OTRS_TICKET_Queue>, <OTRS_TICKET_State>`).

<OTRS_TICKET_DynamicField_*>
Options of ticket dynamic fields internal key values (e. g. `<OTRS_TICKET_DynamicField_TestField>, <OTRS_TICKET_DynamicField_TicketFreeText1>`).

<OTRS_TICKET_DynamicField_*_Value>
Options of ticket dynamic fields display values, useful for Dropdown and Multiselect fields (e. g. `<OTRS_TICKET_DynamicField_TestField_Value>, <OTRS_TICKET_DynamicField_TicketFreeText1_Value>`).

<OTRS_CONFIG_*>
Config options (e. g. `<OTRS_CONFIG_HttpType>`).

Example response:

```
Thanks for your email.

You wrote:
<snip>
<OTRS_CUSTOMER_EMAIL[6]>
```

Fig. 8: Auto Response Variables

```
Ticket#2018101042000012
```

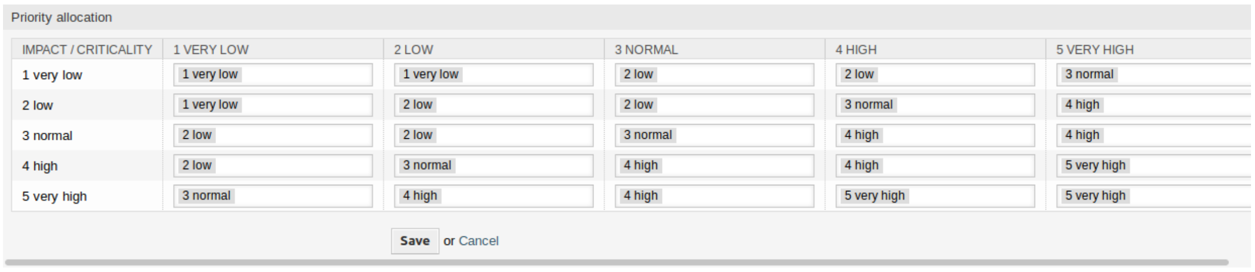
The values of the following variables are translated based on the chosen language of the customer user.

```
<OTRS_TICKET_Type>
<OTRS_TICKET_State>
<OTRS_TICKET_StateType>
<OTRS_TICKET_Lock>
<OTRS_TICKET_Priority>
```

If the language is not supported the default language is applied.

2.3 Criticality   Impact   Priority

Use this screen to manage the criticality   impact   priority matrix. The management screen is available in the *Criticality   Impact   Priority* module of the *Ticket Settings* group.



Priority allocation

IMPACT / CRITICALITY	1 VERY LOW	2 LOW	3 NORMAL	4 HIGH	5 VERY HIGH
1 very low	1 very low	1 very low	2 low	2 low	3 normal
2 low	1 very low	2 low	2 low	3 normal	4 high
3 normal	2 low	2 low	3 normal	4 high	4 high
4 high	2 low	3 normal	4 high	4 high	5 very high
5 very high	3 normal	4 high	4 high	5 very high	5 very high

Fig. 9: Criticality   Impact   Priority Screen

2.4 Dynamic Ticket Templates

Use this screen to add dynamic ticket templates for use in communications. The dynamic ticket template management screen is available in the *Dynamic Ticket Templates* module of the *Ticket Settings* group.

2.4.1 Manage Dynamic Ticket Templates

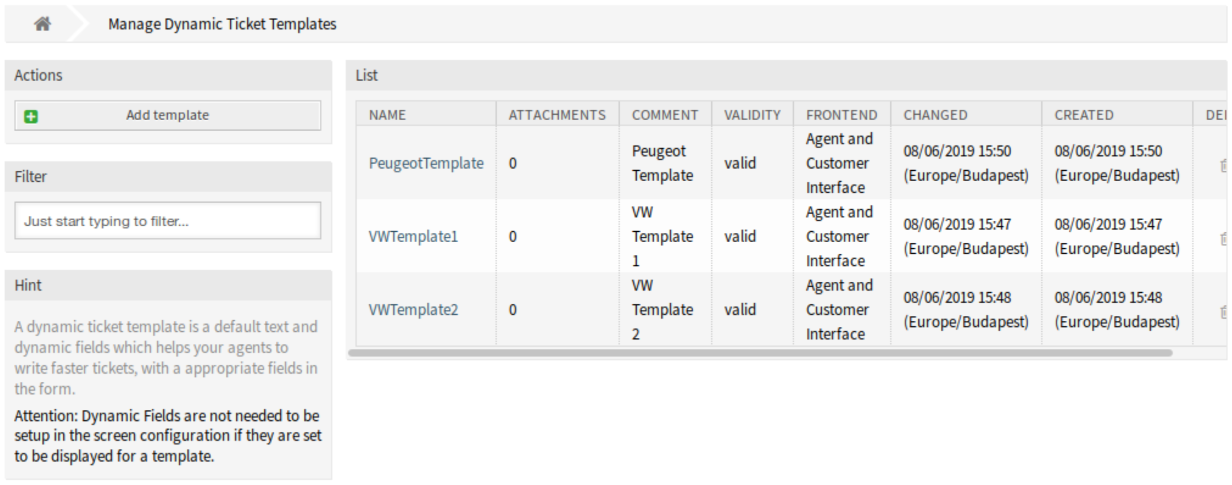
Note: To add attachments to a dynamic ticket template, it needs to create the attachment first in the *Attachment Management* screen.

To add a dynamic ticket template:

1. Click on the *Add Template* button in the left sidebar.
2. Fill in the required fields.
3. Click on the *Save* button.

To edit a dynamic ticket template:

1. Click on a dynamic ticket template in the list of dynamic ticket templates.



Manage Dynamic Ticket Templates

Actions

Add template

Filter

Just start typing to filter...

Hint

A dynamic ticket template is a default text and dynamic fields which helps your agents to write faster tickets, with a appropriate fields in the form.

Attention: Dynamic Fields are not needed to be setup in the screen configuration if they are set to be displayed for a template.

List

NAME	ATTACHMENTS	COMMENT	VALIDITY	FRONTEND	CHANGED	CREATED	DEI
PeugeotTemplate	0	Peugeot Template	valid	Agent and Customer Interface	08/06/2019 15:50 (Europe/Budapest)	08/06/2019 15:50 (Europe/Budapest)	
VWTemplate1	0	VW Template 1	valid	Agent and Customer Interface	08/06/2019 15:47 (Europe/Budapest)	08/06/2019 15:47 (Europe/Budapest)	
VWTemplate2	0	VW Template 2	valid	Agent and Customer Interface	08/06/2019 15:48 (Europe/Budapest)	08/06/2019 15:48 (Europe/Budapest)	

Fig. 10: Dynamic Ticket Templates Management Screen

2. Modify the fields.
3. Click on the *Save* or *Save and finish* button.

To delete a dynamic ticket template:

1. Click on the trash icon in the list of dynamic ticket templates.
2. Click on the *Confirm* button.

Note: If several dynamic ticket templates are added to the system, use the filter box to find a particular dynamic ticket template by just typing to filter.

2.4.2 Dynamic Ticket Template Settings

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

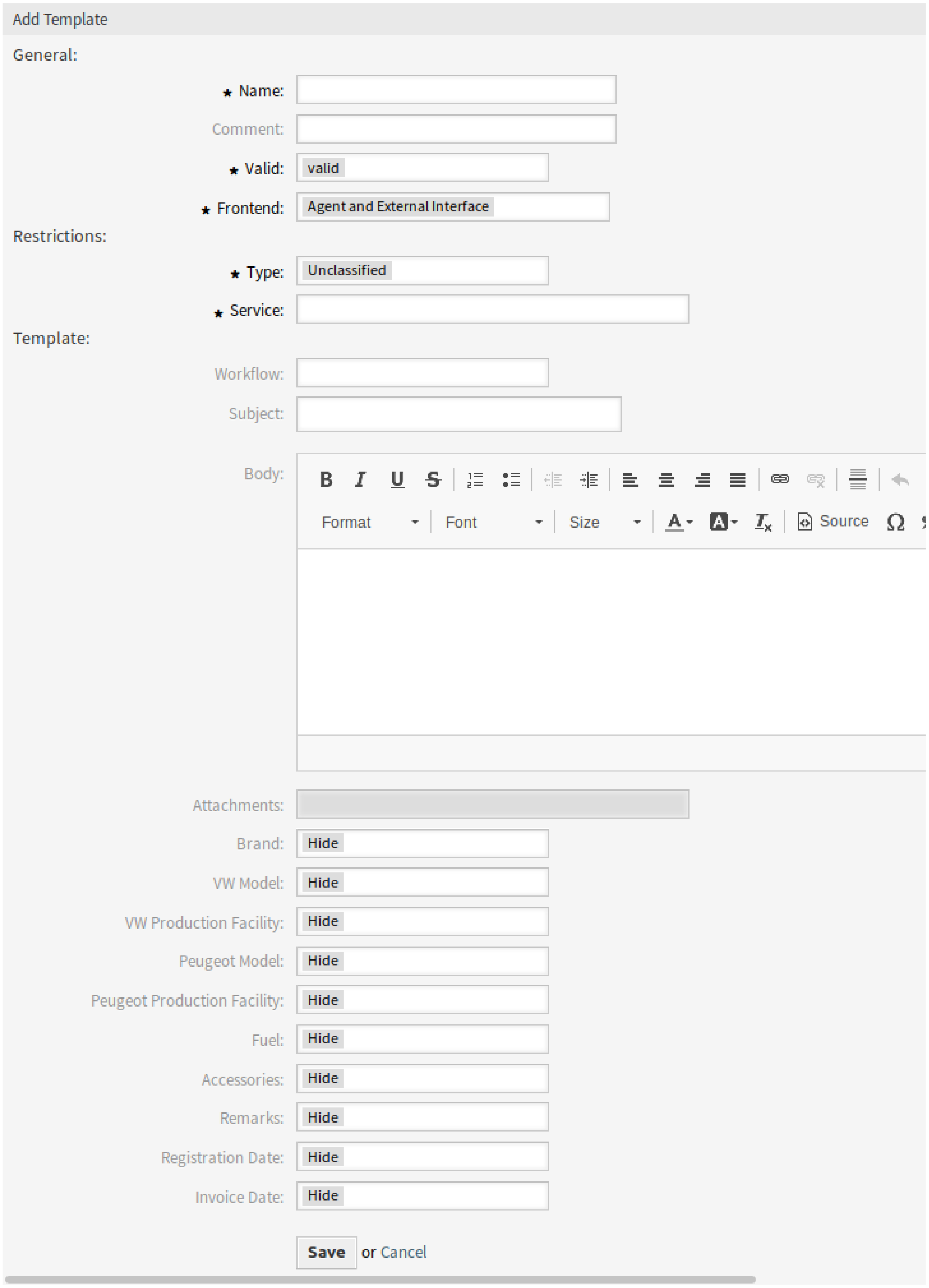
Name *
The name of this resource. Any type of characters can be entered to this field including uppercase letters and spaces. The name will be displayed in the overview table.

Comment
Add additional information to this resource. It is recommended to always fill this field as a description of the resource with a full sentence for better clarity, because the comment will be also displayed in the overview table.

Valid *
Set the validity of this resource. Each resource can be used in OTRS only, if this field is set to *valid*. Setting this field to *invalid* or *invalid-temporarily* will disable the use of the resource.

Frontend *
Defines, where can the dynamic ticket template be used. A dynamic ticket template can be used in the following interfaces:

- Agent Interface


Add Template

General:

⋆ Name:	
Comment:	
⋆ Valid:	valid
⋆ Frontend:	Agent and External Interface

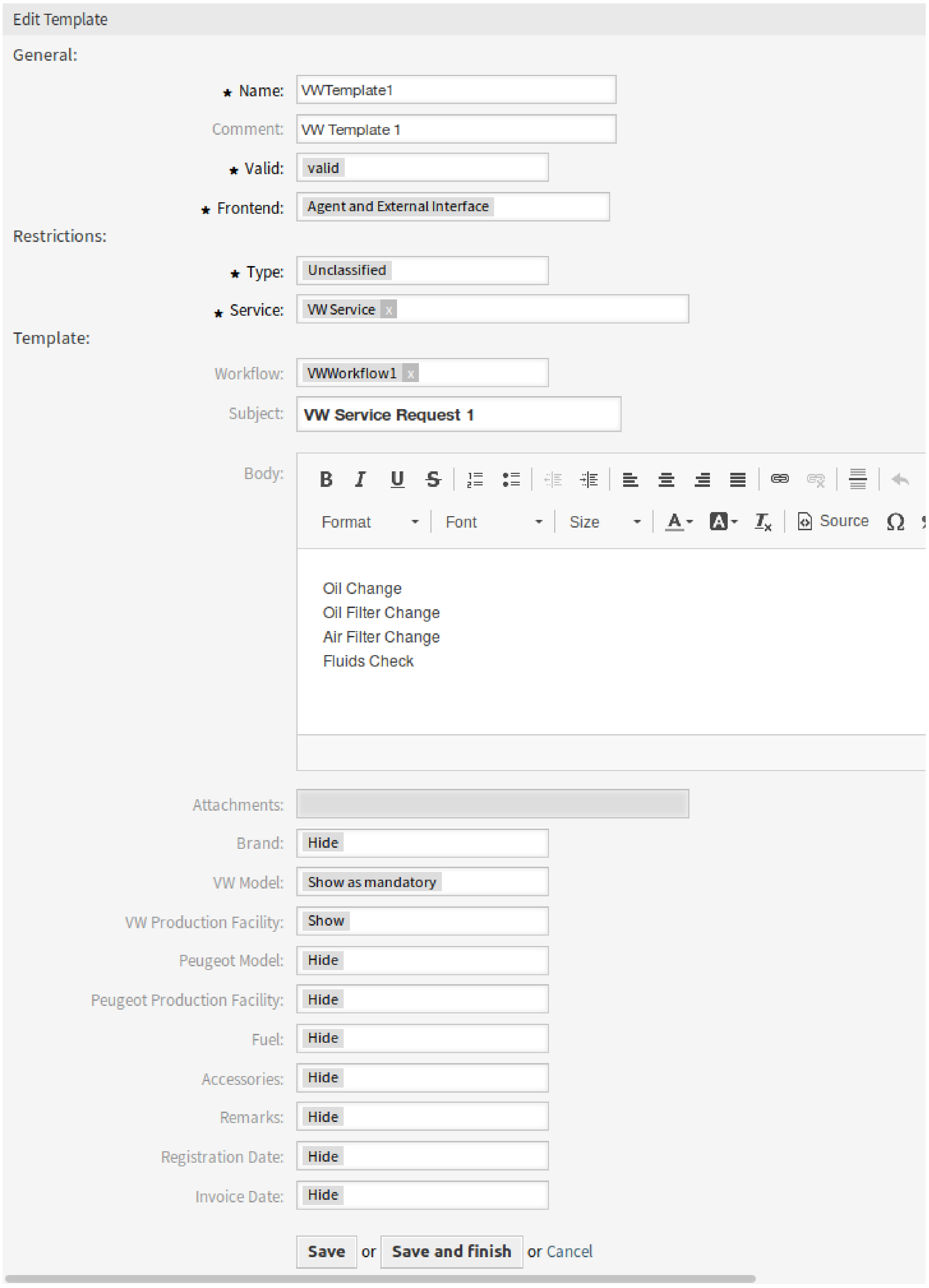
Restrictions:

⋆ Type:	Unclassified
⋆ Service:	

Template:

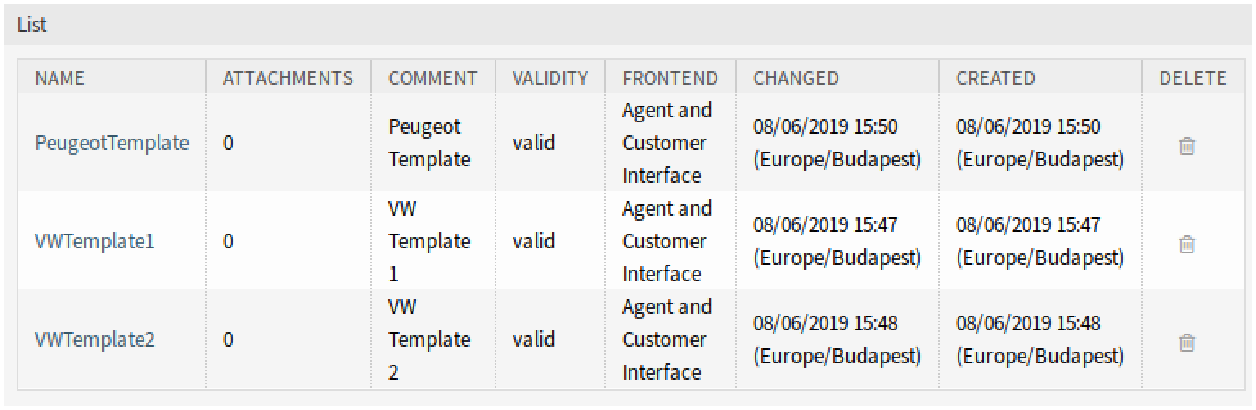
Workflow:	
Subject:	
Body:	
Attachments:	
Brand:	Hide
VW Model:	Hide
VW Production Facility:	Hide
Peugeot Model:	Hide
Peugeot Production Facility:	Hide
Fuel:	Hide
Accessories:	Hide
Remarks:	Hide
Registration Date:	Hide
Invoice Date:	Hide

Fig. 11: Add Dynamic Ticket Template Screen



★ Name:	VWTemplate1
Comment:	VW Template 1
★ Valid:	valid
★ Frontend:	Agent and External Interface
★ Type:	Unclassified
★ Service:	VW Service
Workflow:	VWWorkflow1
Subject:	VW Service Request 1
Body:	Oil Change Oil Filter Change Air Filter Change Fluids Check
Attachments:	
Brand:	Hide
VW Model:	Show as mandatory
VW Production Facility:	Show
Peugeot Model:	Hide
Peugeot Production Facility:	Hide
Fuel:	Hide
Accessories:	Hide
Remarks:	Hide
Registration Date:	Hide
Invoice Date:	Hide

Fig. 12: Edit Dynamic Ticket Template Screen



List

NAME	ATTACHMENTS	COMMENT	VALIDITY	FRONTEND	CHANGED	CREATED	DELETE
PeugeotTemplate	0	Peugeot Template	valid	Agent and Customer Interface	08/06/2019 15:50 (Europe/Budapest)	08/06/2019 15:50 (Europe/Budapest)	🗑
VWTemplate1	0	VW Template 1	valid	Agent and Customer Interface	08/06/2019 15:47 (Europe/Budapest)	08/06/2019 15:47 (Europe/Budapest)	🗑
VWTemplate2	0	VW Template 2	valid	Agent and Customer Interface	08/06/2019 15:48 (Europe/Budapest)	08/06/2019 15:48 (Europe/Budapest)	🗑

Fig. 13: Delete Dynamic Ticket Template Screen

- Agent and External Interface
- External Interface

Type *
Select a ticket type, that has been added to the system.

Service *
Select a service, that has been added to the system.

Subject
The subject of the email sent to the users.

Body
The body of the email sent to the users.

Attachments
It is possible to add one or more attachments to this template. Attachments can be added in the *Attachment Management* screen.

Dynamic Fields
The list of dynamic fields are displayed in this section. Choose for each dynamic field to *Hide*, *Show* or *Show as mandatory* on ticket create screens.

2.4.3 Dynamic Ticket Template Variables

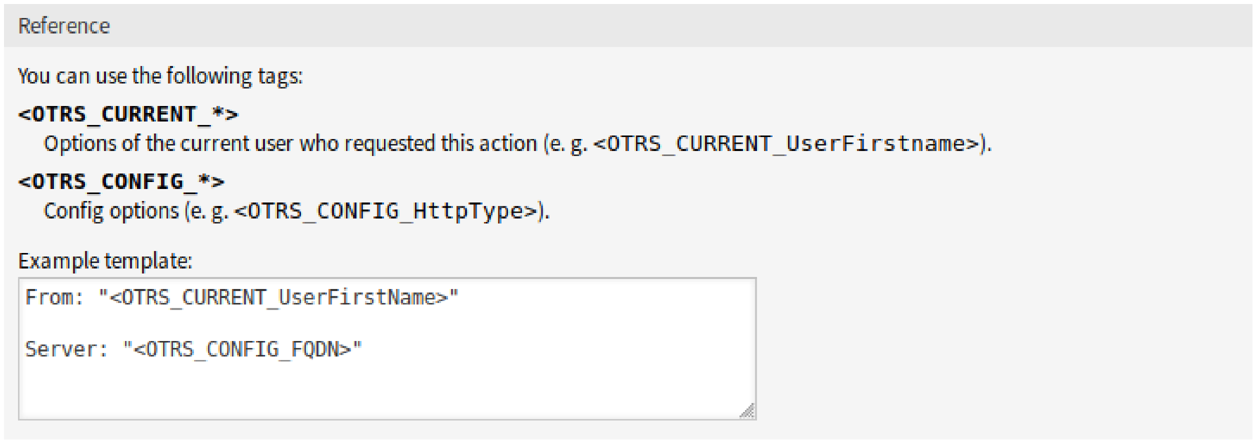
Using variables in the text makes it possible to personalize messages. Variables, known as OTRS tags, are replaced by the system when generating the message. Find a list of available tags for this resource at the bottom of both add and edit screens.

For example, the variable `<OTRS_CURRENT_UserFullname>` expands to the full name of the current user allowing a template to include something like the following.

```
From: <OTRS_CURRENT_UserFullname>
```

This tag expands, for example to:

```
From: John Smith
```



Reference

You can use the following tags:

<OTRS_CURRENT_*>
Options of the current user who requested this action (e. g. <OTRS_CURRENT_UserFirstname>).

<OTRS_CONFIG_*>
Config options (e. g. <OTRS_CONFIG_HttpType>).

Example template:

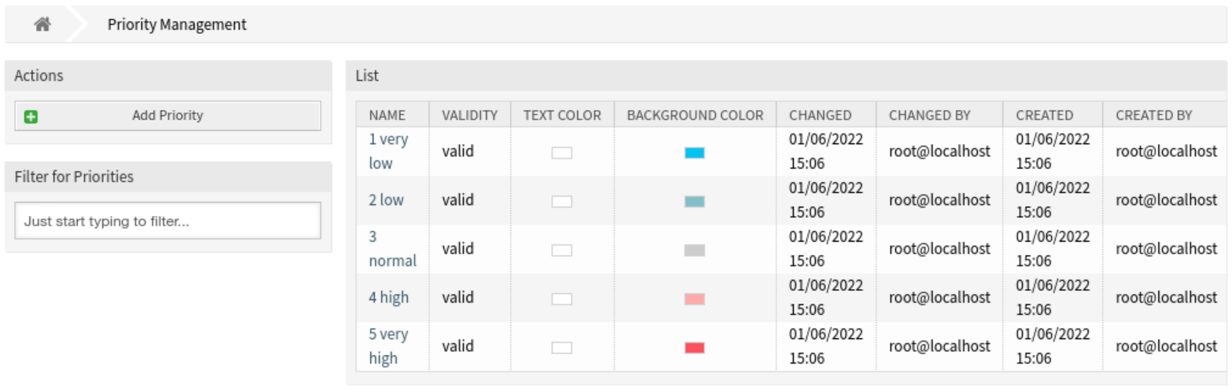
```
From: "<OTRS_CURRENT_UserFirstName>"

Server: "<OTRS_CONFIG_FQDN>"
```

Fig. 14: Dynamic Ticket Template Variables

2.5 Priorities

Sometimes tickets are not equally created. One ticket may need more focus than another. A customer may be given a higher priority by the service desk to help raise customer satisfaction in a pinch or to ensure that a long-running request receives special attention. Keeping track of these higher priority requests is important, as well as handling them quickly.

Use this screen to add priorities to the system. A fresh OTRS installation contains five default priority levels. The priority management screen is available in the *Priorities* module of the *Ticket Settings* group.



NAME	VALIDITY	TEXT COLOR	BACKGROUND COLOR	CHANGED	CHANGED BY	CREATED	CREATED BY
1 very low	valid	⬜	🟦	01/06/2022 15:06	root@localhost	01/06/2022 15:06	root@localhost
2 low	valid	⬜	🟦	01/06/2022 15:06	root@localhost	01/06/2022 15:06	root@localhost
3 normal	valid	⬜	⬜	01/06/2022 15:06	root@localhost	01/06/2022 15:06	root@localhost
4 high	valid	⬜	🟥	01/06/2022 15:06	root@localhost	01/06/2022 15:06	root@localhost
5 very high	valid	⬜	🟥	01/06/2022 15:06	root@localhost	01/06/2022 15:06	root@localhost

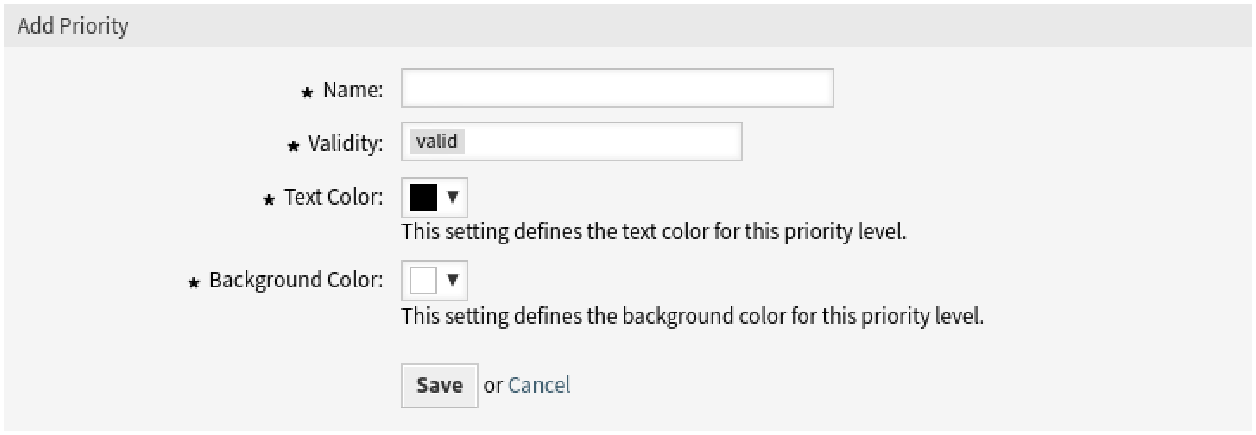
Fig. 15: Priority Management Screen

2.5.1 Manage Priorities

Note: When creating a customized list of priorities, please keep in mind that they are sorted alphabetically in the priority selection box in the user interface.

To add a priority:

1. Click on the *Add Priority* button in the left sidebar.
2. Fill in the required fields.
3. Click on the *Save* button.


Add Priority

★ Name:	
★ Validity:	valid
★ Text Color:	■ ▼ This setting defines the text color for this priority level.
★ Background Color:	□ ▼ This setting defines the background color for this priority level.

Fig. 16: Add Priority Screen

Warning: Priorities can not be deleted from the system. They can only be deactivated by setting the *Validity* option to *invalid* or *invalid-temporarily*.

Note: It’s recommended to limit your system to 5 priorities or less and reuse the current 5 to keep the use of the traffic light system.

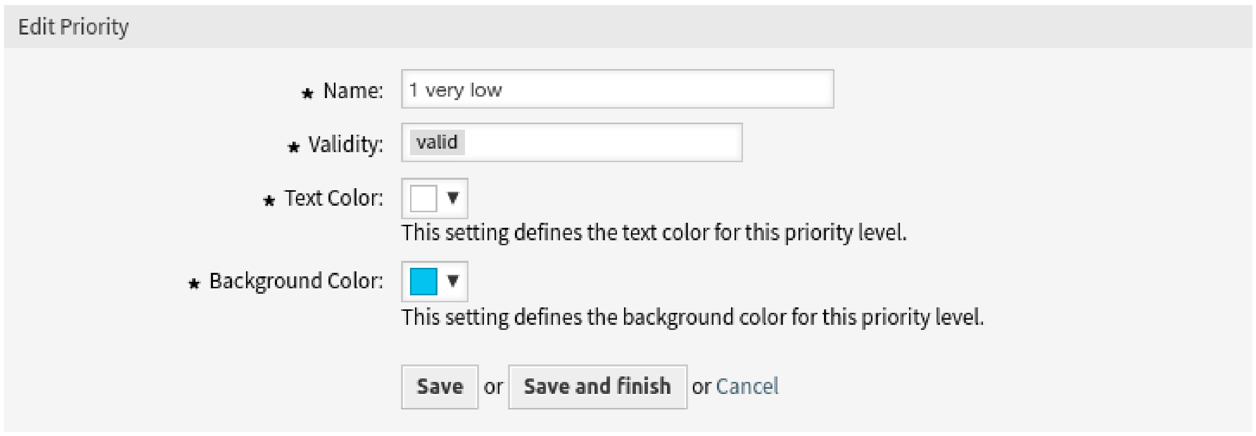
To edit a priority:

1. Click on a priority in the list of priorities.
2. Modify the fields.
3. Click on the *Save* or *Save and finish* button.

Note: If several priorities are added to the system, use the filter box to find a particular priority by just typing the name to filter.

If you change the name of a priority which is used in the system configuration, a validation check will warn you and give you the following options:

Save and update automatically
Apply the change and also update the affected settings.



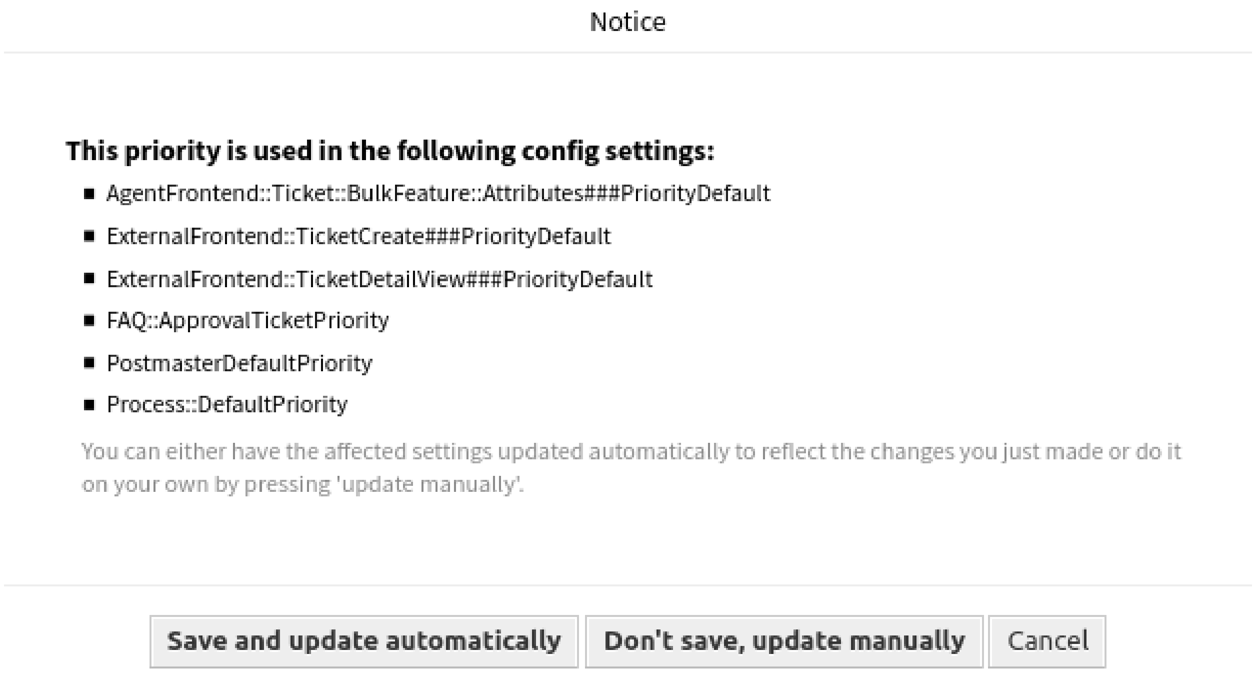
Name: 1 very low

Validity: valid

Text Color: This setting defines the text color for this priority level.

Background Color: This setting defines the background color for this priority level.

Fig. 17: Edit Priority Screen



This priority is used in the following config settings:

- AgentFrontend::Ticket::BulkFeature::Attributes###PriorityDefault
- ExternalFrontend::TicketCreate###PriorityDefault
- ExternalFrontend::TicketDetailView###PriorityDefault
- FAQ::ApprovalTicketPriority
- PostmasterDefaultPriority
- Process::DefaultPriority

You can either have the affected settings updated automatically to reflect the changes you just made or do it on your own by pressing 'update manually'.

Fig. 18: Priority Validation Dialog

Don’t save, update manually
Apply the change, but don’t update the affected settings. The updates need to be done manually.

Cancel
Cancel the action.

Warning: Changing the name of this object should be done with care, the check only provides verification for certain settings and ignores things where the name can’t be verified. Some examples are dashboard filters, access control lists (ACLs), and processes (sequence flow actions) to name a few. Documentation of your setup is key to surviving a name change.

2.5.2 Priority Settings

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

Name *
The name of this resource. Any type of characters can be entered to this field including uppercase letters and spaces. The name will be displayed in the overview table.

Validity *
Set the validity of this resource. Each resource can be used in OTRS only, if this field is set to *valid*. Setting this field to *invalid* or *invalid-temporarily* will disable the use of the resource.

Text Color *
This setting defines the text color for this priority level.

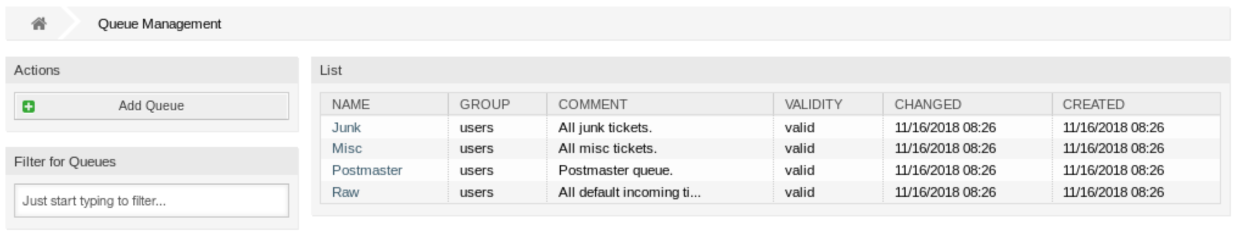
Background Color *
This setting defines the background color for this priority level.

2.6 Queues

Teams need a workspace and the ability to dispatch work based on skill level, security level, department or responsibility just to name a few. Other teams may also need to view or react in these requests as well.

OTRS uses queues to provide your teams with structure. Queues provide a powerful way to divide and disperse the work to the responsible group of people.

Use this screen to add queues to the system. In a fresh OTRS installation there are 4 default queues: *Raw*, *Junk*, *Misc* and *Postmaster*. All incoming messages will be stored in the *Raw* queue if no filter rules are defined. The *Junk* queue can be used to store spam messages. The queue management screen is available in the *Queues* module of the *Ticket Settings* group.



NAME	GROUP	COMMENT	VALIDITY	CHANGED	CREATED
Junk	users	All junk tickets.	valid	11/16/2018 08:26	11/16/2018 08:26
Misc	users	All misc tickets.	valid	11/16/2018 08:26	11/16/2018 08:26
Postmaster	users	Postmaster queue.	valid	11/16/2018 08:26	11/16/2018 08:26
Raw	users	All default incoming ti...	valid	11/16/2018 08:26	11/16/2018 08:26

Fig. 19: Queue Management Screen

2.6.1 Manage Queues

To add a queue:

1. Click on the *Add Queue* button in the left sidebar.
2. Fill in the required fields.
3. Click on the *Save* button.

Warning: Queues can not be deleted from the system. They can only be deactivated by setting the *Validity* option to *invalid* or *invalid-temporarily*.

Warning: The maximum number of 200 *valid* queues should not be exceeded. Exceeding this limit may affect the system performance.

To edit a queue:

1. Click on a queue in the list of queues.
2. Modify the fields.
3. Click on the *Save* or *Save and finish* button.

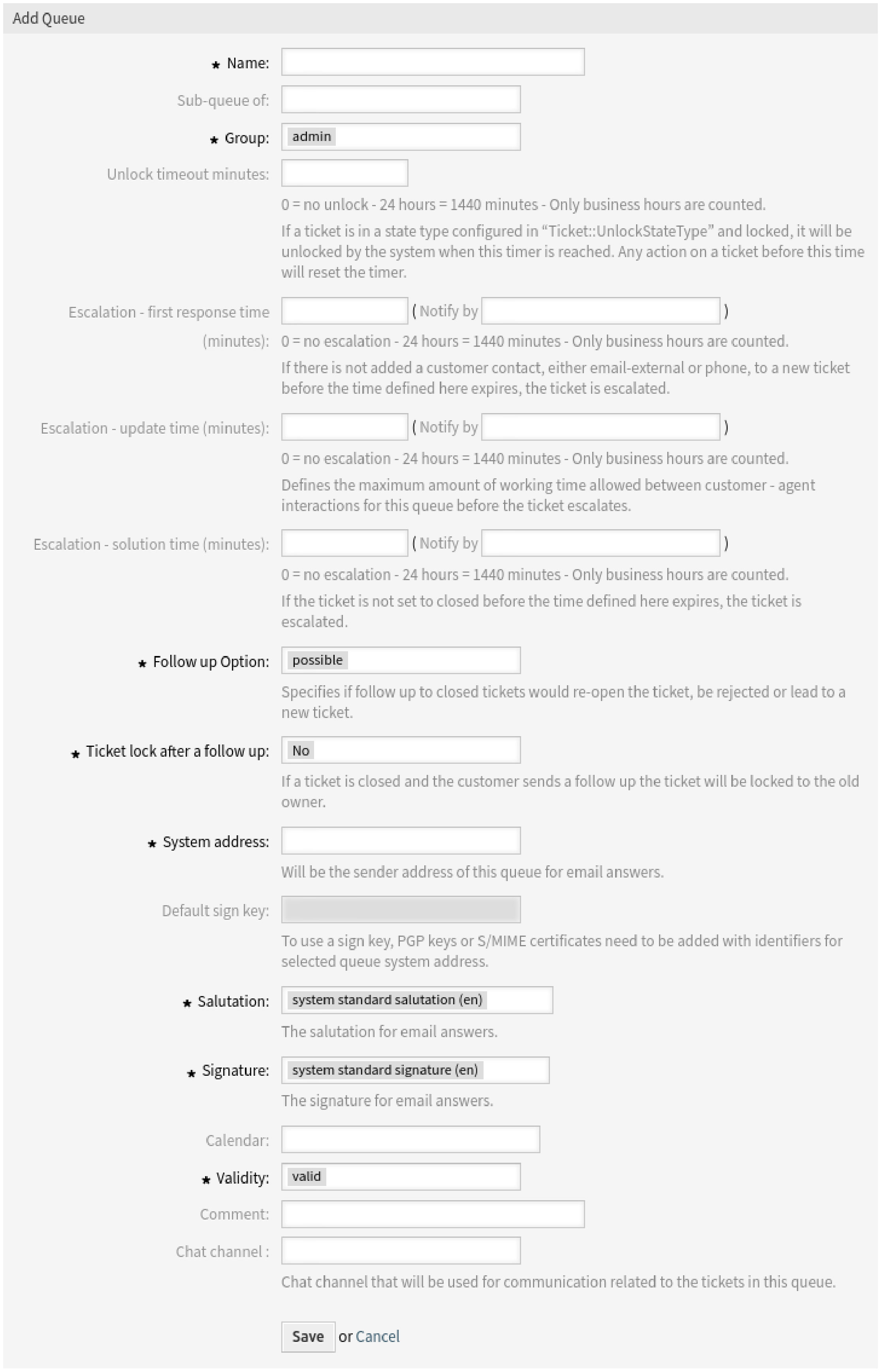
Note: If several queues are added to the system, use the filter box to find a particular queue by just typing the name to filter.

If you change the name of a queue which is used in the system configuration, a validation check will warn you and give you the following options:

Save and update automatically
Apply the change and also update the affected settings.

Don't save, update manually
Apply the change, but don't update the affected settings. The updates need to be done manually.

Cancel
Cancel the action.

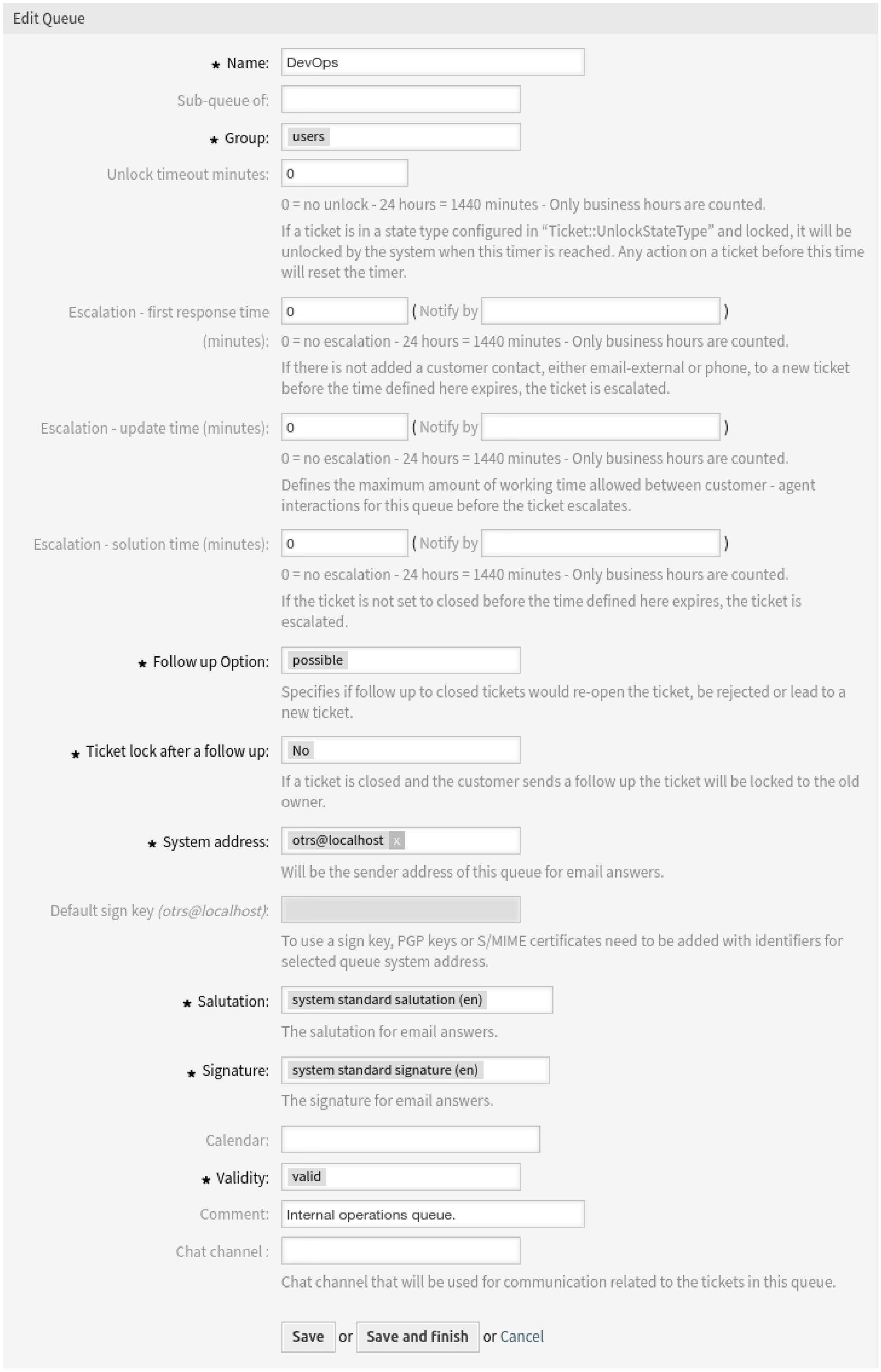


Add Queue

★ Name:	
Sub-queue of:	
★ Group:	admin
Unlock timeout minutes:	0 = no unlock - 24 hours = 1440 minutes - Only business hours are counted. If a ticket is in a state type configured in “Ticket::UnlockStateType” and locked, it will be unlocked by the system when this timer is reached. Any action on a ticket before this time will reset the timer.
Escalation - first response time (minutes):	(Notify by) 0 = no escalation - 24 hours = 1440 minutes - Only business hours are counted. If there is not added a customer contact, either email-external or phone, to a new ticket before the time defined here expires, the ticket is escalated.
Escalation - update time (minutes):	(Notify by) 0 = no escalation - 24 hours = 1440 minutes - Only business hours are counted. Defines the maximum amount of working time allowed between customer - agent interactions for this queue before the ticket escalates.
Escalation - solution time (minutes):	(Notify by) 0 = no escalation - 24 hours = 1440 minutes - Only business hours are counted. If the ticket is not set to closed before the time defined here expires, the ticket is escalated.
★ Follow up Option:	possible — Specifies if follow up to closed tickets would re-open the ticket, be rejected or lead to a new ticket.
★ Ticket lock after a follow up:	No — If a ticket is closed and the customer sends a follow up the ticket will be locked to the old owner.
★ System address:	Will be the sender address of this queue for email answers.
Default sign key:	To use a sign key, PGP keys or S/MIME certificates need to be added with identifiers for selected queue system address.
★ Salutation:	system standard salutation (en) — The salutation for email answers.
★ Signature:	system standard signature (en) — The signature for email answers.
Calendar:	
★ Validity:	valid
Comment:	
Chat channel :	Chat channel that will be used for communication related to the tickets in this queue.

Save or Cancel

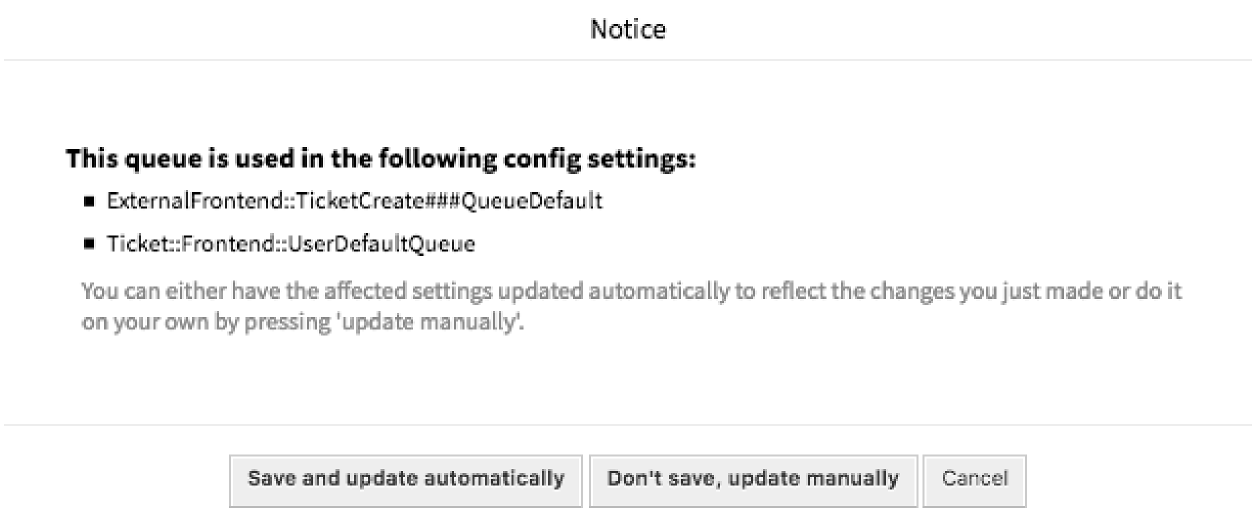
Fig. 20: Add Queue Screen



Edit Queue

⋆ Name:	DevOps
Sub-queue of:	
⋆ Group:	users
Unlock timeout minutes:	0 0 = no unlock - 24 hours = 1440 minutes - Only business hours are counted. If a ticket is in a state type configured in “Ticket::UnlockStateType” and locked, it will be unlocked by the system when this timer is reached. Any action on a ticket before this time will reset the timer.
Escalation - first response time (minutes):	0 (Notify by) 0 = no escalation - 24 hours = 1440 minutes - Only business hours are counted. If there is not added a customer contact, either email-external or phone, to a new ticket before the time defined here expires, the ticket is escalated.
Escalation - update time (minutes):	0 (Notify by) 0 = no escalation - 24 hours = 1440 minutes - Only business hours are counted. Defines the maximum amount of working time allowed between customer - agent interactions for this queue before the ticket escalates.
Escalation - solution time (minutes):	0 (Notify by) 0 = no escalation - 24 hours = 1440 minutes - Only business hours are counted. If the ticket is not set to closed before the time defined here expires, the ticket is escalated.
⋆ Follow up Option:	possible Specifies if follow up to closed tickets would re-open the ticket, be rejected or lead to a new ticket.
⋆ Ticket lock after a follow up:	No If a ticket is closed and the customer sends a follow up the ticket will be locked to the old owner.
⋆ System address:	otrs@localhost x Will be the sender address of this queue for email answers.
Default sign key *(otrs@localhost)*:	 To use a sign key, PGP keys or S/MIME certificates need to be added with identifiers for selected queue system address.
⋆ Salutation:	system standard salutation (en) The salutation for email answers.
⋆ Signature:	system standard signature (en) The signature for email answers.
Calendar:	
⋆ Validity:	valid
Comment:	Internal operations queue.
Chat channel :	 Chat channel that will be used for communication related to the tickets in this queue.

Fig. 21: Edit Queue Screen


Notice

This queue is used in the following config settings:

- ExternalFrontend::TicketCreate###QueueDefault
- Ticket::Frontend::UserDefaultQueue

You can either have the affected settings updated automatically to reflect the changes you just made or do it on your own by pressing 'update manually'.

Fig. 22: Queue Validation Dialog

Warning: Changing the name of this object should be done with care, the check only provides verification for certain settings and ignores things where the name can't be verified. Some examples are dashboard filters, access control lists (ACLs), and processes (sequence flow actions) to name a few. Documentation of your setup is key to surviving a name change.

2.6.2 Queue Settings

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

Name *
The name of this resource. Any type of characters can be entered to this field including uppercase letters and spaces. The name will be displayed in the overview table.

Note: The names of queues cannot be translated, even using a custom language file. *Salutations* and *Signatures* have to be assigned to queues, which depend on a specific language, so translating the name makes no sense.

Sub-queue of
It is possible to add the new queue under an existing one as sub-queue. This will be displayed as *Parent Queue::Child Queue*.

Group *
It is possible to limit access to the selected group. The group creates a permission link between the queue and an agent or a customer user.

Unlock timeout minutes
Any ticket which is in a state type configured in `Ticket::UnlockStateType` and locked by an agent in this queue will automatically unlock after the set amount of minutes. The value *0* (default) means tickets in this queue remain locked.

Escalation - first response time (minutes)
Defines the time span for this queue within the defined business hours, in which an agent has to send the first reply to the customer as a visible article to a ticket created by a customer. The escalation time for the first response is cancelled as soon as the agent sends an article that is visible to the customer.

The first response time is triggered if a ticket has been created by an agent.

Once a ticket has been escalated because of exceeding the defined time span then this ticket stays in the status *escalated*.

The first response time escalation notification will be sent to all agents who have selected the specific queue as preferred queue in the *My Queues* setting and the ticket escalation notification is enabled.

Escalation - update time (minutes)
Defines the time span for this queue within the defined business hours, after which the ticket will be escalated if no further articles are written. This escalation update time is reset whenever an agent writes an article that is visible to the customer user. The escalation update time is also reset when a customer user responds to an agent article for the first time. Additional articles written by the customer user do not affect the escalation update time.

The escalation notification will be sent to all agents who have selected the specific queue as preferred queue in the *My Queues* setting and the ticket escalation notification is enabled.

Escalation - solution time (minutes)
Sets the time span for this queue, within the defined business hours, within the ticket has to be solved. Solved means that the ticket is set to the status *closed*. If the ticket is not set to solved until the time span defined here is reached, the ticket escalates.

The escalation notification will be sent to all agents who have selected the specific queue as preferred queue in the *My Queues* setting and the ticket escalation notification is enabled.

Once resolution time escalation has been triggered for a ticket, the ticket remains escalated.

The solution time will not reset if the ticket is reopened.

Note: The escalation settings of *Service Level Agreements* will overwrite the escalation settings of queues.

Follow up Option *
Specify the handling of a follow up on closed tickets. Possible values:

new ticket
The follow up will create a new ticket.

possible
The follow up will reopen the already closed ticket.

reject
The follow up will be rejected.

See also:

See *Auto Responses* chapter for more information.

Ticket lock after a follow up *
Only applicable if the *Follow up Option* is set to *possible*. Locks the previously closed ticket, upon reopening, to the last owner. This ensures that a follow up for a ticket is processed by the agent that has previously handled that ticket.

> **Warning:** This does not take out-of-office into account. Use this setting with care to ensure or in combination with *Unlock timeout minutes*.

System address *
Select one of the *Email Addresses* as the sender identity for this queue.

> **Note:** This is an ID in the database. Making changes to the *Email Addresses* can have adverse effects here.

Default sign key
This is only active if *PGP Keys* or *S/MIME Certificates* is enabled in the *System Configuration*. Choose the key to sign emails per default.

Salutation *
Select one of the defined *Salutations*.

Signature *
Select one of the defined *Signatures*.

Calendar
Select the calendar which defines working hours for this queue. Calendars are defined in the *System Configuration*.

Validity *
Set the validity of this resource. Each resource can be used in OTRS only, if this field is set to *valid*. Setting this field to *invalid* or *invalid-temporarily* will disable the use of the resource.

Comment
Add additional information to this resource. It is recommended to always fill this field as a description of the resource with a full sentence for better clarity, because the comment will be also displayed in the overview table.

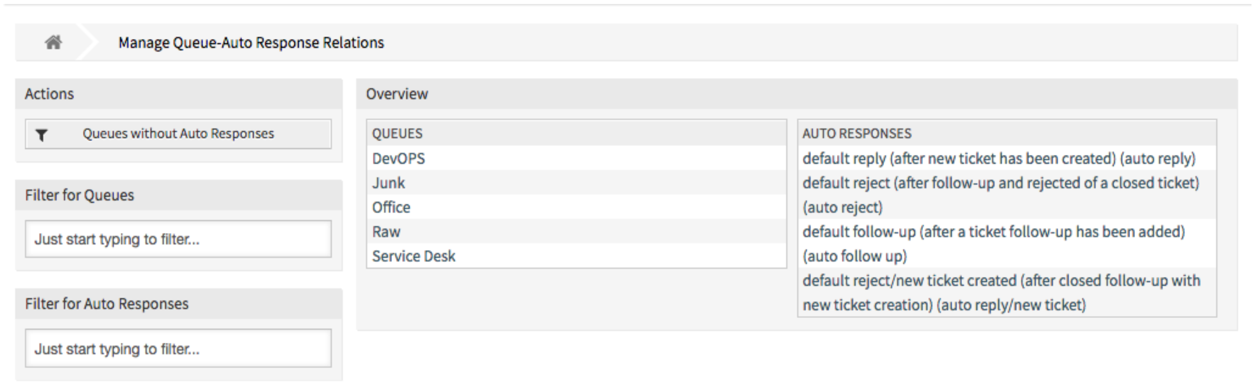
Chat Channel
Chat channel that will be used for communication related to the tickets in this queue.

2.7 Queues   Auto Responses

Communicating a change in, for example, service times, service levels or other information which would be good for a customer to know when opening a ticket can be a tedious and error-ridden task. Attempting to make sure all departments have the correct information and transmitting this to their customers poses a challenge.

OTRS gives you the power to quickly assign the appropriate automatic responses to any queue, containing pertinent service information, ensuring this information reaches your customers before expectations aren’t reached.

Use this screen to add one or more automatic responses to one or more queues. The management screen is available in the *Queues   Auto Responses* module of the *Ticket Settings* group.


Manage Queue-Auto Response Relations

Actions

Queues without Auto Responses

Filter for Queues

Just start typing to filter...

Filter for Auto Responses

Just start typing to filter...

Overview

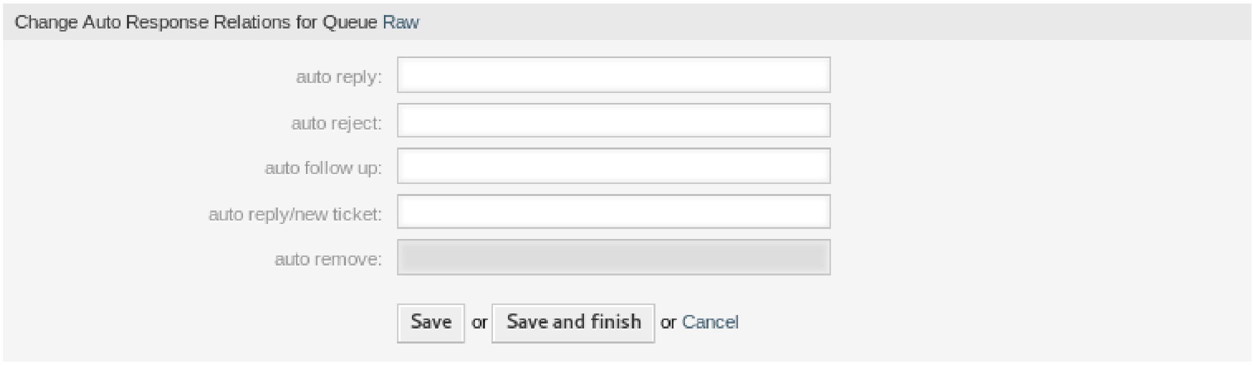
QUEUES	AUTO RESPONSES
DevOPS	default reply (after new ticket has been created) (auto reply)
Junk	default reject (after follow-up and rejected of a closed ticket) (auto reject)
Office	default follow-up (after a ticket follow-up has been added) (auto follow up)
Raw	default reject/new ticket created (after closed follow-up with new ticket creation) (auto reply/new ticket)
Service Desk	

Fig. 23: Manage Queue-Auto Response Relations

2.7.1 Manage Queues   Auto Responses Relations

To assign an automatic response to a queue:

1. Click on a queue in the *Queues* column.
2. Select the automatic responses you would like to add the queue to.
3. Click on the *Save* or *Save and finish* button.


Change Auto Response Relations for Queue Raw

auto reply:	
auto reject:	
auto follow up:	
auto reply/new ticket:	
auto remove:	

Save or Save and finish or Cancel

Fig. 24: Change Auto Response Relations for Queue

Note: It is not possible to assign multiple queues to an automatic response by clicking on the automatic response name. A click on the automatic response will open to the *Auto Responses* screen.

Note: If several automatic responses or queues are added to the system, use the filter box to find a particular automatic response or queue by just typing the name to filter.

2.7.2 Queues Auto Responses Settings

The following settings are available when assigning some automatic responses to a queue.

auto reply
This automatic response will be sent to users, when they send a message to the queue, and the message is not a follow-up message of a ticket.

auto reject
This automatic response will be sent to users, when they send a message to the queue and *reject* is set as *Follow up Option* in the queue settings.

auto follow up
This automatic response will be sent to users, when they send a message to the queue and *possible* is set as *Follow up Option* in the queue settings.

auto reply/new ticket
This automatic response will be sent to users, when they send a message to the queue and *new ticket* is set as *Follow up Option* in the queue settings.

auto remove
This automatic response will be sent to users, when the remove option is enabled.

Note: *Auto reply*, *auto reject* and *auto reply/new ticket* mutually cancel each other based on the *Queues* settings. Only one will take effect per queue.

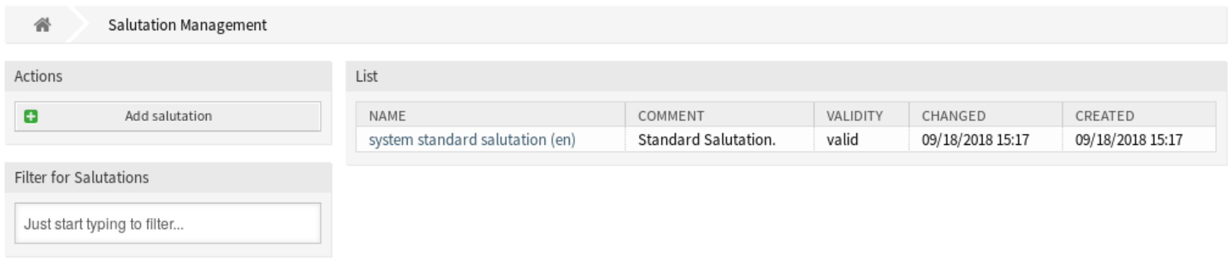
2.8 Salutations

Addressing customers must be done in a standardized way. Your customers may not always be external customers requiring a less formal greeting.

OTRS provides you with the tools needed to create a standardized communication form for any one of your queues. As defined in the *Queue Settings*: *Salutations*, *Templates*, and *Signatures* are combined to ensure a well formed standardized email communication.

Salutations can be linked to one or more *Queues*. A salutation is used only in email answers to tickets.

Use this screen to add salutations to the system. A fresh OTRS installation already contains a standard salutation. The salutation management screen is available in the *Salutations* module of the *Ticket Settings* group.



NAME	COMMENT	VALIDITY	CHANGED	CREATED
system standard salutation (en)	Standard Salutation.	valid	09/18/2018 15:17	09/18/2018 15:17

Fig. 25: Salutation Management Screen

2.8.1 Manage Salutations

To add a salutation:

1. Click on the *Add Salutation* button in the left sidebar.
2. Fill in the required fields.
3. Click on the *Save* button.

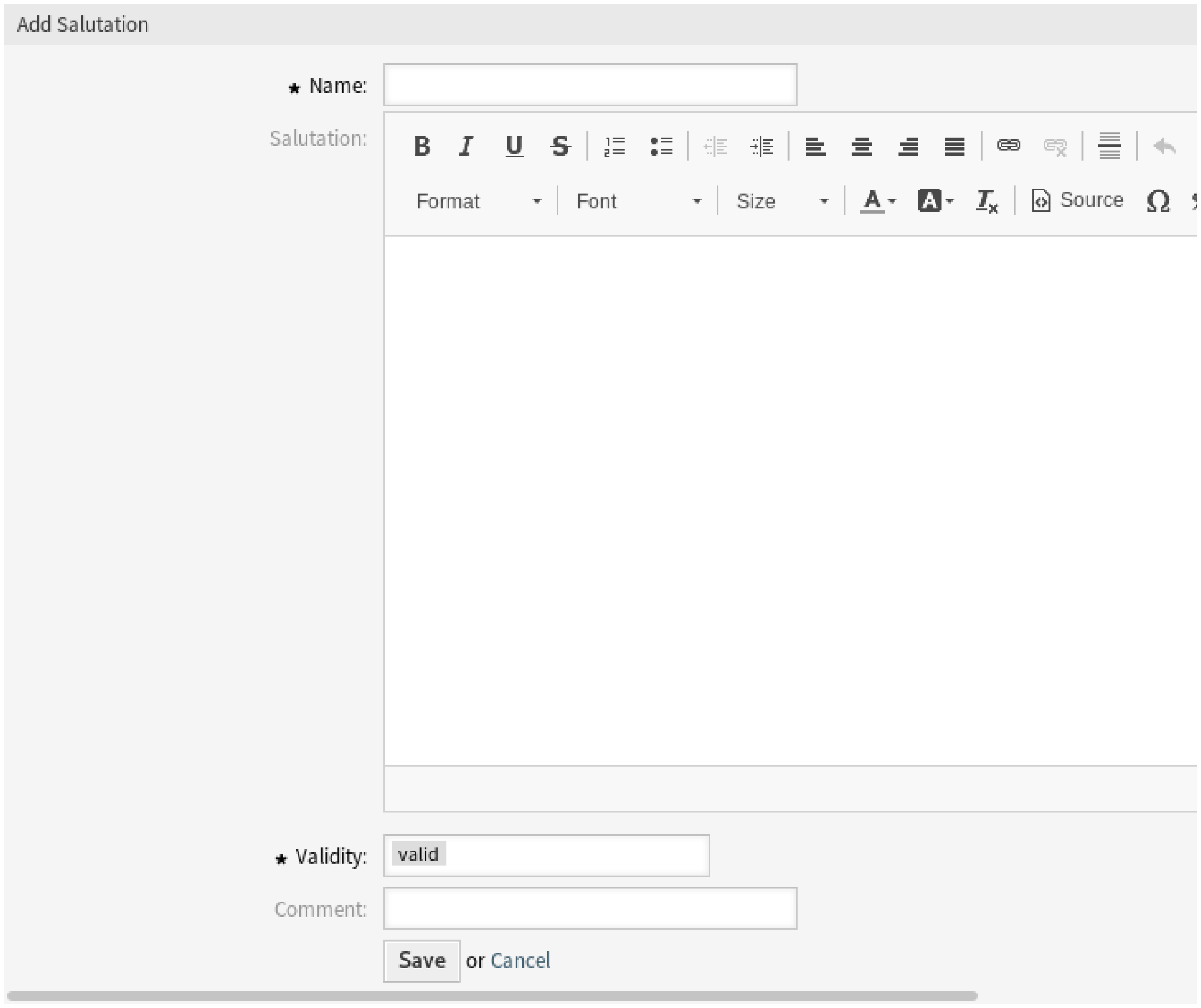


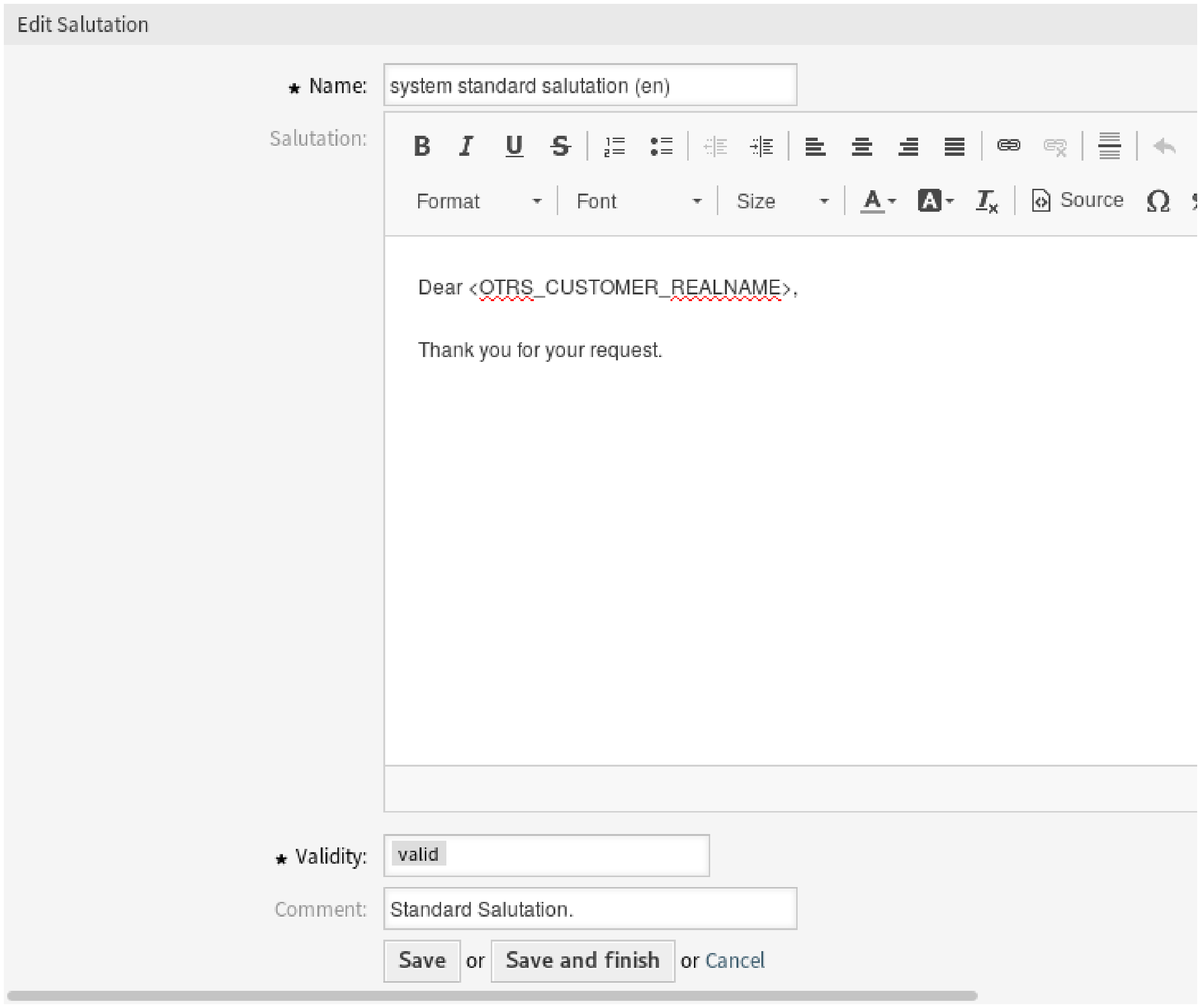
Fig. 26: Add Salutation Screen

Warning: Salutations can not be deleted from the system. They can only be deactivated by setting the *Validity* option to *invalid* or *invalid-temporarily*.

To edit a salutation:

1. Click on a salutation in the list of salutations.
2. Modify the fields.

3. Click on the *Save* or *Save and finish* button.



Edit Salutation

★ Name:	system standard salutation (en)
Salutation:	Dear <OTRS_CUSTOMER_REALNAME>, Thank you for your request.
★ Validity:	valid
Comment:	Standard Salutation.

Fig. 27: Edit Salutation Screen

Note: If several salutations are added to the system, use the filter box to find a particular salutation by just typing the name to filter.

Warning: Before invalidating this object, please go to the *Queues* module of the *Ticket Settings* group and make sure all queues using this setting are using a valid object.

2.8.2 Salutation Settings

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

Name *
The name of this resource. Any type of characters can be entered to this field including uppercase letters and spaces. The name will be displayed in the overview table.

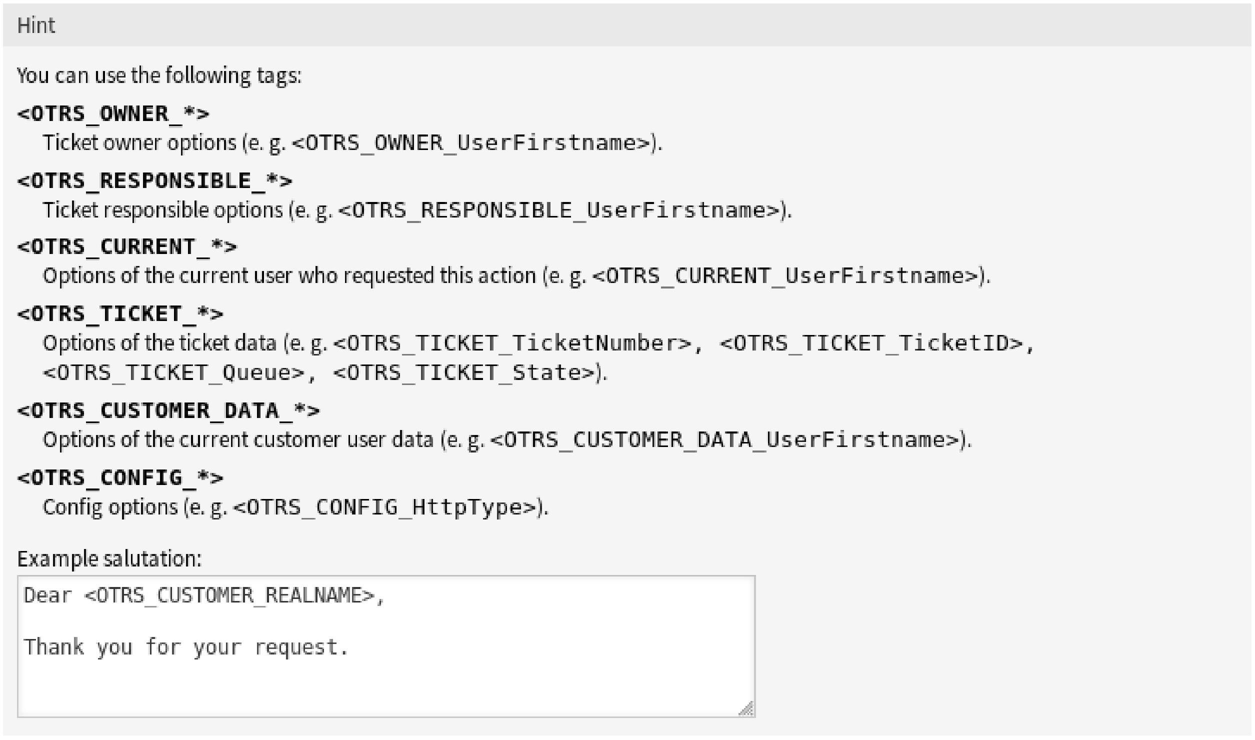
Salutation
The text that will be placed to the beginning of new emails.

Validity *
Set the validity of this resource. Each resource can be used in OTRS only, if this field is set to *valid*. Setting this field to *invalid* or *invalid-temporarily* will disable the use of the resource.

Comment
Add additional information to this resource. It is recommended to always fill this field as a description of the resource with a full sentence for better clarity, because the comment will be also displayed in the overview table.

2.8.3 Salutation Variables

Using variables in the text makes it possible to personalize messages. Variables, known as OTRS tags, are replaced by OTRS when generating the message. Find a list of available tags stems for this resource at the bottom of both add and edit screens.



Hint

You can use the following tags:

<OTRS_OWNER_*>
Ticket owner options (e. g. <OTRS_OWNER_UserFirstname>).

<OTRS_RESPONSIBLE_*>
Ticket responsible options (e. g. <OTRS_RESPONSIBLE_UserFirstname>).

<OTRS_CURRENT_*>
Options of the current user who requested this action (e. g. <OTRS_CURRENT_UserFirstname>).

<OTRS_TICKET_*>
Options of the ticket data (e. g. <OTRS_TICKET_TicketNumber>, <OTRS_TICKET_TicketID>, <OTRS_TICKET_Queue>, <OTRS_TICKET_State>).

<OTRS_CUSTOMER_DATA_*>
Options of the current customer user data (e. g. <OTRS_CUSTOMER_DATA_UserFirstname>).

<OTRS_CONFIG_*>
Config options (e. g. <OTRS_CONFIG_HttpType>).

Example salutation:

```
Dear <OTRS_CUSTOMER_REALNAME>,

Thank you for your request.
```

Fig. 28: Salutation Variables

For example, the variable `<OTRS_CUSTOMER_DATA_UserLastname>` expands to the customer’s last name to be included in something like the following.

```
Dear <OTRS_CUSTOMER_DATA_UserFirstname> <OTRS_CUSTOMER_DATA_UserLastname>,
```

This tag expands, for example to:

```
Dear Lisa Wagner,
```

The values of the following variables are translated based on the chosen language of the customer user.

```
<OTRS_TICKET_Type>
<OTRS_TICKET_State>
<OTRS_TICKET_StateType>
<OTRS_TICKET_Lock>
<OTRS_TICKET_Priority>
```

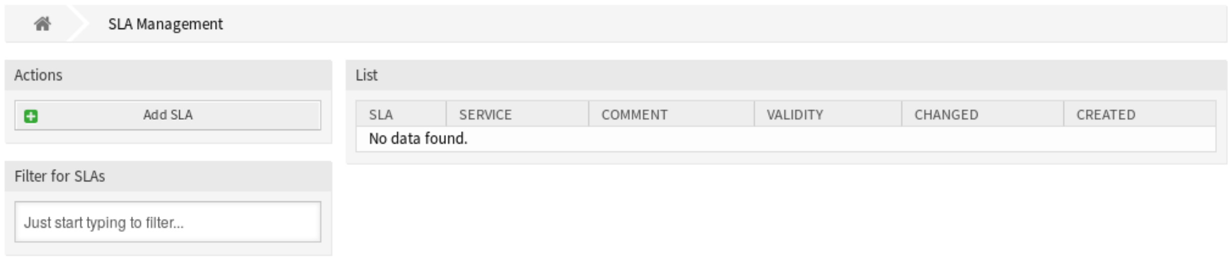
If the language is not supported the default language is applied.

2.9 Service Level Agreements

Your organization must meet the time demands of your customers. Timely service matters. Response to questions, updates on issues, and solutions must be provided in an agreed amount of time. The agent must receive notification of possible breaches, to prevent ticket escalation.

OTRS scales well with your needs and offers management of service level agreements (SLAs). OTRS provides you with the possibility to create numerous service level agreements covering all of your service and customer need. Each *SLA* can cover multiple services and define the availability of service and escalation periods.

Use this screen to add service level agreements to the system. A fresh OTRS installation doesn't contain any service level agreements by default. The service level agreement management screen is available in the *Service Level Agreements* module of the *Ticket Settings* group.



SLA	SERVICE	COMMENT	VALIDITY	CHANGED	CREATED
No data found.					

Fig. 29: Service Level Agreement Management Screen

See also:

To use this feature, Ticket::Service must be activated in the *System Configuration* under the *Administration* group to be selectable in the ticket screens. You may click on the link in the warning message of the notification bar to directly jump to the configuration setting.

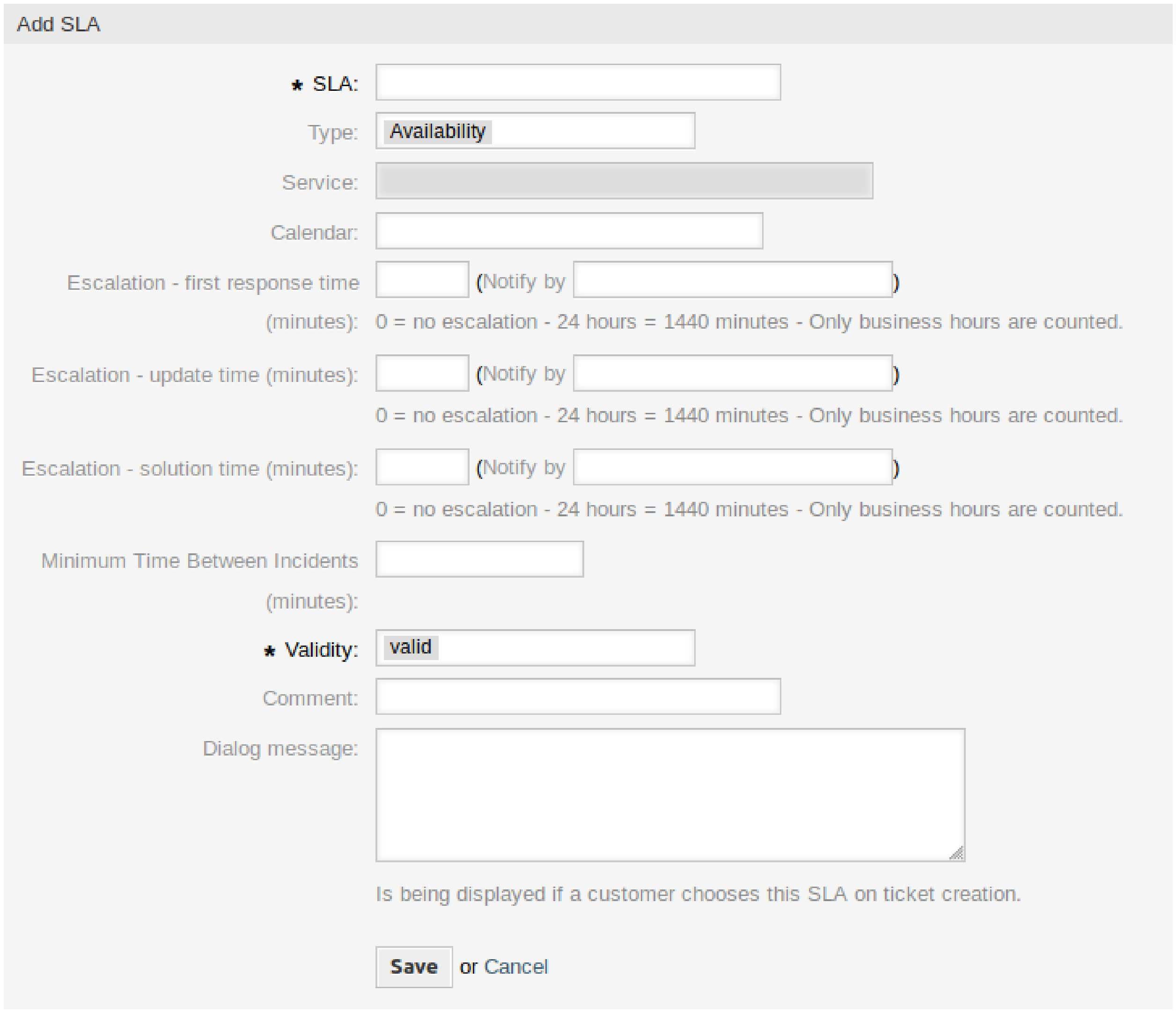
To use this feature on external interface, ExternalFrontend::TicketCreate###SLA must be activated.

2.9.1 Manage Service Level Agreements

Note: Adding service level agreements requires, that at least one service is added to the system. Create services in the *Services* screen.

To add a service level agreement:

1. Click on the *Add SLA* button in the left sidebar.
2. Fill in the required fields.
3. Click on the *Save* button.


Add SLA

* SLA:	
Type:	Availability
Service:	
Calendar:	
Escalation - first response time (minutes):	(Notify by) 0 = no escalation - 24 hours = 1440 minutes - Only business hours are counted.
Escalation - update time (minutes):	(Notify by) 0 = no escalation - 24 hours = 1440 minutes - Only business hours are counted.
Escalation - solution time (minutes):	(Notify by) 0 = no escalation - 24 hours = 1440 minutes - Only business hours are counted.
Minimum Time Between Incidents (minutes):	
* Validity:	valid
Comment:	
Dialog message:	Is being displayed if a customer chooses this SLA on ticket creation.

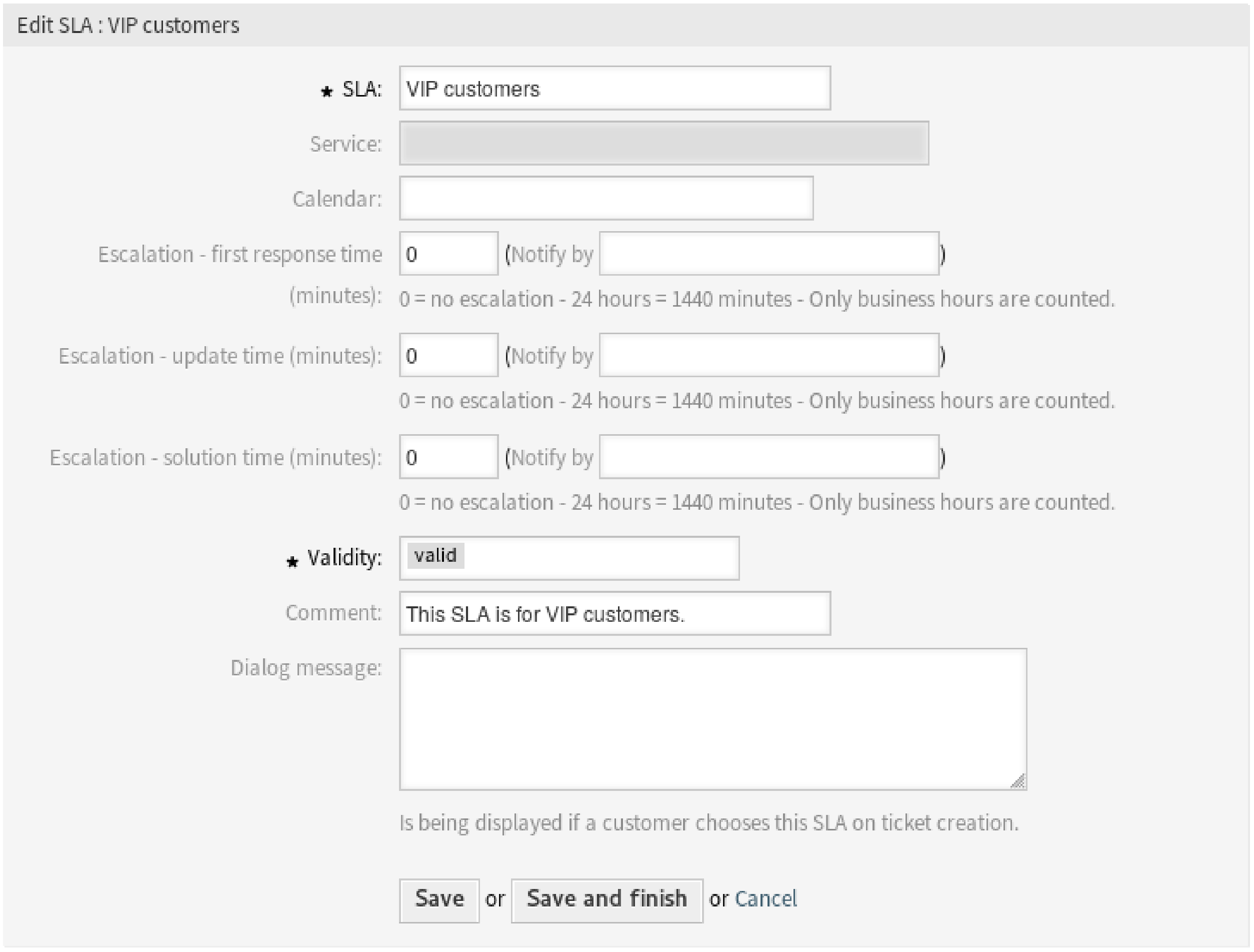
Fig. 30: Add Service Level Agreement Screen

Warning: Service level agreements can not be deleted from the system. They can only be deactivated by setting the *Validity* option to *invalid* or *invalid-temporarily*.

Warning: The maximum number of 50 *valid* service level agreements should not be exceeded. Exceeding this limit may affect the system performance.

To edit a service level agreement:

1. Click on a service level agreement in the list of service level agreements.
2. Modify the fields.
3. Click on the *Save* or *Save and finish* button.


Edit SLA : VIP customers

⋆ SLA:	VIP customers
Service:	
Calendar:	
Escalation - first response time (minutes):	0 (Notify by) 0 = no escalation - 24 hours = 1440 minutes - Only business hours are counted.
Escalation - update time (minutes):	0 (Notify by) 0 = no escalation - 24 hours = 1440 minutes - Only business hours are counted.
Escalation - solution time (minutes):	0 (Notify by) 0 = no escalation - 24 hours = 1440 minutes - Only business hours are counted.
⋆ Validity:	valid
Comment:	This SLA is for VIP customers.
Dialog message:	Is being displayed if a customer chooses this SLA on ticket creation.

Fig. 31: Edit Service Level Agreement Screen

Note: If several service level agreements are added to the system, use the filter box to find a particular service level agreement by just typing the name to filter.

2.9.2 Service Level Agreement Settings

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

SLA *

The name of this resource. Any type of characters can be entered to this field including uppercase letters and spaces. The name will be displayed in the overview table.

Note: The names of service level agreements cannot be translated, even using a custom language file. Service level agreements can use calendars, that depend on time zones. Because of this, different service level agreements have to be created for each time zone and translating their names makes no sense.

Type

Select a type for the service level agreement. The possible values come from *General Catalog*.

Service

Select one or more of the *Services* to be assigned to this SLA.

Calendar

Select the calendar which defines working hours for this queue.

Calendars are defined in the *System Configuration* under the navigation path *Core* → *Time*. You can define up to 10 calendars, a default one and 9 additional calendars for different working days, working hours and public holidays. By default, all calendars contain the same working days, working hours and public holidays.

Escalation - first response time (minutes)

Defines the time span for this SLA within the defined business hours, in which an agent has to send the first reply to the customer as a visible article to a ticket created by a customer. The escalation time for the first response is cancelled as soon as the agent sends an article that is visible to the customer.

The first response time is not triggered if a ticket has been created by an agent.

Once a ticket has been escalated because of exceeding the defined time span then this ticket stays in the status *escalated*.

The first response time escalation notification will be sent to all agents who have selected the specific queue as preferred queue in the *My Queues* setting and the ticket escalation notification is enabled.

Escalation - update time (minutes)

Defines the time span for this SLA within the defined business hours, after which the ticket will be escalated if no further articles are written. This escalation update time is reset whenever an agent writes an article that is visible to the customer user. The escalation update time is also reset when a customer user responds to an agent article for the first time. Additional articles written by the customer user do not affect the escalation update time.

The escalation notification will be sent to all agents who have selected the specific queue as preferred queue in the *My Queues* setting and the ticket escalation notification is enabled.

Escalation - solution time (minutes)

Sets the time span for this SLA within the defined business hours, within the ticket has to be solved. Solved means that the ticket is set to the status *closed*. If the ticket is not set to solved until the time span defined here is reached, the ticket escalates.

The escalation notification will be sent to all agents who have selected the specific queue as preferred queue in the *My Queues* setting and the ticket escalation notification is enabled.

Once resolution time escalation has been triggered for a ticket, the ticket remains escalated.

The solution time will not reset if the ticket is reopened.

Note: The escalation settings of service level agreements will overwrite the escalation settings of *Queues*.

Minimum Time Between Incidents (minutes)
You can define here the minimum time between incidents.

Validity *
Set the validity of this resource. Each resource can be used in OTRS only, if this field is set to *valid*. Setting this field to *invalid* or *invalid-temporarily* will disable the use of the resource.

Comment
Add additional information to this resource. It is recommended to always fill this field as a description of the resource with a full sentence for better clarity, because the comment will be also displayed in the overview table.

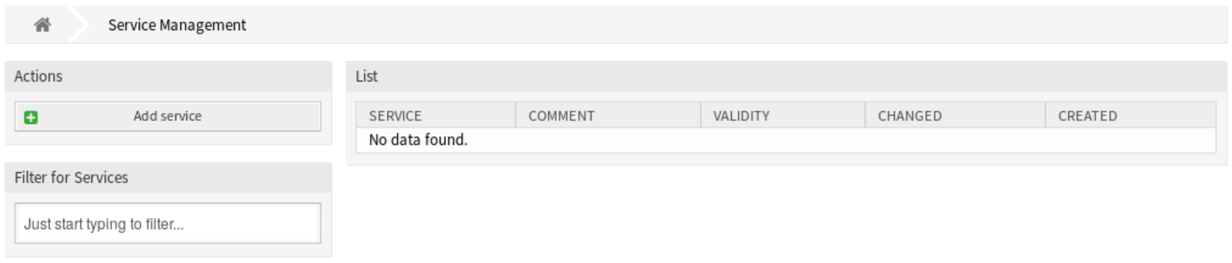
Dialog message
Is being displayed if a customer chooses this SLA on ticket creation (only in the external interface).

2.10 Services

Requests to your service desk should be categorized based on affected *service* to correctly reach a team of experts to deal with the issue. Because not all your agents can deal with specific problems. Lack of experience or access to resources to fix an issue requires defining the service affected in a ticket helps to categorize the issue and target the correct teams.

OTRS allows adding all services offered to your customers. These services may be later bound to *Service Level Agreements* to ensure a timely solution based on customer-specific agreements.

Use this screen to add services to the system. A fresh OTRS installation doesn't contain any services by default. The service management screen is available in the *Services* module of the *Ticket Settings* group.



SERVICE	COMMENT	VALIDITY	CHANGED	CREATED
No data found.				

Fig. 32: Service Management Screen

See also:

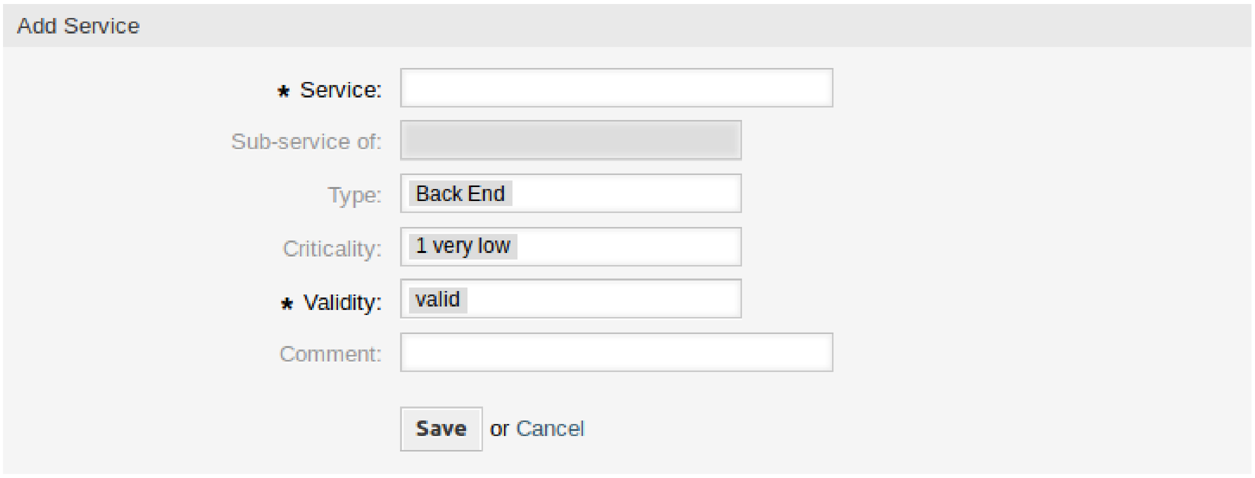
To use this feature, Ticket::Service must be activated in the *System Configuration* under the *Administration* group to be selectable in the ticket screens. You may click on the link in the warning message of the notification bar to directly jump to the configuration setting.

To use this feature on external interface, ExternalFrontend::TicketCreate###Service must be activated.

2.10.1 Manage Services

To add a service:

1. Click on the *Add Service* button in the left sidebar.
2. Fill in the required fields.
3. Click on the *Save* button.


Add Service

⋆ Service:	
Sub-service of:	
Type:	Back End
Criticality:	1 very low
⋆ Validity:	valid
Comment:	

Fig. 33: Add Service Screen

Warning: Services can not be deleted from the system. They can only be deactivated by setting the *Validity* option to *invalid* or *invalid-temporarily*.

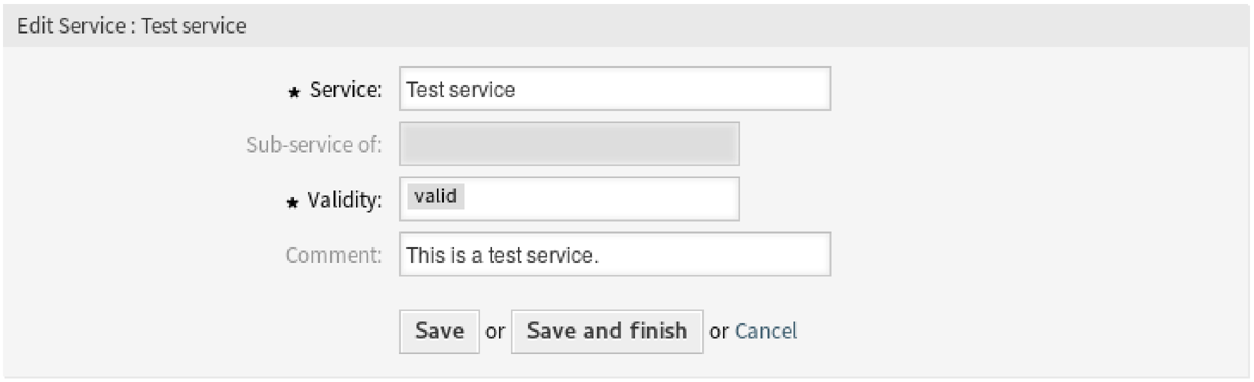
Warning: The maximum number of 500 *valid* services should not be exceeded. Exceeding this limit may affect the system performance.

To edit a service:

1. Click on a service in the list of services.
2. Modify the fields.
3. Click on the *Save* or *Save and finish* button.

Note: If several services are added to the system, use the filter box to find a particular service by just typing the name to filter.

Warning: Changing the name of this object should be done with care, the check only provides verification for certain settings and ignores things where the name can't be verified. Some examples are dashboard filters, access control lists (ACLs), and processes (sequence flow actions) to name a few. Documentation of your setup is key to surviving a name change.


Edit Service : Test service

★ Service:	Test service
Sub-service of:	
★ Validity:	valid
Comment:	This is a test service.

Fig. 34: Edit Service Screen

2.10.2 Service Settings

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

Service *
The name of this resource. Any type of characters can be entered to this field including uppercase letters and spaces. The name will be displayed in the overview table.

Sub-service of
It is possible to add the new service under an existing one as sub-service. This will be displayed as *Parent Service::Child Service*.

Type
Select a type for the service. The possible values come from *General Catalog*.

Criticality
Select a criticality for the service. The possible values come from *General Catalog*.

Validity *
Set the validity of this resource. Each resource can be used in OTRS only, if this field is set to *valid*. Setting this field to *invalid* or *invalid-temporarily* will disable the use of the resource.

Comment
Add additional information to this resource. It is recommended to always fill this field as a description of the resource with a full sentence for better clarity, because the comment will be also displayed in the overview table.

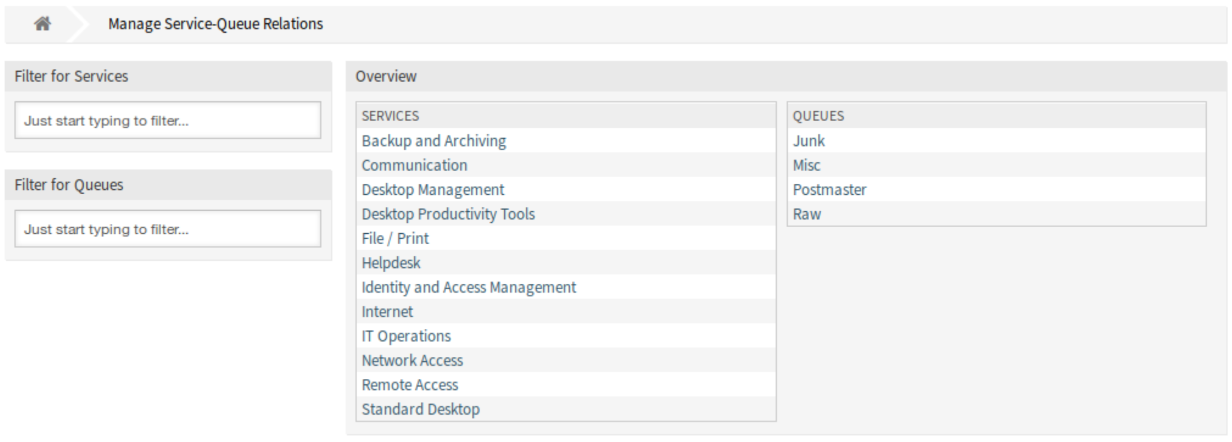
Note: A service can only be selectable in the ticket screens, if the service is assigned to the customer user of the ticket or it is set as default service for all customer users in the *Customer Users* *Services* screen.

2.11 Services   Queues

For a customer, excellent service initially starts with contacting the right service staff member. Being a help desk software, **OTRS** makes this possible with a clever queue and permission concept. However, if a ticket is created in the external or in the agent interface, one has to choose the suitable queue for his/her request from all existing and shown queues –not an easy task for a customer or responsible agent. For example a call center agent may not know which queue structure the service company has for the task on which he is working. This means a long decision-making times with a possible wrong result and the loss of precious working time.

This feature now makes it possible to assign queues to several services, so that if this service is selected, then only the appropriate queues are shown. For example the service *Support Mobile Devices* can be assigned to the queues *IT-2nd Level-Support Hardware* and *IT-2nd Level Support Mobile Applications* so the right selection can be made more easily. This module is configured to make all queues reappear with just one click in the agent interface. The feature is disabled in the external interface by default but can be enabled in the system configuration. Time consuming moves to the right queues and overextended customer users are a thing of the past.

Use this screen to add one or more services to one or more queues. To use this function, at least one service and one queue need to have been added to the system. The management screen is available in the *Services   Queues* module of the *Ticket Settings* group.


Manage Service-Queue Relations

Filter for Services

Just start typing to filter...

Filter for Queues

Just start typing to filter...

Overview

SERVICES	QUEUES
Backup and Archiving	Junk
Communication	Misc
Desktop Management	Postmaster
Desktop Productivity Tools	Raw
File / Print	
Helpdesk	
Identity and Access Management	
Internet	
IT Operations	
Network Access	
Remote Access	
Standard Desktop	

Fig. 35: Manage Service-Queue Relations Screen

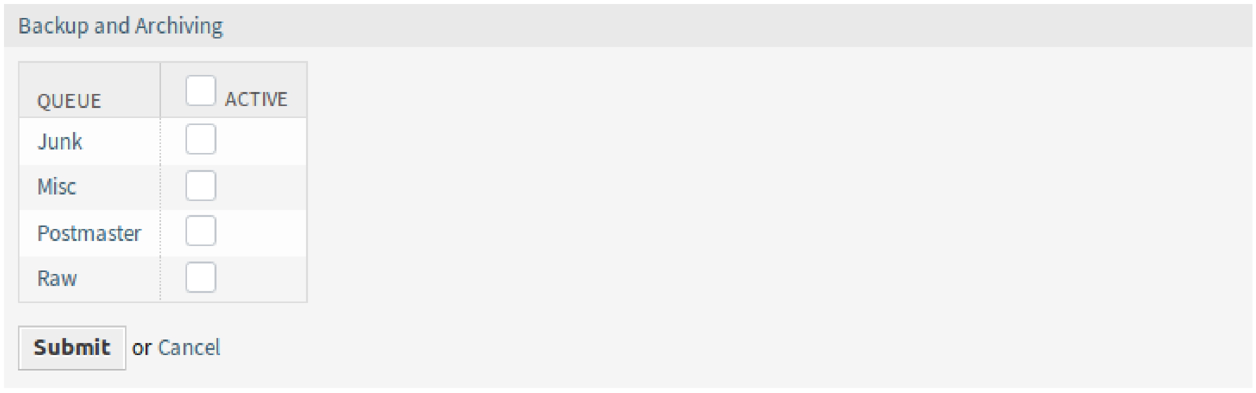
2.11.1 Manage Services   Queues Relations

To assign a service to a queue:

1. Click on a service in the *Services* column.
2. Select the queues you would like to add the service to.
3. Click on the *Save* or *Save and finish* button.

To assign a queue to a service:

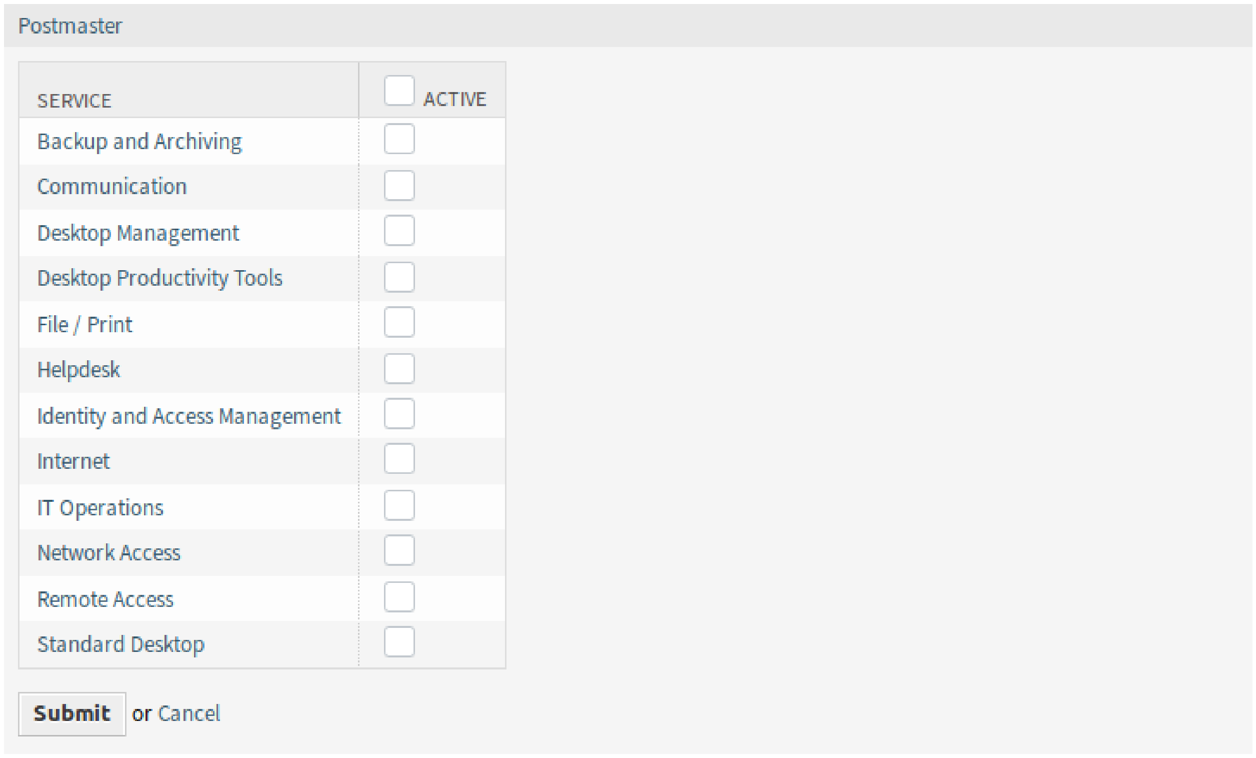
1. Click on a queue in the *Queues* column.
2. Select the services you would like to assign the queue to.


Backup and Archiving

QUEUE	☐ ACTIVE
Junk	☐
Misc	☐
Postmaster	☐
Raw	☐

Fig. 36: Change Queue Relations for Service

3. Click on the *Save* or *Save and finish* button.


Postmaster

SERVICE	☐ ACTIVE
Backup and Archiving	☐
Communication	☐
Desktop Management	☐
Desktop Productivity Tools	☐
File / Print	☐
Helpdesk	☐
Identity and Access Management	☐
Internet	☐
IT Operations	☐
Network Access	☐
Remote Access	☐
Standard Desktop	☐

Fig. 37: Change Service Relations for Queue

Note: If several services or queues are added to the system, use the filter box to find a particular service or queue by just typing the name to filter.

Multiple services or queues can be assigned in both screens at the same time. Additionally clicking on a service or clicking on a queue in the relations screen will open the *Edit Service* screen or the *Edit Queue* screen accordingly.

Warning: Accessing a queue or a service provides no back link to the relations screen.

2.11.2 Change Field Order on Screens

After selecting a service, there is a lookup for configured queues for this service. If there are configured queues for the selected service, only those will be available in the queue selection. If there are no queues configured for the selected service, all queues will be displayed.

However, the *Queue* field is before the *Service* field in the screens and this can cause confusion for the users. They select a queue first then they select a service but the selected queue is unselected if the assignment between the queue and the service is not set. The queue selection now depends on the service selection so the service should be the first input field.

To avoid such confusions it is **recommended** to change the field order of the affected screens in the system configuration.

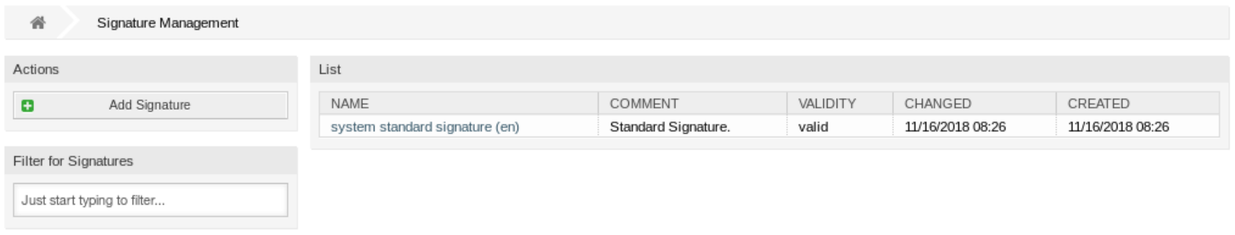
2.12 Signatures

Corporate identity and team information are essential in each communication. The name of the employee writing and other vital details like disclaimers are some examples of necessary information to include in the communication with the customer.

OTRS provides you with the same tools here, as with *Salutations*, to create a standardized communication form for any one of your queues. As defined in the *Queue Settings*: *Salutations*, *Templates*, and *Signatures* are combined to ensure a well formed standardized email communication.

Signatures can be linked to one or more *Queues*. A signature is used only in email answers to tickets.

Use this screen to add signatures to the system. A fresh OTRS installation already contains a standard signature. The signature management screen is available in the *Signatures* module of the *Ticket Settings* group.



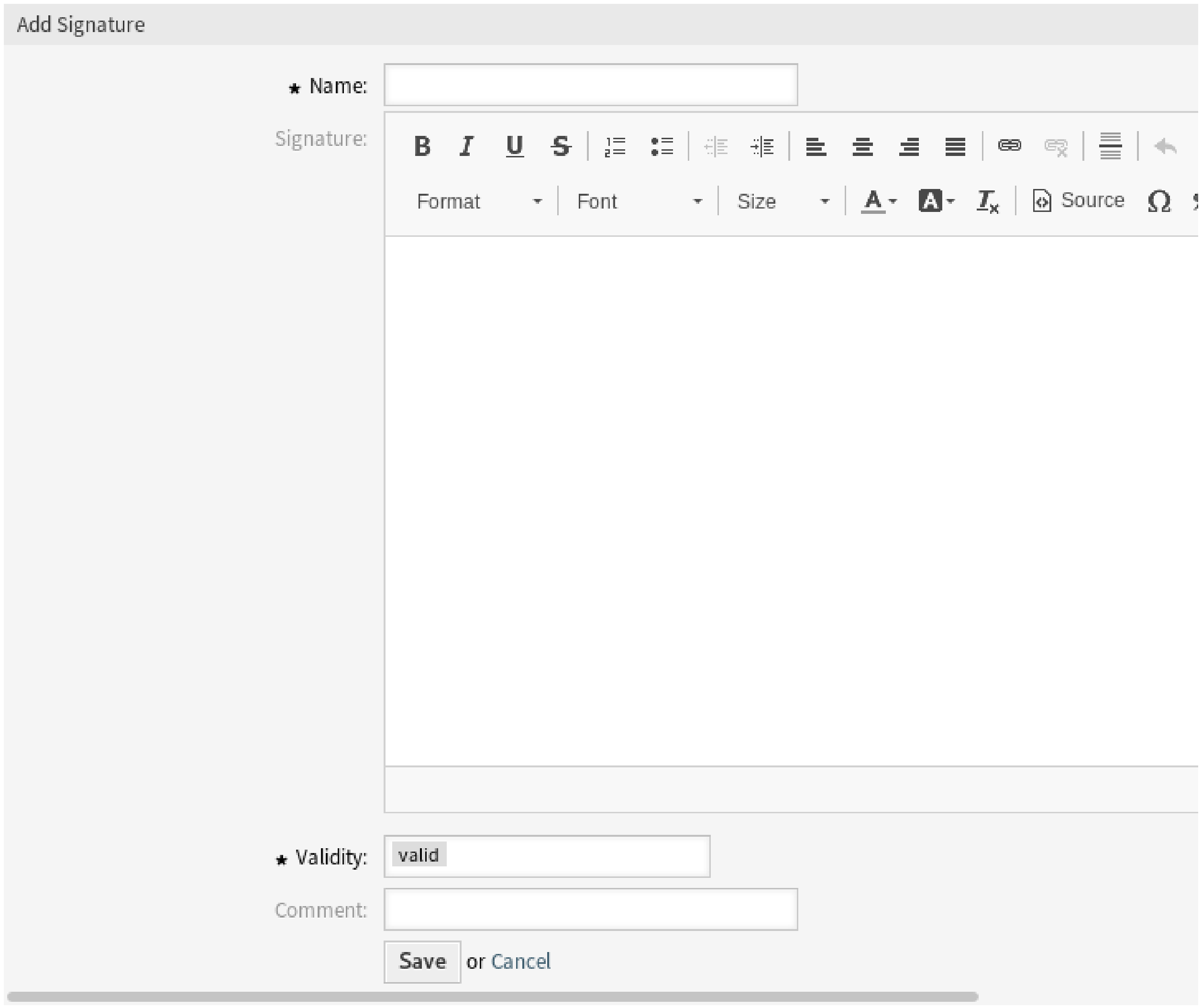
NAME	COMMENT	VALIDITY	CHANGED	CREATED
system standard signature (en)	Standard Signature.	valid	11/16/2018 08:26	11/16/2018 08:26

Fig. 38: Signature Management Screen

2.12.1 Manage Signatures

To add a signature:

1. Click on the *Add Signature* button in the left sidebar.
2. Fill in the required fields.
3. Click on the *Save* button.


Add Signature

★ Name:	
Signature:	
★ Validity:	valid
Comment:	

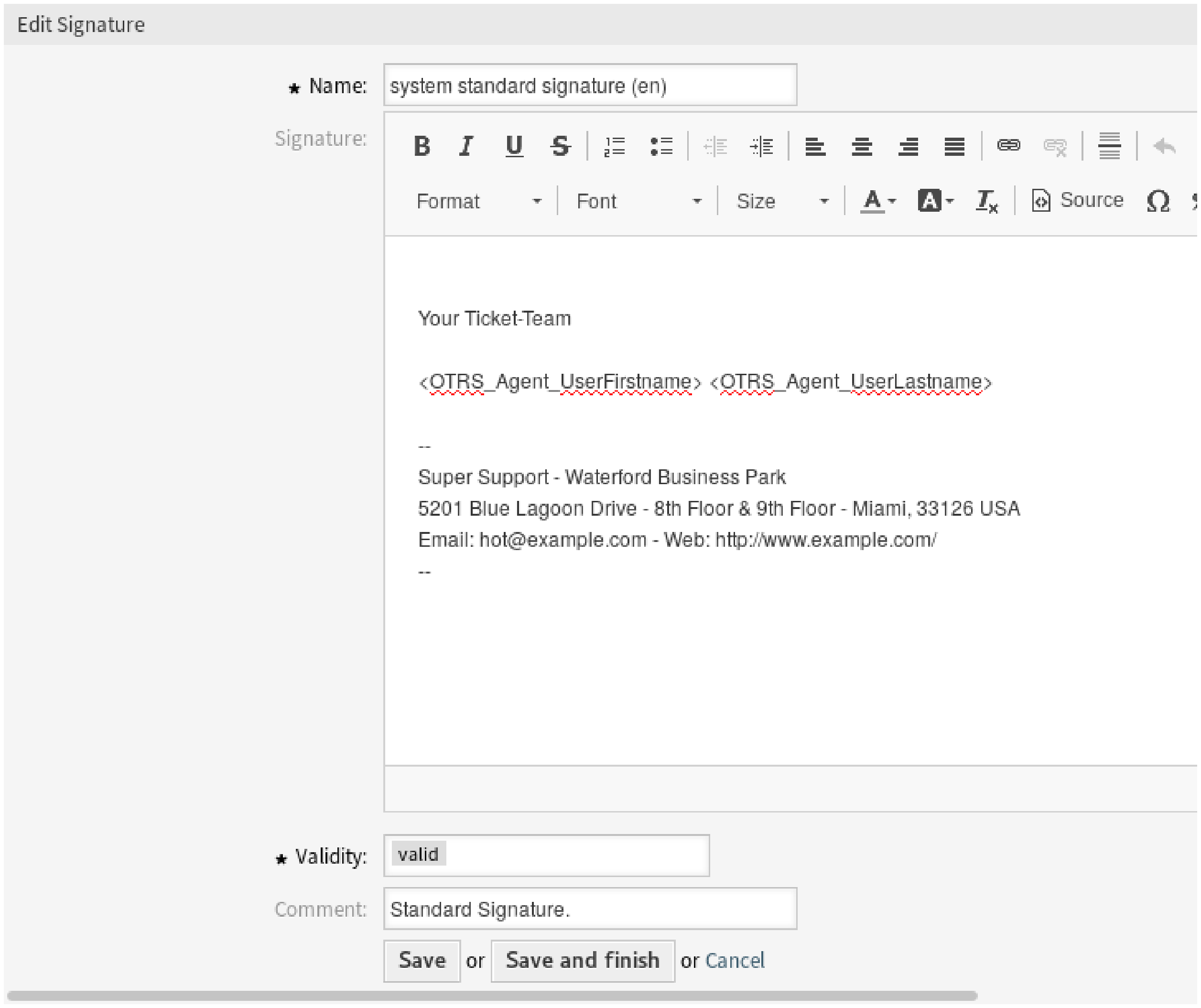
Fig. 39: Add Signature Screen

Warning: Signatures can not be deleted from the system. They can only be deactivated by setting the *Validity* option to *invalid* or *invalid-temporarily*.

To edit a signature:

1. Click on a signature in the list of signatures.
2. Modify the fields.

3. Click on the *Save* or *Save and finish* button.



Name: system standard signature (en)

Signature:

Your Ticket-Team

<OTRS_Agent_UserFirstname> <OTRS_Agent_UserLastname>

--
Super Support - Waterford Business Park
5201 Blue Lagoon Drive - 8th Floor & 9th Floor - Miami, 33126 USA
Email: hot@example.com - Web: http://www.example.com/
--

Validity: valid

Comment: Standard Signature.

Fig. 40: Edit Signature Screen

Note: If several signatures are added to the system, use the filter box to find a particular signature by just typing the name to filter.

Warning: Before invalidating this object, please go to the *Queues* module of the *Ticket Settings* group and make sure all queues using this setting are using a valid object.

2.12.2 Signature Settings

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

Name *
The name of this resource. Any type of characters can be entered to this field including uppercase letters and spaces. The name will be displayed in the overview table.

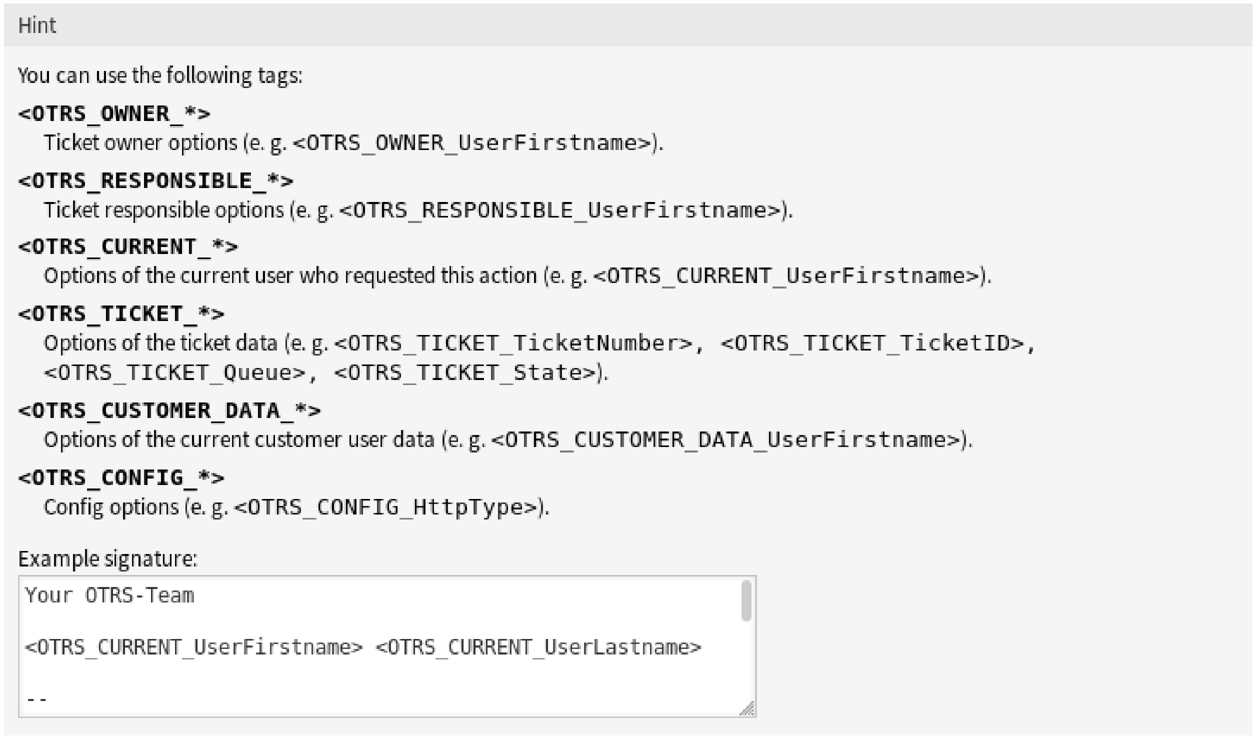
Signature
The text that will be placed to the end of new emails.

Validity *
Set the validity of this resource. Each resource can be used in OTRS only, if this field is set to *valid*. Setting this field to *invalid* or *invalid-temporarily* will disable the use of the resource.

Comment
Add additional information to this resource. It is recommended to always fill this field as a description of the resource with a full sentence for better clarity, because the comment will be also displayed in the overview table.

2.12.3 Signature Variables

Using variables in the text makes it possible to personalize messages. Variables, known as OTRS tags, are replaced by OTRS when generating the message. Find a list of available tags stems for this resource at the bottom of both add and edit screens.


Hint

You can use the following tags:

<OTRS_OWNER_*>
Ticket owner options (e. g. <OTRS_OWNER_UserFirstname>).

<OTRS_RESPONSIBLE_*>
Ticket responsible options (e. g. <OTRS_RESPONSIBLE_UserFirstname>).

<OTRS_CURRENT_*>
Options of the current user who requested this action (e. g. <OTRS_CURRENT_UserFirstname>).

<OTRS_TICKET_*>
Options of the ticket data (e. g. <OTRS_TICKET_TicketNumber>, <OTRS_TICKET_TicketID>, <OTRS_TICKET_Queue>, <OTRS_TICKET_State>).

<OTRS_CUSTOMER_DATA_*>
Options of the current customer user data (e. g. <OTRS_CUSTOMER_DATA_UserFirstname>).

<OTRS_CONFIG_*>
Config options (e. g. <OTRS_CONFIG_HttpType>).

Example signature:

```
Your OTRS-Team

<OTRS_CURRENT_UserFirstname> <OTRS_CURRENT_UserLastname>

--
```

Fig. 41: Signature Variables

For example, the variable `<OTRS_CURRENT_UserFirstname> <OTRS_CURRENT_UserLastname>` expands to the agent’s first and last name allowing a template to include something like the following.

```
Best regards,

<OTRS_CURRENT_UserFirstname> <OTRS_CURRENT_UserLastname>
```

This tag expands, for example to:

```
Best regards,

Steven Weber
```

The values of the following variables are translated based on the chosen language of the customer user.

```
<OTRS_TICKET_Type>
<OTRS_TICKET_State>
<OTRS_TICKET_StateType>
<OTRS_TICKET_Lock>
<OTRS_TICKET_Priority>
```

If the language is not supported the default language is applied.

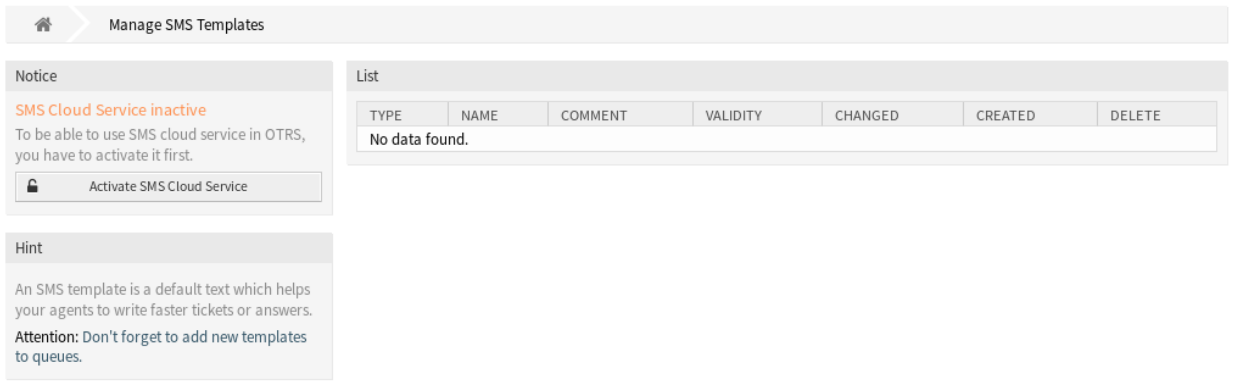
2.13 SMS Templates

An on-call duty should be alarmed about incidents on an email servers, therefore cannot get an email from OTRS. Additionally, in the case where customers have no internet access, it' s imperative to ensure good contact.

OTRS provides SMS as a cloud service and allows, as with email, management of this communication via templates.

An SMS template is a default text which helps your agents to write faster tickets or answers.

Use this screen to add SMS templates for use in communications. A fresh OTRS installation doesn't contain any SMS templates by default. The SMS template management screen is available in the *SMS Templates* module of the *Ticket Settings* group.



SMS Cloud Service inactive

To be able to use SMS cloud service in OTRS, you have to activate it first.

An SMS template is a default text which helps your agents to write faster tickets or answers.

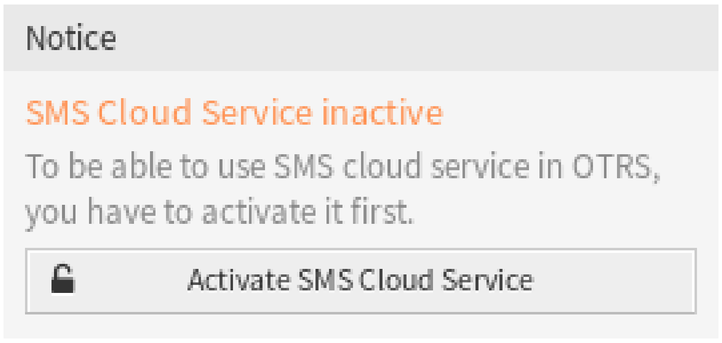
Attention: Don't forget to add new templates to queues.

TYPE	NAME	COMMENT	VALIDITY	CHANGED	CREATED	DELETE
No data found.						

Fig. 42: SMS Template Management Screen

2.13.1 Manage SMS Templates

Note: To be able to use SMS cloud service in OTRS, you have to activate it first in *Cloud Services* module.


Notice

SMS Cloud Service inactive

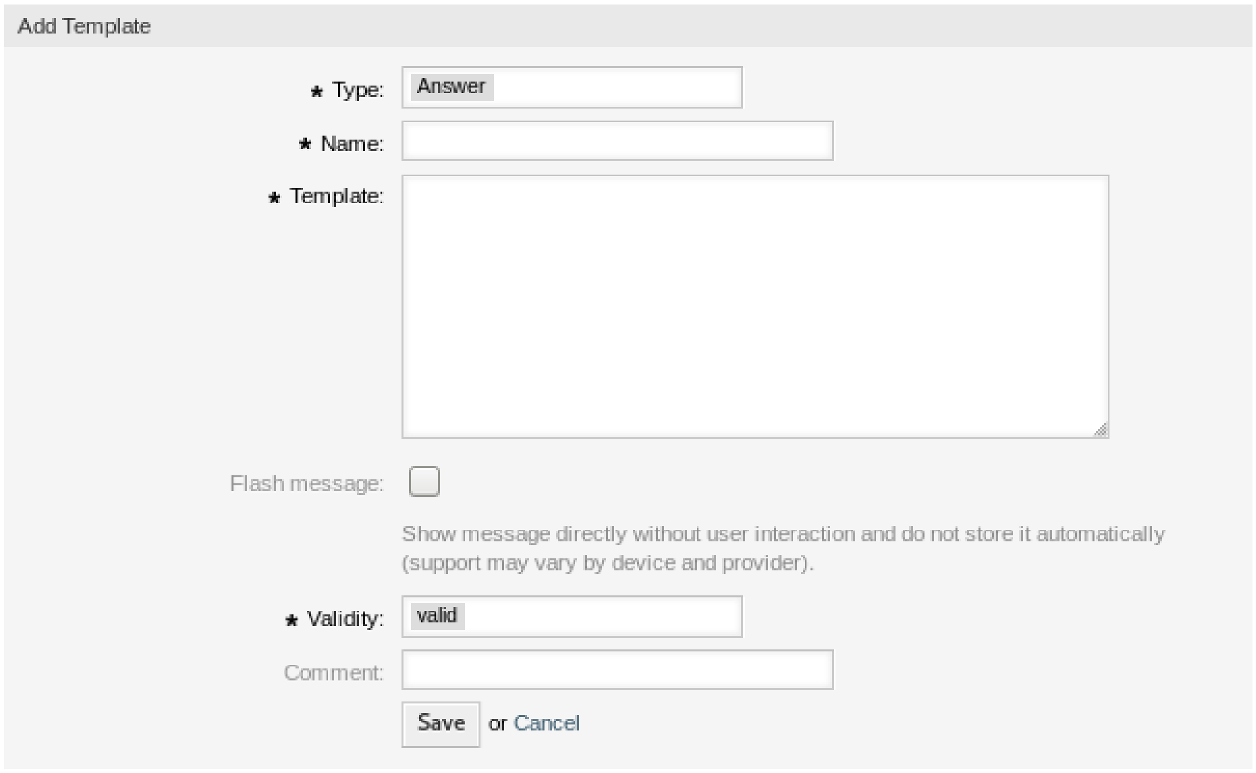
To be able to use SMS cloud service in OTRS, you have to activate it first.

Activate SMS Cloud Service

Fig. 43: Activate SMS Cloud Service

To add an SMS template:

1. Click on the *Add SMS Template* button in the left sidebar.
2. Fill in the required fields.
3. Click on the *Save* button.

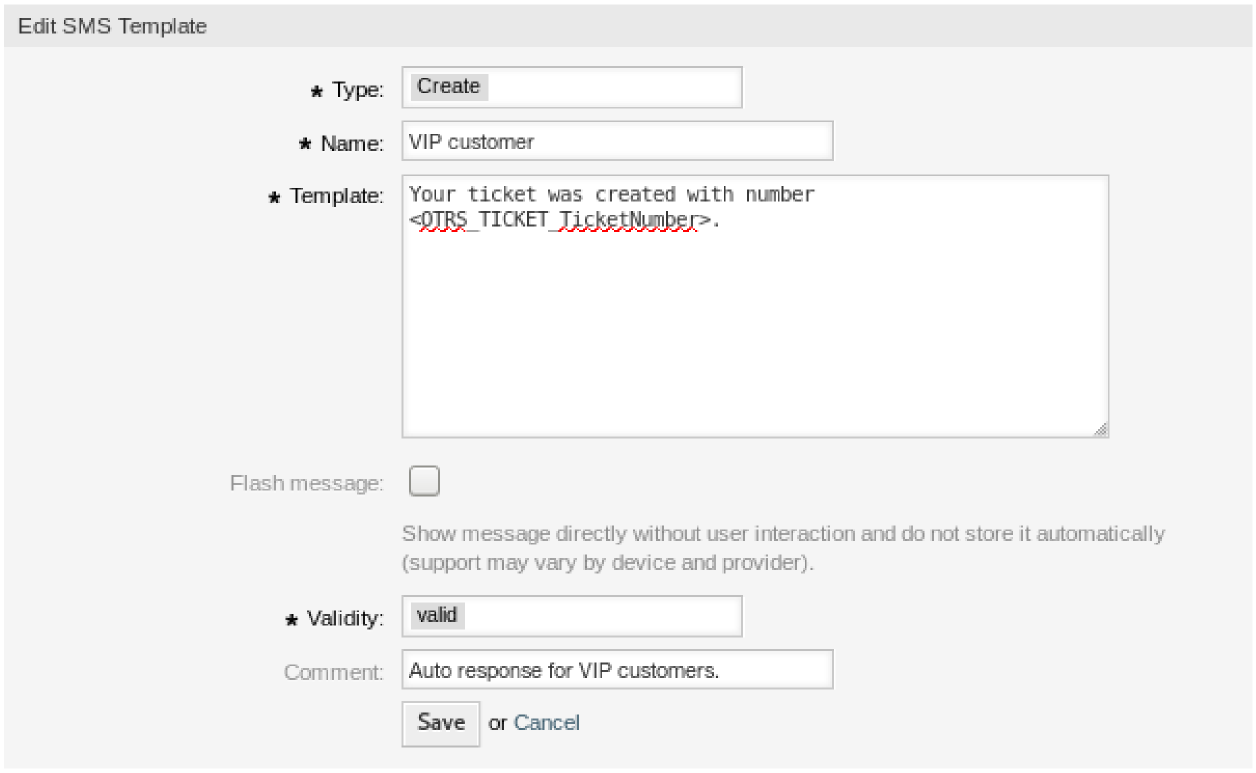

Add Template

* Type:	Answer
* Name:	
* Template:	
Flash message:	☐
	Show message directly without user interaction and do not store it automatically (support may vary by device and provider).
* Validity:	valid
Comment:	

Fig. 44: Add SMS Template Screen

To edit an SMS template:

1. Click on an SMS template in the list of SMS templates.
2. Modify the fields.
3. Click on the *Save* or *Save and finish* button.



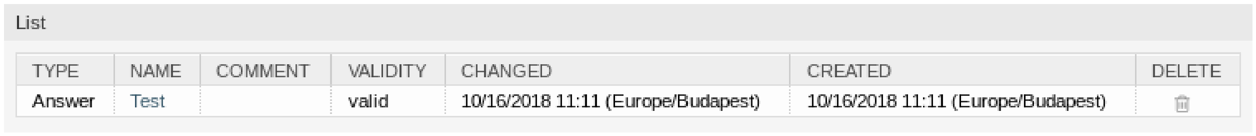
Edit SMS Template

⋆ Type:	Create
⋆ Name:	VIP customer
⋆ Template:	Your ticket was created with number <OTRS_TICKET_TicketNumber>.
Flash message:	☐
	Show message directly without user interaction and do not store it automatically (support may vary by device and provider).
⋆ Validity:	valid
Comment:	Auto response for VIP customers.

Fig. 45: Edit SMS Template Screen

To delete an SMS template:

1. Click on the trash icon in the list of SMS templates.
2. Click on the *Confirm* button.



List

TYPE	NAME	COMMENT	VALIDITY	CHANGED	CREATED	DELETE
Answer	Test		valid	10/16/2018 11:11 (Europe/Budapest)	10/16/2018 11:11 (Europe/Budapest)	🗑

Fig. 46: Delete SMS Template Screen

Note: If several SMS templates are added to the system, a filter box is useful to find a particular SMS template by just typing to filter.

2.13.2 SMS Template Settings

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

Type *
There are different kind of SMS templates that are used for different purposes. A template can be:

Answer
To be used as a ticket response with *Reply via SMS* article action of the ticket detail view.

Create
To be used for a new SMS ticket.

SMSOutbound
To be used for sending a new SMS to a customer user from within the ticket detail view.

Name *
The name of this resource. Any type of characters can be entered to this field including uppercase letters and spaces. The name will be displayed in the overview table.

Template
The body of the email sent to the users.

Warning: The maximum length of text message that you can send is 918 characters. However, if you send more than 160 characters then your message will be broken down in to chunks of 153 characters before being sent to the recipient's handset. We recommend keeping the messages to less than 160 characters.

Flash message
Show message directly without user interaction and do not store it automatically (support may vary by device and provider).

Validity *
Set the validity of this resource. Each resource can be used in OTRS only, if this field is set to *valid*. Setting this field to *invalid* or *invalid-temporarily* will disable the use of the resource.

Comment
Add additional information to this resource. It is recommended to always fill this field as a description of the resource with a full sentence for better clarity, because the comment will be also displayed in the overview table.

2.13.3 SMS Template Variables

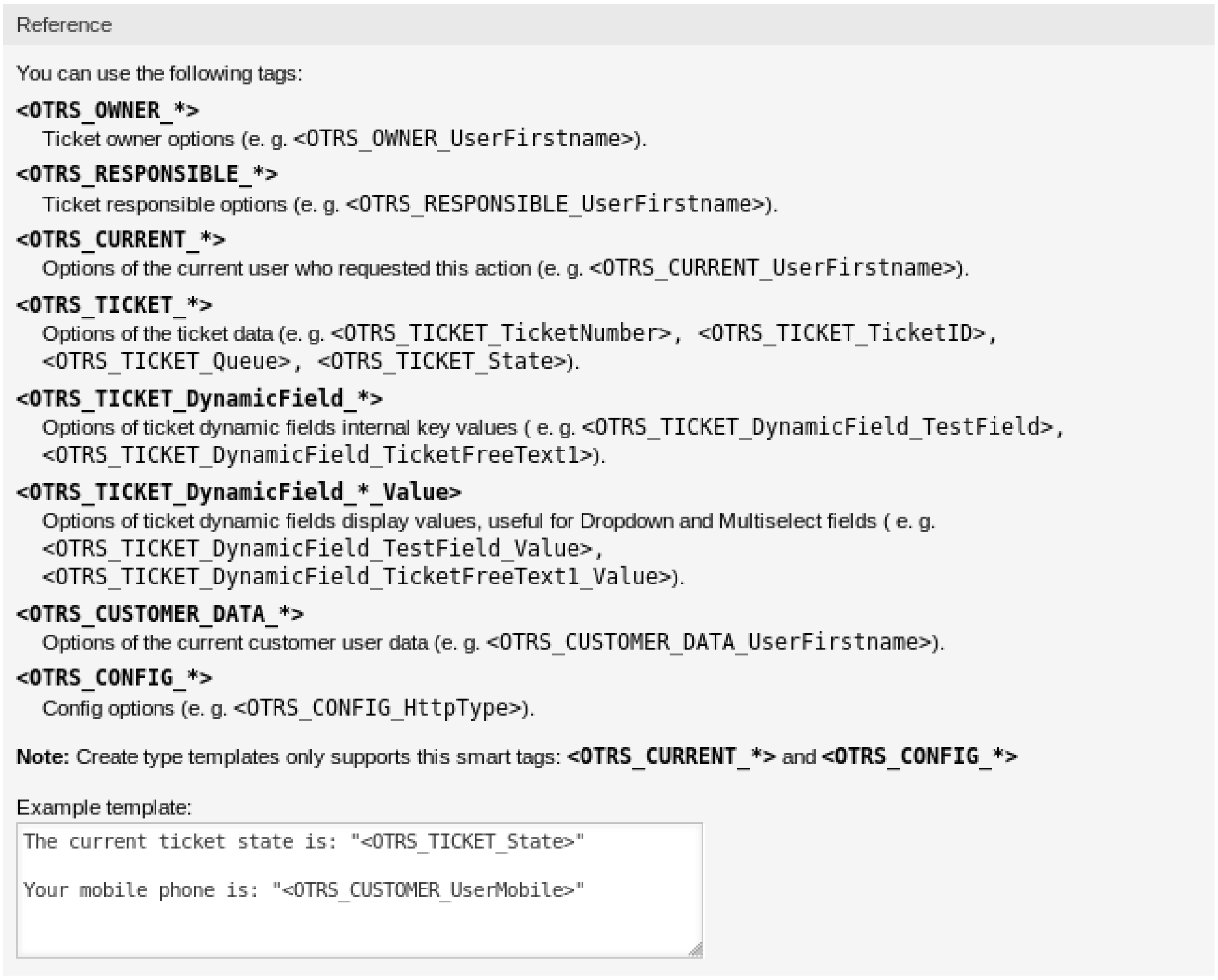
Using variables in the text makes it possible to personalize messages. Variables, known as OTRS tags, are replaced by OTRS when generating the message. Find a list of available tags stems for this resource at the bottom of both add and edit screens.

For example, the variable `<OTRS_TICKET_TicketNumber>` expands to the ticket number allowing an SMS template to include something like the following.

```
Ticket#<OTRS_TICKET_TicketNumber> has been raised in <OTRS_Ticket_Queue>.
```

This tag expands, for example to:

```
Ticket#2018101042000012 has been raised in Postmaster.
```



Reference

You can use the following tags:

<OTRS_OWNER_*>
Ticket owner options (e. g. <OTRS_OWNER_UserFirstname>).

<OTRS_RESPONSIBLE_*>
Ticket responsible options (e. g. <OTRS_RESPONSIBLE_UserFirstname>).

<OTRS_CURRENT_*>
Options of the current user who requested this action (e. g. <OTRS_CURRENT_UserFirstname>).

<OTRS_TICKET_*>
Options of the ticket data (e. g. <OTRS_TICKET_TicketNumber>, <OTRS_TICKET_TicketID>, <OTRS_TICKET_Queue>, <OTRS_TICKET_State>).

<OTRS_TICKET_DynamicField_*>
Options of ticket dynamic fields internal key values (e. g. <OTRS_TICKET_DynamicField_TestField>, <OTRS_TICKET_DynamicField_TicketFreeText1>).

<OTRS_TICKET_DynamicField_*_Value>
Options of ticket dynamic fields display values, useful for Dropdown and Multiselect fields (e. g. <OTRS_TICKET_DynamicField_TestField_Value>, <OTRS_TICKET_DynamicField_TicketFreeText1_Value>).

<OTRS_CUSTOMER_DATA_*>
Options of the current customer user data (e. g. <OTRS_CUSTOMER_DATA_UserFirstname>).

<OTRS_CONFIG_*>
Config options (e. g. <OTRS_CONFIG_HttpType>).

Note: Create type templates only supports this smart tags: **<OTRS_CURRENT_*>** and **<OTRS_CONFIG_*>**

Example template:

```
The current ticket state is: "<OTRS_TICKET_State>"

Your mobile phone is: "<OTRS_CUSTOMER_UserMobile>"
```

Fig. 47: SMS Template Variables

The values of the following variables are translated based on the chosen language of the customer user.

```
<OTRS_TICKET_Type>
<OTRS_TICKET_State>
<OTRS_TICKET_StateType>
<OTRS_TICKET_Lock>
<OTRS_TICKET_Priority>
```

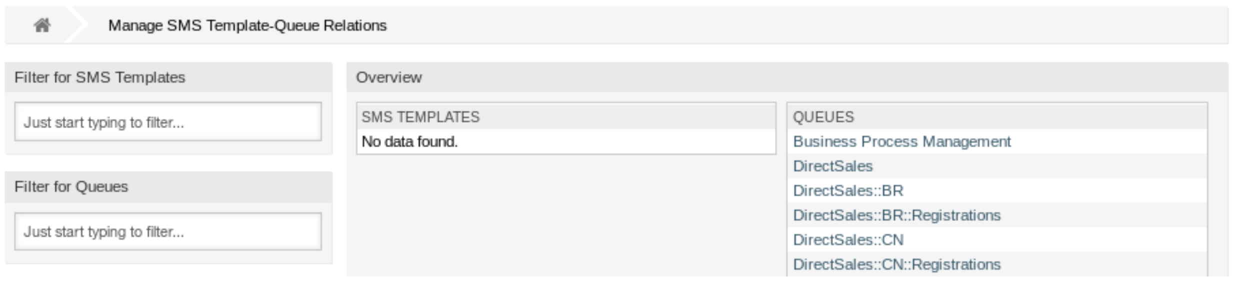
If the language is not supported the default language is applied.

2.14 SMS Templates Queues

Communicating a change to a template, or distribution of a new communication requirement, can be tedious and error-ridden because it requires that all teams pull the information and publish it appropriately.

OTRS gives you the power to quickly assign the appropriate SMS templates to any queue, containing pertinent ticket information, ensuring this information reaches your customers and agents.

Use this screen to add one or more SMS templates to one or more queues. The management screen is available in the *SMS Templates Queues* module of the *Ticket Settings* group.


Manage SMS Template-Queue Relations
Filter for SMS Templates
Just start typing to filter...
Filter for Queues
Just start typing to filter...
Overview

SMS TEMPLATES
No data found.

QUEUES
Business Process Management
DirectSales
DirectSales::BR
DirectSales::BR::Registrations
DirectSales::CN
DirectSales::CN::Registrations

Fig. 48: Manage SMS Template-Queue Relations

2.14.1 Manage SMS Templates Queues Relations

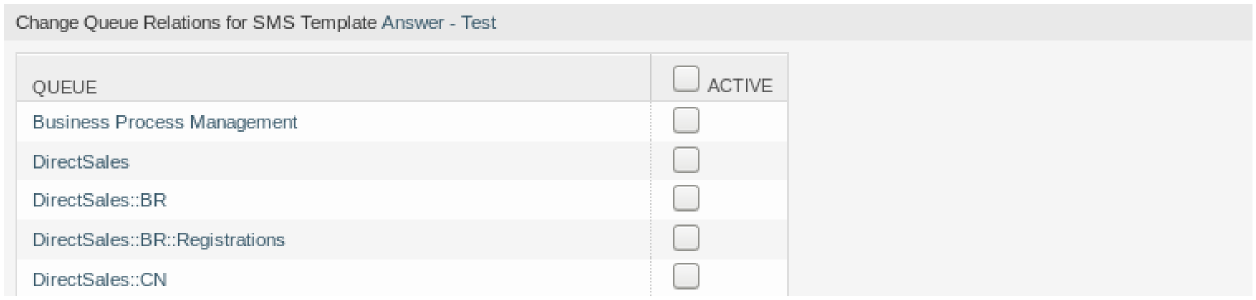
To assign an SMS template to one or more queues:

1. Click on an SMS template in the *SMS Templates* column.
2. Select the queues you would like to add the SMS template to.
3. Click on the *Save* or *Save and finish* button.

To assign a queue to one or more SMS templates:

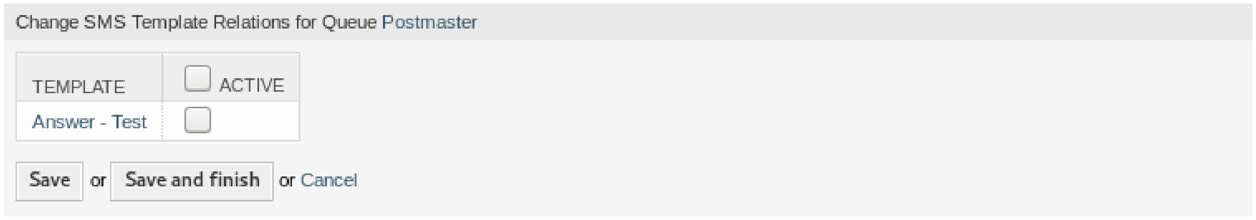
1. Click on a queue in the *Queues* column.
2. Select the SMS templates you would like to add the queue to.
3. Click on the *Save* or *Save and finish* button.

Note: If several SMS templates or queues are added to the system, use the filter box to find a particular SMS template or queue by just typing the name to filter.


Change Queue Relations for SMS Template Answer - Test

QUEUE	ACTIVE
Business Process Management	☐
DirectSales	☐
DirectSales::BR	☐
DirectSales::BR::Registrations	☐
DirectSales::CN	☐

Fig. 49: Change Queue Relations for SMS Template


Change SMS Template Relations for Queue Postmaster

TEMPLATE	ACTIVE
Answer - Test	☐

Fig. 50: Change SMS Template Relations for Queue

Multiple SMS templates or queues can be assigned in both screens at the same time. Additionally clicking on an SMS template or clicking on a queue in the relations screen will open the *Edit SMS Template* screen or the *Edit Queue* screen accordingly.

Warning: Accessing a queue or an SMS template provides no back link to the relations screen.

2.15 States

Active tracking of tickets leads to a better sense of workload and provides metrics as a key performance indicator. Sorting tasks and setting appointments can help to level-off the workload and keep your service desk running.

OTRS uses ticket states to ensure that your agents always know which tickets are being attended to and which not. Additionally, detailed reports on the states of your tickets can be provided by ticket search or reports and personalized sorting is possible using dashboards and queue and service overviews.

Nine states are pre-defined. More states can be added, but the default states are enough to get you going and mostly enough for any situation.

The default states are the following:

closed successful
A ticket is complete. The customer received a solution which worked.

closed unsuccessful
A ticket is complete. The customer received no solution or the solution was not appropriate.

merged
The ticket content is found in a different ticket.

new
The ticket is created by the customer without contact with an agent.

open
The ticket is currently in progress. Customer and agent are in contact with each another.

pending auto close+
The ticket will be set as *closed successful* upon reaching the pending time.

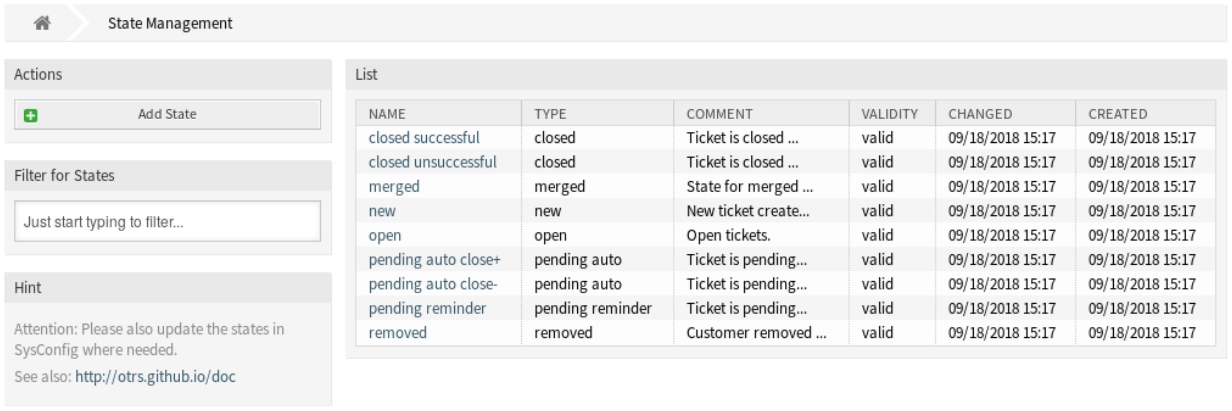
pending auto close-
The ticket will be set as *closed unsuccessful* upon reaching the pending time.

pending reminder
The ticket should be worked on again upon reaching the pending time.

removed
The ticket has been removed from the system.

Note: Pending jobs are checked per default every two hours and forty-five minutes. This time is a static time, which means the times are 02:45, 04:45, 06:45 and so on. The job can be run more often or seldom and are configured in the *System Configuration* module of the *Administration* group.

Use this screen to add states to the system. A fresh OTRS installation contains several states by default. The state management screen is available in the *States* module of the *Ticket Settings* group.



Attention: Please also update the states in SysConfig where needed.

See also: http://otrs.github.io/doc

NAME	TYPE	COMMENT	VALIDITY	CHANGED	CREATED
closed successful	closed	Ticket is closed ...	valid	09/18/2018 15:17	09/18/2018 15:17
closed unsuccessful	closed	Ticket is closed ...	valid	09/18/2018 15:17	09/18/2018 15:17
merged	merged	State for merged ...	valid	09/18/2018 15:17	09/18/2018 15:17
new	new	New ticket create...	valid	09/18/2018 15:17	09/18/2018 15:17
open	open	Open tickets.	valid	09/18/2018 15:17	09/18/2018 15:17
pending auto close+	pending auto	Ticket is pending...	valid	09/18/2018 15:17	09/18/2018 15:17
pending auto close-	pending auto	Ticket is pending...	valid	09/18/2018 15:17	09/18/2018 15:17
pending reminder	pending reminder	Ticket is pending...	valid	09/18/2018 15:17	09/18/2018 15:17
removed	removed	Customer removed ...	valid	09/18/2018 15:17	09/18/2018 15:17

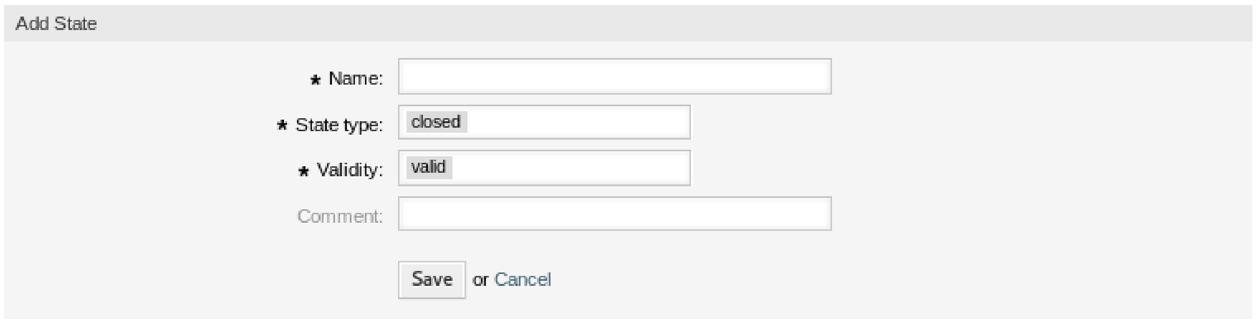
Fig. 51: State Management Screen

2.15.1 Manage States

To add a state:

1. Click on the *Add State* button in the left sidebar.
2. Fill in the required fields.
3. Click on the *Save* button.

Warning: States can not be deleted from the system. They can only be deactivated by setting the *Validity* option to *invalid* or *invalid-temporarily*.


Add State

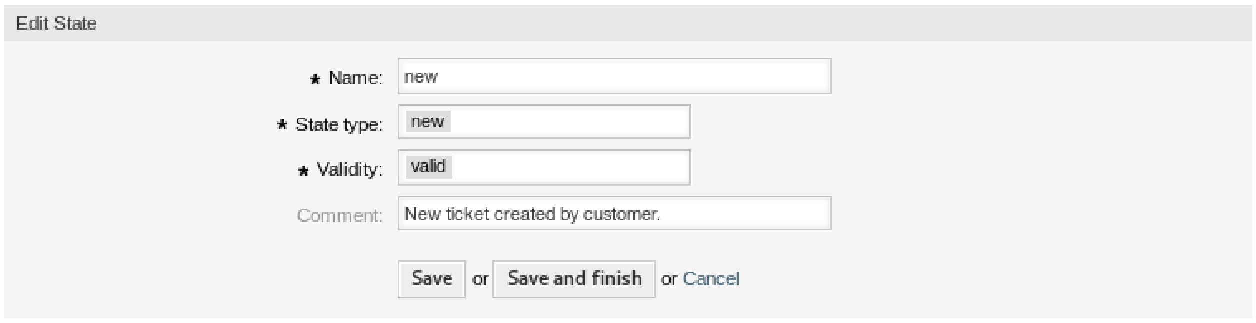
⋆ Name:	
⋆ State type:	closed
⋆ Validity:	valid
Comment:	

Fig. 52: Add State Screen

Warning: The maximum number of 20 *valid* states should not be exceeded. Exceeding this limit may affect the system performance.

To edit a state:

1. Click on a state in the list of states.
2. Modify the fields.
3. Click on the *Save* or *Save and finish* button.


Edit State

⋆ Name:	new
⋆ State type:	new
⋆ Validity:	valid
Comment:	New ticket created by customer.

Fig. 53: Edit State Screen

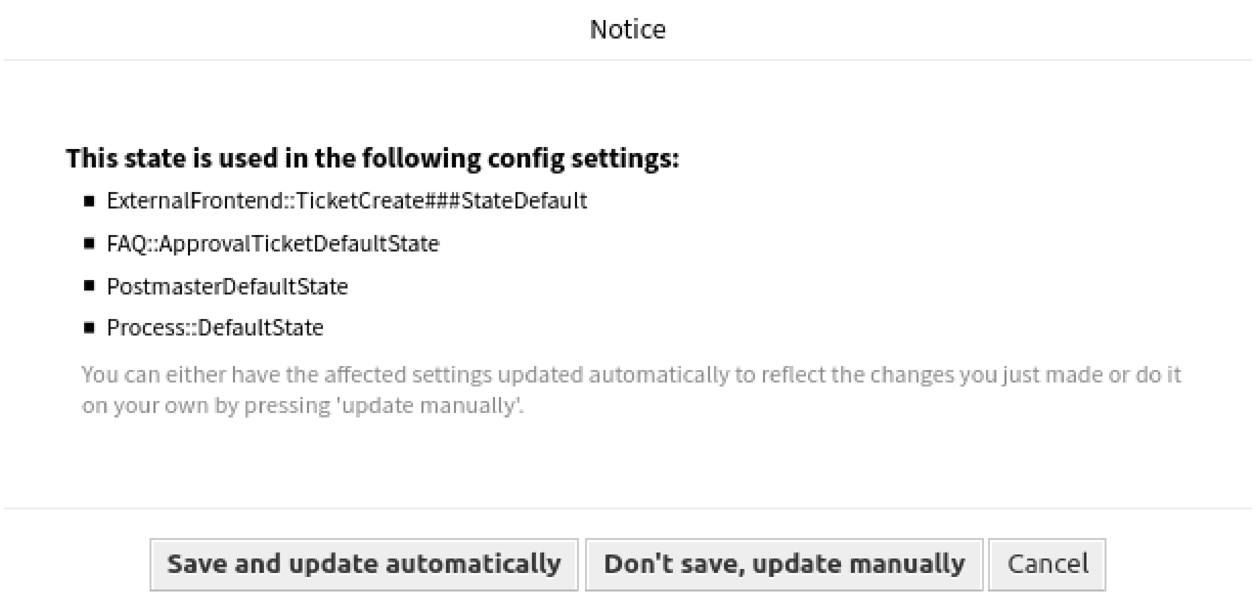
Note: If several states are added to the system, use the filter box to find a particular state by just typing the name to filter.

If you change the name of a state which is used in the system configuration, a validation check will warn you and give you the following options:

Save and update automatically
Apply the change and also update the affected settings.

Don' t save, update manually
Apply the change, but don' t update the affected settings. The updates need to be done manually.

Cancel
Cancel the action.



This state is used in the following config settings:

- ExternalFrontend::TicketCreate###StateDefault
- FAQ::ApprovalTicketDefaultState
- PostmasterDefaultState
- Process::DefaultState

You can either have the affected settings updated automatically to reflect the changes you just made or do it on your own by pressing 'update manually'.

Fig. 54: State Validation Dialog

Warning: Changing the name of this object should be done with care, the check only provides verification for certain settings and ignores things where the name can't be verified. Some examples are dashboard filters, access control lists (ACLs), and processes (sequence flow actions) to name a few. Documentation of your setup is key to surviving a name change.

2.15.2 State Settings

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

Name *
The name of this resource. Any type of characters can be entered to this field including uppercase letters and spaces. The name will be displayed in the overview table.

State type *
Every state is linked to a type, which needs to be specified if a new state is created or an existing one is edited. The following types are available:

- closed
- merged
- new
- open
- pending auto
- pending reminder
- removed

Note: State types are predefined and cannot be changed in the software due to their special mechanics. When adding new states for *pending auto* and *pending reminder* state types you must make further configurations in the *System Configuration* module of the *Administration* group.

Validity *
Set the validity of this resource. Each resource can be used in OTRS only, if this field is set to *valid*. Setting this field to *invalid* or *invalid-temporarily* will disable the use of the resource.

Comment
Add additional information to this resource. It is recommended to always fill this field as a description of the resource with a full sentence for better clarity, because the comment will be also displayed in the overview table.

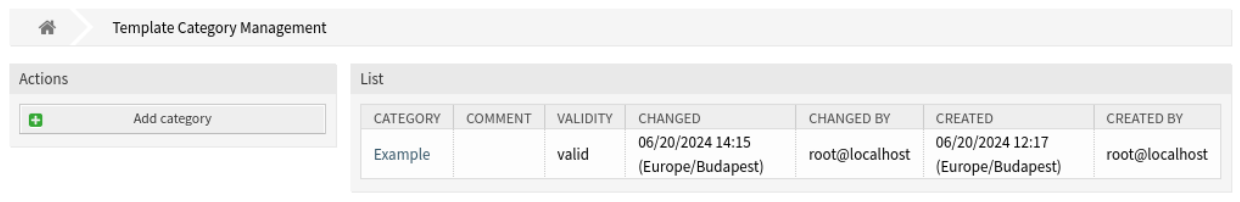
See also:

The following configuration options are relevant and noteworthy when managing states:

- Daemon::SchedulerCronTaskManager::Task###TicketPendingCheck
- Ticket::StateAfterPending

2.16 Template Categories

Use this screen to add categories to the system. The category management screen is available in the *Categories* module of the *Ticket Settings* group.



CATEGORY	COMMENT	VALIDITY	CHANGED	CHANGED BY	CREATED	CREATED BY
Example		valid	06/20/2024 14:15 (Europe/Budapest)	root@localhost	06/20/2024 12:17 (Europe/Budapest)	root@localhost

Fig. 55: Template Category Management Screen

2.16.1 Manage Template Categories

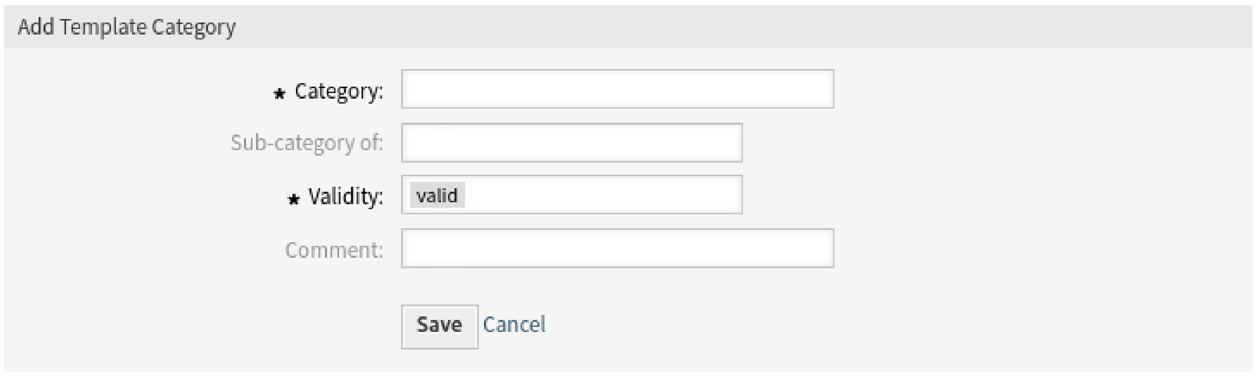
To add a template category:

1. Click on the *Add category* button in the left sidebar.
2. Fill in the required fields.
3. Click on the *Save* button.

Warning: Template categories can not be deleted from the system. They can only be deactivated by setting the *Validity* option to *invalid* or *invalid-temporarily*.

To edit a template category:

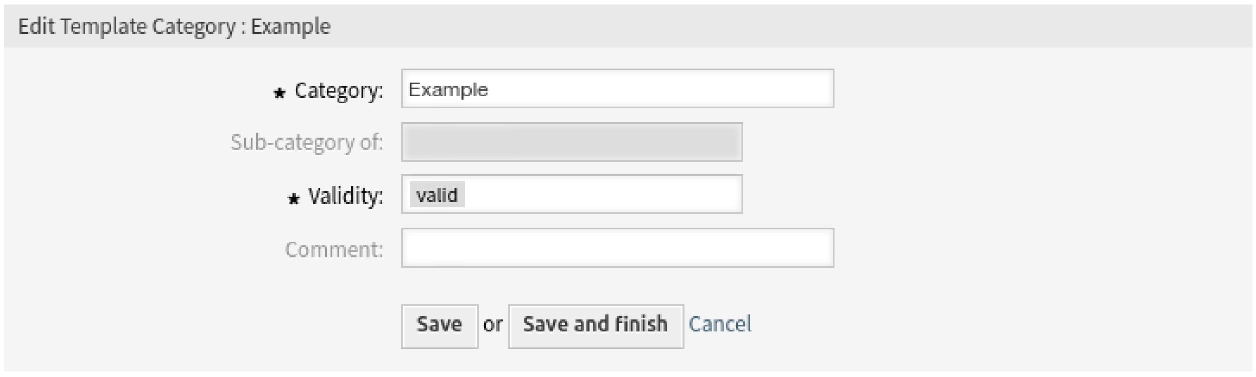
1. Click on a template category in the list of template categories.


Add Template Category

★ Category:	
Sub-category of:	
★ Validity:	valid
Comment:	

Fig. 56: Add Template Category Screen

2. Modify the fields.
3. Click on the *Save* or *Save and finish* button.


Edit Template Category : Example

★ Category:	Example
Sub-category of:	
★ Validity:	valid
Comment:	

Fig. 57: Edit Template Category Screen

2.16.2 Template Category Settings

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

Category *
The name of this resource. Any type of characters can be entered to this field including uppercase letters and spaces. The name will be displayed in the overview table.

Sub-category of
It is possible to add the new category under an existing one as sub-category. This will be displayed as *Parent Category::Child Category*.

Valid *
Set the validity of this resource. Each resource can be used in OTRS only, if this field is set to *valid*. Setting this field to *invalid* or *invalid-temporarily* will disable the use of the resource.

Comment
Add additional information to this resource. It is recommended to always fill this field as a description of the resource with a full sentence for better clarity, because the comment will be also displayed in the overview table.

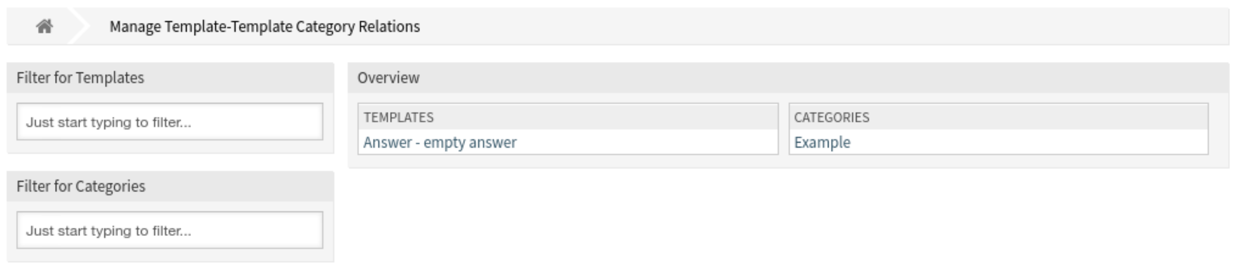
2.17 Template Categories   Templates

Service agents can quickly respond to normal or recurring service requests by leveraging pre-defined templates to answer the tickets. This feature allows you to group certain text modules by category and implements a category folder structure for standard templates.

For example, a typical recurring question is the request to send out a user guide for one of your specific products. Today, you would probably browse down the long list of your pre-defined templates and, once you finally have identified the right one with the user guide attached, you would send it out.

You can consolidate all text modules related to your products' user guides in a separate category like *User Guides*. You can find it quickly, saving a lot of time trying to retrieve this information. This reduces your response time.

Use this screen to add one or more template categories to one or more templates. To use this function, at least one template category and one template need to have been added to the system. The management screen is available in the *Template Categories   Templates* module of the *Ticket Settings* group.


Manage Template-Template Category Relations
Filter for Templates
Just start typing to filter...
Filter for Categories
Just start typing to filter...
Overview

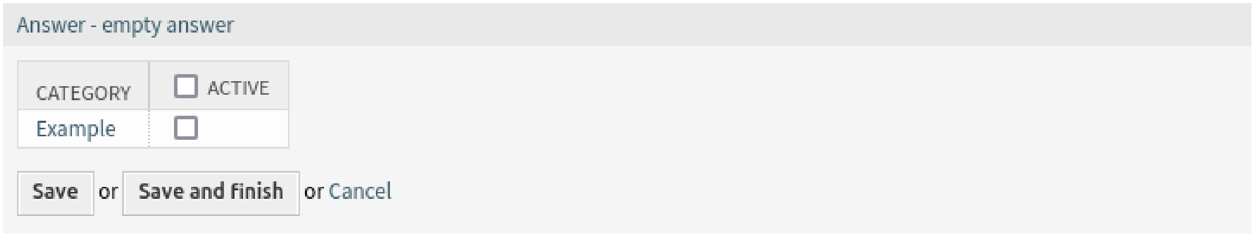
TEMPLATES	CATEGORIES
Answer - empty answer	Example

Fig. 58: Manage Template Category-Template Relations Screen

2.17.1 Manage Template Categories   Templates Relations

To assign a template to a category:

1. Click on a template in the *Templates* column.
2. Select the category you would like to add the template to.
3. Click on the *Save* or *Save and finish* button.



CATEGORY	ACTIVE
Example	☐

Fig. 59: Change Category Relations for Template

To assign a category to a template:

1. Click on a category in the *Categories* column.
2. Select the templates you would like to assign the category to.

3. Click on the *Save* or *Save and finish* button.

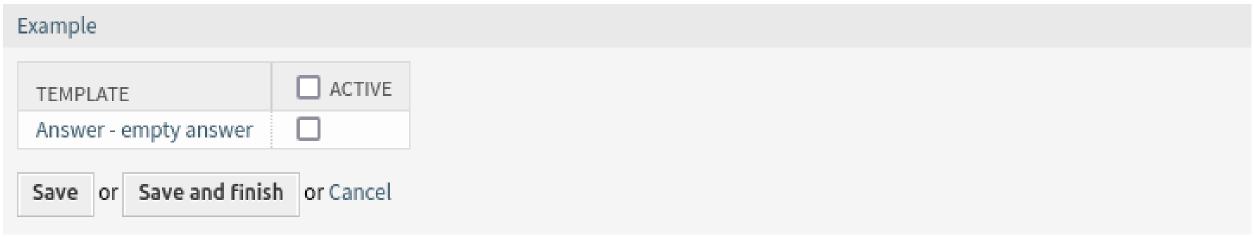


Fig. 60: Change Template Relations for Category

Note: If several templates or categories are added to the system, use the filter box to find a particular template or category by just typing the name to filter.

Multiple templates or categories can be assigned in both screens at the same time. Additionally clicking on a template or clicking on a category in the relations screen will open the *Edit Template* screen or the *Edit Category* screen accordingly.

Warning: Accessing a category or a template provides no back link to the relations screen.

2.18 Templates

Providing the correct and consistent answer all the time regardless of employee or knowledge-level is important to maintain a professional appearance to your customers. Additionally, speed in sending standard answers is key to wading through the masses of requests in growing service desks.

OTRS templates offer you a variety of ways to deal with standardizing communications and help to pre-define texts so that the customer always receives the same level and quality of service from all agents.

Use this screen to add templates for use in communications. A fresh OTRS installation already contains a template by default. The template management screen is available in the *Templates* module of the *Ticket Settings* group.

2.18.1 Manage Templates

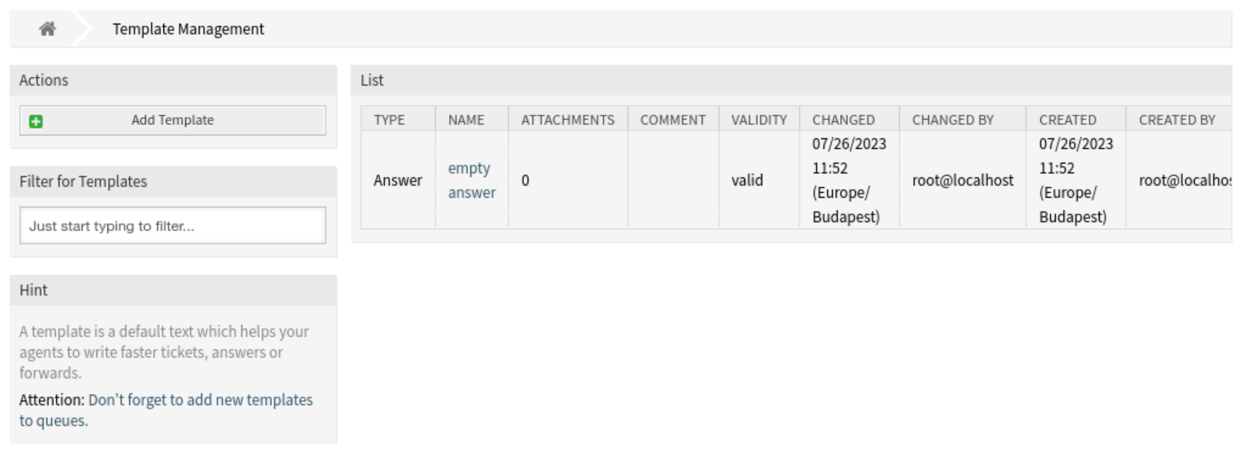
Note: To add attachments to a template, it needs to create the attachment first in the *Attachments* screen.

To add a template:

1. Click on the *Add Template* button in the left sidebar.
2. Fill in the required fields.
3. Click on the *Save* button.

To edit a template:

1. Click on a template in the list of templates.



Hint

A template is a default text which helps your agents to write faster tickets, answers or forwards.

Attention: Don't forget to add new templates to queues.

TYPE	NAME	ATTACHMENTS	COMMENT	VALIDITY	CHANGED	CHANGED BY	CREATED	CREATED BY
Answer	empty answer	0		valid	07/26/2023 11:52 (Europe/ Budapest)	root@localhost	07/26/2023 11:52 (Europe/ Budapest)	root@localhos

Fig. 61: Template Management Screen

2. Modify the fields.
3. Click on the *Save* or *Save and finish* button.

To delete a template:

1. Click on the trash icon in the list of templates.
2. Click on the *Confirm* button.

Note: If several templates are added to the system, a filter box is useful to find a particular template by just typing to filter.

2.18.2 Template Settings

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

Type *
There are different kind of templates that are used for different purposes. A template can be:

Answer
To be used as a ticket response or reply.

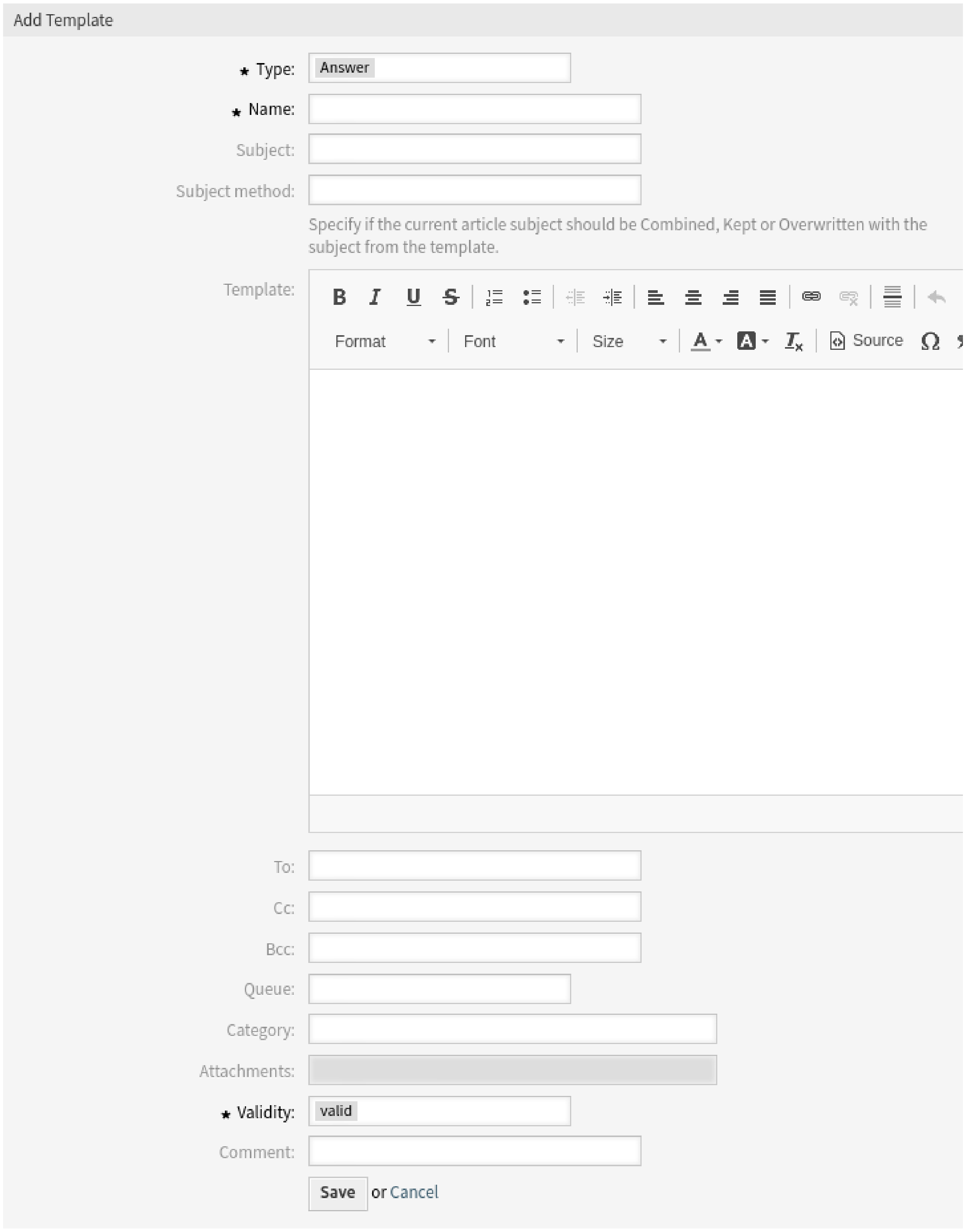
Create
To be used for new phone or email ticket.

Email
To be used for writing an email to a customer user or to someone else.

Forward
To be used to forward an article to someone else.

Note
To be used to create notes.

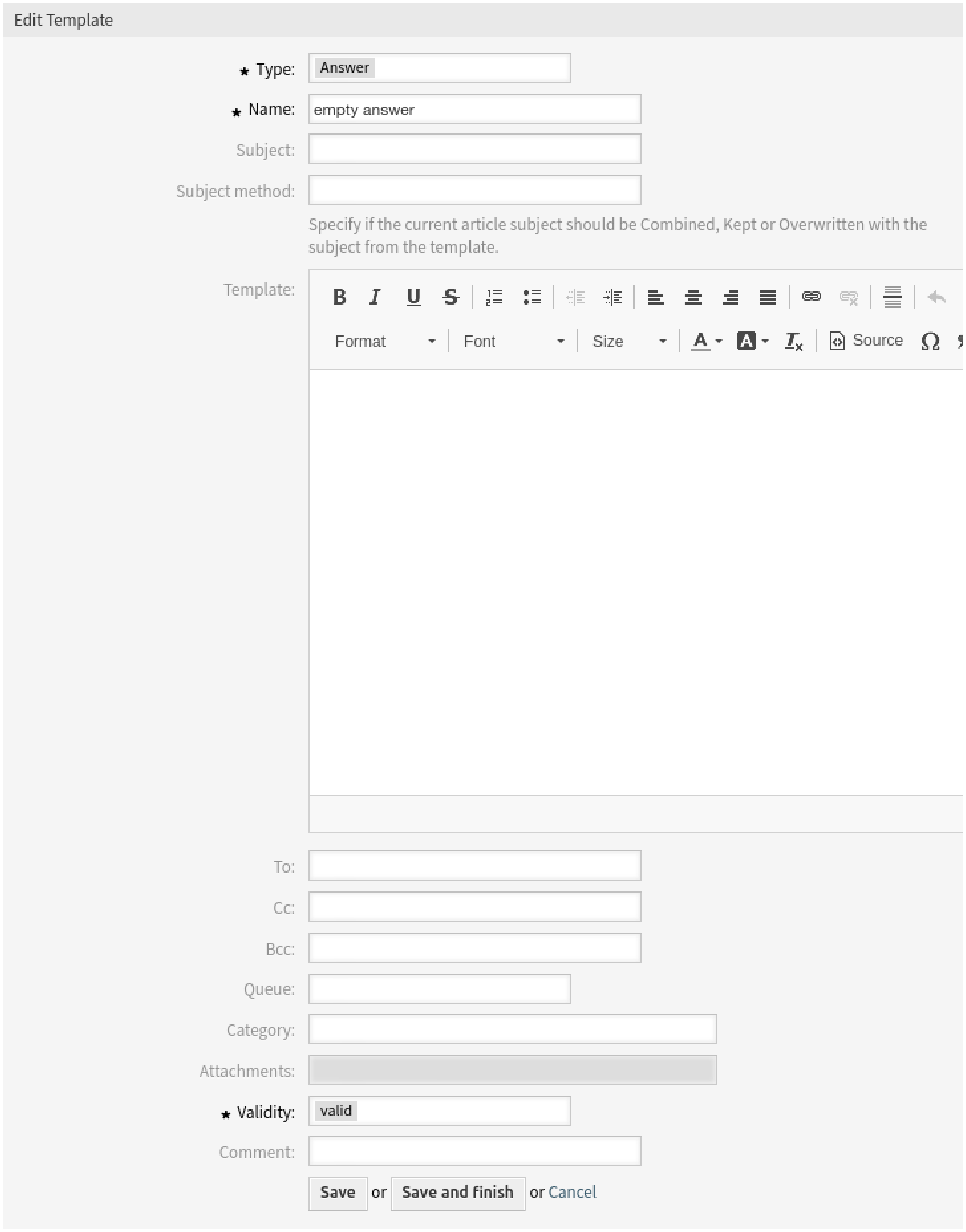
Phone call
To be used for inbound and outbound phone calls.



Add Template

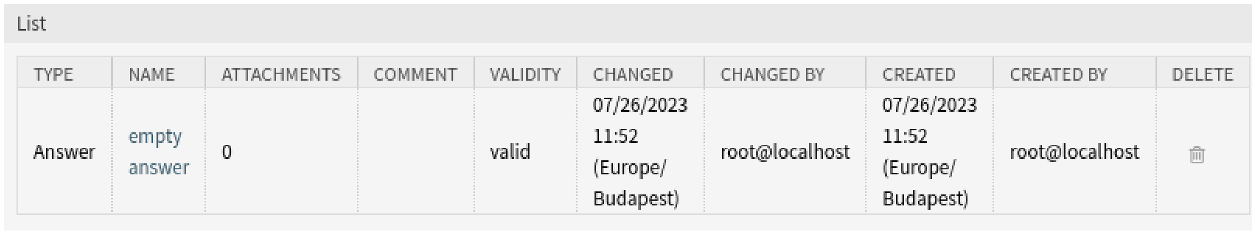
★ Type:	Answer
★ Name:	
Subject:	
Subject method:	Specify if the current article subject should be Combined, Kept or Overwritten with the subject from the template.
Template:	
To:	
Cc:	
Bcc:	
Queue:	
Category:	
Attachments:	
★ Validity:	valid
Comment:	

Fig. 62: Add Template Screen


Edit Template

★ Type:	Answer
★ Name:	empty answer
Subject:	
Subject method:	
	Specify if the current article subject should be Combined, Kept or Overwritten with the subject from the template.
Template:	
To:	
Cc:	
Bcc:	
Queue:	
Category:	
Attachments:	
★ Validity:	valid
Comment:	

Fig. 63: Edit Template Screen


List

TYPE	NAME	ATTACHMENTS	COMMENT	VALIDITY	CHANGED	CHANGED BY	CREATED	CREATED BY	DELETE
Answer	empty answer	0		valid	07/26/2023 11:52 (Europe/ Budapest)	root@localhost	07/26/2023 11:52 (Europe/ Budapest)	root@localhost	🗑

Fig. 64: Delete Template Screen

Process dialog
: To be used in process management to pre-fill text of articles in user task activity dialogs.

Name *
: The name of this resource. Any type of characters can be entered to this field including uppercase letters and spaces. The name will be displayed in the overview table.

Subject
: The subject of the article added by the template.

Subject method
: Specify how the current article subject should be dealt with. The following methods are available:

  Combine
  : The template subject will be added after the current article subject.

  Keep
  : The current article subject will be kept.

  Overwrite
  : The current article subject will be replaced with the template subject.

Template
: The body of the article added by the template.

To
: The person, to whom the ticket is created for. This will be the *To* field of the email.

Cc
: Other addresses as carbon copy of the email, if needed.

Bcc
: Other addresses as blind carbon copy of the email, if needed.

Queue
: The queue where the ticket should be placed to use the template.

State
: Select a ticket state to assign it to the template. This ticket state will be pre-selected on reply actions when answering a ticket using this response template.

  This setting is only available if *Answer* is selected in the *Type* field.

Category
: The category for the text template. The category can be also assigned in the *Template Categories Templates* screen.

Attachments
: It is possible to add one or more attachments to this template. Attachments can be added in the *Attachments* screen.

Validity *
Set the validity of this resource. Each resource can be used in OTRS only, if this field is set to *valid*. Setting this field to *invalid* or *invalid-temporarily* will disable the use of the resource.

Comment
Add additional information to this resource. It is recommended to always fill this field as a description of the resource with a full sentence for better clarity, because the comment will be also displayed in the overview table.

2.18.3 Template Variables

Using variables in the text makes it possible to personalize messages. Variables, known as OTRS tags, are replaced by OTRS when generating the message. Find a list of available tags stems for this resource at the bottom of both add and edit screens.

For example, the variable `<OTRS_TICKET_TicketNumber>` expands to the ticket number allowing a template to include something like the following.

```
Ticket#<OTRS_TICKET_TicketNumber>
```

This tag expands, for example to:

```
Ticket#2018101042000012
```

The values of the following variables are translated based on the chosen language of the customer user.

```
<OTRS_TICKET_Type>
<OTRS_TICKET_State>
<OTRS_TICKET_StateType>
<OTRS_TICKET_Lock>
<OTRS_TICKET_Priority>
```

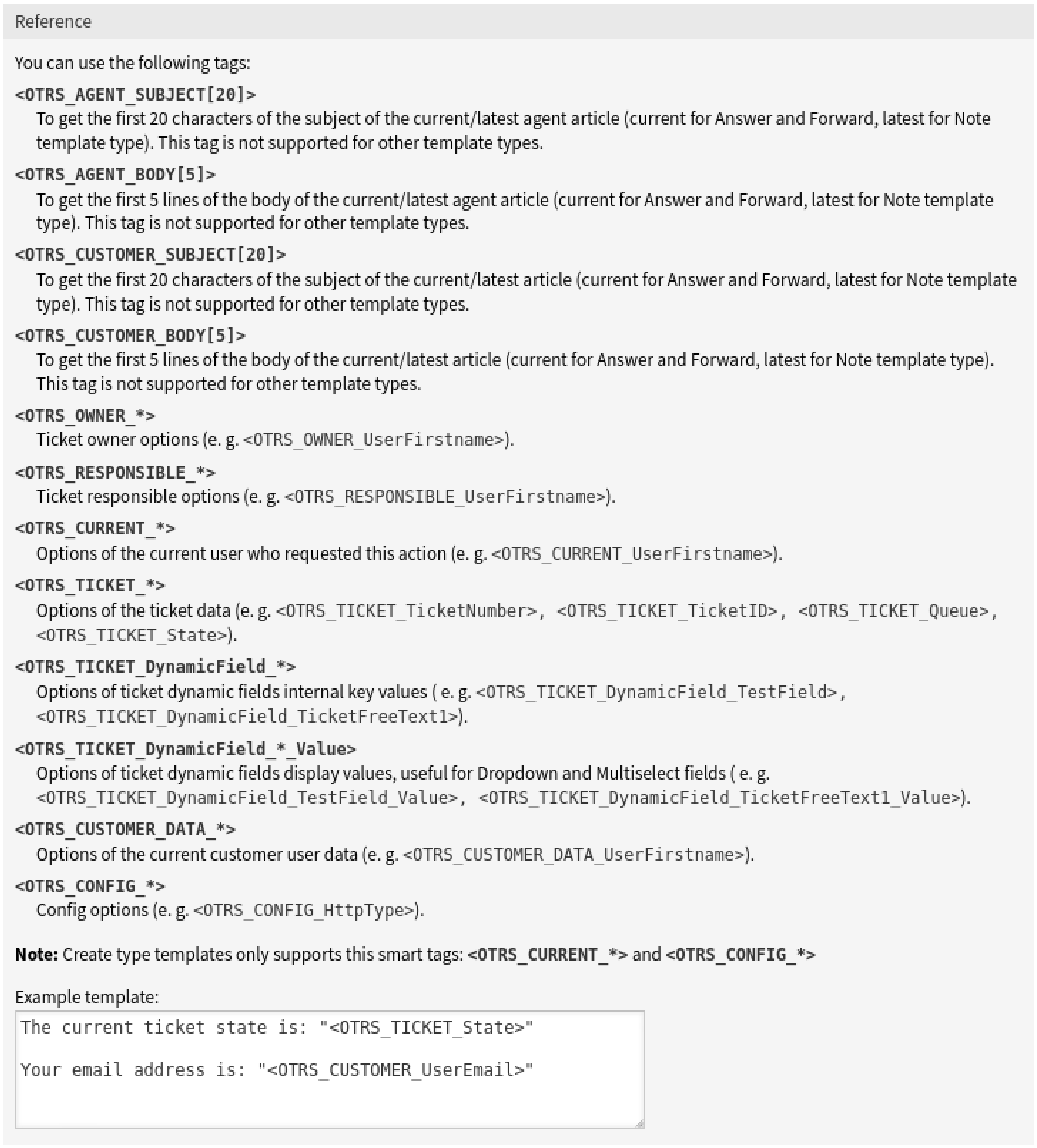
If the language is not supported the default language is applied.

2.19 Templates   Attachments

Making changes to standard attachments is sometimes overwhelming. The question is, “Where were the attachments used?” or “How to quickly update them across the board?”. It’s also important to know who are using which attachments, before updating them. A new attachment can have multiple uses within your organization.

OTRS empowers you to manage this providing you an overview to manage the 1:n relationships and quickly identify the templates using your attachments.

Use this screen to add one or more attachments to one or more templates. To use this function, at least one attachment and one template need to have been added to the system. The management screen is available in the *Templates   Attachments* module of the *Ticket Settings* group.



Reference

You can use the following tags:

<OTRS_AGENT_SUBJECT[20]>
To get the first 20 characters of the subject of the current/latest agent article (current for Answer and Forward, latest for Note template type). This tag is not supported for other template types.

<OTRS_AGENT_BODY[5]>
To get the first 5 lines of the body of the current/latest agent article (current for Answer and Forward, latest for Note template type). This tag is not supported for other template types.

<OTRS_CUSTOMER_SUBJECT[20]>
To get the first 20 characters of the subject of the current/latest article (current for Answer and Forward, latest for Note template type). This tag is not supported for other template types.

<OTRS_CUSTOMER_BODY[5]>
To get the first 5 lines of the body of the current/latest article (current for Answer and Forward, latest for Note template type). This tag is not supported for other template types.

<OTRS_OWNER_*>
Ticket owner options (e. g. `<OTRS_OWNER_UserFirstname>`).

<OTRS_RESPONSIBLE_*>
Ticket responsible options (e. g. `<OTRS_RESPONSIBLE_UserFirstname>`).

<OTRS_CURRENT_*>
Options of the current user who requested this action (e. g. `<OTRS_CURRENT_UserFirstname>`).

<OTRS_TICKET_*>
Options of the ticket data (e. g. `<OTRS_TICKET_TicketNumber>, <OTRS_TICKET_TicketID>, <OTRS_TICKET_Queue>, <OTRS_TICKET_State>`).

<OTRS_TICKET_DynamicField_*>
Options of ticket dynamic fields internal key values (e. g. `<OTRS_TICKET_DynamicField_TestField>, <OTRS_TICKET_DynamicField_TicketFreeText1>`).

<OTRS_TICKET_DynamicField_*_Value>
Options of ticket dynamic fields display values, useful for Dropdown and Multiselect fields (e. g. `<OTRS_TICKET_DynamicField_TestField_Value>, <OTRS_TICKET_DynamicField_TicketFreeText1_Value>`).

<OTRS_CUSTOMER_DATA_*>
Options of the current customer user data (e. g. `<OTRS_CUSTOMER_DATA_UserFirstname>`).

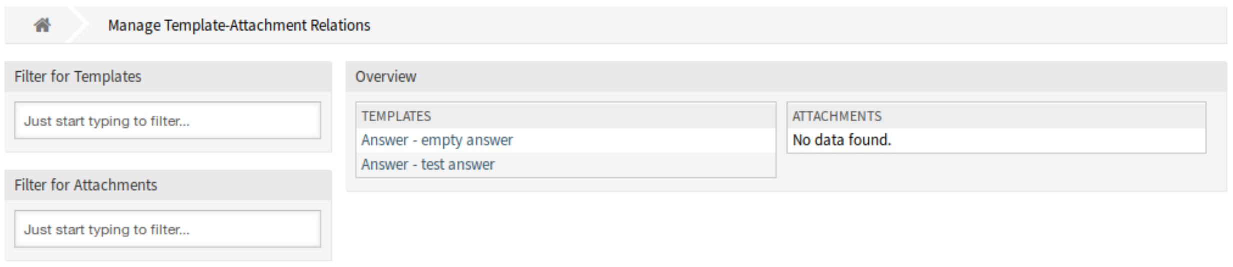
<OTRS_CONFIG_*>
Config options (e. g. `<OTRS_CONFIG_HttpType>`).

Note: Create type templates only supports this smart tags: **<OTRS_CURRENT_*>** and **<OTRS_CONFIG_*>**

Example template:

```
The current ticket state is: "<OTRS_TICKET_State>"

Your email address is: "<OTRS_CUSTOMER_UserEmail>"
```

Fig. 65: Template Variables



Manage Template-Attachment Relations

Filter for Templates

Just start typing to filter...

Filter for Attachments

Just start typing to filter...

Overview

TEMPLATES
Answer - empty answer
Answer - test answer

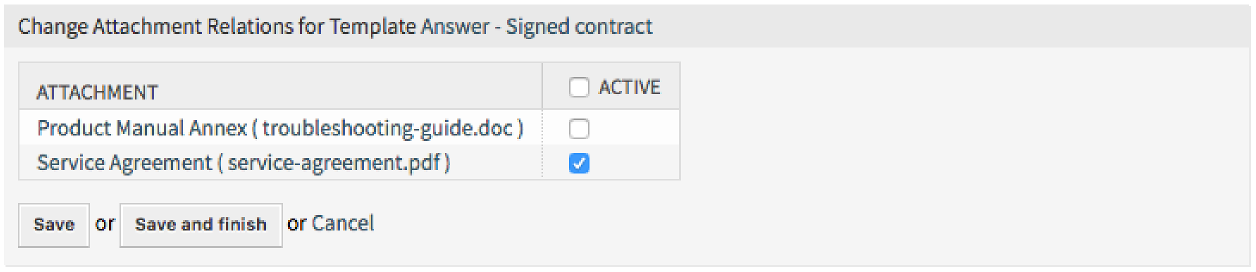
ATTACHMENTS
No data found.

Fig. 66: Manage Template-Attachment Relations

2.19.1 Manage Templates Attachments Relations

To add some attachments to a template:

1. Click on a template in the *Templates* column.
2. Select the attachments you would like to add to the template.
3. Click on the *Save* or *Save and finish* button.



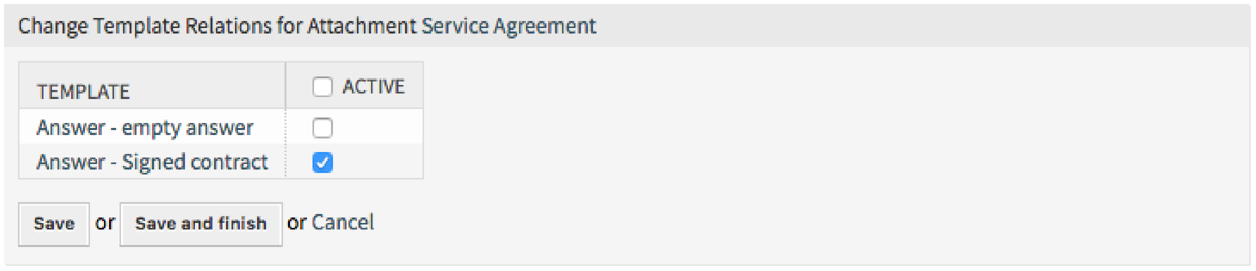
Change Attachment Relations for Template Answer - Signed contract

ATTACHMENT	☐ ACTIVE
Product Manual Annex (troubleshooting-guide.doc)	☐
Service Agreement (service-agreement.pdf)	✓

Fig. 67: Change Attachment Relations for Template

To assign an attachment to some templates:

1. Click on an attachment in the *Attachments* column.
2. Select the templates you would like to assign the attachment to.
3. Click on the *Save* or *Save and finish* button.



Change Template Relations for Attachment Service Agreement

TEMPLATE	☐ ACTIVE
Answer - empty answer	☐
Answer - Signed contract	✓

Fig. 68: Change Template Relations for Attachment

Note: If several templates or attachments are added to the system, use the filter box to find a particular template or attachment by just typing the name to filter.

Multiple templates or attachments can be assigned in both screens at the same time. Additionally clicking on a template or clicking on an attachment in the relations screen will open the *Edit Template* screen or the *Edit Attachment* screen accordingly.

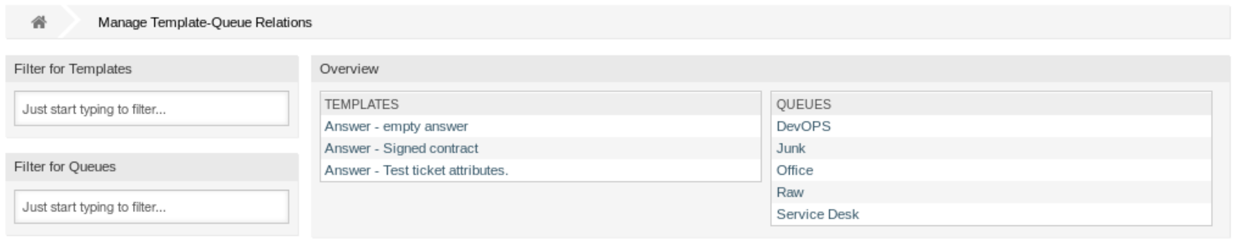
Warning: Accessing a template or an attachment provides no back link to the relations screen.

2.20 Templates Queues

Sharing, distributing and making changes to standard text modules and information for sharing with customers or communication between teams can be an impossible mission.

OTRS can quickly aid you in ensuring that all of your teams have the correct templates available for use by assignment based on the queue.

Use this screen to add one or more templates to one or more queues. To use this function, at least one template and one queue need to have been added to the system. The management screen is available in the *Templates Queues* module of the *Ticket Settings* group.


Manage Template-Queue Relations
Filter for Templates
Just start typing to filter...
Filter for Queues
Just start typing to filter...
Overview

TEMPLATES	QUEUES
Answer - empty answer	DevOPS
Answer - Signed contract	Junk
Answer - Test ticket attributes.	Office
	Raw
	Service Desk

Fig. 69: Manage Template-Queue Relations Screen

2.20.1 Manage Templates Queues Relations

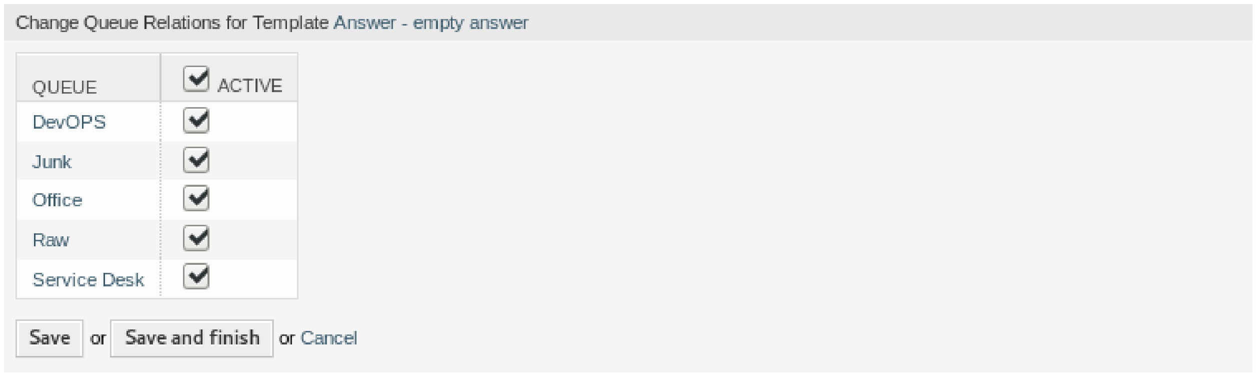
To assign a template to a queue:

1. Click on a template in the *Templates* column.
2. Select the queues you would like to add the template to.
3. Click on the *Save* or *Save and finish* button.

To assign a queue to a template:

1. Click on a queue in the *Queues* column.
2. Select the templates you would like to assign the queue to.
3. Click on the *Save* or *Save and finish* button.

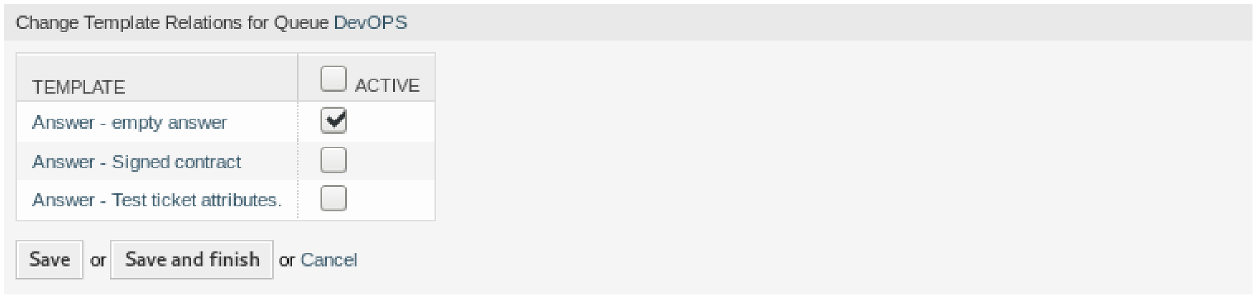
Note: If several templates or queues are added to the system, use the filter box to find a particular template or queue by just typing the name to filter.



Change Queue Relations for Template Answer - empty answer

QUEUE	✓ ACTIVE
DevOPS	✓
Junk	✓
Office	✓
Raw	✓
Service Desk	✓

Fig. 70: Change Queue Relations for Template



Change Template Relations for Queue DevOPS

TEMPLATE	☐ ACTIVE
Answer - empty answer	✓
Answer - Signed contract	☐
Answer - Test ticket attributes.	☐

Fig. 71: Change Template Relations for Queue

Multiple templates or queues can be assigned in both screens at the same time. Additionally clicking on a template or clicking on a queue in the relations screen will open the *Edit Template* screen or the *Edit Queue* screen accordingly.

Warning: Accessing a queue or a template provides no back link to the relations screen.

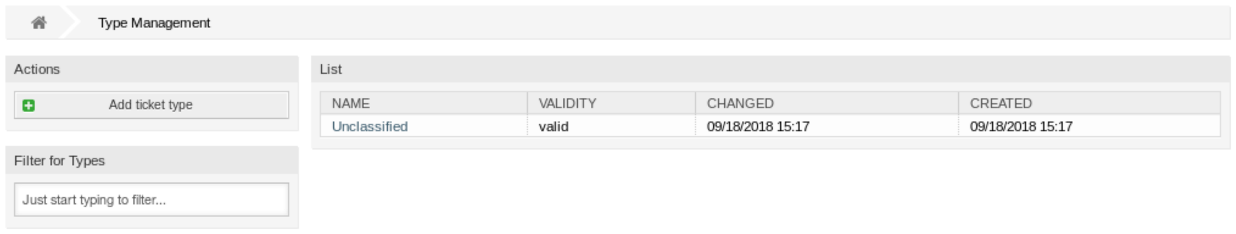
2.21 Types

Good KPIs require knowing the type of work your organization performs. Not all tasks take the same amount of effort even when performed by the same team. Creating a queue structure for this purpose can be overpowered due to the amount of configuration required to create and manage a queue.

OTRS provisions for KPIs with minimal overhead using ticket types. Typical types used in IT service desks are *unclassified*, *incident* and *problem*. You can quickly define new types with ease.

Use this screen to add types to the system. A fresh OTRS installation contains an *unclassified* type by default. The type management screen is available in the *Types* module of the *Ticket Settings* group.

Warning: Services must first be activated via *System Configuration* under the *Administration* group to be selectable in the ticket screens. You may click on the link in the warning message to directly jump to the configuration setting.



NAME	VALIDITY	CHANGED	CREATED
Unclassified	valid	09/18/2018 15:17	09/18/2018 15:17

Fig. 72: Type Management Screen

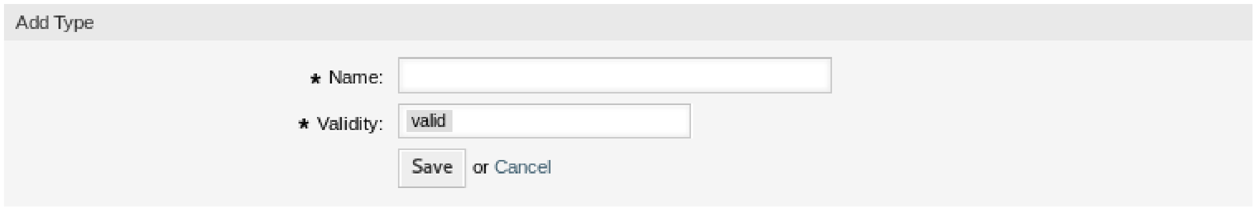
Please activate Type first! →

Fig. 73: Type Activation Warning

2.21.1 Manage Types

To add a type:

1. Click on the *Add Ticket Type* button in the left sidebar.
2. Fill in the required fields.
3. Click on the *Save* button.



* Name:	
* Validity:	valid

Fig. 74: Add Type Screen

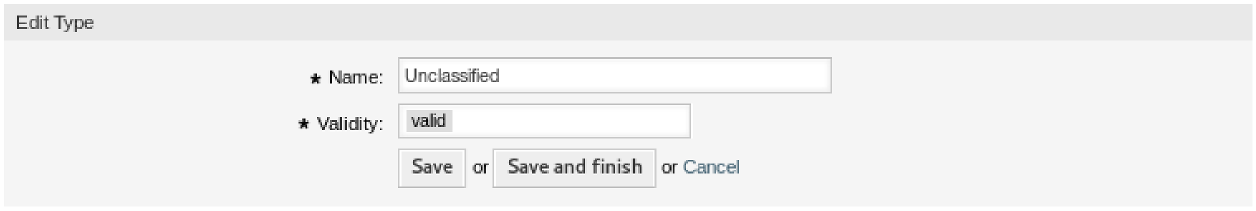
Warning: Types can not be deleted from the system. They can only be deactivated by setting the *Validity* option to *invalid* or *invalid-temporarily*.

Warning: The maximum number of 10 *valid* types should not be exceeded. Exceeding this limit may affect the system performance.

To edit a type:

1. Click on a type in the list of types.
2. Modify the fields.
3. Click on the *Save* or *Save and finish* button.

Note: If several types are added to the system, use the filter box to find a particular type by just typing the name to filter.



* Name: Unclassified
* Validity: valid

Fig. 75: Edit Type Screen

2.21.2 Type Settings

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

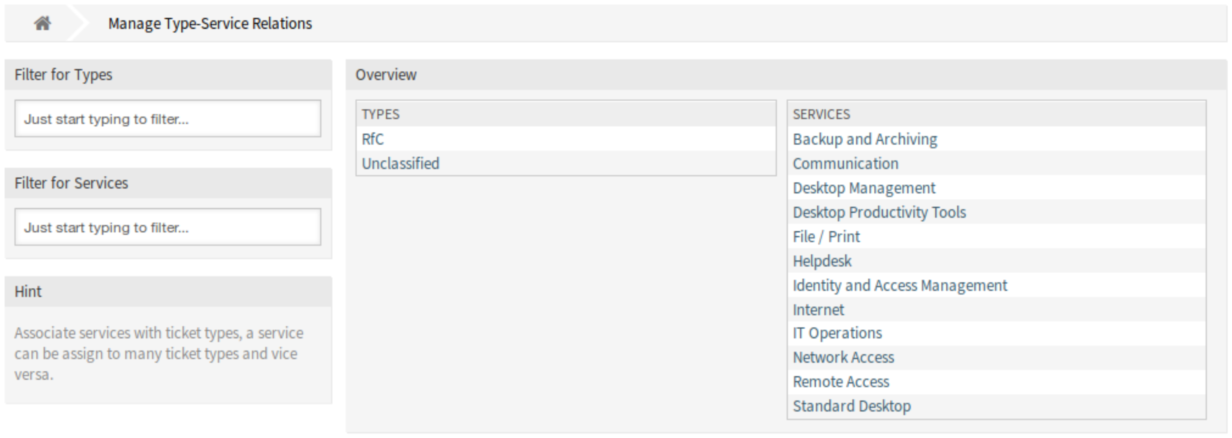
Name *
The name of this resource. Any type of characters can be entered to this field including uppercase letters and spaces. The name will be displayed in the overview table.

Validity *
Set the validity of this resource. Each resource can be used in OTRS only, if this field is set to *valid*. Setting this field to *invalid* or *invalid-temporarily* will disable the use of the resource.

2.22 Types Services

If an agent or a customer creates a ticket and defines the ticket type, this feature makes it easier to automatically display only the linked services from there on. The linking of ticket types and ticket services is carried out by an administrator with ticket configuration permissions, and it is possible to link a service to more than one ticket type, as well as a ticket type to more than one service. Ticket types and services can be additionally filtered and sorted in a drop-down menu.

Use this screen to add one or more types to one or more services. To use this function, at least one type and one service need to have been added to the system. The management screen is available in the *Types Services* module of the *Ticket Settings* group.


Manage Type-Service Relations
Filter for Types
Just start typing to filter...
Filter for Services
Just start typing to filter...
Hint
Overview

Associate services with ticket types, a service can be assign to many ticket types and vice versa.

TYPES
RfC
Unclassified

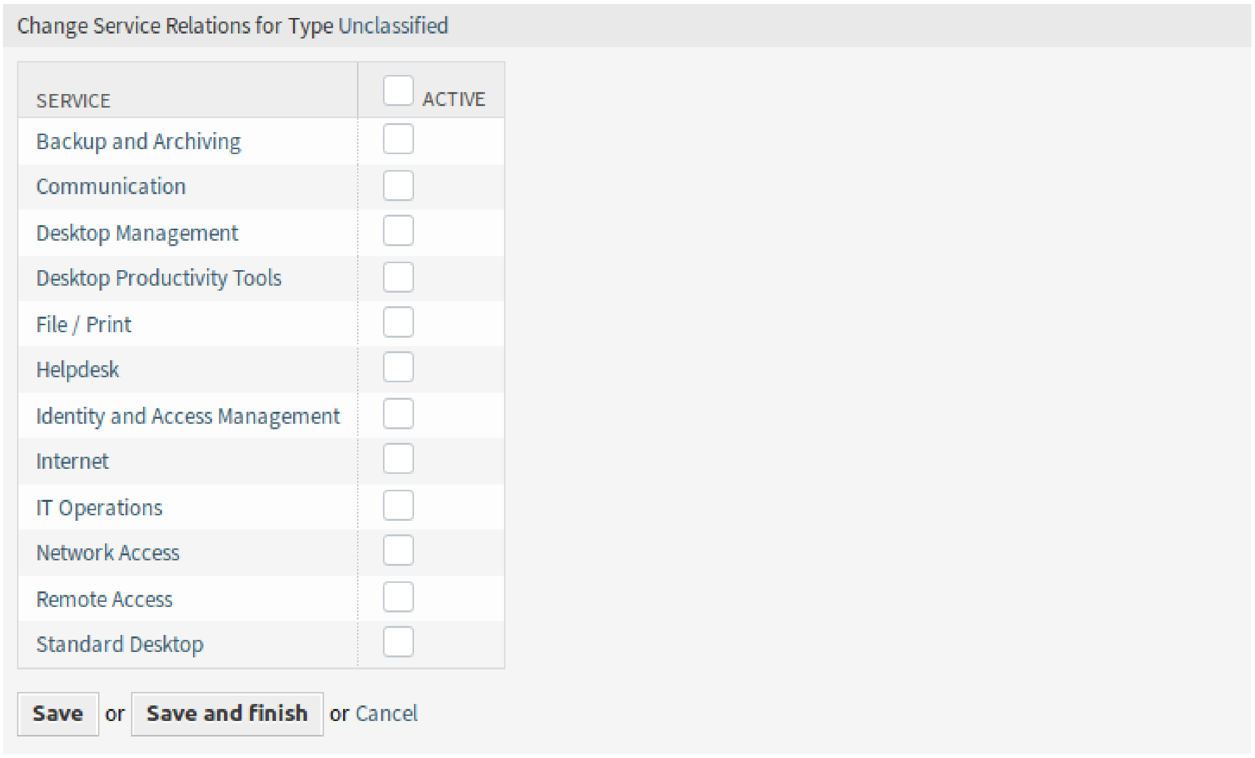
SERVICES
Backup and Archiving
Communication
Desktop Management
Desktop Productivity Tools
File / Print
Helpdesk
Identity and Access Management
Internet
IT Operations
Network Access
Remote Access
Standard Desktop

Fig. 76: Manage Type-Service Relations Screen

2.22.1 Manage Types Services Relations

To assign a type to a service:

1. Click on a type in the *Types* column.
2. Select the services you would like to add the type to.
3. Click on the *Save* or *Save and finish* button.



Change Service Relations for Type Unclassified

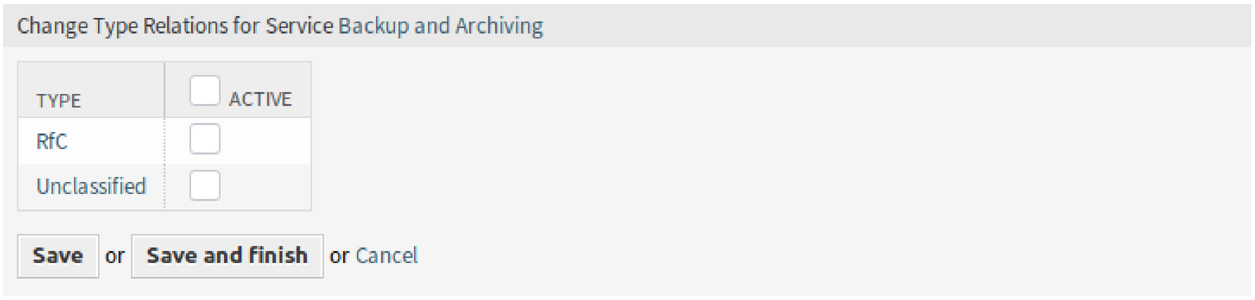
SERVICE	☐ ACTIVE
Backup and Archiving	☐
Communication	☐
Desktop Management	☐
Desktop Productivity Tools	☐
File / Print	☐
Helpdesk	☐
Identity and Access Management	☐
Internet	☐
IT Operations	☐
Network Access	☐
Remote Access	☐
Standard Desktop	☐

Save or Save and finish or Cancel

Fig. 77: Change Service Relations for Type

To assign a service to a type:

1. Click on a service in the *Services* column.
2. Select the types you would like to assign the service to.
3. Click on the *Save* or *Save and finish* button.



Change Type Relations for Service Backup and Archiving

TYPE	☐ ACTIVE
RfC	☐
Unclassified	☐

Save or Save and finish or Cancel

Fig. 78: Change Type Relations for Service

Note: If several types or services are added to the system, use the filter box to find a particular type or service by just typing the name to filter.

Multiple types or services can be assigned in both screens at the same time. Additionally clicking on a type or clicking on a service in the relations screen will open the *Edit Type* screen or the *Edit Service* screen accordingly.

Warning: Accessing a type or a service provides no back link to the relations screen.

CHAPTER

THREE

COMMUNICATION & NOTIFICATIONS

Clear, complete, and transparent communication is key to ensuring that your organization offers outstanding service to your customers. Your organization would like to give your customers the most consistent service and ensure that your service is highly recognizable, every time. Your teams want to have an identity which instills a specific sense of security and familiarity.

OTRS provides you with the tools to build teams, trust and security to better and uniformly serve your customer. With system addresses you can assign your inbound mails to certain teams and allow them all to use the same address without causing confusion by using team mailboxes via a mail client. Templates, automatic answers, and attachment management let you leverage central management of the most important communications. Many other tools are also available and covered in the sections below.

3.1 Admin Notification

Corporations may need to make general announcements to everyone or publish news to several groups of agents or individuals. Also, OTRS administrator may need to contact with specific agents regarding an event.

OTRS gives the administration the tool *Admin Notification* making sending announcements and news to the masses of users accurate and timely, to target group of people, simple. Administrators can send notifications based on a specific list of recipients, or a group of users inside OTRS, with powerful text editor enhanced content.

Use this screen to send administrative messages to specific agents, group or role members. The message compose screen is available in the *Admin Notification* module of the *Communication & Notifications* group.

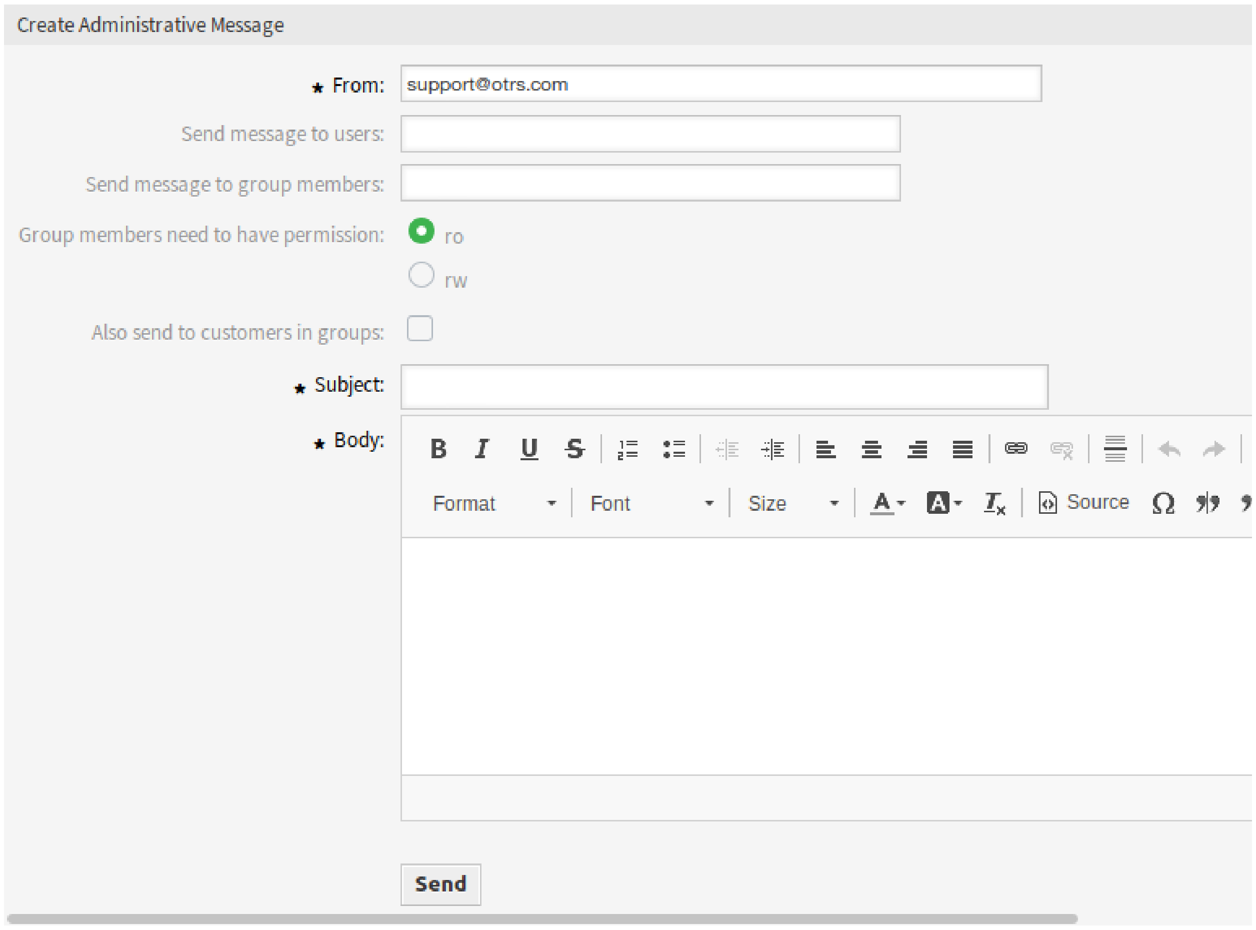
3.1.1 Administrative Message Settings

The following settings are available when composing an administrative message. The fields marked with an asterisk are mandatory.

From *
This email address will be added into the *From* field of the message.

Send message to users
One or more *Agents* can be selected in this field, to whom the message will be sent.

Send message to group members
One or more *Groups* can be selected in this field, to whose members the message will be sent.


Create Administrative Message

⋆ From:	support@otrs.com
Send message to users:	
Send message to group members:	
Group members need to have permission:	[✓ ro] rw
Also send to customers in groups:	☐
⋆ Subject:	
⋆ Body:	

Fig. 1: Create Administrative Message Screen

Group members need to have permission
With these radio buttons can be selected, if a group member need read-only or read-write permissions to receive the message.

Also send to customers in groups
Select this checkbox to send the message also for customers in groups.

Note: This option is available only, if the CustomerGroupSupport setting is enabled.

Subject *
The subject of the messages.

Body *
The body text of the message.

3.2 Appointment Notifications

Missing appointments can damage your image with a customer. Once there is an appointment assigned in the calendar, it' s normal to receive notification:

- Upon a new or changed event
- Upon cancellation of an event
- Before the event, as a reminder

Notification relieves the agent the stress of mentally tracking appointments.

OTRS appointment notifications satisfies this need. Here an administrator can easily set notifications with general rules, including trigger events and filters. Afterward, appointments fitting the bill notify the correct users at the correct time.

Use this screen to add appointment notifications to the system. In a fresh OTRS installation an appointment reminder notification is already added by default. The appointment notification management screen is available in the *Appointment Notifications* module of the *Communication & Notifications* group.

3.2.1 Manage Appointment Notifications

To add an appointment notification:

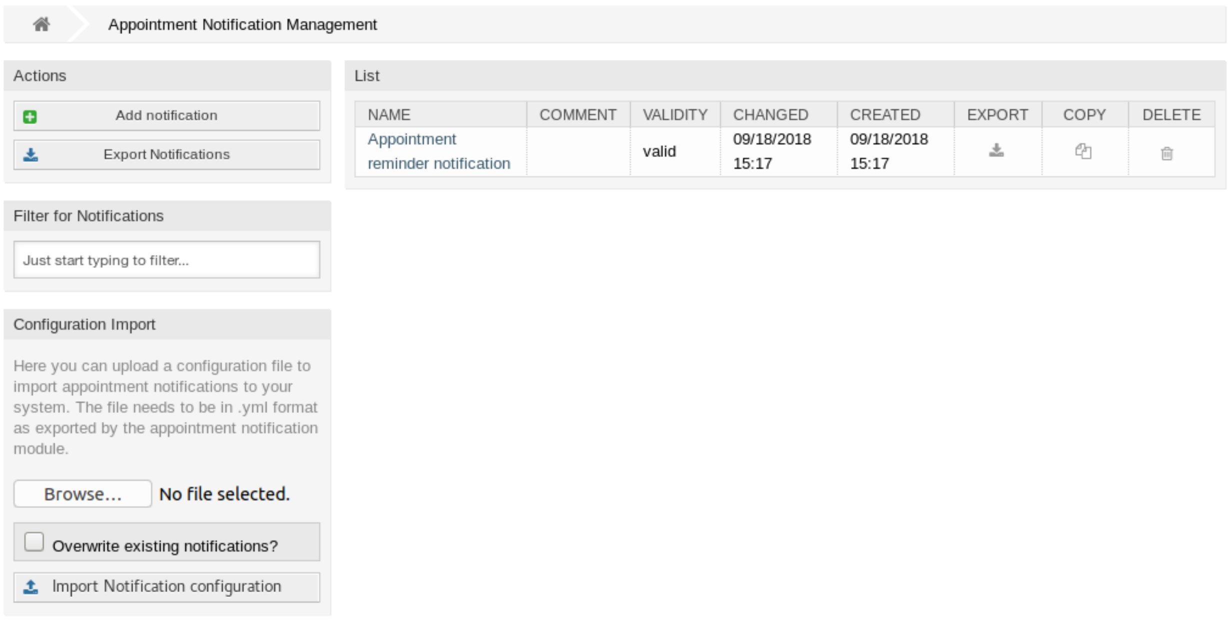
1. Click on the *Add Notification* button in the left sidebar.
2. Fill in the required fields as explained in *Appointment Notification Settings*.
3. Click on the *Save* button.

To edit an appointment notification:

1. Click on an appointment notification in the list of appointment notifications.
2. Modify the fields as explained in *Appointment Notification Settings*.
3. Click on the *Save* or *Save and finish* button.

To delete an appointment notification:

1. Click on the trash icon in the list of appointment notifications.

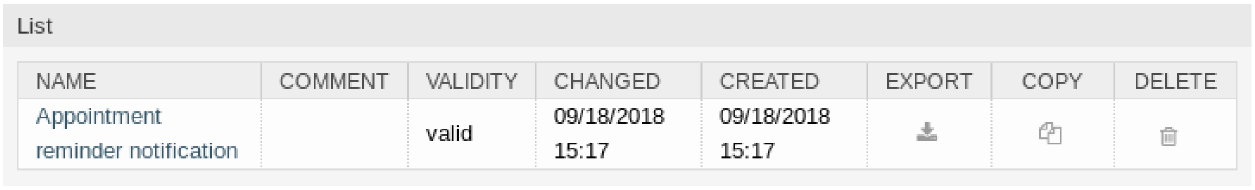


Here you can upload a configuration file to import appointment notifications to your system. The file needs to be in .yml format as exported by the appointment notification module.

☐ Overwrite existing notifications?

NAME	COMMENT	VALIDITY	CHANGED	CREATED	EXPORT	COPY	DELETE
Appointment reminder notification		valid	09/18/2018 15:17	09/18/2018 15:17			

Fig. 2: Appointment Notification Management Screen



NAME	COMMENT	VALIDITY	CHANGED	CREATED	EXPORT	COPY	DELETE
Appointment reminder notification		valid	09/18/2018 15:17	09/18/2018 15:17			

Fig. 3: Delete Appointment Notification Screen

2. Click on the *Confirm* button.

To export all appointment notifications:

1. Click on the *Export Notifications* button in the left sidebar.
2. Choose a location in your computer to save the `Export_Notification.yml` file.

To import appointment notifications:

1. Click on the *Browse…* button in the left sidebar.
2. Select a previously exported `.yml` file.
3. Click on the *Overwrite existing notifications?* checkbox, if you would like to overwrite the existing notifications.
4. Click on the *Import Notification configuration* button.

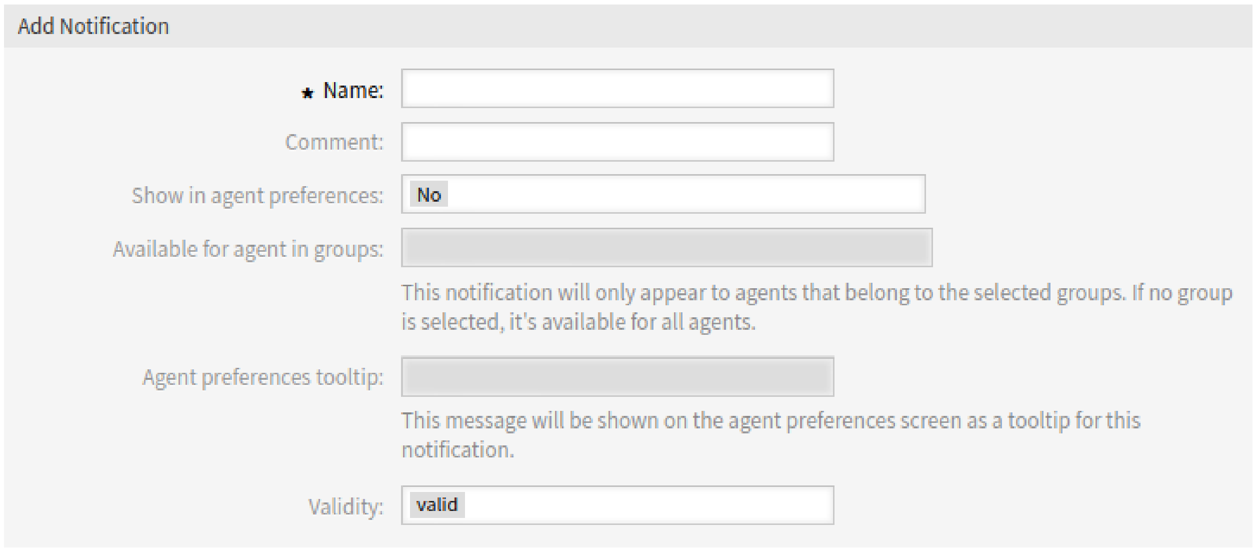
3.2.2 Appointment Notification Settings

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

See also:

For an example, see the default appointment reminder notification which is included in a fresh OTRS installation.

Basic Appointment Notification Settings


Add Notification

★ Name:	
Comment:	
Show in agent preferences:	No
Available for agent in groups:	
	This notification will only appear to agents that belong to the selected groups. If no group is selected, it's available for all agents.
Agent preferences tooltip:	
	This message will be shown on the agent preferences screen as a tooltip for this notification.
Validity:	valid

Fig. 4: Appointment Notification Settings - Basic

Name *
The name of this resource. Any type of characters can be entered to this field including uppercase letters and spaces. The name will be displayed in the overview table.

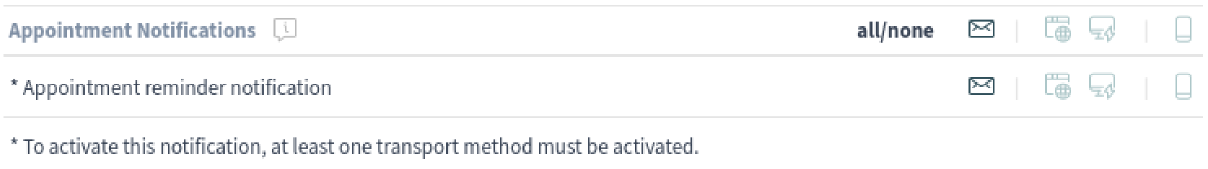
Comment
Add additional information to this resource. It is recommended to always fill this field as a description of the resource with a full sentence for better clarity, because the comment will be also displayed in the overview table.

Show in agent preferences
Define how the notification should be displayed in agent preferences. The following options are available:

No
The notification won't be displayed in agent preferences. The notification is sent to all appropriate agents by the defined method.

Yes
The notification will be displayed in agent preferences for selection. The agents may opt-in or opt-out.

Yes, but require at least one active notification method.
The notification will be displayed in agent preferences, but require at least one active notification method. This is annotated by an asterisk next to the name.


Appointment Notifications — all/none

* Appointment reminder notification

* To activate this notification, at least one transport method must be activated.

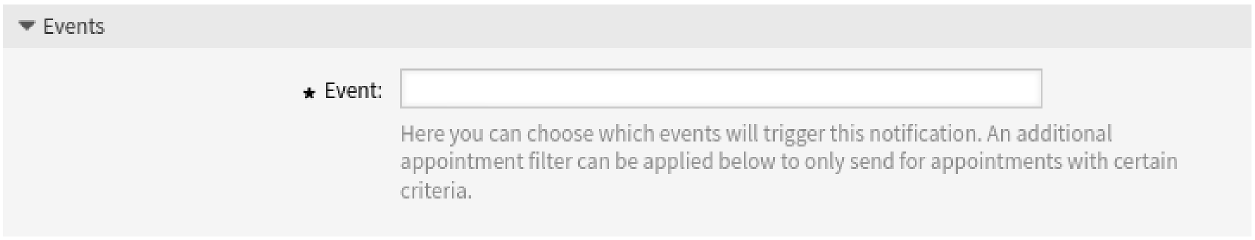
Fig. 5: Personal Appointment Notification Settings

Available for agent in groups
The notification will only appear to agents that belong to the selected groups. If no group is selected, it is available for all agents.

Agent preferences tooltip
This message will be shown on the agent preferences screen as a tooltip for this notification.

Validity *
Set the validity of this resource. Each resource can be used in OTRS only, if this field is set to *valid*. Setting this field to *invalid* or *invalid-temporarily* will disable the use of the resource.

Appointment Events


Events

* Event:

Here you can choose which events will trigger this notification. An additional appointment filter can be applied below to only send for appointments with certain criteria.

Fig. 6: Appointment Notification Settings - Events

Event
Here you can choose which events will trigger this notification. An additional appointment filter can be applied below to only send for appointments with certain criteria.

Possible events are:

`AppointmentCreate`
Executed after an appointment has been created.

`AppointmentUpdate`
Executed after an appointment has been updated.

`AppointmentDelete`
Executed after an appointment has been deleted.

`AppointmentNotification`
This is a special appointment event that will be executed by the OTRS daemon in time. If an appointment contains a date/time value for notifications, as already described in this documentation, and such a notification date is reached, the OTRS daemon will execute this kind of event for every related appointment separately.

`CalendarCreate`
Executed after a calendar has been created.

`CalendarUpdate`
Executed after a calendar has been updated.

Appointment Filter

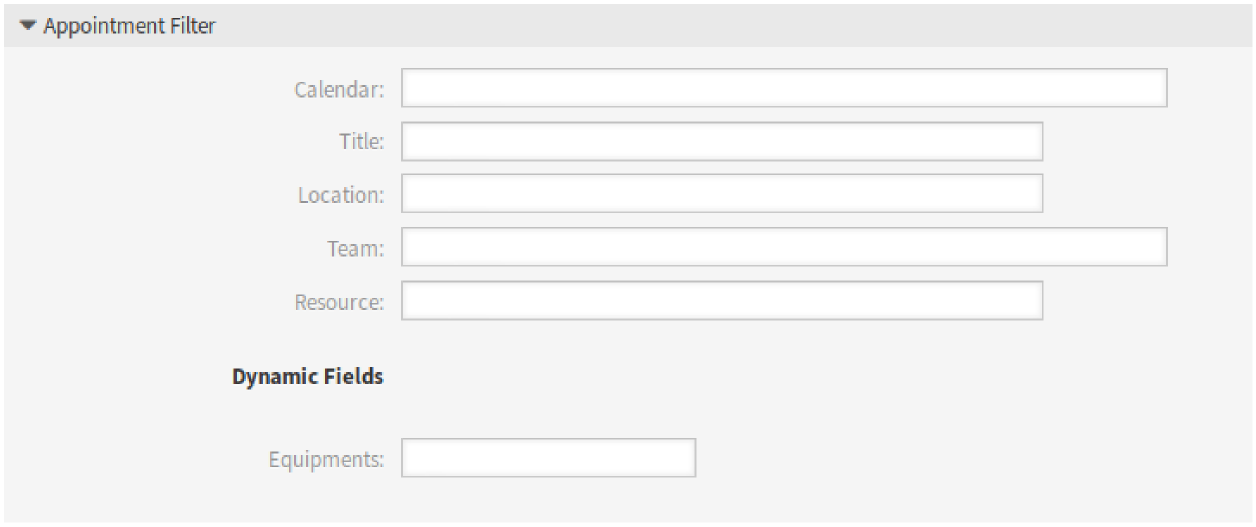


Fig. 7: Appointment Notification Settings - Appointment Filter

This widget can optionally be used to narrow the list of appointments by matching configured values:

Calendar
Select which calendar the related appointment needs to be part of.

Title
Filter for a part or complete title of the appointment.

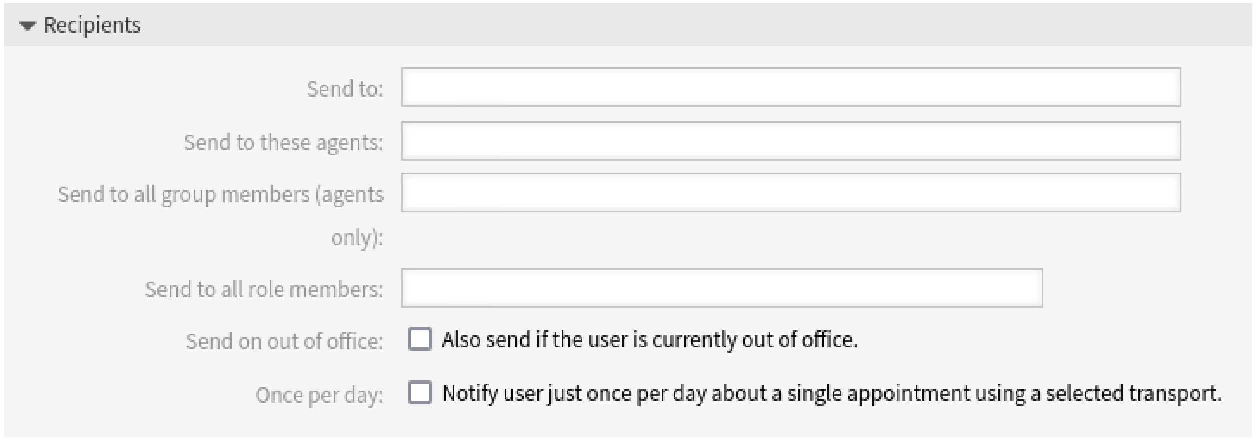
Location
Filter for a part or complete location of the appointment.

Team
Select which team the related appointment needs to be assigned to.

Resource
Choose from a list of teams or resources assigned to the appointments.

Dynamic fields
This section is displayed only if at least one dynamic field for appointments is created in *Dynamic Fields* screen.

Appointment Notification Recipients



Send to:

Send to these agents:

Send to all group members (agents only):

Send to all role members:

Send on out of office: ☐ Also send if the user is currently out of office.

Once per day: ☐ Notify user just once per day about a single appointment using a selected transport.

Fig. 8: Appointment Notification Settings - Recipients

Send to
Select which agents should receive the notifications. Possible values are:

- Agent (resources), who are selected within the appointment
- All agents with (at least) read permission for the appointment (calendar)
- All agents with write permission for the appointment (calendar)

Send to these agents
One or more agents can be selected who should receive the notifications.

Send to all group members (agents only)
One or more groups can be selected whom agents should receive the notifications.

Send to all role members
One or more roles can be selected whom agents should receive the notifications.

Send on out of office
If this option is checked, the notification will be sent even if the agent is currently out of office.

Once per day
Notify users just once per day about a single appointment using a selected transport. If this is the first notification about an appointment, then the notification will be sent. If a notification was already sent before and this option is checked, the OTRS daemon will check the time the last notification was sent. If there was no notification sent in the last 24 hours, the notification will be sent again.

Note: The notifications respect the permissions of the agent. Notifications are sent to agents only if the agent has at least read permissions for the appointment in that moment in time when the notification is triggered.

Appointment Notification Methods

Enable this notification method
Enable or disable this notification method. A notification method can be email, web view with browser notification support or SMS.

Enable browser notification
This option is available for web view only. If checked, the system sends a browser notification, too. Web view notifications will be displayed in the notifications area of the agent interface while a browser notification is a native browser notification which needs to be enabled in the web browser at first time.

Note: To use the SMS notification method, *Cloud Services* need to be enabled.

Active by default in agent preferences
This is the default value for assigned recipient agents who didn't make a choice for this notification in their preferences yet. If the box is enabled, the notification will be sent to such agents.

Note: This field is displayed only if *Yes* is selected in the *Show in agent preferences* setting above.

Additional recipient email addresses
Additional recipients can be added here. Use comma or semicolon to separate the email addresses.

Article visible to customer
An article will be created if the notification is sent to the customer or an additional email address.

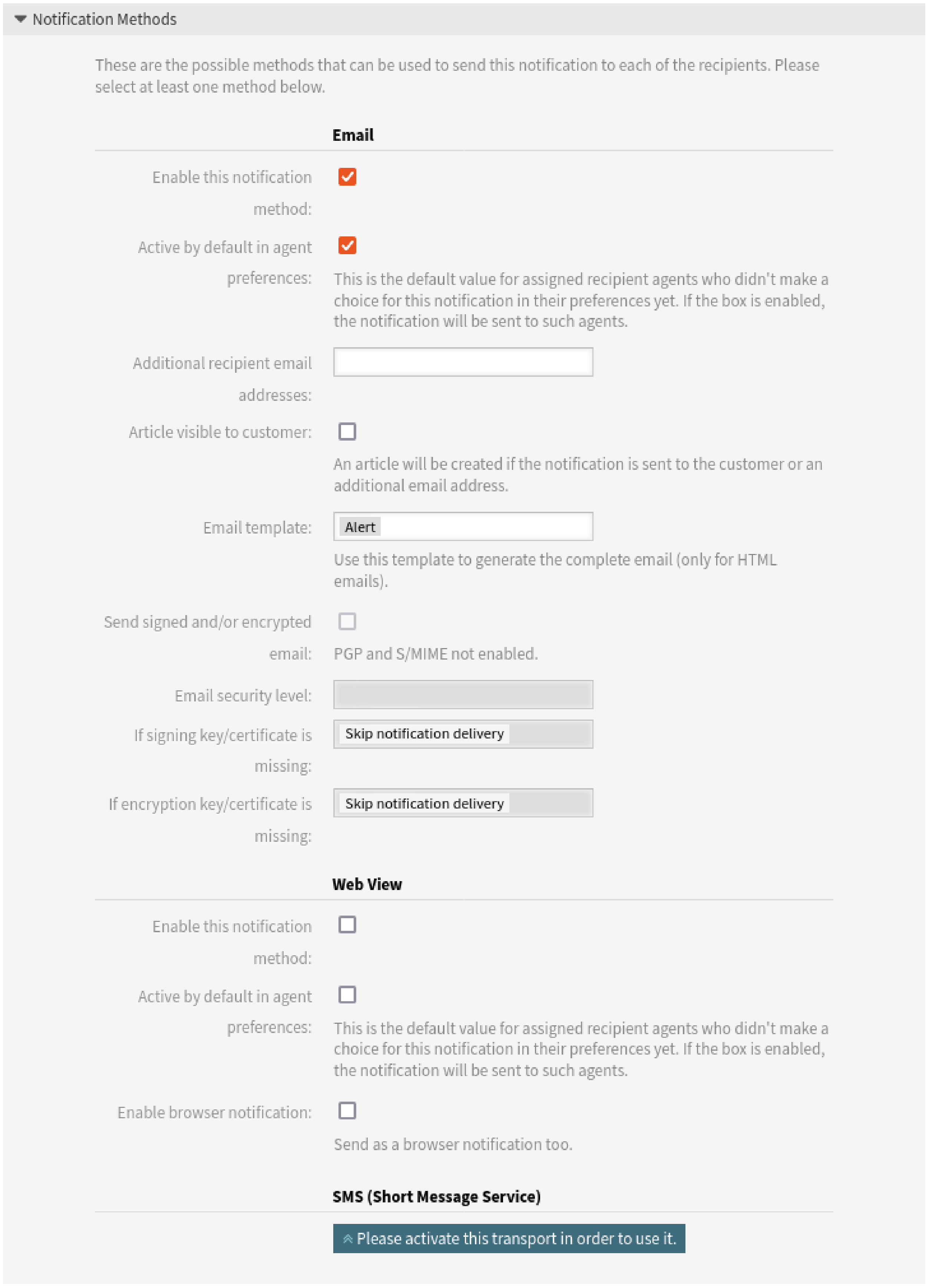
Send ICS file
It is possible to include an ICS file to the appointment notification sent to users. In the *Events* widget above select `AppointmentCreate`, `AppointmentUpdate` or `AppointmentDelete` as event before activating this feature.

You can even configure to use the additional shown information out of the *Custom* timeline view (configurable per calendar) in the ICS file generation as well. Just activate the setting `AppointmentCalendar::ICSFiles::UseAppointmentDescription` to combine the regular description with the configured text.

Email template
Select which email template should be used for the notification.

Note: Additional email templates can be added by placing a `.tt` file into the folder `<OTRS_Home>/Kernel/Output/HTML/Templates/Standard/NotificationEvent/Email/`. See the existing email templates for an example.

Send signed and/or encrypted email
Checking this option will encrypt the notification email.



Notification Methods

These are the possible methods that can be used to send this notification to each of the recipients. Please select at least one method below.

Email

Enable this notification method:	✓
Active by default in agent preferences:	✓ This is the default value for assigned recipient agents who didn't make a choice for this notification in their preferences yet. If the box is enabled, the notification will be sent to such agents.
Additional recipient email addresses:	
Article visible to customer:	☐ An article will be created if the notification is sent to the customer or an additional email address.
Email template:	Alert Use this template to generate the complete email (only for HTML emails).
Send signed and/or encrypted email:	☐ PGP and S/MIME not enabled.
Email security level:	
If signing key/certificate is missing:	Skip notification delivery
If encryption key/certificate is missing:	Skip notification delivery

Web View

Enable this notification method:	☐
Active by default in agent preferences:	☐ This is the default value for assigned recipient agents who didn't make a choice for this notification in their preferences yet. If the box is enabled, the notification will be sent to such agents.
Enable browser notification:	☐ Send as a browser notification too.

SMS (Short Message Service)

Fig. 9: Appointment Notification Settings - Notification Methods

Note: To use this feature, *PGP Keys* or *S/MIME Certificates* need to be enabled.

Email security level
If *Enable email security* is checked, then this setting is activated. The following options are available:

PGP sign only
Sign only the notification email with PGP key. If no PGP keys have been added to the system, this option is not visible.

PGP encrypt only
Encrypt only the notification email with PGP key. If no PGP keys have been added to the system, this option is not visible.

PGP sign and encrypt
Sign and encrypt the notification email with PGP key. If no PGP keys have been added to the system, this option is not visible.

S/MIME sign only
Sign only the notification email with S/MIME certificate. If no S/MIME certificates have been added to the system, this option is not visible.

S/MIME encrypt only
Encrypt only the notification email with S/MIME certificate. If no S/MIME certificates have been added to the system, this option is not visible.

S/MIME sign and encrypt
Sign and encrypt the notification email with S/MIME certificate. If no S/MIME certificates have been added to the system, this option is not visible.

Note: To use this feature, *PGP Keys* or *S/MIME Certificates* need to be enabled.

If signing key/certificate is missing
Select the method, that should be used if signing key or certificate is missing.

If encryption key/certificate is missing:
Select the method, that should be used if encryption key or certificate is missing.

Appointment Notification Text

The main content of a notification can be added for each languages with localized subject and body text. It is also possible to define static text content mixed with OTRS smart tags.

Subject *
The localized subject for a specific language.

Text *
The localized body text for a specific language.

Add new notification language
Select which languages should be added to create localized notifications. The language of the customer or agent will be used as found in the customer and agent preferences. Secondarily, the system default language will be chosen. The fall back will always be English.

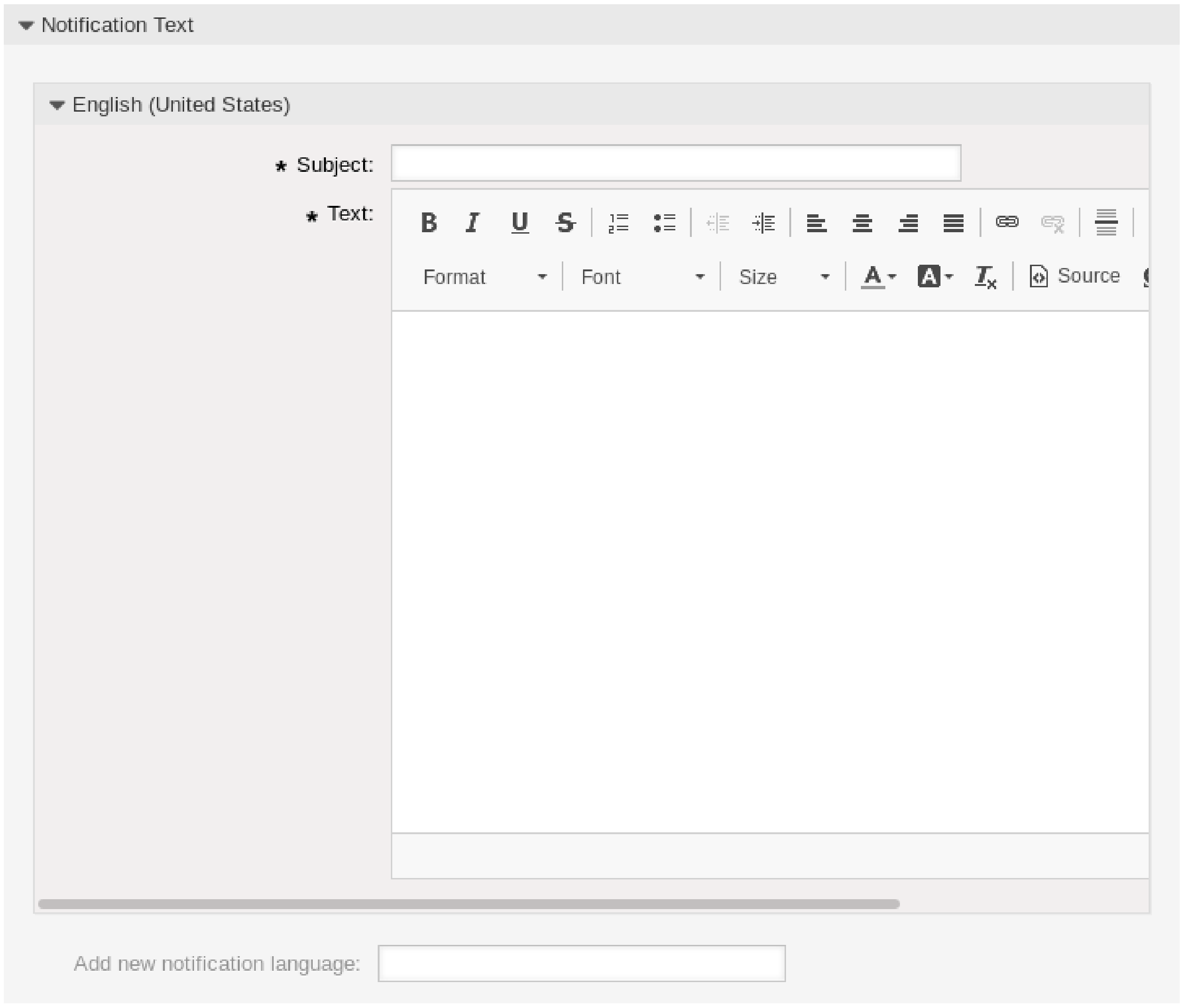
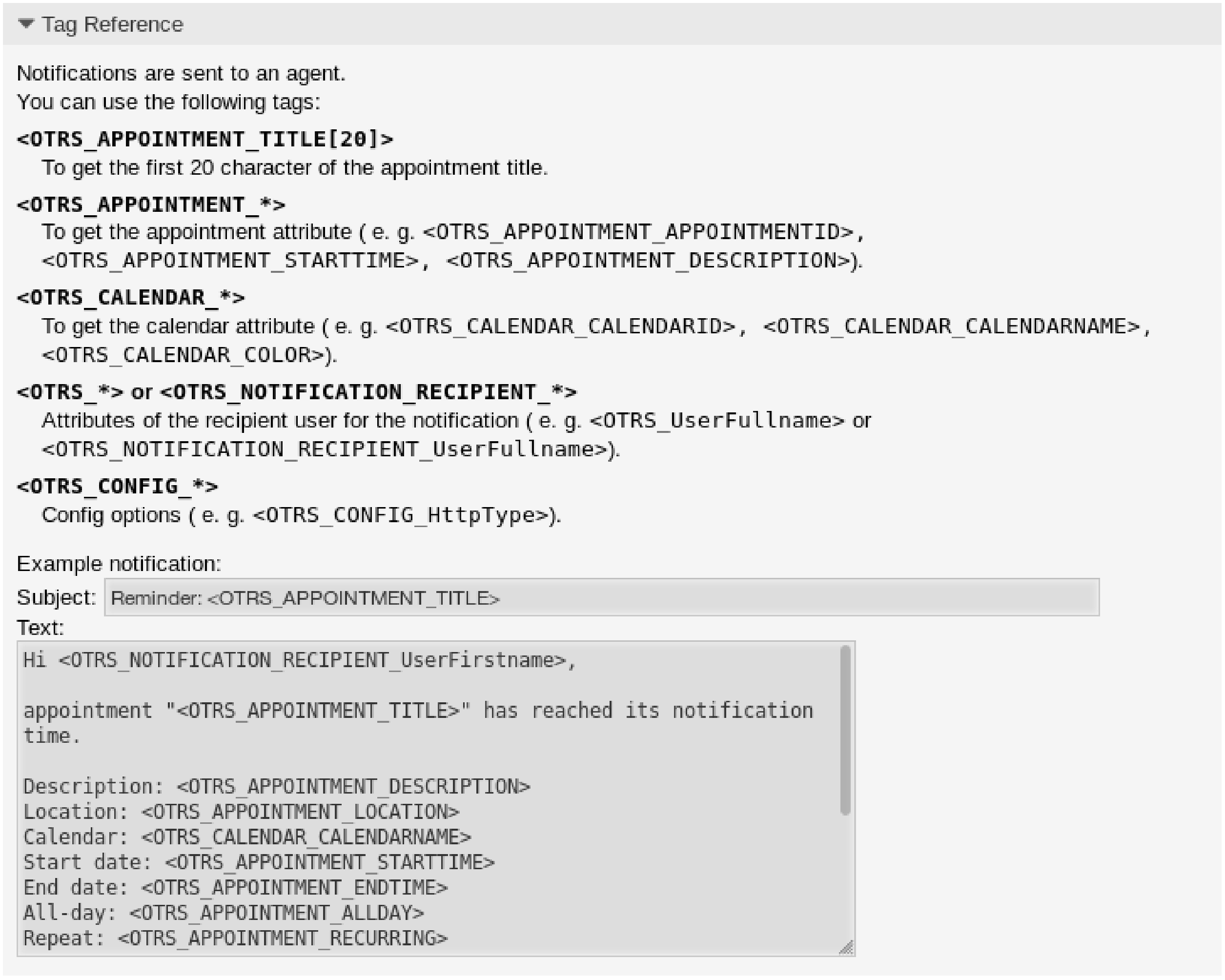


Fig. 10: Appointment Notification Settings - Notification Text

Warning: Deleting a language from the DefaultUsedLanguages setting that already has a notification text here will make the notification text unusable. If a language is not present or enabled on the system, the corresponding notification text could be deleted if it is not needed anymore.

3.2.3 Appointment Notification Variables

Using variables in the text makes it possible to personalize messages. Variables, known as OTRS tags, are replaced by OTRS when generating the message. Find a list of available tags stems for this resource at the bottom of both add and edit screens.



▼ Tag Reference

Notifications are sent to an agent.
You can use the following tags:

<OTRS_APPOINTMENT_TITLE[20]>
To get the first 20 character of the appointment title.

<OTRS_APPOINTMENT_*>
To get the appointment attribute (e. g. <OTRS_APPOINTMENT_APPOINTMENTID>, <OTRS_APPOINTMENT_STARTTIME>, <OTRS_APPOINTMENT_DESCRIPTION>).

<OTRS_CALENDAR_*>
To get the calendar attribute (e. g. <OTRS_CALENDAR_CALENDARID>, <OTRS_CALENDAR_CALENDARNAME>, <OTRS_CALENDAR_COLOR>).

<OTRS_*> or <OTRS_NOTIFICATION_RECIPIENT_*>
Attributes of the recipient user for the notification (e. g. <OTRS_UserFullname> or <OTRS_NOTIFICATION_RECIPIENT_UserFullname>).

<OTRS_CONFIG_*>
Config options (e. g. <OTRS_CONFIG_HttpType>).

Example notification:
Subject: Reminder: <OTRS_APPOINTMENT_TITLE>
Text:

```
Hi <OTRS_NOTIFICATION_RECIPIENT_UserFirstname>,

appointment "<OTRS_APPOINTMENT_TITLE>" has reached its notification
time.

Description: <OTRS_APPOINTMENT_DESCRIPTION>
Location: <OTRS_APPOINTMENT_LOCATION>
Calendar: <OTRS_CALENDAR_CALENDARNAME>
Start date: <OTRS_APPOINTMENT_STARTTIME>
End date: <OTRS_APPOINTMENT_ENDTIME>
All-day: <OTRS_APPOINTMENT_ALLDAY>
Repeat: <OTRS_APPOINTMENT_RECURRING>
```

Fig. 11: Appointment Notification Variables

For example, the variable `<OTRS_APPOINTMENT_TITLE[20]>` expands to the first 20 characters of the title allowing a template to include something like the following.

```
Title: <OTRS_APPOINTMENT_TITLE[20]>
```

This tag expands, for example to:

```
Title: Daily meeting in the…
```

The values of the following variables are translated based on the chosen language of the customer user.

```
<OTRS_TICKET_Type>
<OTRS_TICKET_State>
<OTRS_TICKET_StateType>
<OTRS_TICKET_Lock>
<OTRS_TICKET_Priority>
```

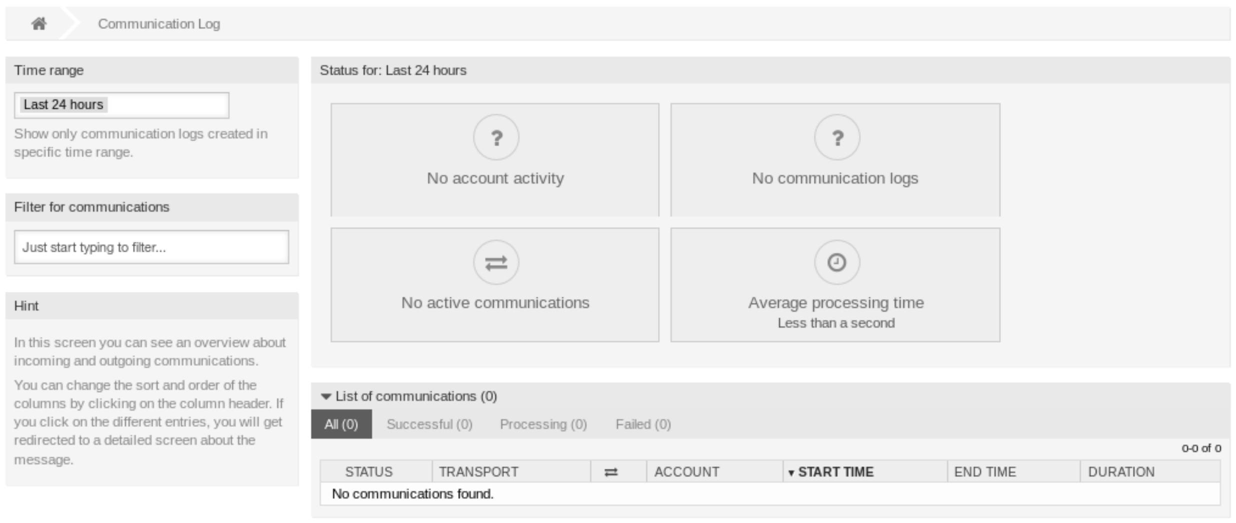
If the language is not supported the default language is applied.

3.3 Communication Log

Managers, leaders, team leads, and system administrators may need to track past communication to follow up specific messages. In some cases, issues arise, and a target recipient did not receive a message. Without access to the mail server logs, tracking the communication is difficult.

OTRS introduces the *Communication Log* module. It's designed to track the communication: building and spooling the mail and the connection between client and server.

Use this screen to inspect the internal logs about communication handling. The communication log overview screen is available in the *Communication Log* module of the *Communication & Notifications* group.



In this screen you can see an overview about incoming and outgoing communications.

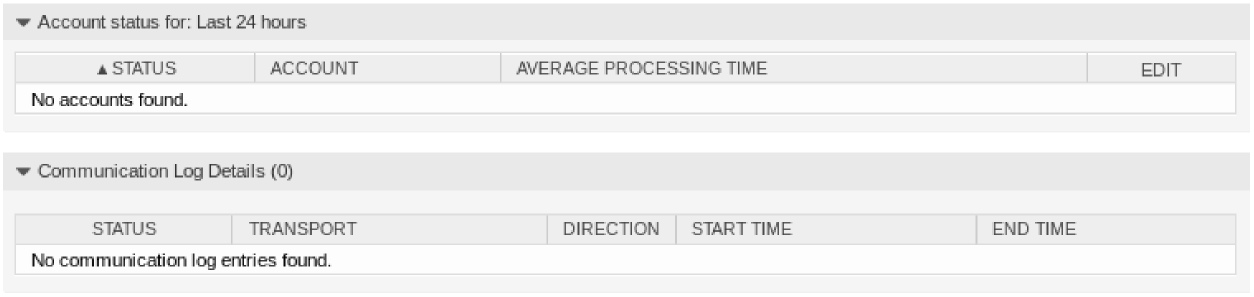
You can change the sort and order of the columns by clicking on the column header. If you click on the different entries, you will get redirected to a detailed screen about the message.

STATUS	TRANSPORT	⇄	ACCOUNT	▾ START TIME	END TIME	DURATION
No communications found.						

Fig. 12: Communication Log Overview Screen

3.3.1 Communication Log Overview

The communication log overview page is a dashboard-like screen with several metrics indicating the overall health of the system, depending on filtered communications.



Account status for: Last 24 hours

▲STATUS	ACCOUNT	AVERAGE PROCESSING TIME	EDIT
No accounts found.			

Communication Log Details (0)

STATUS	TRANSPORT	DIRECTION	START TIME	END TIME
No communication log entries found.				

Fig. 13: Account Status Screen

Account status
This widget will signal if you have any issues with configured accounts used for fetching or sending messages.

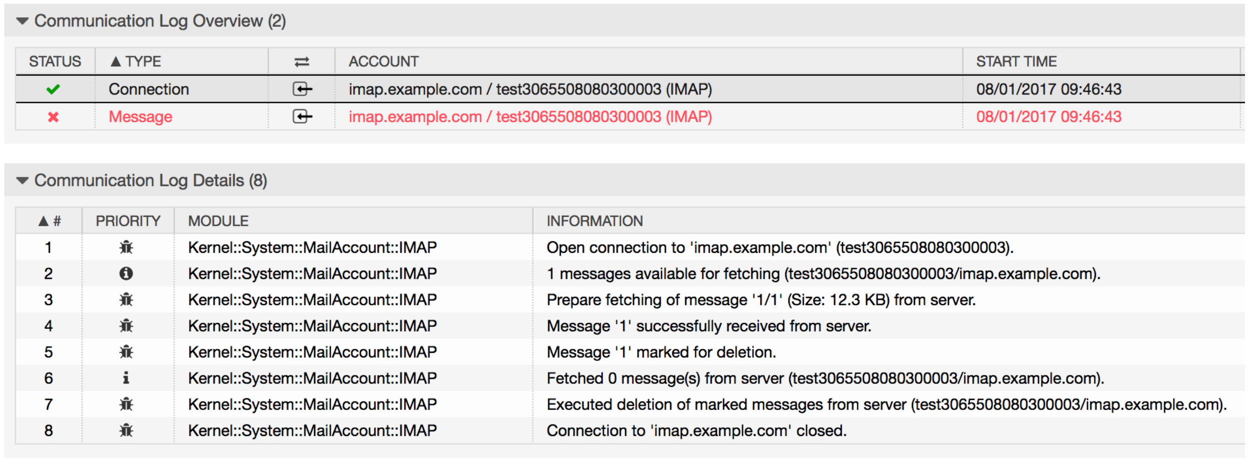
Communication status
This widget will notify you if there are any errors with either account connections or message processing.

Communication state
This widget will display if there are any active communications currently in the system.

Average processing time
This is a cumulative time statistic that is needed to complete a communication.

You can select the time range in the left sidebar in order to filter communications depending on their creation time. In addition to this, you can also dynamically filter for any keywords, state of the communication, and you can sort the overview table by all columns.

If you click on a communication row in any table, you will be presented with a detailed view screen.



Communication Log Overview (2)

STATUS	▲TYPE	⇄	ACCOUNT	START TIME
✓	Connection	⇥	imap.example.com / test3065508080300003 (IMAP)	08/01/2017 09:46:43
✗	Message	⇥	imap.example.com / test3065508080300003 (IMAP)	08/01/2017 09:46:43

Communication Log Details (8)

▲#	PRIORITY	MODULE	INFORMATION
1	🐞	Kernel::System::MailAccount::IMAP	Open connection to 'imap.example.com' (test3065508080300003).
2	●	Kernel::System::MailAccount::IMAP	1 messages available for fetching (test3065508080300003/imap.example.com).
3	🐞	Kernel::System::MailAccount::IMAP	Prepare fetching of message '1/1' (Size: 12.3 KB) from server.
4	🐞	Kernel::System::MailAccount::IMAP	Message '1' successfully received from server.
5	🐞	Kernel::System::MailAccount::IMAP	Message '1' marked for deletion.
6	i	Kernel::System::MailAccount::IMAP	Fetched 0 message(s) from server (test3065508080300003/imap.example.com).
7	🐞	Kernel::System::MailAccount::IMAP	Executed deletion of marked messages from server (test3065508080300003/imap.example.com).
8	🐞	Kernel::System::MailAccount::IMAP	Connection to 'imap.example.com' closed.

Fig. 14: Communication Log Detailed View Screen

Every communication can contain one or more logs, which can be of *Connection* or *Message* type.

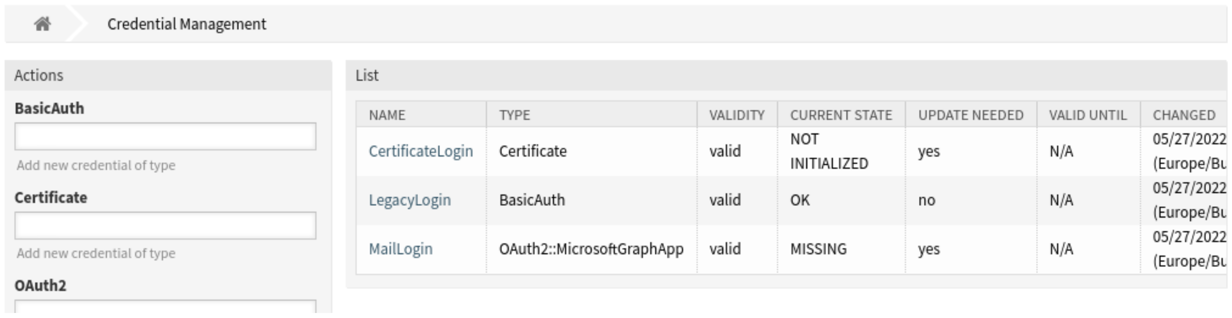
Connection
This type of logs will contain any log messages coming from the modules responsible for connecting to your accounts and fetching/receiving messages.

Message
This type of logs will contain any log messages related to specific message processing. Any module working on message themselves can log their actions in this log, giving you a clear overview of what' s going on.

You can filter log entries based on their priority, by choosing desired priority in the left sidebar. Log level rules apply: by selecting a specific priority, you will get log entries that have that priority set and higher, with *Error* being the highest.

3.4 Credentials

Use this screen to manage credentials in the system. The credential management screen is available in the *Credentials* module of the *Communication & Notifications* group.



NAME	TYPE	VALIDITY	CURRENT STATE	UPDATE NEEDED	VALID UNTIL	CHANGED
CertificateLogin	Certificate	valid	NOT INITIALIZED	yes	N/A	05/27/2022 (Europe/Bu
LegacyLogin	BasicAuth	valid	OK	no	N/A	05/27/2022 (Europe/Bu
MailLogin	OAuth2::MicrosoftGraphApp	valid	MISSING	yes	N/A	05/27/2022 (Europe/Bu

Fig. 15: Credential Management Screen

3.4.1 Manage Credentials

To add a new credential:

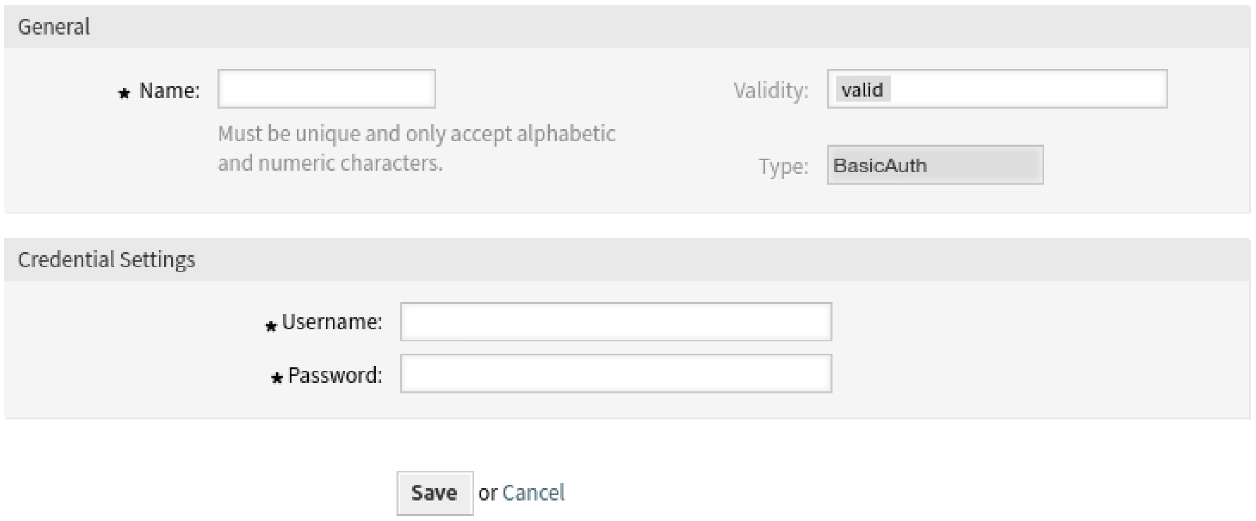
1. Choose a method in the left sidebar and select a type from its drop-down.
2. Fill in the required fields.
3. Click on the *Save* button.

To edit a credential:

1. Click on a credential in the list of credentials.
2. Modify the fields.
3. Click on the *Save* or *Save and finish* button.

To delete a credential:

1. Click on the trash icon in the second last column of the overview table.
2. Click on the *Confirm* button.

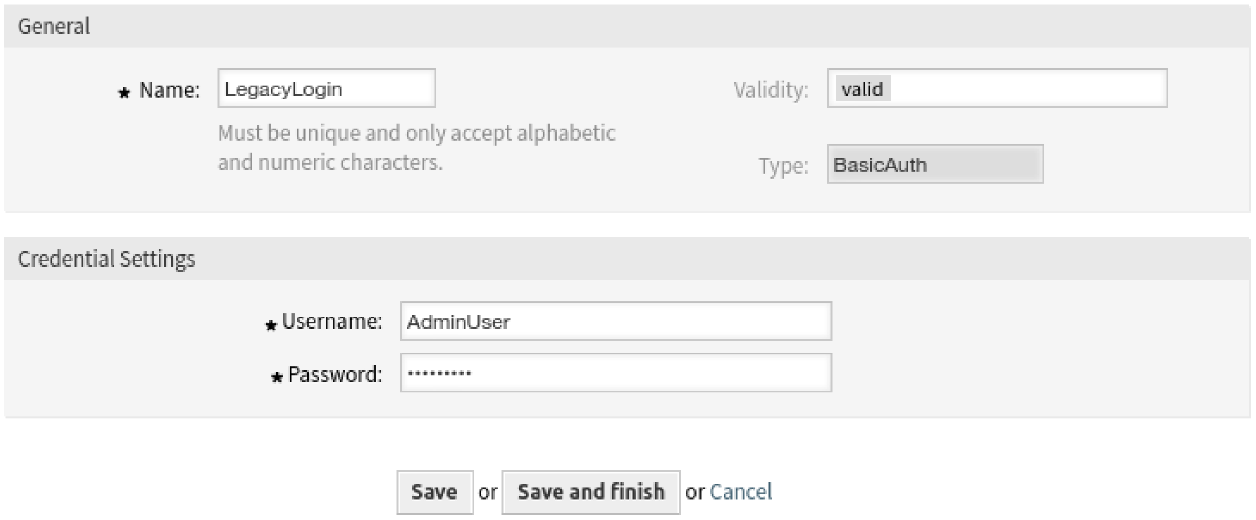

General

★ Name:	
	Must be unique and only accept alphabetic and numeric characters.
Validity:	valid
Type:	BasicAuth

Credential Settings

★ Username:	
★ Password:	

Fig. 16: Add BasicAuth Credential Screen

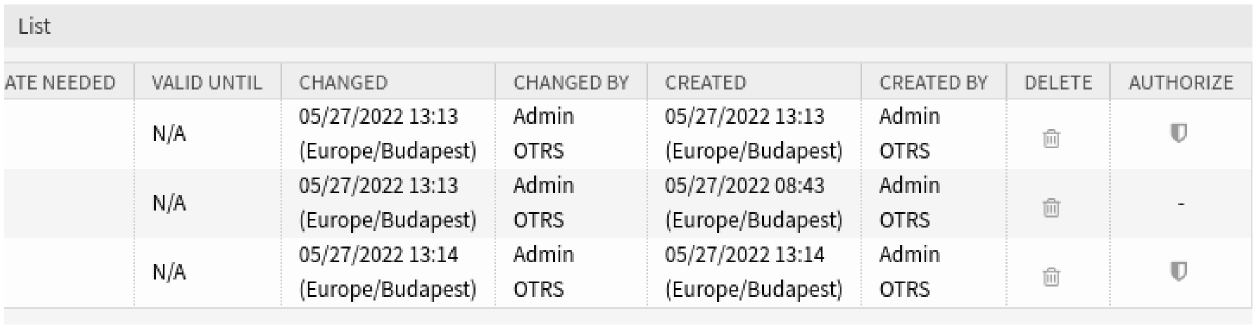

General

★ Name:	LegacyLogin
	Must be unique and only accept alphabetic and numeric characters.
Validity:	valid
Type:	BasicAuth

Credential Settings

★ Username:	AdminUser
★ Password:	••••••••••

Fig. 17: Edit BasicAuth Credential Screen


List

ATE NEEDED	VALID UNTIL	CHANGED	CHANGED BY	CREATED	CREATED BY	DELETE	AUTHORIZE
	N/A	05/27/2022 13:13 (Europe/Budapest)	Admin OTRS	05/27/2022 13:13 (Europe/Budapest)	Admin OTRS	🗑	🛡
	N/A	05/27/2022 13:13 (Europe/Budapest)	Admin OTRS	05/27/2022 08:43 (Europe/Budapest)	Admin OTRS	🗑	-
	N/A	05/27/2022 13:14 (Europe/Budapest)	Admin OTRS	05/27/2022 13:14 (Europe/Budapest)	Admin OTRS	🗑	🛡

Fig. 18: Delete Credential Screen

Note: If several credentials are added to the system, use the filter box to find a particular credential by just typing the name to filter.

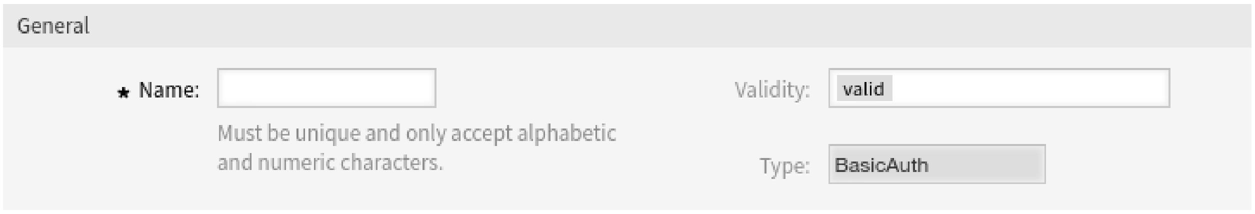
Note: If the emails cannot be fetched and *yes* is displayed in the *Update Needed* column, click the shield icon to authorize the credentials manually. If this happens regularly, please contact the Customer Solutions Team for further analysis.

3.4.2 Credential Settings

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

General Credential Settings

These settings are the same for all types of credentials.


General

★ Name:

Must be unique and only accept alphabetic and numeric characters.

Validity: valid

Type: BasicAuth

Fig. 19: Credential General Screen

Name *
The name of this resource. Must be unique and only accept alphabetic and numeric characters. The name will be displayed in the overview table.

Validity
Set the validity of this resource. Each resource can be used in OTRS only, if this field is set to *valid*. Setting this field to *invalid* or *invalid-temporarily* will disable the use of the resource.

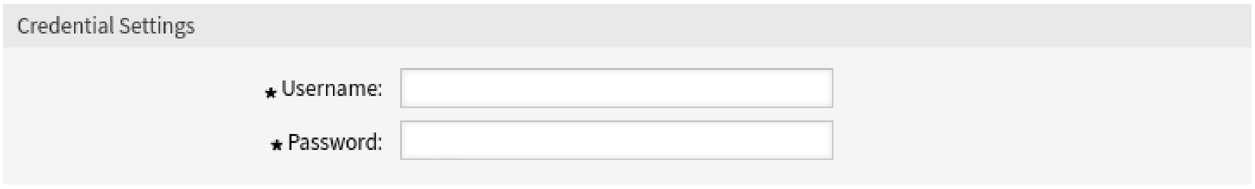
Type
This type have been selected in the previous page and can not be changed here anymore. This is a read-only field.

BasicAuth Credential Settings

The following settings are available if *BasicAuth* was selected in the credential management screen. The fields marked with an asterisk are mandatory.

Username *
The username for the credential.

Password *
The password for the credential.


Credential Settings

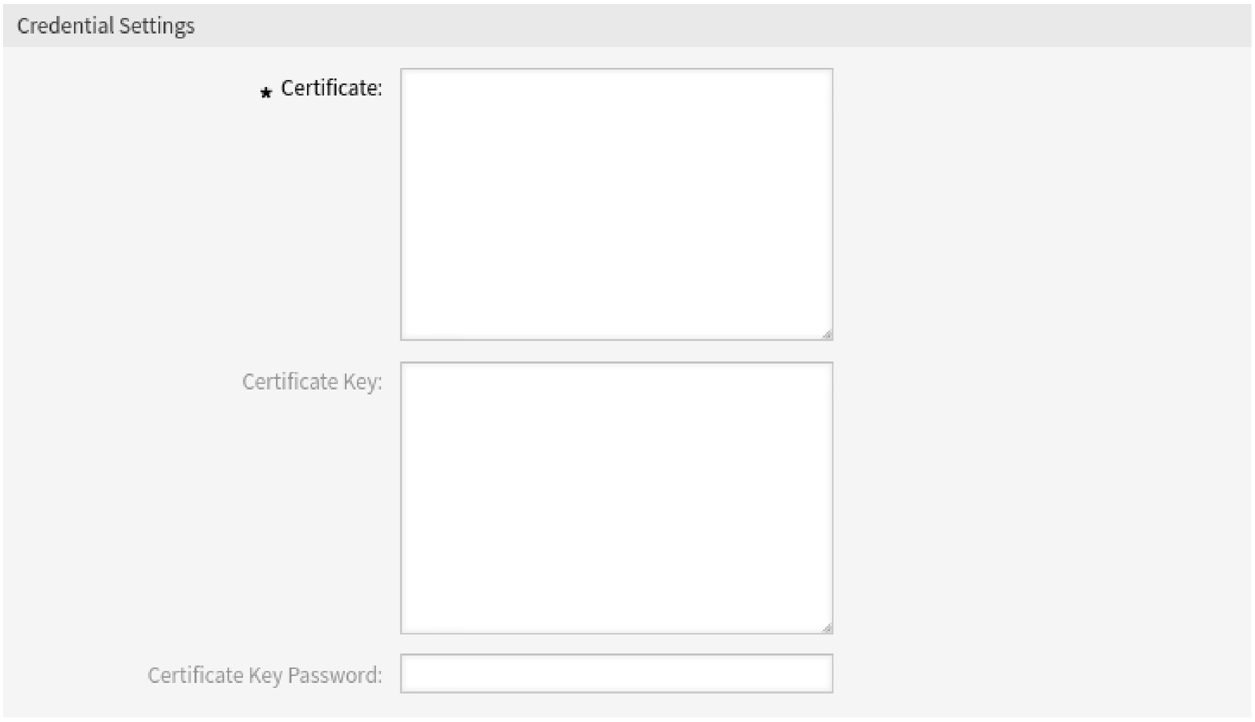
★ Username:

★ Password:

Fig. 20: BasicAuth Credential Settings Screen

Certificate Credential Settings

The following settings are available if *Certificate* was selected in the credential management screen. The fields marked with an asterisk are mandatory.


Credential Settings

★ Certificate:

Certificate Key:

Certificate Key Password:

Fig. 21: Certificate Credential Settings Screen

Certificate *
The certificate itself as a string. A certificate should look like this:

```
-----BEGIN CERTIFICATE-----
bjJ4b05tdXZkZnZ1ZUF5S0lwZlJUTUlYY0dGa0l4OFpHc0xPbU93UkhZTG5Pc3BE
NDJEUUM0NFpJZFRrUFdubGJ6UUJiYmNtdnh3ZVRiSG0zUVhwVXVVdXpXZzRaREN0
...
ZW9NeWpsRXo2Z1VLTk9pT0RoVmJOVE5TbTFGTFA2TXRScW16dWxvUWloWERWaTQy
UVNTdkhQN0lHVjNJT2JTWnpWRWxqc1ZHMFVjSWxXOGNxUW9ySHhwVm5lbQo=
-----END CERTIFICATE-----
```

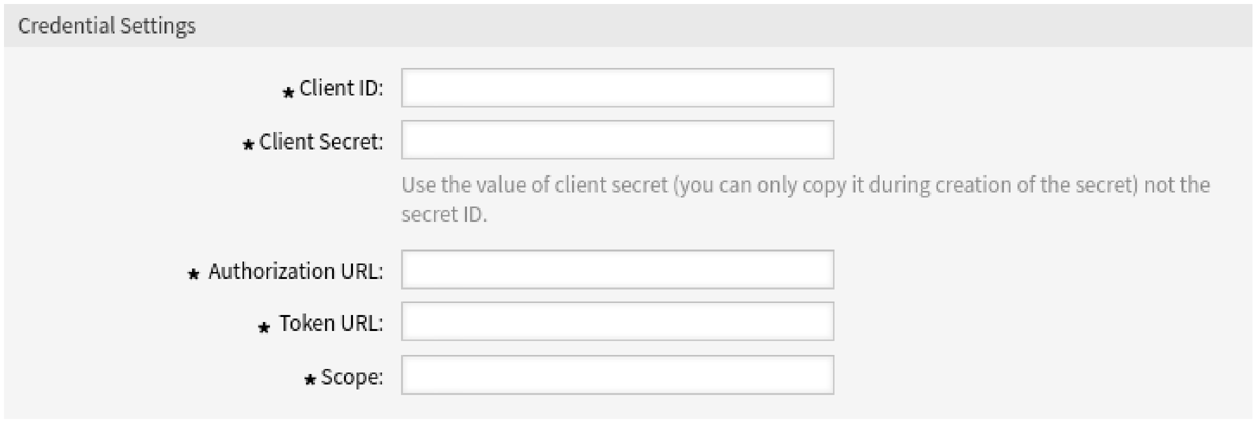
Certificate Key
The private key of the certificate. A private key should look like this:

```
-----BEGIN RSA PRIVATE KEY-----
MFZFUnBqbUhYd2pIZFNEZ29oV01BdTBGek84UkRaQWxQeVR1azE5OXJEVG45OWUz
b25GeWRsNzlBV1BHTjFzOWs2NVFLdHFTSDRYWnFjc3oxS09Rd3Y2NFc1ZkRFNGhp
...
TUZaRlVuQnFiVWhZZDJwSVpGTkVaMjlvVjBsQmRUQkdlazg0VWtSYVFXeFFlVlIx
T1dVemIyNUdlV1JzTnpsQ1YxQkhUakZ6T1dzMk5WRkxkSEZUU0RSWQo=
-----END RSA PRIVATE KEY-----
```

Certificate Key Password
If the certificate key has a password, you can enter it here.

Authorization Code Flow Credential Settings

The following settings are available if *Authorization Code Flow* was selected in the credential management screen. The fields marked with an asterisk are mandatory.


Credential Settings

★ Client ID:	
★ Client Secret:	
	Use the value of client secret (you can only copy it during creation of the secret) not the secret ID.
★ Authorization URL:	
★ Token URL:	
★ Scope:	

Fig. 22: Authorization Code Flow Credential Settings Screen

Client ID *
The client ID for the application.

Client Secret *
The client secret for the application. Use the value of client secret (you can only copy it during creation of the secret) not the secret ID.

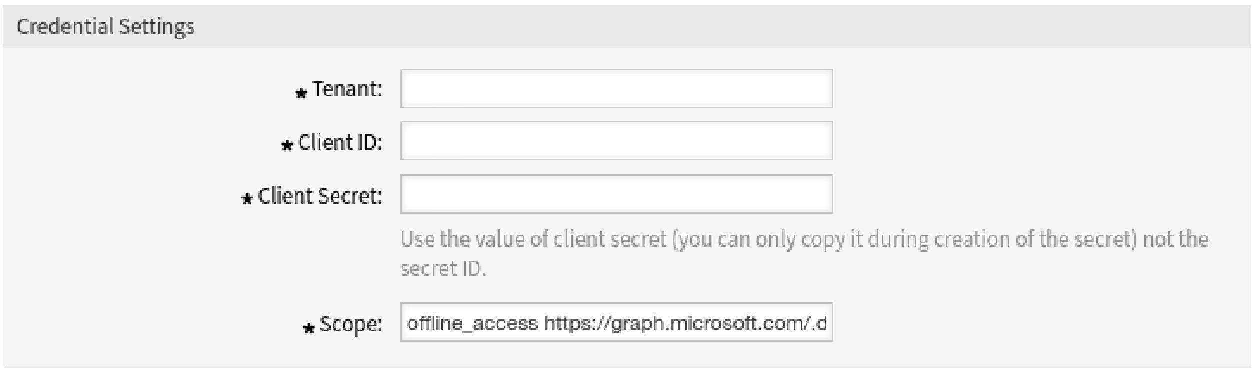
Authentication URL *
The authentication URL for the application.

Token URL *
The token URL for the application.

Scope *
The scope for the application.

Microsoft Graph App Credential Settings

The following settings are available if *Microsoft Graph App* was selected in the credential management screen. The fields marked with an asterisk are mandatory.


Credential Settings

★ Tenant:	
★ Client ID:	
★ Client Secret:	
	Use the value of client secret (you can only copy it during creation of the secret) not the secret ID.
★ Scope:	offline_access https://graph.microsoft.com/.d

Fig. 23: Microsoft Graph App Credential Settings Screen

Tenant *
: The tenant for the application.

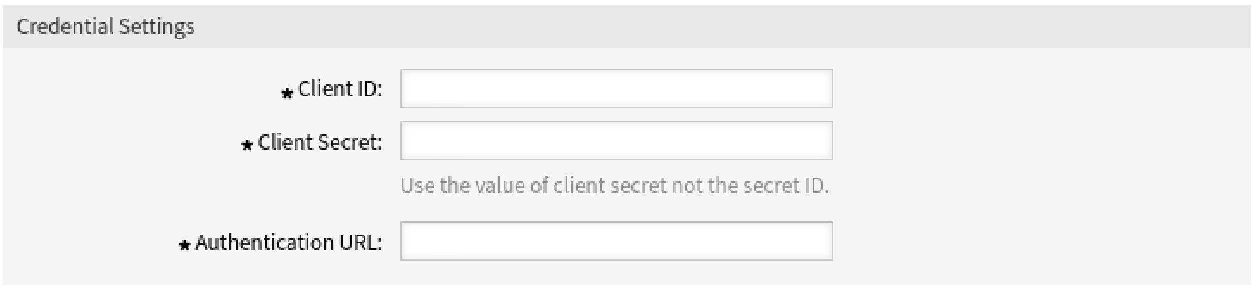
Client ID *
: The client ID for the application.

Client Secret *
: The client secret for the application. Use the value of client secret (you can only copy it during creation of the secret) not the secret ID.

Scope *
: The scope for the application.

Resource Owner Password Basic (Cisco) Credential Settings

The following settings are available if *Resource Owner Password Basic (Cisco)* was selected in the credential management screen. The fields marked with an asterisk are mandatory.


Credential Settings

★ Client ID:	
★ Client Secret:	
	Use the value of client secret not the secret ID.
★ Authentication URL:	

Fig. 24: Resource Owner Password Basic (Cisco) Credential Settings Screen

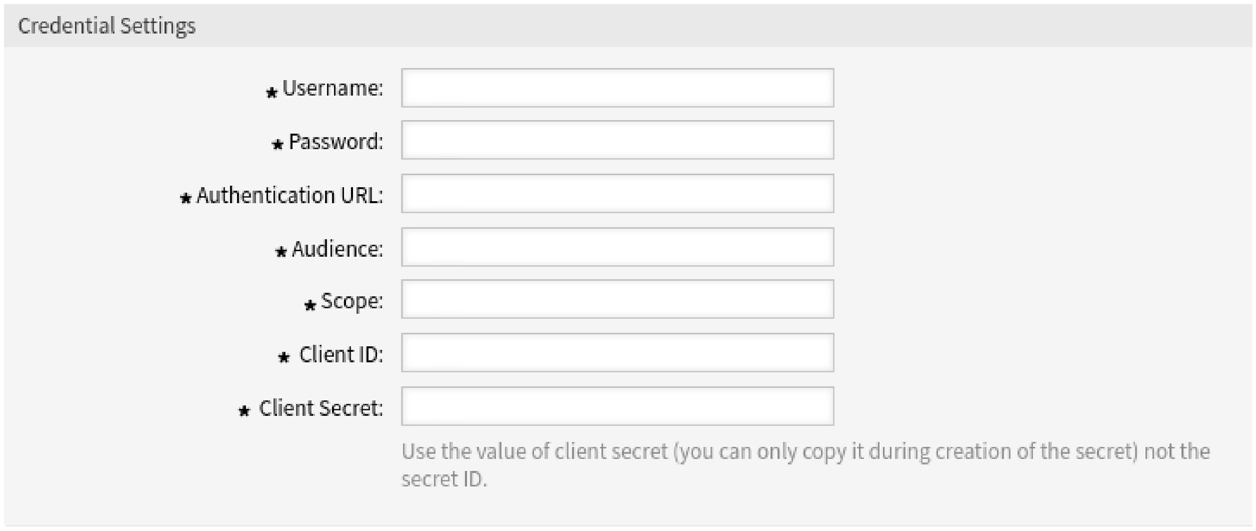
Client ID *
: The client ID for the application.

Client Secret *
: The client secret for the application. Use the value of client secret not the secret ID.

Authentication URL *
The authentication URL for the application.

Resource Owner Password Flow Credential Settings

The following settings are available if *Resource Owner Password Flow* was selected in the credential management screen. The fields marked with an asterisk are mandatory.



Credential Settings

★ Username:	
★ Password:	
★ Authentication URL:	
★ Audience:	
★ Scope:	
★ Client ID:	
★ Client Secret:	

Use the value of client secret (you can only copy it during creation of the secret) not the secret ID.

Fig. 25: Resource Owner Password Flow Credential Settings Screen

Username *
The username for the application.

Password *
The password for the application.

Authentication URL *
The authentication URL for the application.

Audience *
The audience for the application.

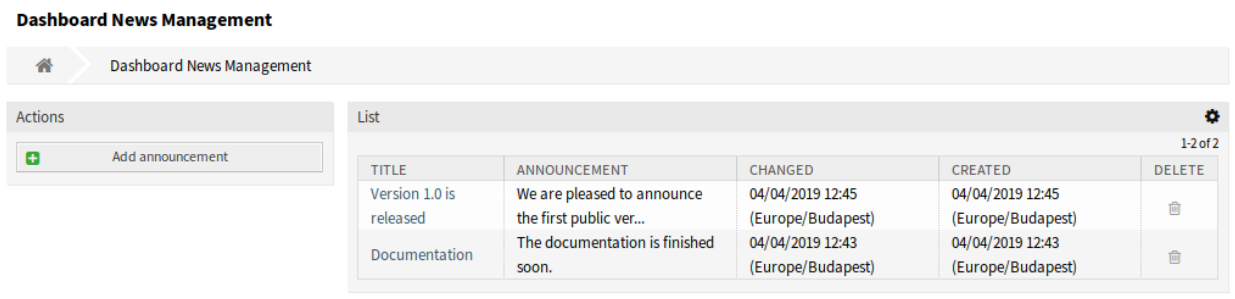
Scope *
The scope for the application.

Client ID *
The client ID for the application.

Client Secret *
The client secret for the application. Use the value of client secret (you can only copy it during creation of the secret) not the secret ID.

3.5 Dashboard News

Use this screen to add announcements to agent dashboard. The announcement management screen is available in the *Dashboard News* module of the *Communication & Notifications* group.



TITLE	ANNOUNCEMENT	CHANGED	CREATED	DELETE
Version 1.0 is released	We are pleased to announce the first public ver...	04/04/2019 12:45 (Europe/Budapest)	04/04/2019 12:45 (Europe/Budapest)	🗑
Documentation	The documentation is finished soon.	04/04/2019 12:43 (Europe/Budapest)	04/04/2019 12:43 (Europe/Budapest)	🗑

Fig. 26: Dashboard News Management Screen

3.5.1 Manage Announcements

To add an announcement:

1. Click on the *Add announcement* button in the left sidebar.
2. Fill in the required fields.
3. Click on the *Save* button.

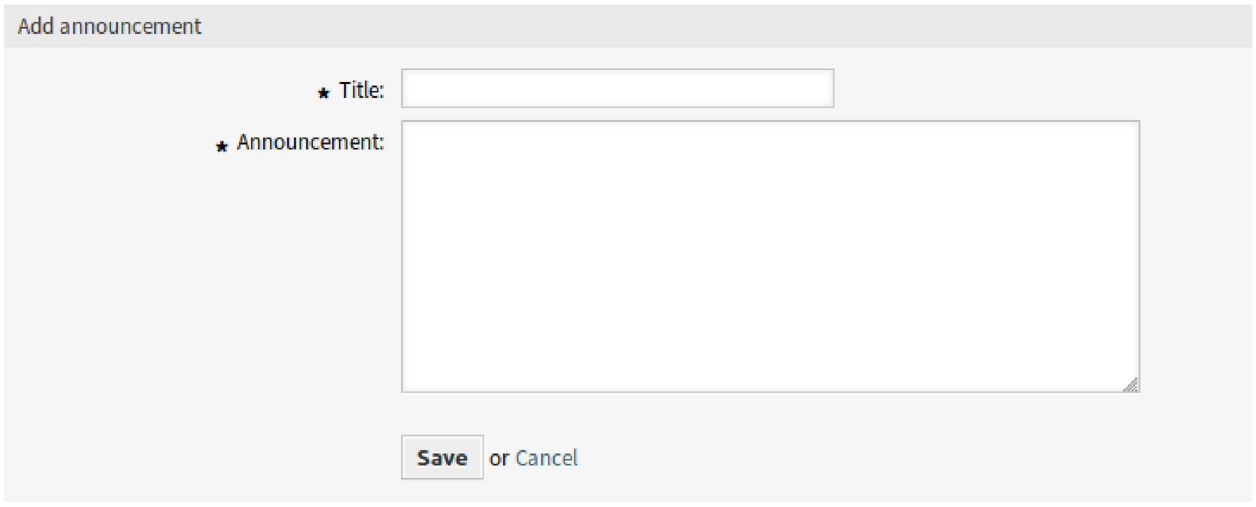
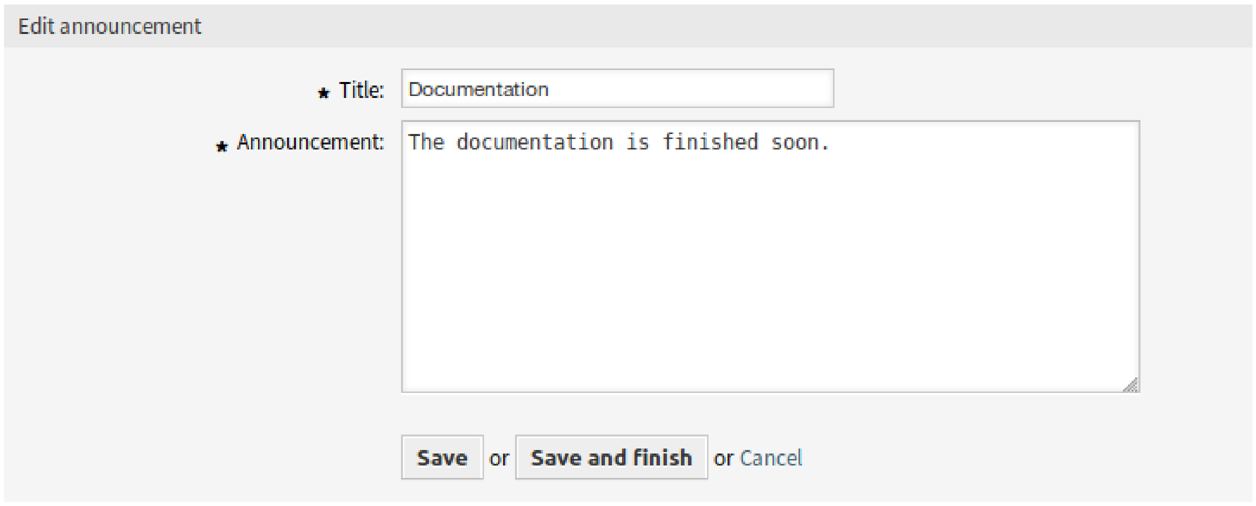


Fig. 27: Add Announcement Screen

To edit an announcement:

1. Click on an announcement in the list of announcements.
2. Modify the fields.
3. Click on the *Save* or *Save and finish* button.

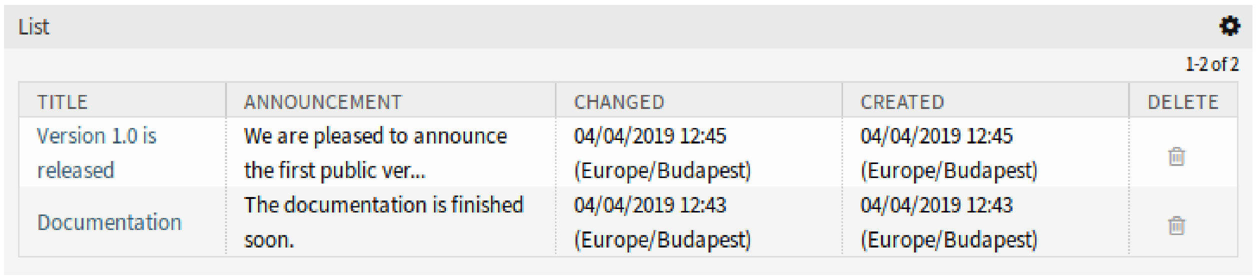

Edit announcement

★ Title:	Documentation
★ Announcement:	The documentation is finished soon.

Fig. 28: Edit Announcement Screen

To delete an announcement:

1. Click on the trash icon in the list of announcements.
2. Click on the *OK* button in the confirmation dialog.


List

1-2 of 2

TITLE	ANNOUNCEMENT	CHANGED	CREATED	DELETE
Version 1.0 is released	We are pleased to announce the first public ver...	04/04/2019 12:45 (Europe/Budapest)	04/04/2019 12:45 (Europe/Budapest)	🗑
Documentation	The documentation is finished soon.	04/04/2019 12:43 (Europe/Budapest)	04/04/2019 12:43 (Europe/Budapest)	🗑

Fig. 29: Delete Announcement Screen

To limit the number of displayed announcements per page:

1. Click on the gear icon in the top right corner of the overview header.
2. Select the maximum number of announcements displayed per page.
3. Click on the *Submit* button.

3.5.2 Announcement Settings

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

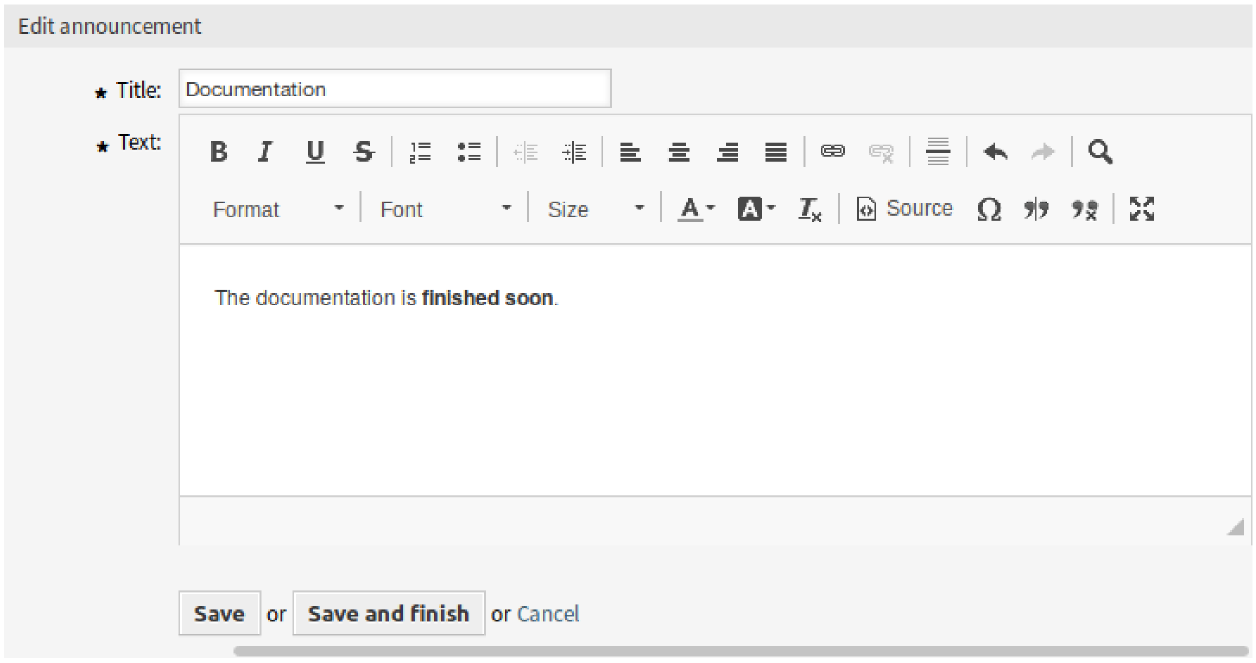
Title *
The title of the announcement. This will be displayed in the dashboard widget of the agent interface.

Announcement *
The body text of the announcement. This will be displayed in the overview screens and in the announcement details screen.

Rich Text editor can be enabled for the announcement text.

See also:

The `DashboardNewsWidget::RichTextField` setting needs to be enabled to use this feature.


Edit announcement

Title: Documentation

Text: The documentation is **finished soon**.

Fig. 30: Edit Announcement Screen with Rich Text Editor

Warning: If announcements were created using a Rich Text editor and the way to manage announcements was changed to normal text in the meantime, the content of the related announcements will be displayed with HTML tags and should be improved afterwards.

3.6 Email Addresses

The main channel of communication with the customers is often email. An organization consists of multiple department or teams. Email addresses differ for each group which is servicing your customer. You may have the following:

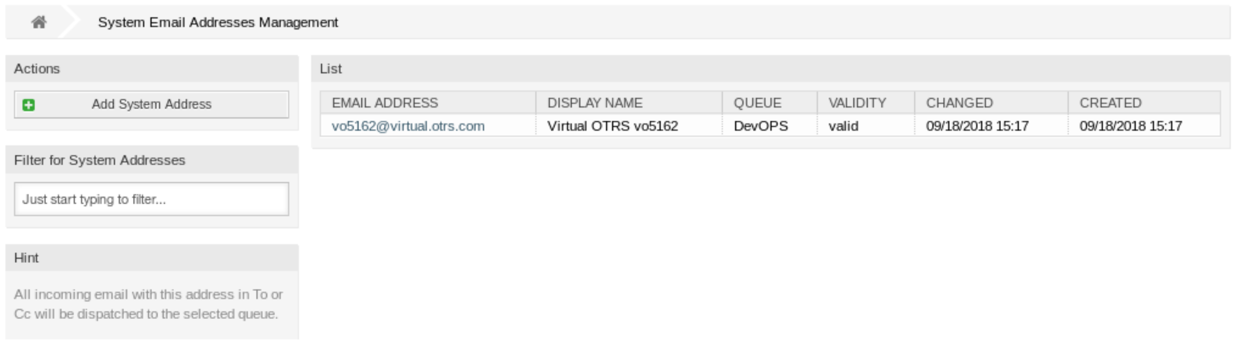
```
support@example.org
hr@exapmle.org
sales@example.org
```

These addresses are just some examples, and you may have many more. You use these channels to receive and send messages, and in mail clients, one can often send with the wrong address.

OTRS manages as many email addresses for your teams as needed. All your email addresses, whether for sending or receiving, are kept and configured nicely in one place. In the *Queue Settings*, the correct address is always chosen preventing that someone working in multiple roles sends an email out with the wrong account.

To enable OTRS to send emails, you need a valid email address to be used by the system. OTRS is capable of working with multiple email addresses, since many support installations need to use more than one. A queue can be linked to many email addresses, and vice versa. The address used for outgoing messages from a queue can be set when the queue is created.

Use this screen to add system email addresses to the system. An email address is already added to the system at installation time of OTRS. The email address management screen is available in the *Email Addresses* module of the *Communication & Notifications* group.


System Email Addresses Management
Actions
Add System Address
Filter for System Addresses
Just start typing to filter...
Hint

All incoming email with this address in To or Cc will be dispatched to the selected queue.

List

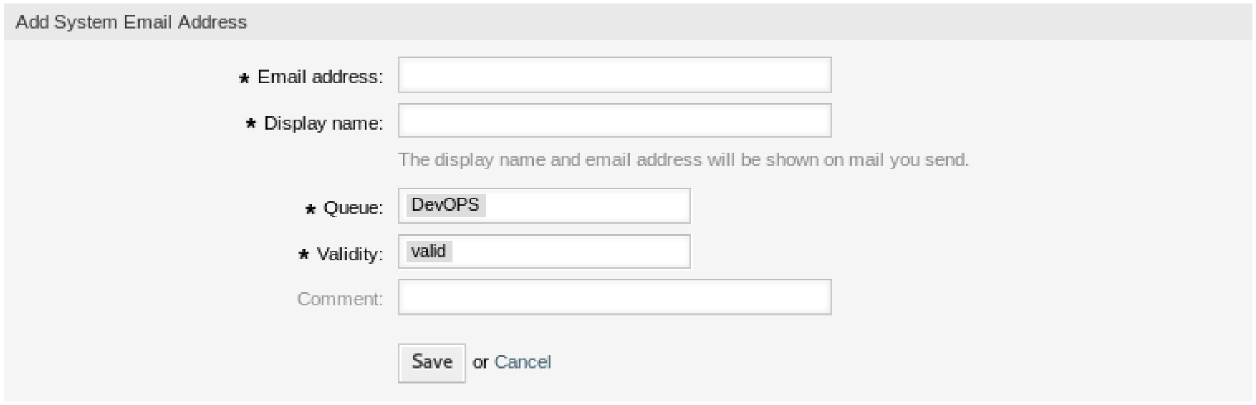
EMAIL ADDRESS	DISPLAY NAME	QUEUE	VALIDITY	CHANGED	CREATED
vo5162@virtual.otrs.com	Virtual OTRS vo5162	DevOPS	valid	09/18/2018 15:17	09/18/2018 15:17

Fig. 31: Email Address Management Screen

3.6.1 Manage Email Addresses

To add an email address:

1. Click on the *Add System Address* button in the left sidebar.
2. Fill in the required fields.
3. Click on the *Save* button.



* Email address:	
* Display name:	
	The display name and email address will be shown on mail you send.
* Queue:	DevOPS
* Validity:	valid
Comment:	

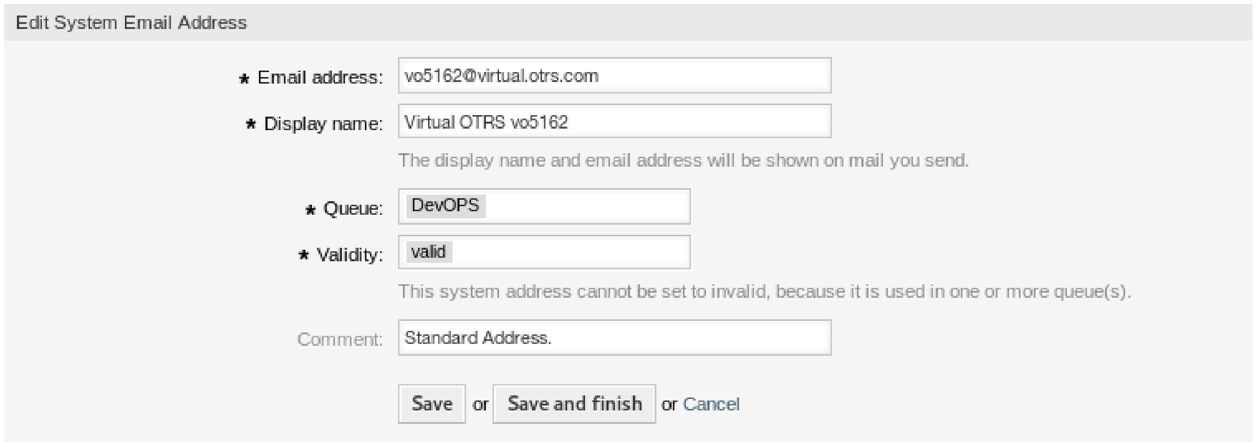
Fig. 32: Email Address Add Screen

Warning: Email addresses can not be deleted from the system. They can only be deactivated by setting the *Validity* option to *invalid* or *invalid-temporarily*.

Note: Once an email address is added and set to valid, OTRS cannot send an email to this address. This prevents loopbacks which could crash your system. If you need to transfer information between departments please use the split article action. This will allow you to create a new ticket to another team for assigning a task, for example.

To edit an email address:

1. Click on an email address in the list of email addresses.
2. Modify the fields.
3. Click on the *Save* or *Save and finish* button.


Edit System Email Address

⋆ Email address:	vo5162@virtual.otrs.com
⋆ Display name:	Virtual OTRS vo5162
	The display name and email address will be shown on mail you send.
⋆ Queue:	DevOPS
⋆ Validity:	valid
	This system address cannot be set to invalid, because it is used in one or more queue(s).
Comment:	Standard Address.

Fig. 33: Email Address Edit Screen

Note: If several email addresses are added to the system, use the filter box to find a particular email address by just typing the name to filter.

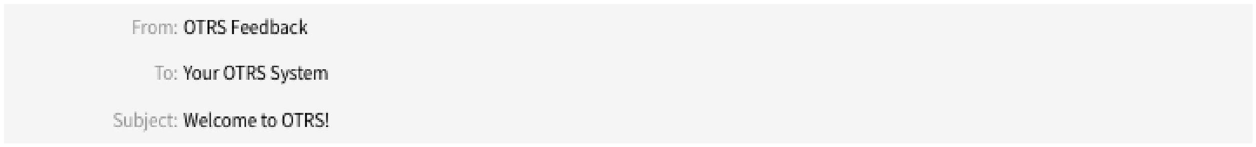
3.6.2 Email Address Settings

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

Email address *
The email address to be added.

Display name *
The name of this resource. Any type of characters can be entered to this field including uppercase letters and spaces. The name will be displayed in the sender information of the article.

Queue *
The queue, to which the email address will be added as default email address.



From: OTRS Feedback

To: Your OTRS System

Subject: Welcome to OTRS!

Fig. 34: Sender Information

Note: This setting will apply if the email is distributed via the recipient address. This setting can be overridden by *Postmaster Filters* or in the *Mail Account Settings* when *Dispatching by selected Queue* is chosen.

Validity *
Set the validity of this resource. Each resource can be used in OTRS only, if this field is set to *valid*. Setting this field to *invalid* or *invalid-temporarily* will disable the use of the resource.

Note: An email address can only be set to *invalid* or *invalid-temporarily*, if it is not assigned to any queue.

Comment
Add additional information to this resource. It is recommended to always fill this field as a description of the resource with a full sentence for better clarity, because the comment will be also displayed in the overview table.

3.6.3 Setting up Outgoing Emails

For outgoing emails, OTRS can be configured in a variety of ways. The best option to do this might depend on your specific circumstances.

Sending Mails via Local MTA

Using a local mail transfer agent (MTA) to send mails directly or relay mails to a smart host is the **recommended** option. It provides local mail queuing in the case of network connectivity issues and provides you with the capability to authenticate against multiple outgoing mail accounts or accounts using other than basic authentication methods.

A default setup could use the *Sendmail* binary and *Postfix*.

Sending emails via SMTP (Simple Mail Transfer Protocol) relay via a smart host can be used in case you are using another mail server in your network, or on the internet. Your local MTA must be configured with the credentials for authentication and the target MTA must allow the user to send mails as other users of the domain.

Sending emails in the name of another domain requires either a sender policy framework (SPF) record, allowing the local server to send as the user, or in the case that the local mail server should act as the mail server for a target domain, mail exchange resource record (MX) and domain name service (DNS) C-name records for the local server are required. The MTA must, in both cases, be correctly configured to send as the target domain.

Note: An SPF record is not necessarily required. However, if the sender domain has an SPF record set, the MTA' s IP address must be listed in it, otherwise the SPF check will fail. Alternatively, you can send outbound mails to a relay host, and in that case the relay' s IP address must be listed in the SPF record of the sender domain. If the sender domain has no SPF record, there is nothing wrong with the OTRS MTA delivering outbound mails directly.

Sending Mails via External Email Provider

Alternatively, it is also possible to use an external email provider for outgoing emails.

Note: This feature is only available to *On-Premise* customers. If you are a *Managed* customer, this feature is taken care of by the *Customer Solutions Team* in **OTRS**. Please contact us via support@otrs.com or in the OTRS Portal.

Warning: Using Smarthost (sender relay host) is deprecated for new *Managed* customers since September 1, 2022. All existing instances will be modified until July 1, 2023.

To setup the system to use *Office 365* as email provider:

1. Read the POP, IMAP, and SMTP settings for Outlook.com and Add your Outlook.com account to another mail app or smart device chapters in the official documentation.
2. Copy the one-time application password shown in the administrator area of **Office 365**.
3. Go to the *System Configuration* screen of **OTRS** and add the following settings:
   - `SendmailModule` → `Kernel::System::Email::SMTPTLS`
   - `SendmailModule::AuthPassword` → the application password from Office 365
   - `SendmailModule::AuthUser` → *email-address@company.tld*
   - `SendmailModule::Host` → *SMTP.office365.com*
   - `SendmailModule::Port` → *587*

To setup the system to use *Gmail* as email provider:

1. Read the Send email from a printer, scanner, or app chapter in the official Help Center.
2. Go to the *System Configuration* screen of **OTRS** and add the following settings:
   - `SendmailModule` → `Kernel::System::Email::SMTPTLS`
   - `SendmailModule::AuthPassword` → your email password
   - `SendmailModule::AuthUser` → *email-address@company.tld*
   - `SendmailModule::Host` → *smtp-relay.gmail.com*
   - `SendmailModule::Port` → *587*

To setup the system to use another external email provider:

1. Read the official documentation of the external provider.
2. Go to the *System Configuration* screen of **OTRS** and add the following settings:

- `SendmailModule` → `Kernel::System::Email::SMTPTLS`
- `SendmailModule::AuthPassword` → your email password
- `SendmailModule::AuthUser` → your email address or username
- `SendmailModule::Host` → the SMTP address of the external provider
- `SendmailModule::Port` → the port for the SMTP server

See also:

You can find detailed instructions in the *PostMaster Mail Accounts* chapter about how to configure incoming emails.

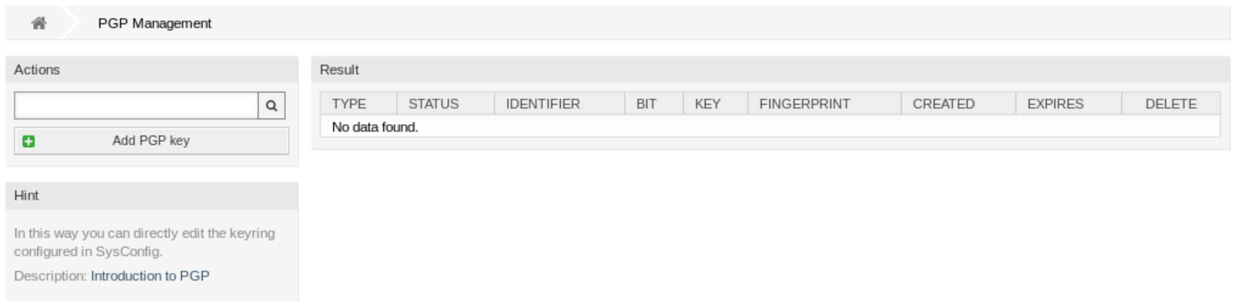
3.7 PGP Keys

Secure communications protect your customers and you. In the GDPR encryption is explicitly mentioned as one of the security and personal data protection measures in a few articles. Although under the GDPR encryption is not mandatory, it is indeed essential in some areas.

OTRS empowers you to encrypt communications where needed by means of *S/MIME Certificates* or *PGP Keys*.

Note: Setup of services and software required for encryption are not covered here because of independence to this software.

Use this screen to add PGP keys to the system. The PGP management screen is available in the *PGP Keys* module of the *Communication & Notifications* group.



In this way you can directly edit the keyring configured in SysConfig.

Description: Introduction to PGP

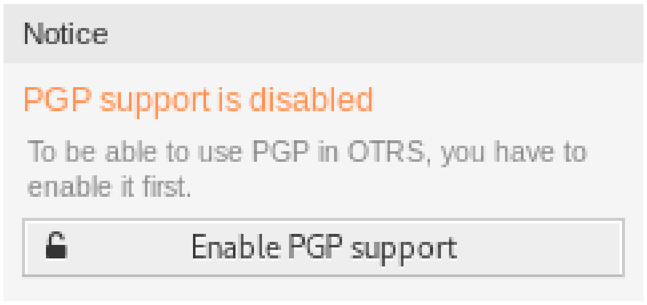
Result

TYPE	STATUS	IDENTIFIER	BIT	KEY	FINGERPRINT	CREATED	EXPIRES	DELETE
No data found.								

Fig. 35: PGP Management Screen

3.7.1 Manage PGP Keys

Note: To be able to use PGP keys in OTRS, you have to activate its setting first.



Notice

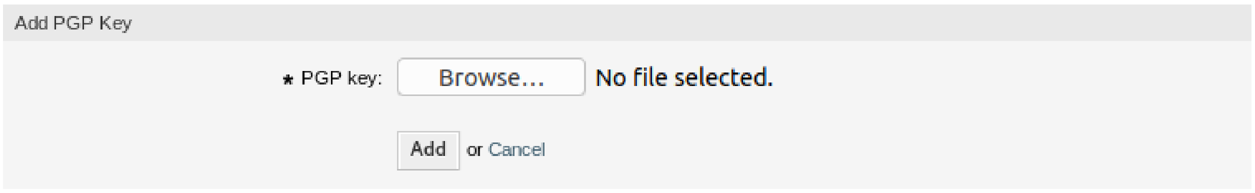
PGP support is disabled

To be able to use PGP in OTRS, you have to enable it first.

Fig. 36: Enable PGP Support

To add a PGP key:

1. Click on the *Add PGP Key* button in the left sidebar.
2. Click on *Browse*··· button to open the file dialog.
3. Select a PGP key from the file system.
4. Click on the *Add* button.



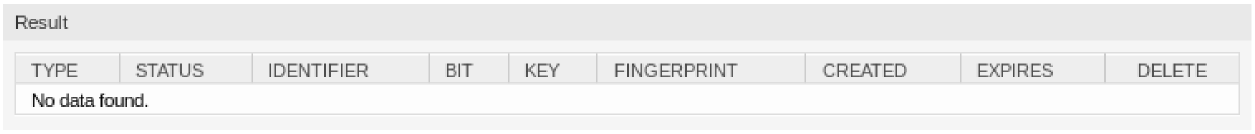
Add PGP Key

* PGP key: Browse... No file selected.

Fig. 37: Add PGP Key Screen

To delete a PGP key:

1. Click on the trash icon in the list of PGP keys.
2. Click on the *Confirm* button.



Result

TYPE	STATUS	IDENTIFIER	BIT	KEY	FINGERPRINT	CREATED	EXPIRES	DELETE
No data found.								

Fig. 38: Delete PGP Key Screen

Note: If several PGP keys are added to the system, use the search box to find a particular PGP key.

To adjust the PGP settings of the system:

1. Go to the *System Configuration* screen.

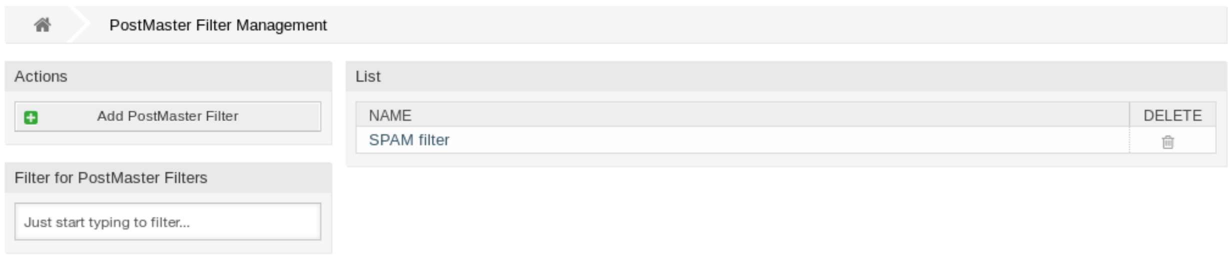
2. Navigate to *Core* → *Crypt* → *PGP* in the navigation tree.
3. Review the settings.

3.8 Postmaster Filters

Pre-sorting standard mail done in a mail room takes care that not every piece of mail sent to the office goes to the same group of people. After a second look at the envelope, rerouting occurs where needed.

OTRS uses so-called *postmaster filters* to read the email's envelope and take further action. Depending upon, for example, a subject or sender, an email bound for the service desk could land in a sub-queue or be redirected to a completely different team to create transparency and give your customer the fastest service possible.

Use this screen to add postmaster filters to the system. The postmaster filter management screen is available in the *PostMaster Filters* module of the *Communication & Notifications* group.


PostMaster Filter Management
Actions
Add PostMaster Filter
Filter for PostMaster Filters
Just start typing to filter...
List

NAME	DELETE
SPAM filter	

Fig. 39: Postmaster Filter Management Screen

3.8.1 Manage Postmaster Filters

Note: When adding or editing a postmaster filter, please keep in mind that they are evaluated in *ASCIIbetical* order by name.

To add a postmaster filter:

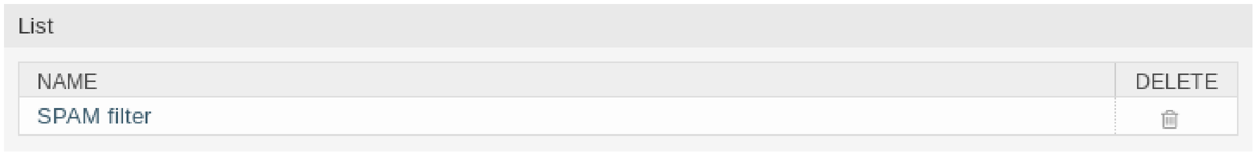
1. Click on the *Add PostMaster Filter* button in the left sidebar.
2. Fill in the required fields.
3. Click on the *Save* button.

To edit a postmaster filter:

1. Click on a postmaster filter in the list of postmaster filters.
2. Modify the fields.
3. Click on the *Save* or *Save and finish* button.

To delete a postmaster filter:

1. Click on the trash icon in the list of postmaster filters.
2. Click on the *Confirm* button.


List

NAME	DELETE
SPAM filter	🗑

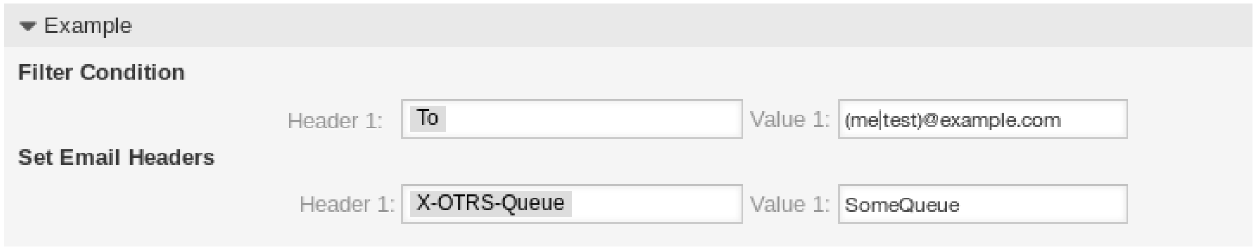
Fig. 40: Delete Postmaster Filter Screen

Note: If several postmaster filters are added to the system, a filter box is useful to find a particular postmaster filter by just typing to filter.

Warning: The maximum number of 50 postmaster filters should not be exceeded. Exceeding this limit may affect the system performance.

3.8.2 Postmaster Filter Settings

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.


Example

Filter Condition

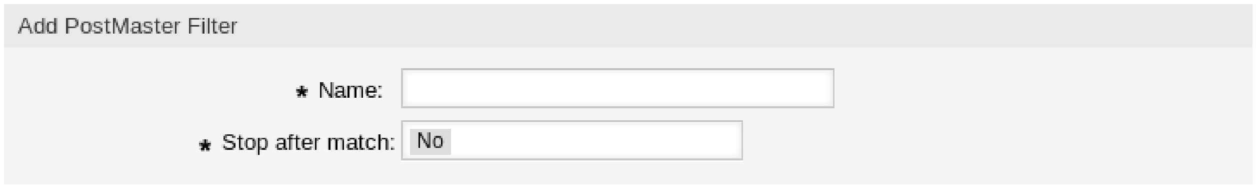
Header 1: To Value 1: (me|test)@example.com

Set Email Headers

Header 1: X-OTRS-Queue Value 1: SomeQueue

Fig. 41: Postmaster Filter Settings Example

Basic Postmaster Filter Settings


Add PostMaster Filter

* Name:
* Stop after match: No

Fig. 42: Postmaster Filter Settings - Basic

Name *
The name of this resource. Any type of characters can be entered to this field including uppercase letters and spaces. The name will be displayed in the overview table.

Note: When adding or editing one of the postmaster filters, remember multiple filters may apply to a single mail. Rules are executed and sorted by the ASCII value of the names. Based on the sorted order in the overview, they are applied from top to bottom. Look at the ASCII table to see how to sort your names based on the *ASCIIbetical* order.

Stop after match *
Postmaster filters are evaluated in *ASCIIbetical* order. This setting defines the evaluation of the subsequent postmaster filters.

No
All postmaster filters are executed.

Yes
The current postmaster filter is still evaluated, but evaluation of the remaining filters is canceled.

Filter Condition

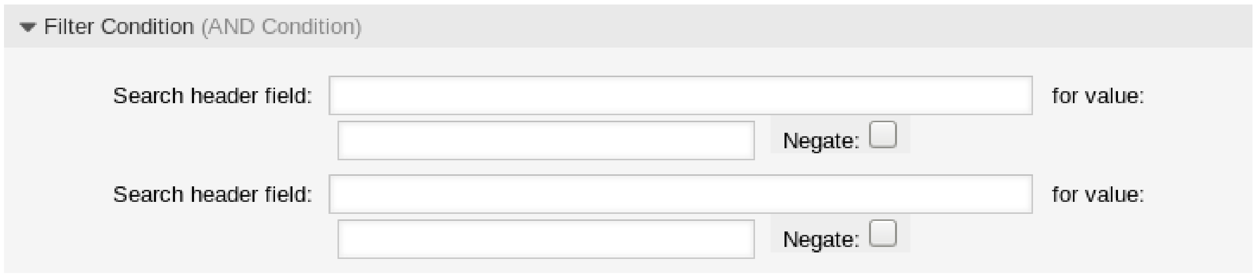


Fig. 43: Postmaster Filter Settings - Filter Condition

A postmaster filter consists of one or more conditions that must be met in order for the defined actions to be executed on the email. Filter conditions can be defined for specific mail header entries or for strings in the mail body.

Search header field …for value
Select a mail header or an `X-OTRS` header from the first drop-down list, and enter a value as search term for the selected mail header to the second field. Even regular expressions can be used for extended pattern matching.

A list of mail header entries can be found in RFC5322. It is also possible to define `X-OTRS` headers as filter condition. The different `X-OTRS` headers and their meaning are the following:

`X-OTRS-AttachmentCount`
This contains as value the number of attachments which are contained in the email (e.g. *0* for mails without attachments).

`X-OTRS-AttachmentExists`
Depending on whether attachments are included in the email this `X-OTRS` header is set to `yes`, or it has a `no` value if no attachments are included.

`X-OTRS-BodyDecrypted`
If the incoming mail was encrypted, it is possible to add a search term to look for the body of the incoming encrypted mail.

`X-OTRS-CustomerNo`
Set the customer ID for the ticket.

`X-OTRS-CustomerUser`
Set the customer user for the ticket.

`X-OTRS-DynamicField-<DynamicFieldName>`
: Saves an additional information value for the ticket on *<DynamicFieldName>* dynamic field. The possible values depend on dynamic field configuration (e.g. text: *Notebook*, date: *2010-11-20 00:00:00*, integer: *1*).

`X-OTRS-FollowUp-*`
: These headers are the same as the ones without the `FollowUp` prefix, but these headers are applied only for follow-up mails.

`X-OTRS-FollowUp-State-Keep`
: If set to *1*, the incoming follow-up message will not change the ticket state. For this purpose the header can be customized in the system configuration using option `KeepStateHeader`.

`X-OTRS-Ignore`
: If set to `Yes` or `True`, the incoming message will completely be ignored and never delivered to the system.

`X-OTRS-IsVisibleForCustomer`
: Controls if the article is shown to customer users. Possible values are *0* or *1*.

`X-OTRS-Lock`
: Set the lock state of a ticket. Possible values are `lock` or `unlock`.

`X-OTRS-Loop`
: If set to `Yes` or `True`, no auto answer is delivered to the sender of the message (mail loop protection).

`X-OTRS-Owner`
: Set the agent as owner for the ticket.

`X-OTRS-OwnerID`
: Set the agent ID as owner for the ticket.

`X-OTRS-Priority`
: Set the priority for the ticket.

`X-OTRS-Queue`
: Defines the queue in which the ticket should be sorted. If a queue is set with this header, this setting has priority over all other filter rules that refer to queues. If you use a sub-queue, specify it as *Parent::Sub*.

`X-OTRS-Responsible`
: Set the agent as responsible for the ticket.

`X-OTRS-ResponsibleID`
: Set the agent ID as responsible for the ticket.

`X-OTRS-SenderType`
: Set the sender type for the ticket. Possible values are `agent`, `system` or `customer`.

`X-OTRS-Service`
: Set the service for the ticket. If you use a sub-service, specify it as *Parent::Sub*.

`X-OTRS-SLA`
: Set the service level agreement for the ticket.

`X-OTRS-State`
: Set the state for the ticket.

`X-OTRS-State-PendingTime`
: Set the pending time for the ticket (you should sent a pending state via `X-OTRS-State`). You can specify absolute dates like *2010-11-20 00:00:00* or relative dates, based on the arrival time

of the email. Use the form `+ $Number $Unit`, where `$Unit` can be *s* (seconds), *m* (minutes), *h* (hours) or *d* (days). Only one unit can be specified. Examples of valid settings: *+50s* (pending in 50 seconds), *+30m* (30 minutes), *+12d* (12 days).

Note: Settings like *+1d 12h* are not possible. You can specify *+36h* instead.

`X-OTRS-Title`
Set the title for the ticket.

`X-OTRS-Type`
Set the type for the ticket.

Warning: These headers must be manually injected into the mail by means not provided for by OTRS. OTRS only accepts `X-OTRS` headers from trusted sources.

See also:

The *Mail Account Settings* defines the trust level.

Negate
If checked, the condition will use the negate search term.

Set Email Headers

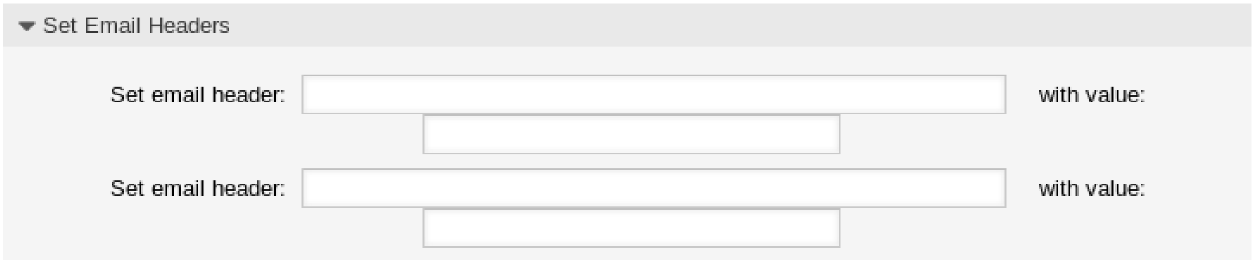


Fig. 44: Postmaster Filter Settings - Set Email Headers

In this section you can choose the actions that are triggered if the filter rules match.

Set email header …with value
Select an `X-OTRS` header from the first drop-down list, and add a value to the second field that should be set as value of the selected `X-OTRS` header.

See also:

The `X-OTRS` headers are already described above.

3.8.3 Filter Modules

OTRS provides a set of postmaster filter modules, that processes incoming email messages before they result in ticket articles and might perform actions during the communication flow. The filter modules are located in `Kernel/System/PostMaster/Filter/` and can be configured with system configuration options in most of the cases.

See also:

The majority of the configuration settings are below the name space `PostMaster::PreFilterModule`. Please see the configuration reference documentation for more information.

Follow-up Article Visibility Check

The postmaster filter module `FollowUpArticleVisibilityCheck` (located in `Kernel/System/PostMaster/Filter/FollowUpArticleVisibilityCheck.pm`) checks, if arrived emails should be marked as internal messages. If the email matches certain criteria, the filter module is able to set email headers about `IsVisibleForCustomer` status (`X-OTRS-IsVisibleForCustomer`) and `SenderType` (`X-OTRS-SenderType`) for further processing.

To give an overview about how OTRS detects and handles the visibility of incoming emails to related ticket customers, enclosed a summary, that describes how such messages are processed.

If an email message arrives at the OTRS system, the following circumstances will lead to an article, that is *visible* to the ticket customer user:

- The message cannot be detected as a follow-up and will lead to a new ticket.
- The `SenderType` was set to another value, than `customer` (maybe through a postmaster filter using header `X-OTRS-FollowUp-SenderType`, or the sender type is `system` because of notifications).
- The customer visibility was explicitly set to a positive value, with an email header `X-OTRS-FollowUp-IsVisibleForCustomer`.
- The ticket customer itself is the sender of the message.
- The sender is external and its `SenderType` is detected as `customer`, but did not have any correspondence yet (no email reference in any article of the particular ticket). This happens, when the *From* field is an unknown email address, that was not a recipient address before (i.e. used within a previous outbound email).

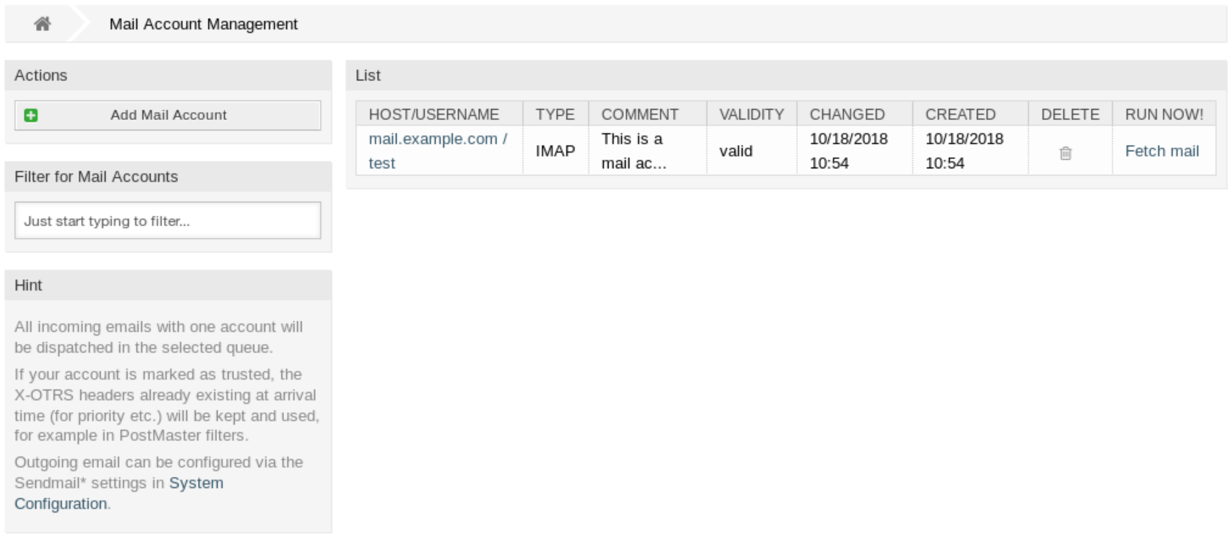
3.9 PostMaster Mail Accounts

Just as a company doesn't just have one department which receives traditional mail, your service desk will also serve multiple teams. Each team can have its physical email mailbox.

OTRS eases setup for email mailboxes. OTRS manages polling one or multiple email mailboxes of any internet standard type.

Use this screen to add mail accounts to the system. The mail account management screen is available in the *PostMaster Mail Accounts* module of the *Communication & Notifications* group.

Warning: When fetching mail, OTRS deletes the mail from the POP or IMAP server. There is no option to also keep a copy on the server. If you want to retain a copy on the server, you should create forwarding rules on your mail server. Please consult your mail server documentation for details.



HOST/USERNAME	TYPE	COMMENT	VALIDITY	CHANGED	CREATED	DELETE	RUN NOW!
mail.example.com / test	IMAP	This is a mail ac...	valid	10/18/2018 10:54	10/18/2018 10:54	🗑	Fetch mail

Hint

All incoming emails with one account will be dispatched in the selected queue.

If your account is marked as trusted, the X-OTRS headers already existing at arrival time (for priority etc.) will be kept and used, for example in PostMaster filters.

Outgoing email can be configured via the Sendmail* settings in System Configuration.

Fig. 45: Mail Account Management Screen

Note: If you choose IMAP, you can specify a folder for collection. Selective dispatching of mails is then possible.

All data for the mail accounts are saved in the OTRS database. The `bin/otrs.Console.pl Maint::PostMaster::MailAccountFetch` command uses the settings in the database and fetches the mail. You can execute it manually to check if all your mail settings are working properly.

On a default installation, the mail is fetched every 10 minutes when the OTRS daemon is running.

3.9.1 Manage Mail Accounts

To add a mail account:

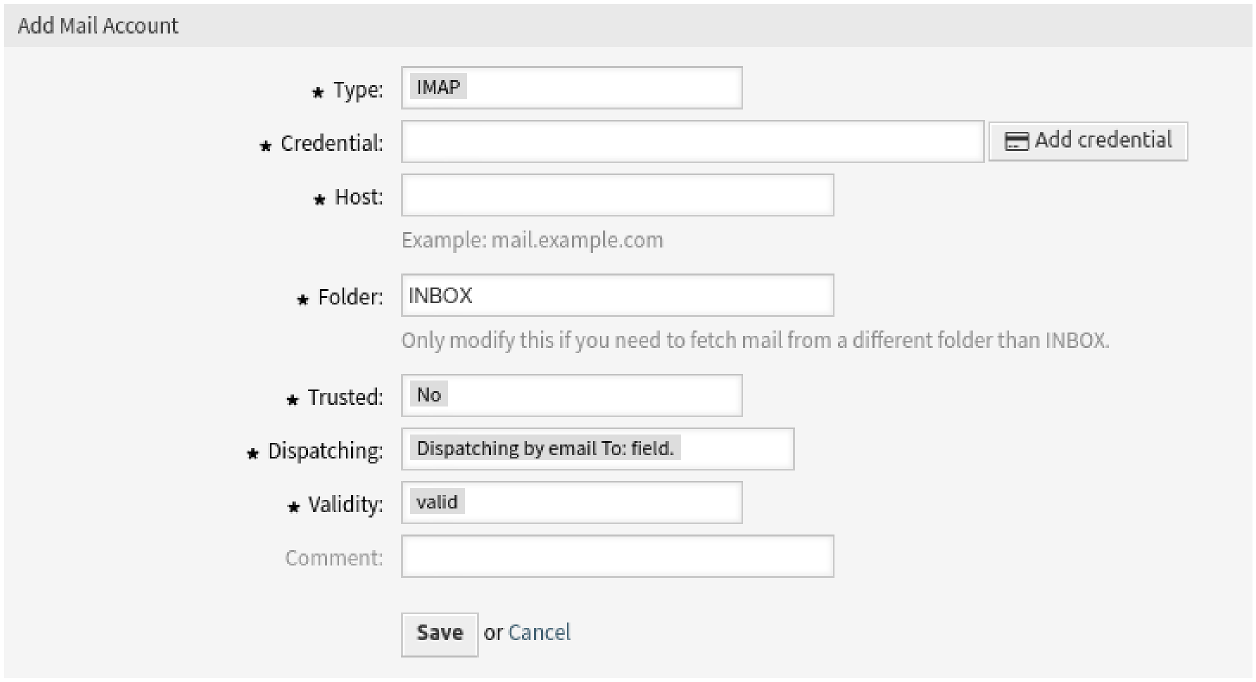
1. Click on the *Add Mail Account* button in the left sidebar.
2. Fill in the required fields.
3. Click on the *Save* button.

To edit a mail account:

1. Click on a mail account in the list of mail accounts.
2. Modify the fields.
3. Click on the *Save* or *Save and finish* button.

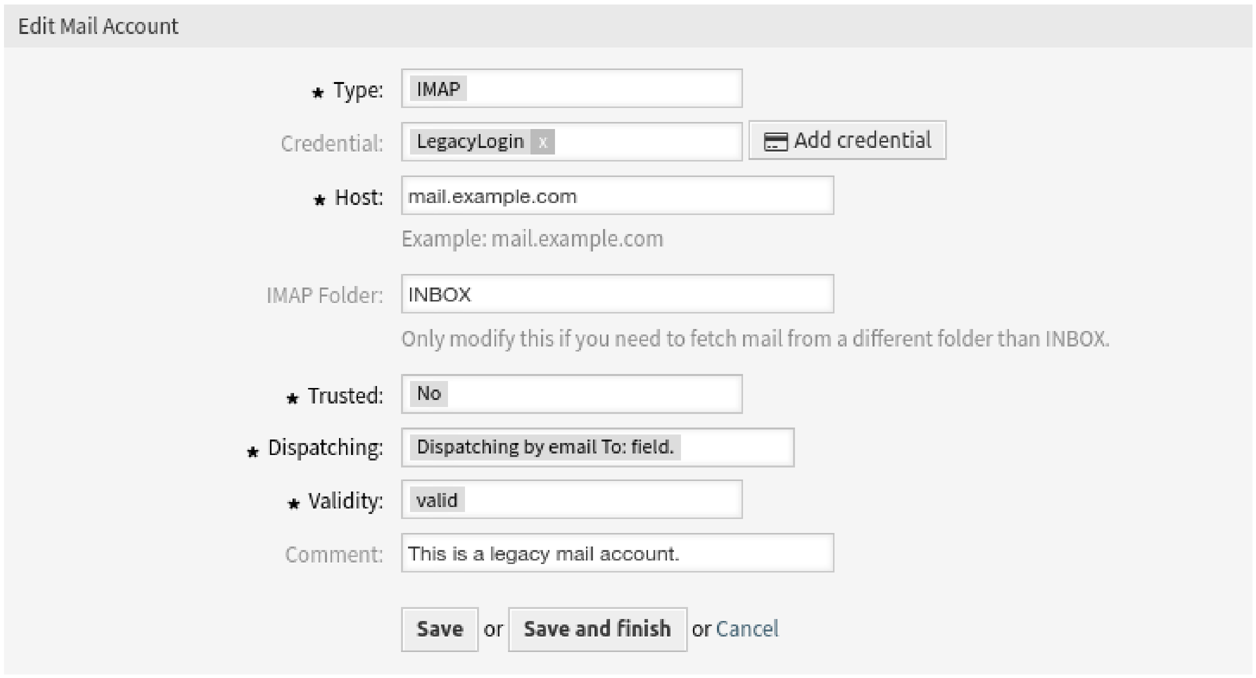
To delete a mail account:

1. Click on the trash icon in the list of mail accounts.
2. Click on the *Confirm* button.


Add Mail Account

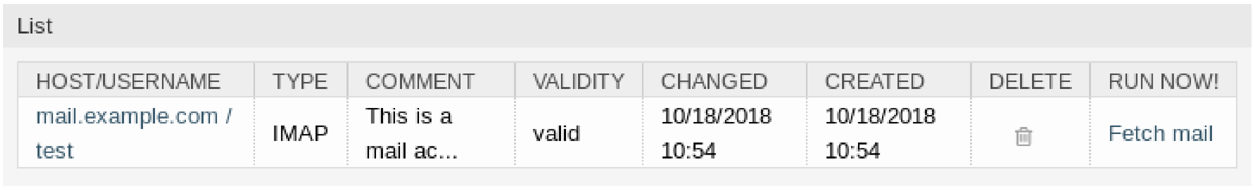
★ Type:	IMAP
★ Credential:	
★ Host:	
	Example: mail.example.com
★ Folder:	INBOX
	Only modify this if you need to fetch mail from a different folder than INBOX.
★ Trusted:	No
★ Dispatching:	Dispatching by email To: field.
★ Validity:	valid
Comment:	

Fig. 46: Add Mail Account Screen


Edit Mail Account

★ Type:	IMAP
Credential:	LegacyLogin x
★ Host:	mail.example.com
	Example: mail.example.com
IMAP Folder:	INBOX
	Only modify this if you need to fetch mail from a different folder than INBOX.
★ Trusted:	No
★ Dispatching:	Dispatching by email To: field.
★ Validity:	valid
Comment:	This is a legacy mail account.

Fig. 47: Edit Mail Account Screen



List

HOST/USERNAME	TYPE	COMMENT	VALIDITY	CHANGED	CREATED	DELETE	RUN NOW!
mail.example.com / test	IMAP	This is a mail ac...	valid	10/18/2018 10:54	10/18/2018 10:54	🗑	Fetch mail

Fig. 48: Delete Mail Account Screen

Note: If several mail accounts are added to the system, a filter box is useful to find a particular mail account by just typing to filter.

Warning: The maximum number of 10 *valid* mail accounts should not be exceeded. Exceeding this limit may affect the system performance.

3.9.2 Mail Account Settings

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

Type *
There are different kind of protocols that are used for fetching mail. The following protocols are supported:

- IMAP
- IMAPS
- IMAPTLS
- Outlook365
- POP3
- POP3S
- POP3TLS

Credential *
Select a credential that has been added in the *Credentials* screen. Click on the *Add credential* button to open the credential management screen.

E-mail Address *
The mail address of the mail account.

This setting is only available if *Outlook365* is selected in the *Type* field.

Host *
The host name of the mail account.

This setting is only available if *IMAP* or *POP3* is selected in the *Type* field.

Folder *
The folder in the mail account to be fetched. Other folders remain untouched. If you use a sub-folder, specify it as *Parent/Sub*.

This setting is only available if *IMAP* or *Outlook365* is selected in the *Type* field.

SSL Verify
If *yes* is selected, OTRS checks if the server certificate is valid. If *no* is selected OTRS does not check for a valid server certificate.

This setting is only available if *IMAPS*, *IMAPTLS*, *POP3S* or *POP3TLS* is selected in the *Type* field.

SSL Fingerprint
In cases where a self-signed certificate or a certificate issued by an unknown CA needs to be accepted without disabling the verification at all you can specify the fingerprint of the certificate as `algo$hex_fingerprint`, where `algo` is a fingerprint algorithm supported by OpenSSL, e.g. `sha1`, `sha256` etc. and `hex_fingerprint` is the hexadecimal representation of the binary fingerprint. Any colons inside the hex string will be ignored.

At the moment we only support one fingerprint.

This setting is only available if *IMAPS*, *IMAPTLS*, *POP3S* or *POP3TLS* is selected in the *Type* field.

SSL VerifyCN Name
In cases where the common name of the certificate is different from the configured host name you can specify the different CN name here.

This setting is only available if *IMAPS*, *IMAPTLS*, *POP3S* or *POP3TLS* is selected in the *Type* field.

See also:

The SSL settings are passed through `IO::Socket:SSL`. For detailed information read its official documentation.

Trusted *
If *Yes* is selected, any `X-OTRS` headers attached to an incoming message are evaluated and executed. Because the `X-OTRS` header can execute some actions in the ticket system, you should set this option to *Yes* only for known senders.

See also:

The `X-OTRS` headers are explained in the filter conditions of *Postmaster Filters*.

Dispatching *
The distribution of incoming messages can be controlled if they need to be sorted by queue or by the content of the *To:* field.

Dispatching by email To: field
The system checks if a queue is linked with the address in the *To:* field of the incoming mail. You can link an address to a queue in the *Email Addresses* screen. If the address in the *To:* field is linked with a queue, the new message will be sorted into the linked queue. If no link is found between the address in the *To:* field and any queue, then the message flows into the *Raw* queue in the system, which is the postmaster default queue after a default installation.

See also:

The postmaster default queue can be changed in system configuration setting PostmasterDefaultQueue.

Dispatching by selected Queue
All incoming messages will be sorted into the specified queue. The address where the mail was sent to is disregarded in this case.

Validity *
Set the validity of this resource. Each resource can be used in OTRS only, if this field is set to *valid*. Setting this field to *invalid* or *invalid-temporarily* will disable the use of the resource.

Comment
Add additional information to this resource. It is recommended to always fill this field as a description of the resource with a full sentence for better clarity, because the comment will be also displayed in the overview table.

3.9.3 Setting up Incoming Emails

For incoming emails, OTRS can be configured in a variety of ways.The best option to do this might depend on your specific circumstances.

Receiving Mails via Local MDA

Using a local mail transfer agent (MTA) to receive mails directly is the **recommended** option. It provides near to real time mail delivery and the setup is more stable than when you poll your mails via IMAP(S) or POP(S). Authentication is also no longer an issue.

Redirecting mails to another address
Your current mail server must relay the emails to the local MTA.

SMTP Transport
An alternative router is setup on the receiving MTA to deliver mails from specific users to an external SMTP server.

In both cases, the use of *Procmail* is required as the mail delivery agent (MDA), as well as the proper `$HOME` setup for the user *otrs* on your system. An example file `.procmailrc.dist` can be used directly without modification. Just remove the `.dist` ending. Mails sent to the local *otrs* user are then piped to the console command for processing.

Warning: Please note that all "push" mail transferred directly to the `otrs` user (i.e. via the console command) is always trusted. This means `X-OTRS` headers in emails can and will affect the distribution of mails and setting of ticket properties. This includes the `X-OTRS-FollowUp` headers.

For more information, please read the *Postmaster Filters* chapter.

Note: Errors will send the mail to the directory `var/spool/problem-email`.

Read the Postfix documentation or the Exim4 documentation for more information on the setup.

Note: This feature is only available to *On-Premise* customers. If you are a *Managed* customer, this feature is taken care of by the *Customer Solutions Team* in **OTRS**. Please contact us via support@otrs.com or in the OTRS Portal.

Receiving Mails via External Email Provider

Alternatively, it is also possible to use an external email provider for incoming emails.

To setup the system to use *Office 365* as email provider:

1. Read the POP and IMAP email settings for Outlook chapter in the official documentation.
2. Fill the form with the following data:
   - Type: select the desired protocol
   - Credential: your credential
   - Host: *outlook.office365.com*

Warning: On October 1st, 2022 Microsoft will stop supporting and retired basic authentication for POP3 and IMAP. Only OAuth2 can be used for authentication. For more information, please read the original blog post.

To setup the system to use *Gmail* as email provider:

1. Read the Check Gmail through other email platforms chapter in the official documentation.
2. Fill the form with the following data:
   - Type: select the desired protocol
   - Credential: your credential
   - Host: *imap.gmail.com*

To setup the system to use another external email provider:

1. Read the official documentation of the external provider.
2. Fill the form with the following data:
   - Type: select the desired protocol
   - Credential: your credential
   - Host: the POP3 or IMAP address of the external provider

See also:

You can find detailed instructions in the *Email Addresses* chapter about how to configure outgoing emails.

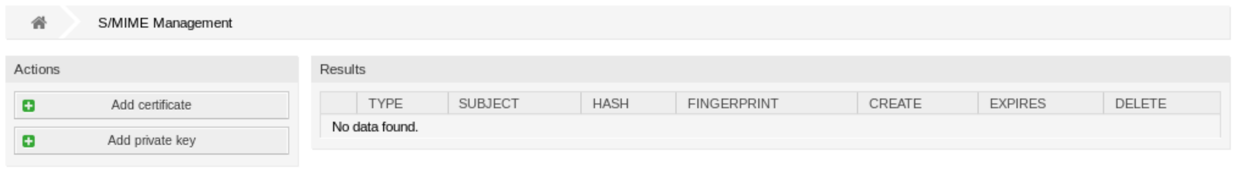
3.10 S/MIME Certificates

Faculty and staff have key roles safeguarding critical information by implementing information security policies, standards, and controls. Safe email communication is a vital part of protecting this communication.

OTRS empowers you to encrypt communications where needed by means of *S/MIME Certificates* or *PGP Keys*.

Note: Setup of services and software required for encryption are not covered here because of independence to this software.

Use this screen to add S/MIME certificates to the system. The S/MIME management screen is available in the *S/MIME Certificates* module of the *Communication & Notifications* group.

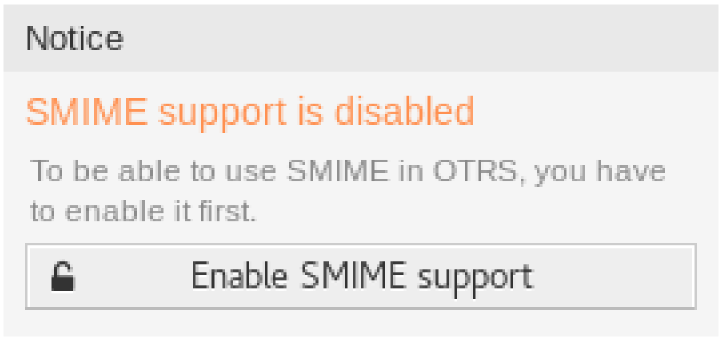


TYPE	SUBJECT	HASH	FINGERPRINT	CREATE	EXPIRES	DELETE
No data found.						

Fig. 49: S/MIME Management Screen

3.10.1 Manage S/MIME Certificates

Note: To be able to use S/MIME certificates in OTRS, you have to activate its setting first.


Notice

SMIME support is disabled

To be able to use SMIME in OTRS, you have to enable it first.

Fig. 50: Enable S/MIME Support

To add an S/MIME certificate:

1. Click on the *Add Certificate* button in the left sidebar.
2. Click on *Browse…* button to open the file dialog.
3. Select an S/MIME certificate from the file system.
4. Click on the *Add* button.

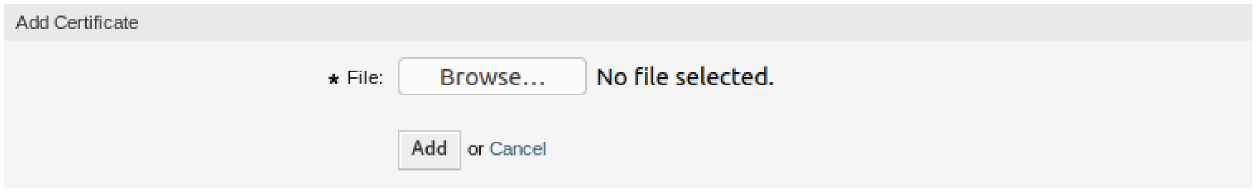


Fig. 51: Add S/MIME Certificate Screen

Note: Only non binary keys contain ASCII (Base64) armored data started with a `----- BEGIN CERTIFICATE -----` line can be uploaded which are most commonly `key.pem` or `root.crt`. Conversion of other formats like `cert.p7b` must be done using OpenSSL.

To add a private key:

1. Click on the *Add Private Key* button in the left sidebar.
2. Click on *Browse…* button to open the file dialog.
3. Select a private key from the file system.
4. Click on the *Submit* button.

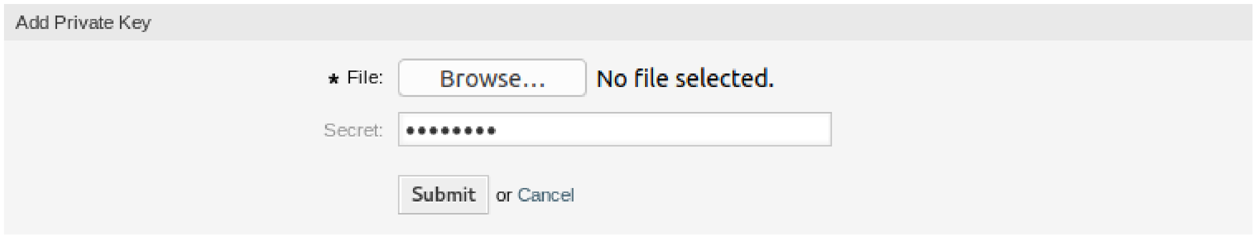
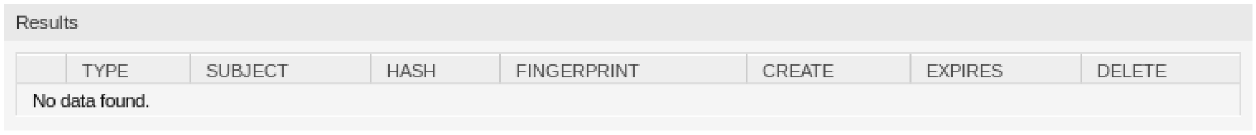


Fig. 52: Add S/MIME Private Key Screen

To delete an S/MIME certificate:

1. Click on the trash icon in the list of S/MIME certificates.
2. Click on the *Confirm* button.


Results

	TYPE	SUBJECT	HASH	FINGERPRINT	CREATE	EXPIRES	DELETE
No data found.							

Fig. 53: Delete S/MIME Certificate Screen

Note: If several S/MIME certificates are added to the system, use the filter box to find a particular S/MIME certificate by just typing the name to filter.

To adjust the S/MIME certificate settings of the system:

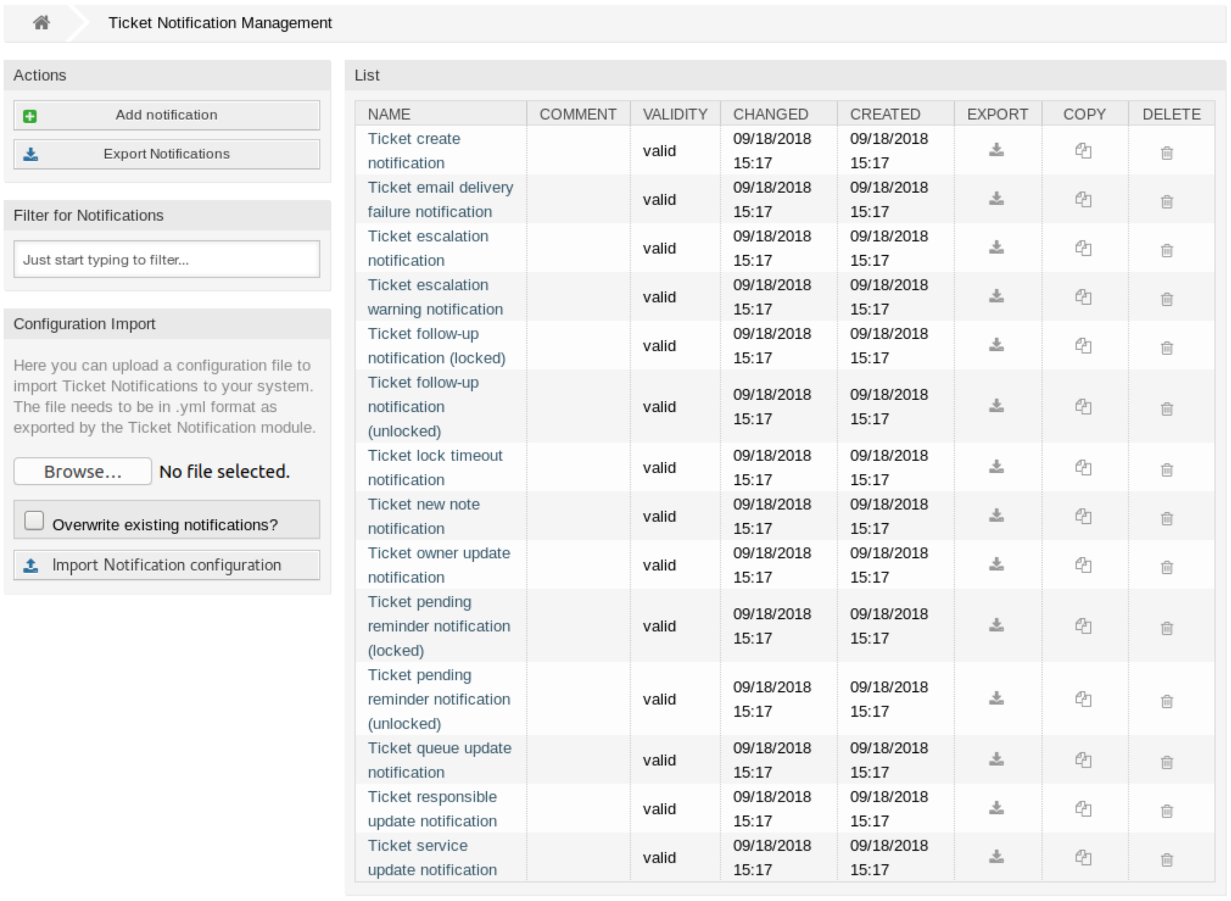
1. Go to the *System Configuration* screen.
2. Navigate to *Core* → *Crypt* → *SMIME* in the navigation tree.
3. Review the settings.

3.11 Ticket Notifications

Streamlining communication can save hours of labor and prevent mistakes. Sending certain messages at pre-defined stages of communication not only keeps the customer and agents informed about specific events, but it can also aid your agents by programmatically doing automated updates to the customer.

The flexible OTRS is an industry leader in email communication and offers you complete control of notifications based on any event in your system.

Use this screen to add ticket notifications to the system. In a fresh OTRS installation several ticket notifications are already added by default. The ticket notification management screen is available in the *Ticket Notifications* module of the *Communication & Notifications* group.



Ticket Notification Management

Actions

Add notification

Export Notifications

Filter for Notifications

Just start typing to filter...

Configuration Import

Here you can upload a configuration file to import Ticket Notifications to your system. The file needs to be in .yml format as exported by the Ticket Notification module.

Browse... No file selected.

Overwrite existing notifications?

Import Notification configuration

List

NAME	COMMENT	VALIDITY	CHANGED	CREATED	EXPORT	COPY	DELETE
Ticket create notification		valid	09/18/2018 15:17	09/18/2018 15:17			
Ticket email delivery failure notification		valid	09/18/2018 15:17	09/18/2018 15:17			
Ticket escalation notification		valid	09/18/2018 15:17	09/18/2018 15:17			
Ticket escalation warning notification		valid	09/18/2018 15:17	09/18/2018 15:17			
Ticket follow-up notification (locked)		valid	09/18/2018 15:17	09/18/2018 15:17			
Ticket follow-up notification (unlocked)		valid	09/18/2018 15:17	09/18/2018 15:17			
Ticket lock timeout notification		valid	09/18/2018 15:17	09/18/2018 15:17			
Ticket new note notification		valid	09/18/2018 15:17	09/18/2018 15:17			
Ticket owner update notification		valid	09/18/2018 15:17	09/18/2018 15:17			
Ticket pending reminder notification (locked)		valid	09/18/2018 15:17	09/18/2018 15:17			
Ticket pending reminder notification (unlocked)		valid	09/18/2018 15:17	09/18/2018 15:17			
Ticket queue update notification		valid	09/18/2018 15:17	09/18/2018 15:17			
Ticket responsible update notification		valid	09/18/2018 15:17	09/18/2018 15:17			
Ticket service update notification		valid	09/18/2018 15:17	09/18/2018 15:17			

Fig. 54: Ticket Notification Management Screen

3.11.1 Manage Ticket Notifications

There are some pre-defined ticket notifications added by default. The list of built-in ticket notifications are the follows:

- Customer ticket article notification
- Customer ticket create notification
- Customer ticket state update (closed)
- Customer ticket state update (other than closed)
- Missing DTT assignment for CustomerGroup
- Ticket create notification
- Ticket email delivery failure notification
- Ticket escalation notification
- Ticket escalation warning notification
- Ticket follow-up notification (locked)
- Ticket follow-up notification (unlocked)
- Ticket lock timeout notification
- Ticket new note notification
- Ticket owner update notification
- Ticket pending reminder notification (locked)
- Ticket pending reminder notification (unlocked)
- Ticket queue update notification
- Ticket responsible update notification
- Ticket service update notification
- Ticket watcher reminder notification
- Watchlist agent article notification
- Watchlist customer article notification
- Watchlist move notification
- Watchlist owner change notification
- Watchlist queue change notification
- Watchlist state change notification

To add a ticket notification:

1. Click on the *Add Notification* button in the left sidebar.
2. Fill in the required fields as explained in *Ticket Notification Settings*.
3. Click on the *Save* button.

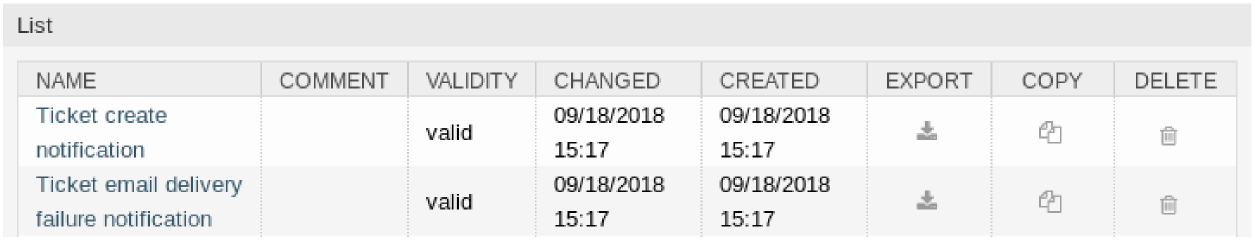
To edit a ticket notification:

1. Click on a ticket notification in the list of ticket notifications.
2. Modify the fields as explained in *Ticket Notification Settings*.

3. Click on the *Save* or *Save and finish* button.

To delete a ticket notification:

1. Click on the trash icon in the list of ticket notifications.
2. Click on the *Confirm* button.



List

NAME	COMMENT	VALIDITY	CHANGED	CREATED	EXPORT	COPY	DELETE
Ticket create notification		valid	09/18/2018 15:17	09/18/2018 15:17			
Ticket email delivery failure notification		valid	09/18/2018 15:17	09/18/2018 15:17			

Fig. 55: Delete Ticket Notification Screen

To export all ticket notifications:

1. Click on the *Export Notifications* button in the left sidebar.
2. Choose a location in your computer to save the `Export_Notification.yml` file.

Warning: Certain settings are exported as numeric IDs and will break when importing to a system where these settings do not appear or reference other named items.

To import ticket notifications:

1. Click on the *Browse*… button in the left sidebar.
2. Select a previously exported `.yml` file.
3. Click on the *Overwrite existing notifications?* checkbox, if you would like to overwrite the existing notifications.
4. Click on the *Import Notification configuration* button.

3.11.2 Sending Articles Created in the External Interface as Emails

To activate sending emails to custom contact field customer users:

1. Click on the *Add Notification* button in the left sidebar.
2. Fill in the required fields.
3. In the *Add Notification* section, choose a name and set it in the *Name* field.
4. In the *Events* section, select *ArticleCreate*.
5. In the *Article Filter* section:
   - Set agent and customer in the *Article Sender Type* field.
   - Set *Visible to customer* in the *Customer visibility* field.
   - Select *Email* in the *Communication channel* field.
   - Switch *Include attachments to notification* to *Yes*.
6. In the *Recipients* section set a dynamic contact field in the *Send to* field.

7. In the *Notification Methods* section:

   - Make sure, that the *Email* method is enabled.

     Note: Joined notifications are only sent by *Email*.

   - Select the checkbox in the *Article visible to customer* field.

8. In the *Notification Text* section:

   - If you want the contacts of the custom contact field to get the title of the article as mail subject add `<OTRS_CUSTOMER_SUBJECT>` into the *Subject* field.
   - If you want to send the article body as mail body add `<OTRS_CUSTOMER_BODY>` into the *Text* field.

9. Click on the *Save* button.

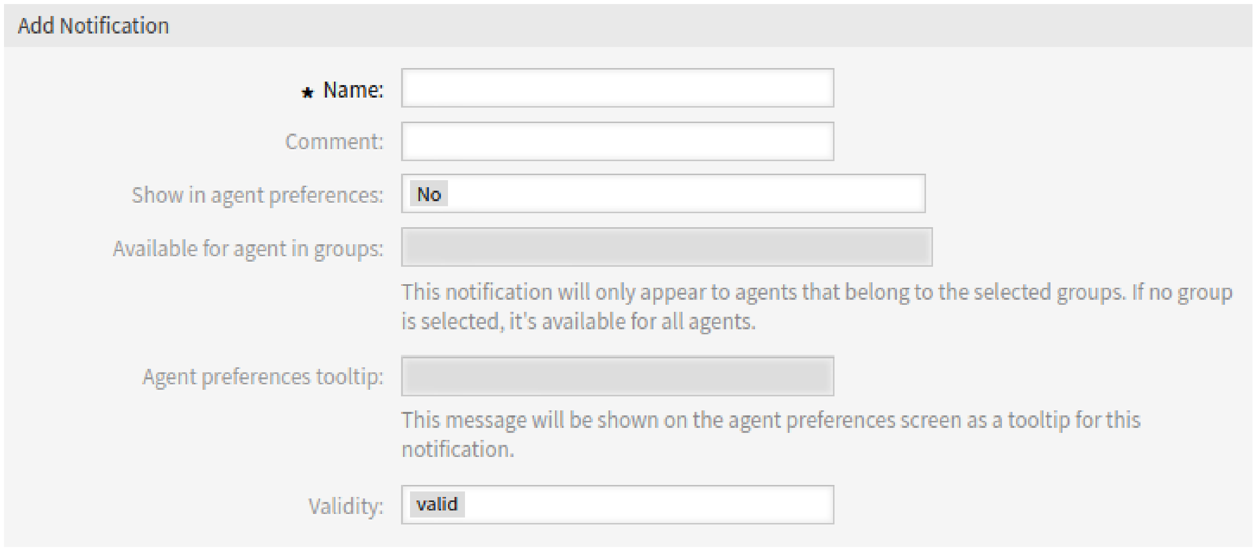
3.11.3 Ticket Notification Settings

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

See also:

For an example, see a default ticket notification which is included in a fresh OTRS installation.

Basic Ticket Notification Settings



Add Notification

★ Name:	
Comment:	
Show in agent preferences:	No
Available for agent in groups:	
	This notification will only appear to agents that belong to the selected groups. If no group is selected, it's available for all agents.
Agent preferences tooltip:	
	This message will be shown on the agent preferences screen as a tooltip for this notification.
Validity:	valid

Fig. 56: Ticket Notification Settings - Basic

Name *
: The name of this resource. Any type of characters can be entered to this field including uppercase letters and spaces. The name will be displayed in the overview table.

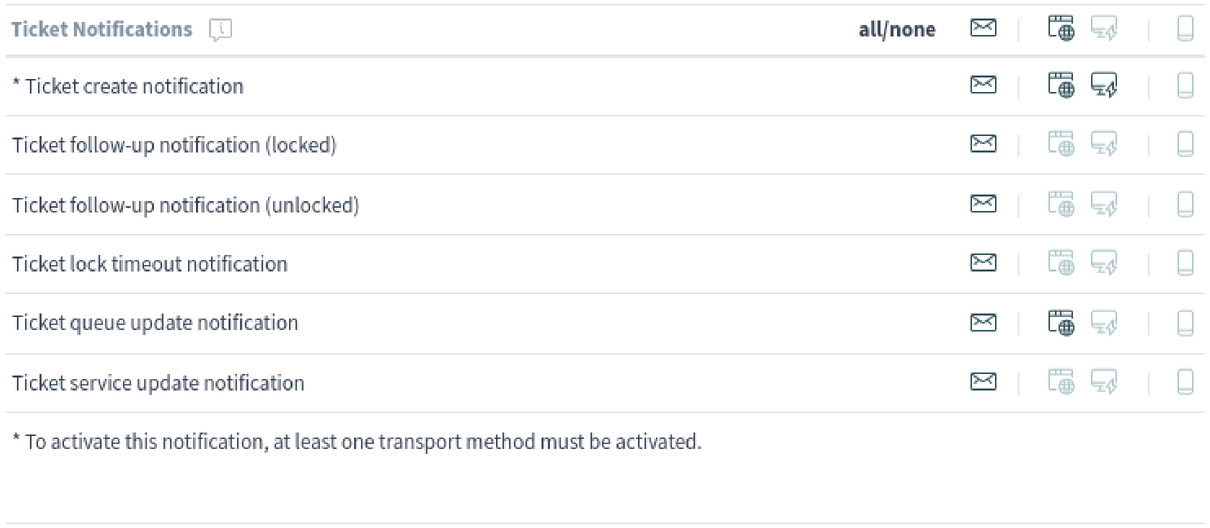
Comment
Add additional information to this resource. It is recommended to always fill this field as a description of the resource with a full sentence for better clarity, because the comment will be also displayed in the overview table.

Show in agent preferences
Define how the notification should be displayed in agent preferences. The following options are available:

No
The notification won't be displayed in agent preferences. The notification is sent to all appropriate agents by the defined method.

Yes
The notification will be displayed in agent preferences for selection. The agents may opt-in or opt-out.

Yes, but require at least one active notification method.
The notification will be displayed in agent preferences, but require at least one active notification method. This is annotated by an asterisk next to the name.



Ticket Notifications	all/none
* Ticket create notification	
Ticket follow-up notification (locked)	
Ticket follow-up notification (unlocked)	
Ticket lock timeout notification	
Ticket queue update notification	
Ticket service update notification	

* To activate this notification, at least one transport method must be activated.

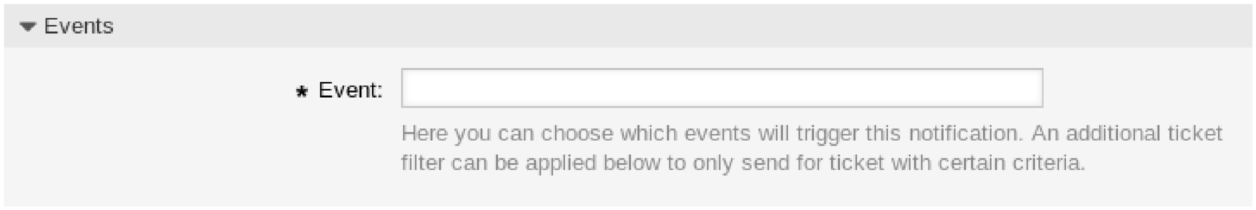
Fig. 57: Personal Ticket Notification Settings

Available for agent in groups
The notification will only appear to agents that belong to the selected groups. If no group is selected, it is available for all agents.

Agent preferences tooltip
This message will be shown on the agent preferences screen as a tooltip for this notification.

Validity *
Set the validity of this resource. Each resource can be used in OTRS only, if this field is set to *valid*. Setting this field to *invalid* or *invalid-temporarily* will disable the use of the resource.

Events


Events

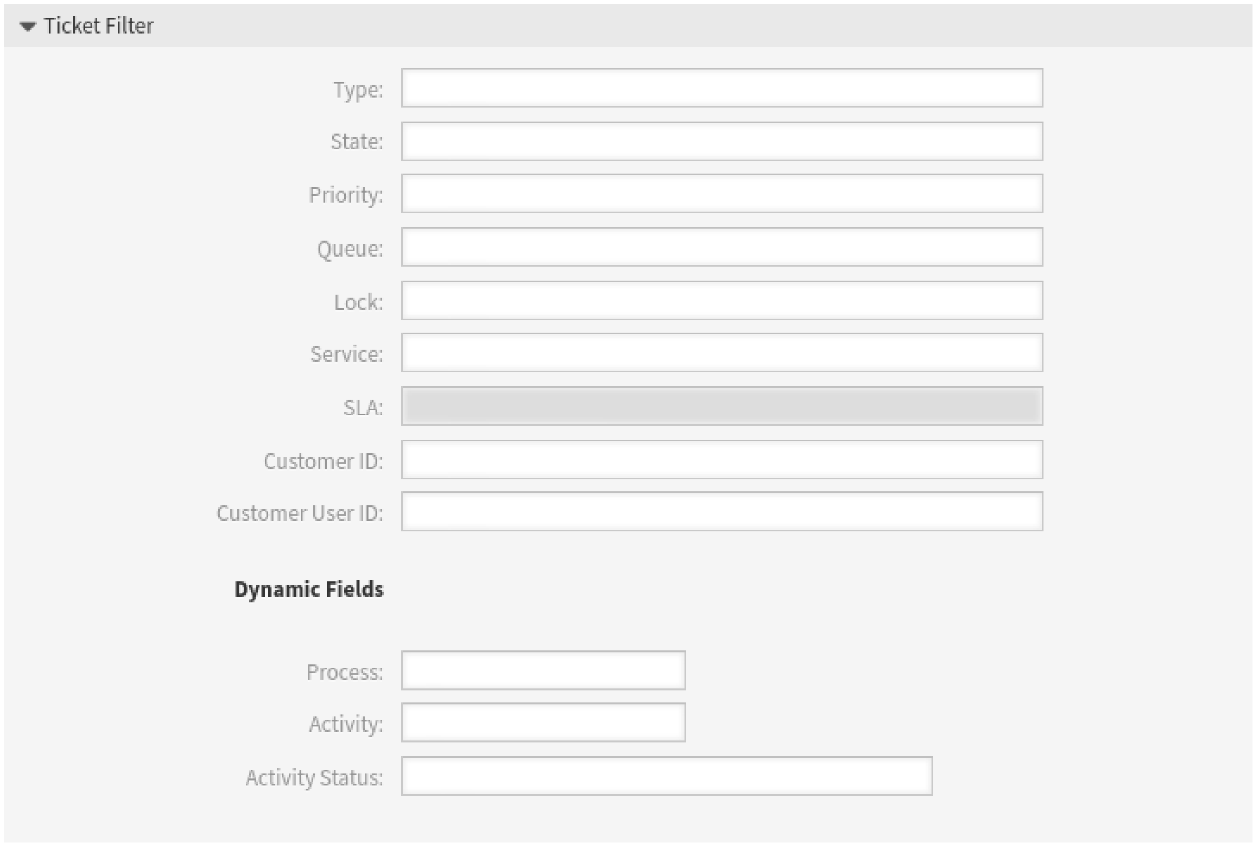
★ Event:

Here you can choose which events will trigger this notification. An additional ticket filter can be applied below to only send for ticket with certain criteria.

Fig. 58: Ticket Notification Settings - Events

Event
Here you can choose which events will trigger this notification. An additional ticket filter can be applied below to only send for tickets with certain criteria.

Ticket Filter[1]


Ticket Filter

Type:

State:

Priority:

Queue:

Lock:

Service:

SLA:

Customer ID:

Customer User ID:

Dynamic Fields

Process:

Activity:

Activity Status:

Fig. 59: Ticket Notification Settings - Ticket Filter

This widget can optionally be used to narrow the list of tickets by matching configured values:

Note: The values in this list can grow as your system grows. The more *Dynamic Fields* are and features

[1] Use of regular expressions as a filter do not work here.

you have in your system, the longer the list will be.

Type
Filter for a type of the ticket.

Note: This field is displayed only if the `Ticket::Type` system configuration setting is enabled and *Types* are added.

State
Filter for a state of the ticket.

Priority
Filter for a priority of the ticket.

Queue
Filter for a queue in which the ticket is located.

Lock
Filter for a lock state of the ticket.

Service
Filter for a service of the ticket.

Note: This field is displayed only if the `Ticket::Service` system configuration setting is enabled and *Services* are added.

SLA
Filter for a SLA of the ticket.

Note: This field is displayed only if the `Ticket::Service` system configuration setting is enabled and *Service Level Agreements* are added.

Customer ID
Filter for a customer ID of the ticket.

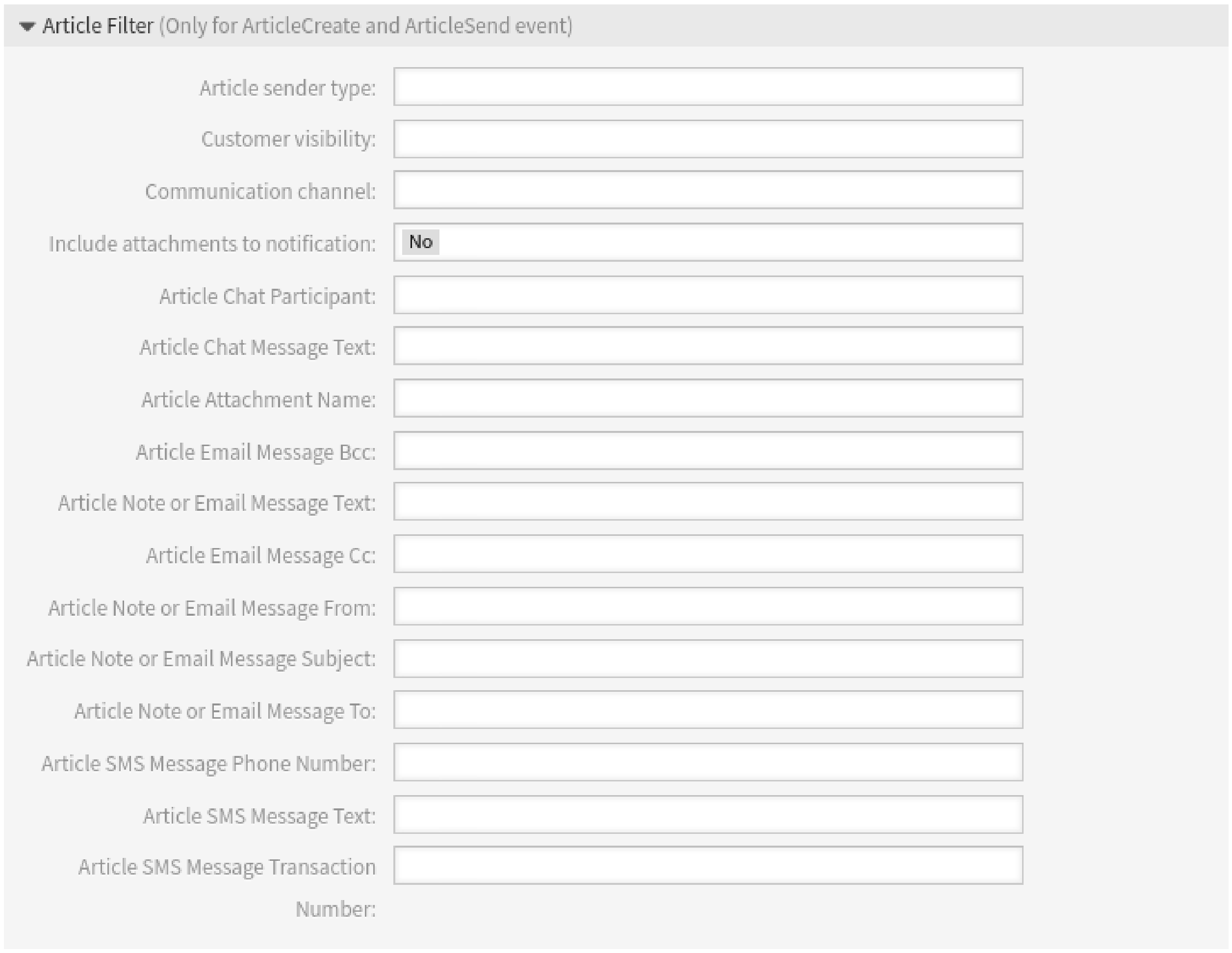
Customer User ID
Filter for a customer user ID of the ticket.

Dynamic Fields
Filter for some dynamic fields added to the system. For the complete list of dynamic fields see the *Dynamic Fields* chapter.

Article Filter[1]

Note: This widget works only if `ArticleCreate` or `ArticleSend` is selected in the *Events* widget.

`ArticleSend` is the default for *Email* communication channel and `ArticleCreate` is the default for all other channels. While `ArticleCreate` can be specified for all communication channels, `ArticleSend` is only allowed for *Email* communication channel and only if all sender or recipient fields are invalid.


Article Filter (Only for ArticleCreate and ArticleSend event)

Article sender type:	
Customer visibility:	
Communication channel:	
Include attachments to notification:	No
Article Chat Participant:	
Article Chat Message Text:	
Article Attachment Name:	
Article Email Message Bcc:	
Article Note or Email Message Text:	
Article Email Message Cc:	
Article Note or Email Message From:	
Article Note or Email Message Subject:	
Article Note or Email Message To:	
Article SMS Message Phone Number:	
Article SMS Message Text:	
Article SMS Message Transaction Number:	

Fig. 60: Ticket Notification Settings - Article Filter

Article sender type
Filter for the sender type of the ticket. Possible values are *agent*, *system* or *customer*.

Customer visibility
Filter for the customer visibility. Possible values are *Invisible to customer* or *Visible to customer*.

Communication channel
Filter for the communication channel. Possible values are *Chat*, *Email*, *OTRS*, *Phone* or *SMS*.

Include attachments to notification
If *Yes* is selected, attachments will be included to notification. Selecting *No* will not use this feature.

Article Chat Participant
Filter for chat participants.

Article Chat Message Text
Filter for chat message text.

Article Attachment Name
Filter for attachment name.

Article Email Message Bcc
Filter for blind carbon copy field.

Article Note or Email Message Text
Filter for body text.

Article Email Message Cc
Filter for carbon copy field.

Article Note or Email Message From
Filter for the sender field.

Article Note or Email Message Subject
Filter for the subject field.

Article Note or Email Message To
Filter for the main recipients field.

Article SMS Message Phone Number
Filter for an SMS phone number.

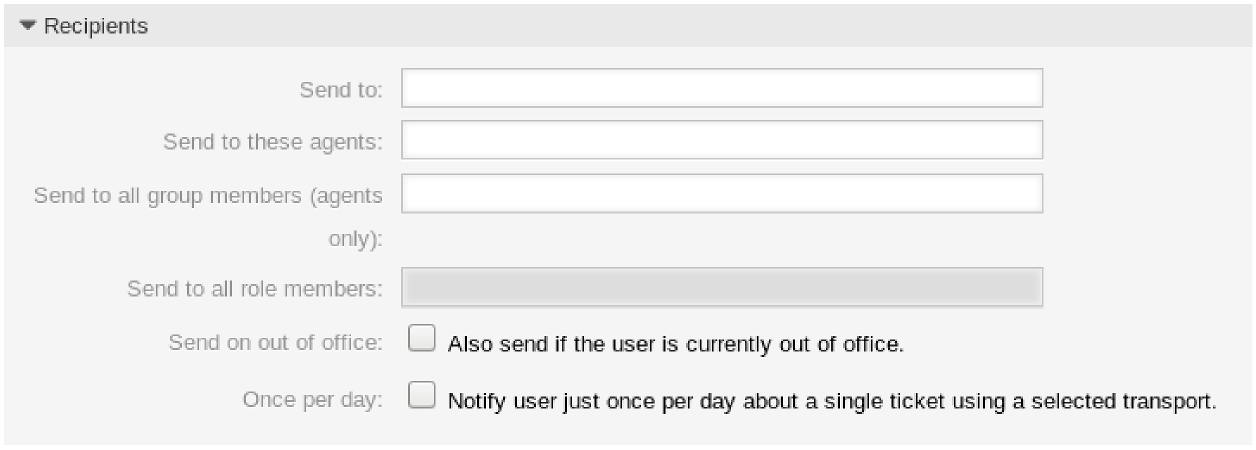
Article SMS Message Text
Filter for the SMS text.

Article SMS Message Transaction Number
Filter for an SMS transaction number.

Ticket Notification Recipients

Send to
Select which agents should receive the notifications. Possible values are:

- Agent who created the ticket
- Agent who is responsible for the ticket
- Agent who owns the ticket
- All agents subscribed to both the ticket’s queue and service
- All agents subscribed to the ticket’s queue


Recipients

Send to:	
Send to these agents:	
Send to all group members (agents only):	
Send to all role members:	
Send on out of office:	☐ Also send if the user is currently out of office.
Once per day:	☐ Notify user just once per day about a single ticket using a selected transport.

Fig. 61: Ticket Notification Settings - Recipients

- All agents subscribed to the ticket’s service
- All agents watching the ticket
- All agents who focus on the owner
- All agents who focus on the priority
- All agents who focus on the queue
- All agents who focus on the responsible
- All agents who focus on the service
- All agents who focus on the SLA
- All agents who focus on the state
- All agents who focus on the type
- All agents who focus on the value
- All agents with write permission for the ticket
- All recipients of the first article
- All recipients of the last article
- Customer user of the ticket

See also:

To be able to use all options in focus topic settings, you need to enable the following system configuration options:

- `AgentPersonalPreference###FocusTopics::Owner`
- `AgentPersonalPreference###FocusTopics::Priority`
- `AgentPersonalPreference###FocusTopics::Queue`
- `AgentPersonalPreference###FocusTopics::Responsible`
- `AgentPersonalPreference###FocusTopics::Service`
- `AgentPersonalPreference###FocusTopics::SLA`
- `AgentPersonalPreference###FocusTopics::State`

- `AgentPersonalPreference###FocusTopics::Type`
- `AgentPersonalPreference###FocusTopics::DynamicFields`
- `Ticket::Responsible`
- `Ticket::Service`
- `Ticket::Type`

Send to these agents
One or more agents can be selected who should receive the notifications.

Send to all group members (agents only)
One or more groups can be selected whom agents should receive the notifications.

Send to all role members
One or more roles can be selected whom agents should receive the notifications.

Send on out of office
If this option is checked, the notification will be sent even if the agent is currently out of office.

Once per day
Notify users just once per day about a single ticket using a selected transport. If this is the first notification about a ticket, then the notification will be sent. If a notification was already sent before and this option is checked, the OTRS daemon will check the time the last notification was sent. If there was no notification sent in the last 24 hours, the notification will be sent again.

Note: The notifications respect the permissions of the agent. Notifications are sent to agents only if the agent has at least read permissions for the ticket in that moment in time when the notification is triggered.

Ticket Notification Methods

Enable this notification method
Enable or disable this notification method. A notification method can be email, web view with browser notification support or SMS.

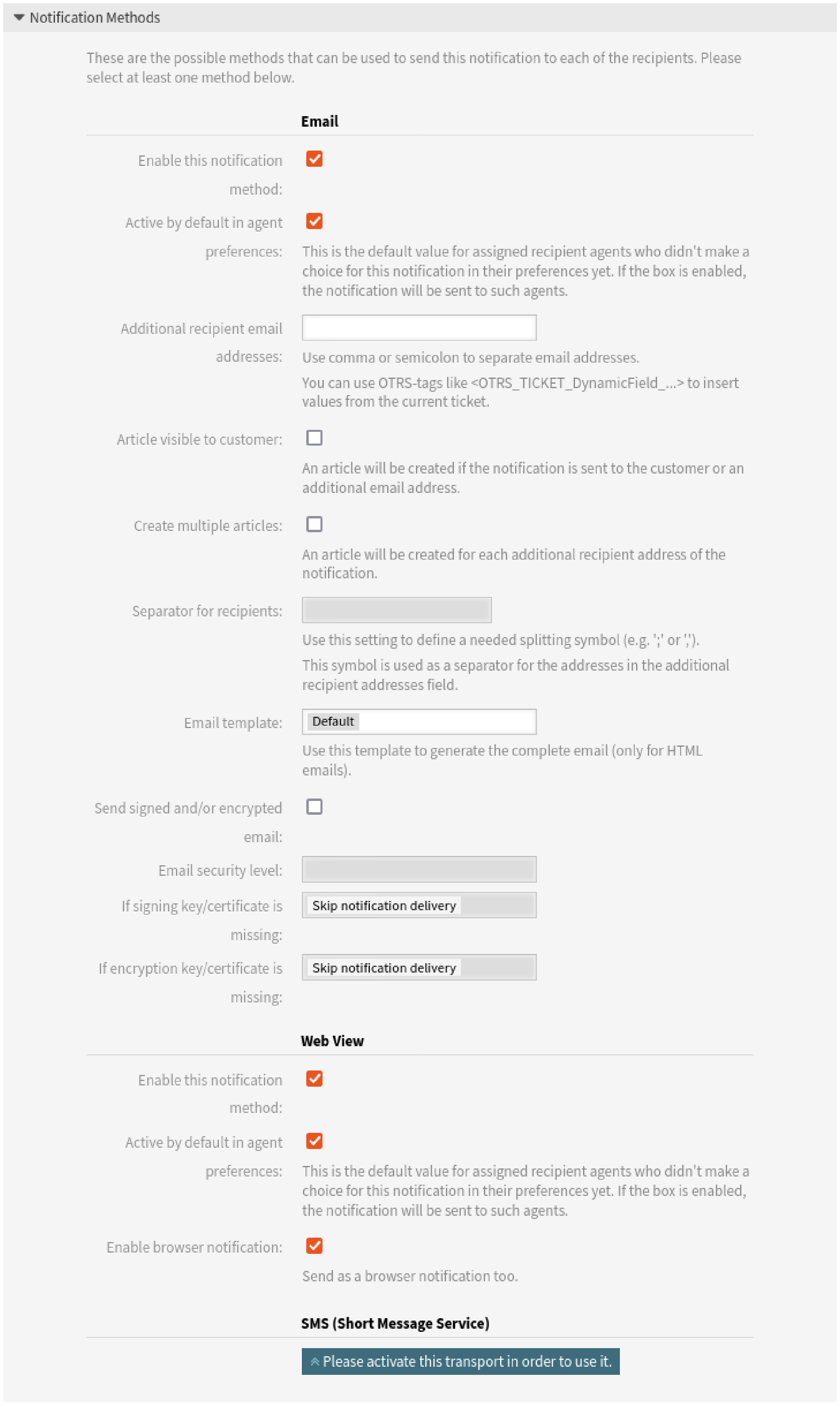
Enable browser notification
This option is available for web view only. If checked, the system sends a browser notification, too. Web view notifications will be displayed in the notifications area of the agent interface while a browser notification is a native browser notification which needs to be enabled in the web browser at first time.

Note: To use the SMS notification method, *Cloud Services* need to be enabled.

Active by default in agent preferences
This is the default value for assigned recipient agents who didn't make a choice for this notification in their preferences yet. If the box is enabled, the notification will be sent to such agents.

Note: This field is displayed only if *Yes* is selected in the *Show in agent preferences* setting above.

Additional recipient email addresses
Additional recipients can be added here. Use comma or semicolon to separate the email addresses.



Notification Methods

These are the possible methods that can be used to send this notification to each of the recipients. Please select at least one method below.

Email

Enable this notification method:	✓
Active by default in agent preferences:	✓ This is the default value for assigned recipient agents who didn't make a choice for this notification in their preferences yet. If the box is enabled, the notification will be sent to such agents.
Additional recipient email addresses:	Use comma or semicolon to separate email addresses. You can use OTRS-tags like <OTRS_TICKET_DynamicField_...> to insert values from the current ticket.
Article visible to customer:	☐ An article will be created if the notification is sent to the customer or an additional email address.
Create multiple articles:	☐ An article will be created for each additional recipient address of the notification.
Separator for recipients:	Use this setting to define a needed splitting symbol (e.g. ';' or ','). This symbol is used as a separator for the addresses in the additional recipient addresses field.
Email template:	Default. Use this template to generate the complete email (only for HTML emails).
Send signed and/or encrypted email:	☐
Email security level:	
If signing key/certificate is missing:	Skip notification delivery
If encryption key/certificate is missing:	Skip notification delivery

Web View

Enable this notification method:	✓
Active by default in agent preferences:	✓ This is the default value for assigned recipient agents who didn't make a choice for this notification in their preferences yet. If the box is enabled, the notification will be sent to such agents.
Enable browser notification:	✓ Send as a browser notification too.

SMS (Short Message Service)

Please activate this transport in order to use it.

Fig. 62: Ticket Notification Settings - Notification Methods

Article visible to customer
An article will be created if the notification is sent to the customer or an additional email address.

Create multiple articles
An article will be created for each additional recipient address of the notification.

Separator for recipients
This symbol is used as a separator for the addresses in the additional recipient addresses field.

Email template
Select which email template should be used for the notification.

Note: Additional email templates can be added by placing a `.tt` file into the folder `<OTRS_Home>/Kernel/Output/HTML/Templates/Standard/NotificationEvent/Email/`. See the existing email templates for an example.

Send signed and/or encrypted email
Checking this option will encrypt the notification email.

Note: To use this feature, *PGP Keys* or *S/MIME Certificates* need to be enabled.

Email security level
If *Enable email security* is checked, then this setting is activated. The following options are available:

PGP sign only
Sign only the notification email with PGP key. If no PGP keys have been added to the system, this option is not visible.

PGP encrypt only
Encrypt only the notification email with PGP key. If no PGP keys have been added to the system, this option is not visible.

PGP sign and encrypt
Sign and encrypt the notification email with PGP key. If no PGP keys have been added to the system, this option is not visible.

S/MIME sign only
Sign only the notification email with S/MIME certificate. If no S/MIME certificates have been added to the system, this option is not visible.

S/MIME encrypt only
Encrypt only the notification email with S/MIME certificate. If no S/MIME certificates have been added to the system, this option is not visible.

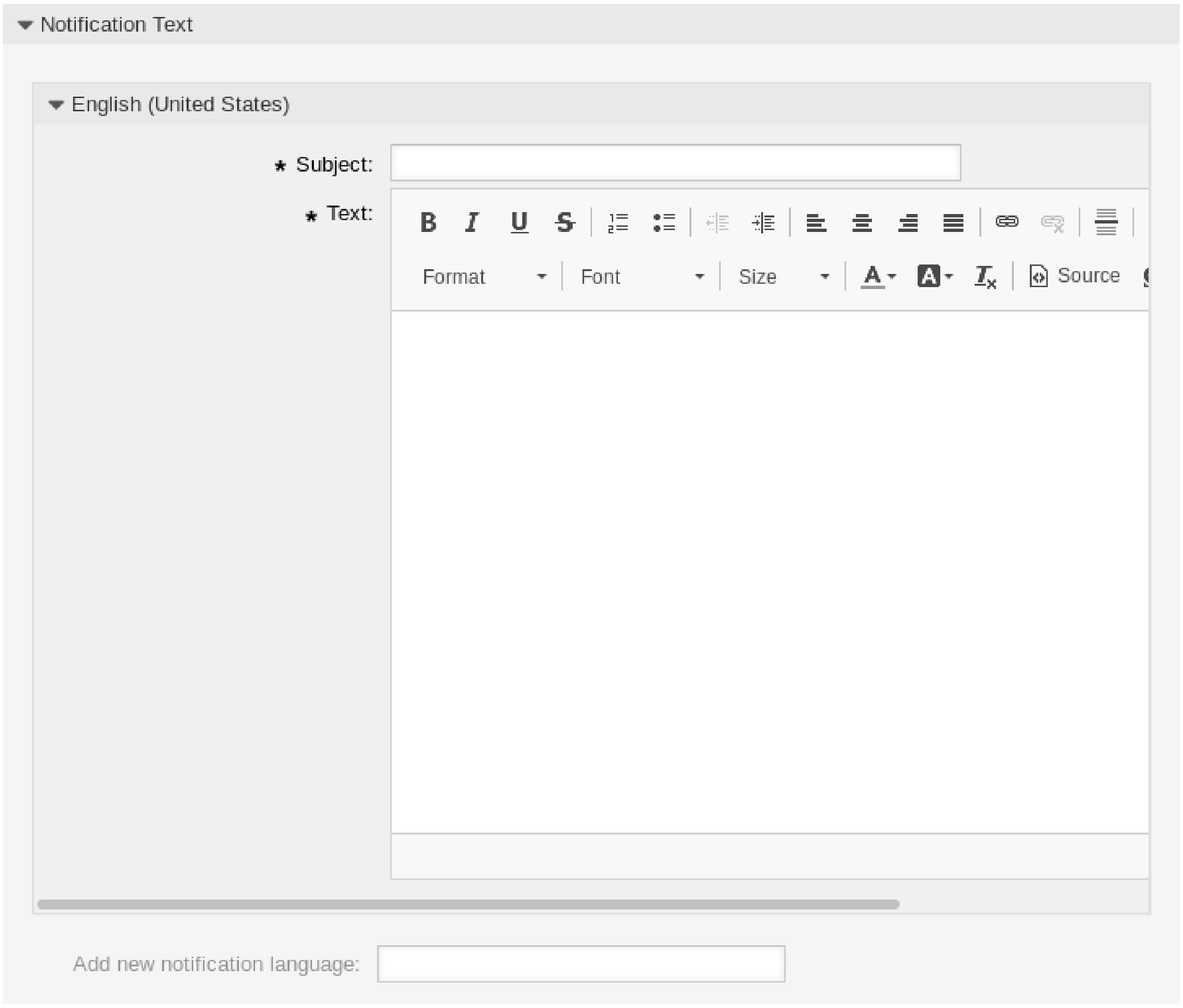
S/MIME sign and encrypt
Sign and encrypt the notification email with S/MIME certificate. If no S/MIME certificates have been added to the system, this option is not visible.

Note: To use this feature, *PGP Keys* or *S/MIME Certificates* need to be enabled.

If signing key/certificate is missing
Select the method, that should be used if signing key or certificate is missing.

If encryption key/certificate is missing:
Select the method, that should be used if encryption key or certificate is missing.

Notification Text



* Subject:

* Text:

Add new notification language:

Fig. 63: Ticket Notification Settings - Notification Text

The main content of a notification can be added for each languages with localized subject and body text. It is also possible to define static text content mixed with OTRS smart tags.

Subject *
: The localized subject for a specific language.

Text *
: The localized body text for a specific language.

Add new notification language
: Select which languages should be added to create localized notifications. The language of the customer or agent will be used as found in the customer and agent preferences. Secondarily, the system default language will be chosen. The fall back will always be English.

Warning: Deleting a language from the DefaultUsedLanguages setting that already has a notification text here will make the notification text unusable. If a language is not present or enabled on the system, the corresponding notification text could be deleted if it is not needed anymore.

3.11.4 Ticket Notification Variables

Using variables in the text makes it possible to personalize messages. Variables, known as OTRS tags, are replaced by OTRS when generating the message. Find a list of available tags stems for this resource at the bottom of both add and edit screens.

For example, the variable `<OTRS_TICKET_TicketNumber>` expands to the ticket number allowing a template to include something like the following.

```
Ticket#<OTRS_TICKET_TicketNumber>
```

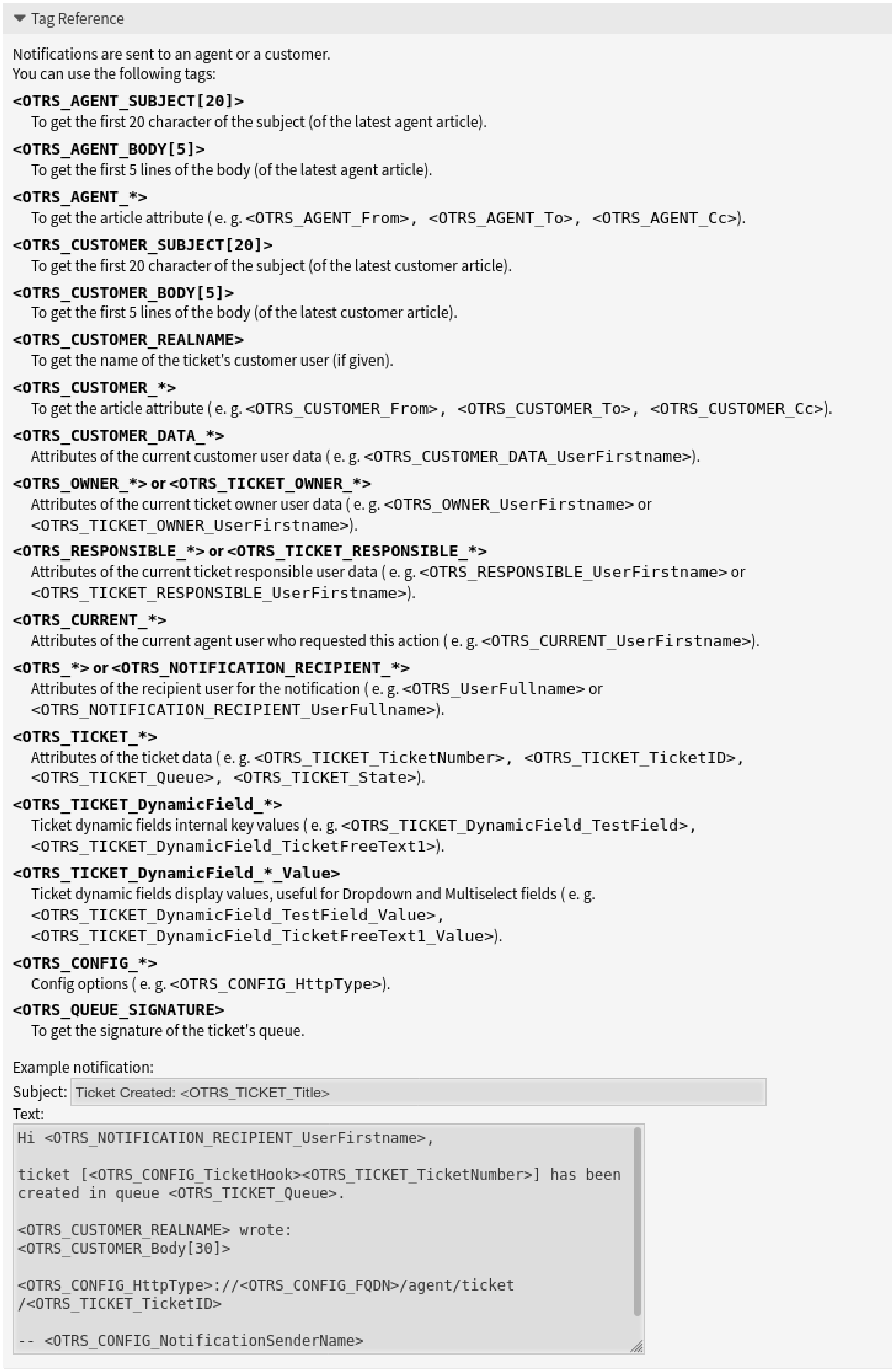
This tag expands, for example to:

```
Ticket#2018101042000012
```

The values of the following variables are translated based on the chosen language of the customer user.

```
<OTRS_TICKET_Type>
<OTRS_TICKET_State>
<OTRS_TICKET_StateType>
<OTRS_TICKET_Lock>
<OTRS_TICKET_Priority>
```

If the language is not supported the default language is applied.



Tag Reference

Notifications are sent to an agent or a customer.
You can use the following tags:

<OTRS_AGENT_SUBJECT[20]>
To get the first 20 character of the subject (of the latest agent article).

<OTRS_AGENT_BODY[5]>
To get the first 5 lines of the body (of the latest agent article).

<OTRS_AGENT_*>
To get the article attribute (e. g. `<OTRS_AGENT_From>, <OTRS_AGENT_To>, <OTRS_AGENT_Cc>`).

<OTRS_CUSTOMER_SUBJECT[20]>
To get the first 20 character of the subject (of the latest customer article).

<OTRS_CUSTOMER_BODY[5]>
To get the first 5 lines of the body (of the latest customer article).

<OTRS_CUSTOMER_REALNAME>
To get the name of the ticket's customer user (if given).

<OTRS_CUSTOMER_*>
To get the article attribute (e. g. `<OTRS_CUSTOMER_From>, <OTRS_CUSTOMER_To>, <OTRS_CUSTOMER_Cc>`).

<OTRS_CUSTOMER_DATA_*>
Attributes of the current customer user data (e. g. `<OTRS_CUSTOMER_DATA_UserFirstname>`).

<OTRS_OWNER_*> or <OTRS_TICKET_OWNER_*>
Attributes of the current ticket owner user data (e. g. `<OTRS_OWNER_UserFirstname>` or `<OTRS_TICKET_OWNER_UserFirstname>`).

<OTRS_RESPONSIBLE_*> or <OTRS_TICKET_RESPONSIBLE_*>
Attributes of the current ticket responsible user data (e. g. `<OTRS_RESPONSIBLE_UserFirstname>` or `<OTRS_TICKET_RESPONSIBLE_UserFirstname>`).

<OTRS_CURRENT_*>
Attributes of the current agent user who requested this action (e. g. `<OTRS_CURRENT_UserFirstname>`).

<OTRS_*> or <OTRS_NOTIFICATION_RECIPIENT_*>
Attributes of the recipient user for the notification (e. g. `<OTRS_UserFullname>` or `<OTRS_NOTIFICATION_RECIPIENT_UserFullname>`).

<OTRS_TICKET_*>
Attributes of the ticket data (e. g. `<OTRS_TICKET_TicketNumber>, <OTRS_TICKET_TicketID>, <OTRS_TICKET_Queue>, <OTRS_TICKET_State>`).

<OTRS_TICKET_DynamicField_*>
Ticket dynamic fields internal key values (e. g. `<OTRS_TICKET_DynamicField_TestField>, <OTRS_TICKET_DynamicField_TicketFreeText1>`).

<OTRS_TICKET_DynamicField_*_Value>
Ticket dynamic fields display values, useful for Dropdown and Multiselect fields (e. g. `<OTRS_TICKET_DynamicField_TestField_Value>, <OTRS_TICKET_DynamicField_TicketFreeText1_Value>`).

<OTRS_CONFIG_*>
Config options (e. g. `<OTRS_CONFIG_HttpType>`).

<OTRS_QUEUE_SIGNATURE>
To get the signature of the ticket's queue.

Example notification:

Subject: Ticket Created: <OTRS_TICKET_Title>

Text:

```
Hi <OTRS_NOTIFICATION_RECIPIENT_UserFirstname>,

ticket [<OTRS_CONFIG_TicketHook><OTRS_TICKET_TicketNumber>] has been
created in queue <OTRS_TICKET_Queue>.

<OTRS_CUSTOMER_REALNAME> wrote:
<OTRS_CUSTOMER_Body[30]>

<OTRS_CONFIG_HttpType>://<OTRS_CONFIG_FQDN>/agent/ticket
/<OTRS_TICKET_TicketID>

-- <OTRS_CONFIG_NotificationSenderName>
```

Fig. 64: Ticket Notification Variables

CHAPTER

FOUR

USERS, GROUPS & ROLES

Simple and complex organizations need a flexible way to control access to their service desk. Access, resources, and permissions must be orchestrated so that, the users and customers have access to the resources they need and their data is protected as needed by the system.

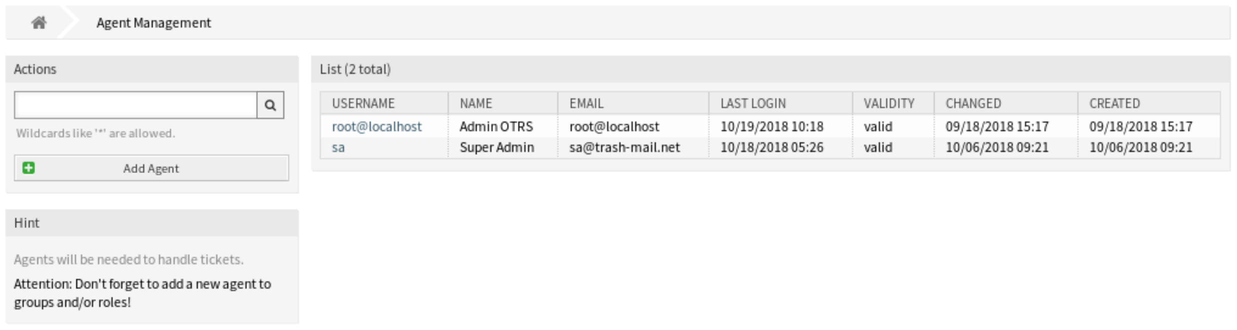
OTRS provides powerful tools for this purpose and their use is described in the following chapter.

4.1 Agents

Agent and access management for your service desk should be easy. Flexibility in adding, editing, invalidating access and a quick overview of which permissions a user has will help you maintain a clean permissions system and record of your setup in OTRS.

OTRS aids you giving you the power to manage agents within OTRS across multiple back ends. OTRS can use up-to ten back end sources, even marking some as read-only. Managing user settings centrally, an administrator can quickly invalidate a compromised account or set an account to out-of-office in case of an unexpected illness.

Use this screen to add agents to the system. A fresh OTRS installation contains an agent with administrator privileges by default. The agent management screen is available in the *Agents* module of the *Users, Groups & Roles* group.



Agent Management

Actions

Wildcards like '*' are allowed.

Add Agent

Hint

Agents will be needed to handle tickets.

Attention: Don't forget to add a new agent to groups and/or roles!

List (2 total)

USERNAME	NAME	EMAIL	LAST LOGIN	VALIDITY	CHANGED	CREATED
root@localhost	Admin OTRS	root@localhost	10/19/2018 10:18	valid	09/18/2018 15:17	09/18/2018 15:17
sa	Super Admin	sa@trash-mail.net	10/18/2018 05:26	valid	10/06/2018 09:21	10/06/2018 09:21

Fig. 1: Agent Management Screen

Warning: The superuser account username is *root@localhost*. Don't use the superuser account to work with OTRS! Create new agents and work with these accounts instead. One of the adverse effects is that *Access Control Lists (ACL)* will not have an effect on this user.

4.1.1 Manage Agents

Note: Adding or editing an agent is possible only by using database back end. Using explicitly external directory services like LDAP and, based on configuration, some databases are read-only. Personal preferences like out-of-office can still be set.

To add an agent:

1. Click on the *Add Agent* button in the left sidebar.
2. Fill in the required fields.
3. Click on the *Save* button.

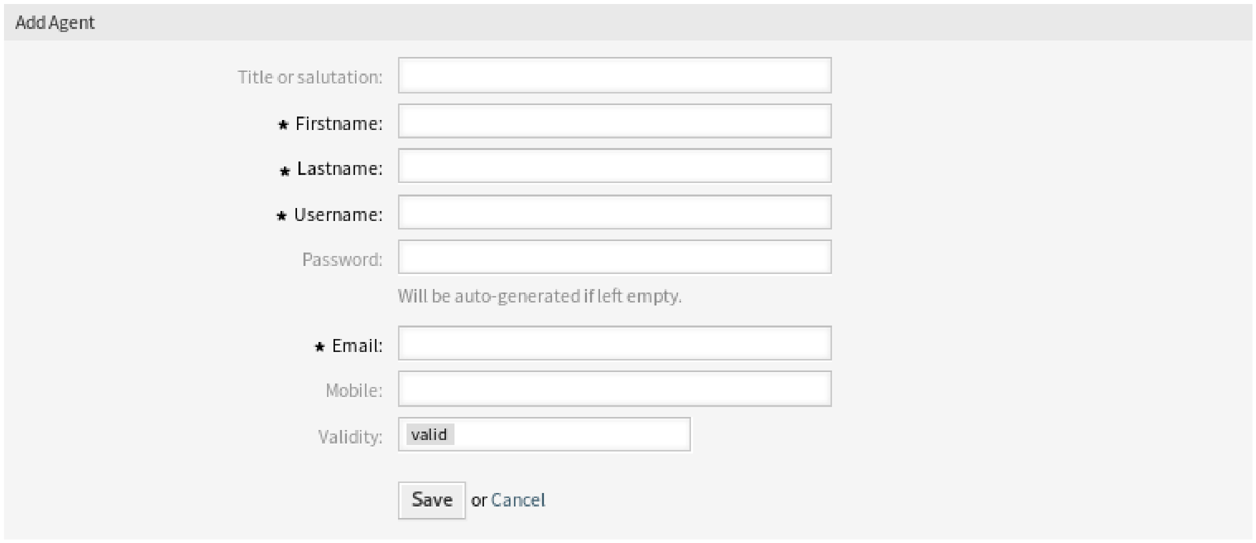


Fig. 2: Add Agent Screen

Warning: Agents can not be deleted from the system. They can only be deactivated by setting the *Validity* option to *invalid* or *invalid-temporarily*.

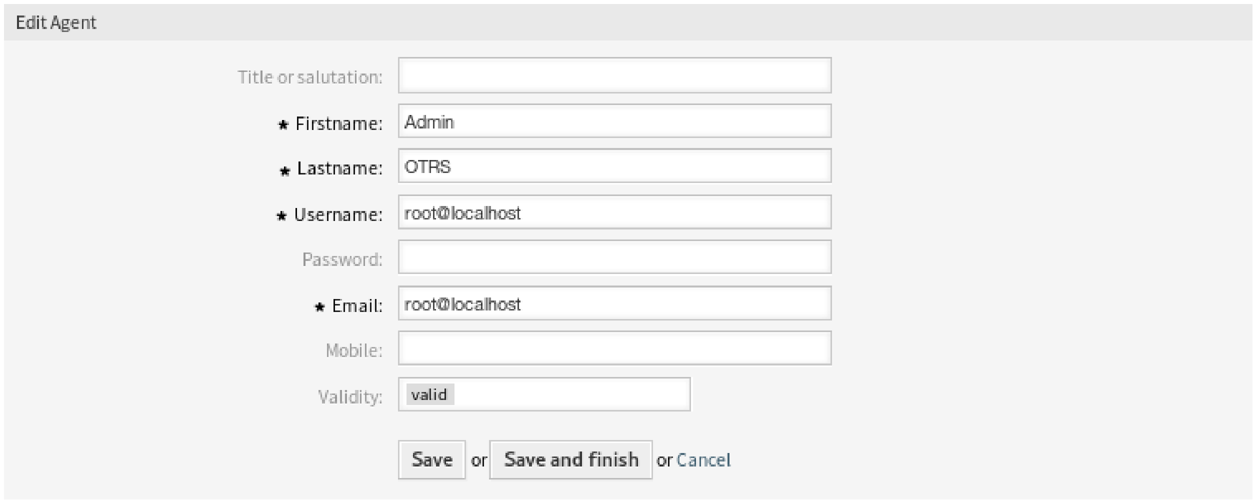
To edit an agent:

1. Click on an agent in the list of agents.
2. Modify the fields.
3. Click on the *Save* or *Save and finish* button.

It is also possible to edit the agent personal preferences. To do this, click on the *Edit personal preferences for this agent* button in the left sidebar of the *Edit Agent* screen.

To find an agent:

1. Enter a search term to the search box in the left sidebar.
2. Click on the magnifying glass icon in the right part of the field or hit an `Enter`.



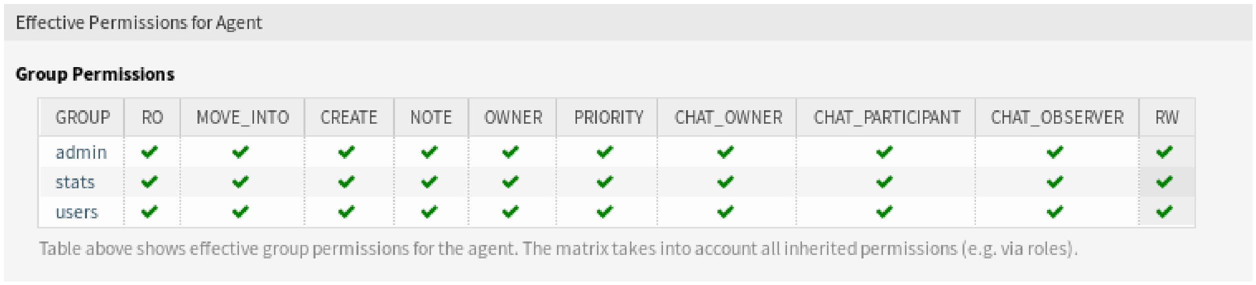
Edit Agent

Title or salutation:	
⋆ Firstname:	Admin
⋆ Lastname:	OTRS
⋆ Username:	root@localhost
Password:	
⋆ Email:	root@localhost
Mobile:	
Validity:	valid

Fig. 3: Edit Agent Screen

Note: If several agents are added to the system, use the search box to find a particular agent. Only the first 1000 agents are listed by default.

The agent permissions can be controlled by adding an agent to *Groups* or *Roles*. This can result a complex matrix of permissions. The effective permissions for an agent can be checked in the bottom of the *Edit Agent* screen. If roles (recommended) are used, this screen will reflect the combined permissions as dictated by the roles.



Effective Permissions for Agent

Group Permissions

GROUP	RO	MOVE_INTO	CREATE	NOTE	OWNER	PRIORITY	CHAT_OWNER	CHAT_PARTICIPANT	CHAT_OBSERVER	RW
admin	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
stats	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
users	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table above shows effective group permissions for the agent. The matrix takes into account all inherited permissions (e.g. via roles).

Fig. 4: Effective Permissions for Agent Widget

4.1.2 Agent Settings

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

Note: These are the default fields available for the internal database table.

Title or salutation
Some name prefix can be added here like *Mr.*, *Dr.*, *Jr.*, etc.

Firstname *
The first name of the agent.

Lastname *
The last name of the agent.

See also:

The agent display name can be set via the system configuration setting `FirstnameLastnameOrder`.

Username *
The username of the agent to login to the system.

Password
The password of the agent. This will be auto-generated if left empty.

Email *
The email address of the agent.

Note: The email syntax and validity of an available MX record could prevent you from submitting this form. For some systems it may be acceptable to turn off these checks.

Mobile
The mobile phone number of the agent.

Validity *
Set the validity of this resource. Each resource can be used in OTRS only, if this field is set to *valid*. Setting this field to *invalid* or *invalid-temporarily* will disable the use of the resource.

4.1.3 Alternate Login for Agents

It is possible to create an alternate user login for agents. This feature is turned off by default.

To activate the feature:

1. Go to the *System Configuration* screen.
2. Search for the setting `AgentPersonalPreference###AnonymousLogin`.
3. Enable the setting and set the `Active` field to `1`.
4. Deploy the modified system configuration.

To set an alternate user for an agent:

1. Go to the *Agents* screen.
2. Click on an agent in the list of agents.
3. Click on the *Edit personal preferences for this agent* button in the left sidebar.
4. Click on the *User Profile* in the *Preferences* widget.
5. Enter the username of the alternate user in the *Alternate User* section.

An agent for which an alternate login exists can now log in with his personal credentials and after logging into the system he will be automatically switched to the alternate login account.

Warning: Use this function with care! Enabling the feature may have undesirable side effects on auditing and compliance, especially on the traceability of actions in the system. If you are not sure, do not use this feature or ask the Customer Solutions Team before using it.

4.1.4 Agent Back Ends

Agents can be read and synchronized from an Active Directory® or LDAP server.

The administrator interface does not support the configuration of external back ends. Administrators need to edit the file `Kernel/Config.pm` by copying and pasting code snippets from `Kernel/Config/Defaults.pm` manually in case of using *On-Premise* system.

If you already have agent back end (e.g. SAP), it is possible to write a module that uses it.

Warning: Do not modify the file `Kernel/Config/Defaults.pm`, it will be overwritten after upgrading the system! Copy and paste the code snippets into `Kernel/Config.pm` instead.

Note: This feature is only available to *On-Premise* customers. If you are a *Managed* customer, this feature is taken care of by the *Customer Solutions Team* in **OTRS**. Please contact us via support@otrs.com or in the OTRS Portal.

Agent Back End - Database

The default user authentication back end for agents is the OTRS database. With this back end, all agent data can be edited via the administrator interface.

```
# This is the auth. module against the otrs db
$Self->{AuthModule} = 'Kernel::System::Auth::DB';

# defines AuthSyncBackend (AuthSyncModule) for AuthModule
# if this key exists and is empty, there won't be a sync.
# example values: AuthSyncBackend, AuthSyncBackend2
#    $Self->{'AuthModule::UseSyncBackend'} = '';

# password crypt type (bcrypt|sha2|sha1|md5|apr1|crypt|plain)
#    $Self->{'AuthModule::DB::CryptType'} = 'sha2';

# If "bcrypt" was selected for CryptType, use cost specified here for bcrypt hashing.
#   Currently max. supported cost value is 31.
# $Self->{'AuthModule::DB::bcryptCost'} = 12;
```

Agent Back End - LDAP

If you have an LDAP directory with all your agent data, you can use the LDAP module to authenticate your agents. Because this module has only read-access to the LDAP back end, it is not possible to edit the agent data via the administrator interface.

```
# This is an example configuration for an LDAP auth. backend.
# (take care that Net::LDAP is installed!)
#    $Self->{AuthModule} = 'Kernel::System::Auth::LDAP';
#    $Self->{'AuthModule::LDAP::Host'} = 'ldap.example.com';
#    $Self->{'AuthModule::LDAP::BaseDN'} = 'dc=example,dc=com';
#    $Self->{'AuthModule::LDAP::UID'} = 'uid';
```

(continues on next page)

(continued from previous page)

```
# Check if the user is allowed to auth in a posixGroup
# (e. g. user needs to be in a group xyz to use otrs)
#    $Self->{'AuthModule::LDAP::GroupDN'} = 'cn=otrsallow,ou=posixGroups,dc=example,
↪dc=com';
#    $Self->{'AuthModule::LDAP::AccessAttr'} = 'memberUid';
# for ldap posixGroups objectclass (just uid)
#    $Self->{'AuthModule::LDAP::UserAttr'} = 'UID';
# for non ldap posixGroups objectclass (with full user dn)
#    $Self->{'AuthModule::LDAP::UserAttr'} = 'DN';

# The following is valid but would only be necessary if the
# anonymous user do NOT have permission to read from the LDAP tree
#    $Self->{'AuthModule::LDAP::SearchUserDN'} = '';
#    $Self->{'AuthModule::LDAP::SearchUserPw'} = '';

# in case you want to add always one filter to each ldap query, use
# this option. e. g. AlwaysFilter => '(mail=*)' or AlwaysFilter => '(objectclass=user)
↪'
# or if you want to filter with a locigal OR-Expression, like AlwaysFilter =>
↪'(|(mail=*abc.com)(mail=*xyz.com))'
#    $Self->{'AuthModule::LDAP::AlwaysFilter'} = '';

# in case you want to add a suffix to each login name, then
# you can use this option. e. g. user just want to use user but
# in your ldap directory exists user@domain.
#    $Self->{'AuthModule::LDAP::UserSuffix'} = '@domain.com';

# In case you want to convert all given usernames to lower letters you
# should activate this option. It might be helpful if databases are
# in use that do not distinguish selects for upper and lower case letters
# (Oracle, postgresql). User might be synched twice, if this option
# is not in use.
#    $Self->{'AuthModule::LDAP::UserLowerCase'} = 0;

# In case you need to use OTRS in iso-charset, you can define this
# by using this option (converts utf-8 data from LDAP to iso).
#    $Self->{'AuthModule::LDAP::Charset'} = 'iso-8859-1';

# Net::LDAP new params (if needed - for more info see perldoc Net::LDAP)
#    $Self->{'AuthModule::LDAP::Params'} = {
#        port    => 389,
#        timeout => 120,
#        async   => 0,
#        version => 3,
#    };

# Die if backend can't work, e. g. can't connect to server.
#    $Self->{'AuthModule::LDAP::Die'} = 1;
```

The minimum required to connect to a directory server is:

```
$Self->{AuthModule} = 'Kernel::System::Auth::LDAP';
$Self->{'AuthModule::LDAP::Host'} = 'ldap.example.com';
$Self->{'AuthModule::LDAP::BaseDN'} = 'dc=example,dc=com';
$Self->{'AuthModule::LDAP::UID'} = 'uid';
```

Host

The DNS name or IP of your directory server.

BaseDN
The starting point in your directory tree.

UID
The attribute used for login and identification.

Note: This is `sAMAccountName` for an Active Directory.

It is possible to connect to an LDAP via secure connection. In this case the `ldaps://` protocol has to be added to the host parameter and the port has to be changed.

```
$Self->{'AuthModule::LDAP::Host'} = 'ldaps://secure.example.com';
$Self->{'AuthModule::LDAP::Params'}->{port} = 636;
```

To use multiple back ends, add an additional section of the example code to the `Config.pm`. Please make sure to add a numeric value [1-9] to all settings to indicate which settings belong to which back end.

```
### Backend One
$Self->{AuthModule} = 'Kernel::System::Auth::LDAP';
$Self->{'AuthModule::LDAP::Host'} = 'ldap.example.com';
$Self->{'AuthModule::LDAP::BaseDN'} = 'dc=example,dc=com';
$Self->{'AuthModule::LDAP::UID'} = 'uid';

### Backend Two
$Self->{AuthModule1} = 'Kernel::System::Auth::LDAP';
$Self->{'AuthModule::LDAP::Host1'} = 'ldap.example.com';
$Self->{'AuthModule::LDAP::BaseDN1'} = 'dc=example,dc=com';
$Self->{'AuthModule::LDAP::UID1'} = 'uid';
```

Warning: All back ends will are used in succession. The UID must be unique to all back ends, otherwise some side effects may occur.

If an LDAP server is not available another one should be used as fallback. Since there is only one host setting in each back end configuration, the servers have to be added in an array.

```
$Self->{'AuthModule::LDAP::Host'} = ['ldaps://ldapserver_one.com', 'ldaps://
↪ldapserver_two.com'];
$Self->{'AuthModule::LDAP::Die'} = 0;
```

Please note that there is no real fallback functionality but this workaround should work in most cases. The system will always try the first LDAP server first and after a timeout the second one. The recommended way is to configure a fallback or redundant LDAP at server side.

To synchronize with a specific directory server, you must add the appropriate setting to your agent authentication back end. To achieve this copy the following block from the `Defaults.pm` and paste it into the `Config.pm`.

```
$Self->{'AuthModule::UseSyncBackend'} = 'AuthSyncBackend';
```

To use multiple back ends, add an additional section of the example code to the `Config.pm`. Please make sure to add a numeric value [1-9] to all settings to indicate which settings belong to which back end.

```
$Self->{'AuthModule::UseSyncBackend1'} = 'AuthSyncBackend1';
```

Reuse of an agent synchronization back end is also possible.

```
$Self->{'AuthModule::UseSyncBackend1'} = 'AuthSyncBackend';
```

It is advisable to synchronize the agent data so that agents need not be manually added to the users table prior to authorization. Additionally, groups and roles can be added automatically using security objects of the directory server.

Note: Multiple agent synchronization back end blocks can be used. Please make sure to add a numeric value [1-9] to all settings to indicate which settings belong to which back end. Each `AuthSyncModule` must be explicitly used in an agent authentication back end.

Syncing user data upon login. To achieve this copy the following block from the `Defaults.pm` and paste it into the `Config.pm`.

```
# This is an example configuration for an LDAP auth sync. backend.
# (take care that Net::LDAP is installed!)
$Self->{AuthSyncModule} = 'Kernel::System::Auth::Sync::LDAP';
$Self->{'AuthSyncModule::LDAP::Host'} = 'ldap.example.com';
$Self->{'AuthSyncModule::LDAP::BaseDN'} = 'dc=example,dc=com';
$Self->{'AuthSyncModule::LDAP::UID'} = 'uid';

# The following is valid but would only be necessary if the
# anonymous user do NOT have permission to read from the LDAP tree
$Self->{'AuthSyncModule::LDAP::SearchUserDN'} = '';
$Self->{'AuthSyncModule::LDAP::SearchUserPw'} = '';

# in case you want to add always one filter to each ldap query, use
# this option. e. g. AlwaysFilter => '(mail=*)' or AlwaysFilter => '(objectclass=user)
↪'
# or if you want to filter with a logical OR-Expression, like AlwaysFilter =>
↪'(|(mail=*abc.com)(mail=*xyz.com))'
$Self->{'AuthSyncModule::LDAP::AlwaysFilter'} = '';

# AuthSyncModule::LDAP::UserSyncMap
# (map if agent should create/synced from LDAP to DB after successful login)
# you may specify LDAP-Fields as either
#   * list, which will check each field. first existing will be picked ( ["givenName",
↪"cn","_empty"] )
#   * name of an LDAP-Field (may return empty strings) ("givenName")
#   * fixed strings, prefixed with an underscore: "_test", which will always return␣
↪this fixed string
$Self->{'AuthSyncModule::LDAP::UserSyncMap'} = {
    # DB -> LDAP
    UserFirstname => 'givenName',
    UserLastname  => 'sn',
    UserEmail     => 'mail',
};

# In case you need to use OTRS in iso-charset, you can define this
# by using this option (converts utf-8 data from LDAP to iso).
$Self->{'AuthSyncModule::LDAP::Charset'} = 'iso-8859-1';
```

(continues on next page)

(continued from previous page)

```
# Net::LDAP new params (if needed - for more info see perldoc Net::LDAP)
$Self->{'AuthSyncModule::LDAP::Params'} = {
    port    => 389,
    timeout => 120,
    async   => 0,
    version => 3,
};

# Die if backend can't work, e. g. can't connect to server.
$Self->{'AuthSyncModule::LDAP::Die'} = 1;

# Attributes needed for group syncs
# (attribute name for group value key)
$Self->{'AuthSyncModule::LDAP::AccessAttr'} = 'memberUid';
# (attribute for type of group content UID/DN for full ldap name)
$Self->{'AuthSyncModule::LDAP::UserAttr'} = 'UID';
$Self->{'AuthSyncModule::LDAP::UserAttr'} = 'DN';

# AuthSyncModule::LDAP::UserSyncInitialGroups
# (sync following group with rw permission after initial create of first agent
# login)
$Self->{'AuthSyncModule::LDAP::UserSyncInitialGroups'} = [
    'users',
];

# AuthSyncModule::LDAP::UserSyncGroupsDefinition
# (If "LDAP" was selected for AuthModule and you want to sync LDAP
# groups to otrs groups, define the following.)
$Self->{'AuthSyncModule::LDAP::UserSyncGroupsDefinition'} = {
    # ldap group
    'cn=agent,o=otrs' => {
        # otrs group
        'admin' => {
            # permission
            rw => 1,
            ro => 1,
        },
        'faq' => {
            rw => 0,
            ro => 1,
        },
    },
    'cn=agent2,o=otrs' => {
        'users' => {
            rw => 1,
            ro => 1,
        },
    }
};

# AuthSyncModule::LDAP::UserSyncRolesDefinition
# (If "LDAP" was selected for AuthModule and you want to sync LDAP
# groups to otrs roles, define the following.)
$Self->{'AuthSyncModule::LDAP::UserSyncRolesDefinition'} = {
    # ldap group
    'cn=agent,o=otrs' => {
```

(continues on next page)

(continued from previous page)

```
        # otrs role
        'role1' => 1,
        'role2' => 0,
    },
    'cn=agent2,o=otrs' => {
        'role3' => 1,
    }
};

# AuthSyncModule::LDAP::UserSyncAttributeGroupsDefinition
# (If "LDAP" was selected for AuthModule and you want to sync LDAP
# attributes to otrs groups, define the following.)
$Self->{'AuthSyncModule::LDAP::UserSyncAttributeGroupsDefinition'} = {
    # ldap attribute
    'LDAPAttribute' => {
        # ldap attribute value
        'LDAPAttributeValue1' => {
            # otrs group
            'admin' => {
                # permission
                rw => 1,
                ro => 1,
            },
            'faq' => {
                rw => 0,
                ro => 1,
            },
        },
    },
    'LDAPAttribute2' => {
        'LDAPAttributeValue' => {
            'users' => {
                rw => 1,
                ro => 1,
            },
        },
     }
};

# AuthSyncModule::LDAP::UserSyncAttributeRolesDefinition
# (If "LDAP" was selected for AuthModule and you want to sync LDAP
# attributes to otrs roles, define the following.)
$Self->{'AuthSyncModule::LDAP::UserSyncAttributeRolesDefinition'} = {
    # ldap attribute
    'LDAPAttribute' => {
        # ldap attribute value
        'LDAPAttributeValue1' => {
            # otrs role
            'role1' => 1,
            'role2' => 1,
        },
    },
    'LDAPAttribute2' => {
        'LDAPAttributeValue1' => {
            'role3' => 1,
        },
```

(continues on next page)

(continued from previous page)

```
    },
};
```

The minimum required to connect to a directory server is:

```
$Self->{AuthSyncModule} = 'Kernel::System::Auth::Sync::LDAP';
$Self->{'AuthSyncModule::LDAP::Host'} = 'ldap.example.com';
$Self->{'AuthSyncModule::LDAP::BaseDN'} = 'dc=example,dc=com';
$Self->{'AuthSyncModule::LDAP::UID'} = 'uid';
```

Host
The DNS name or IP of your directory server.

BaseDN
The starting point in your directory tree.

UID
The attribute used for login and identification.

Note: This is `sAMAccountName` for an Active Directory.

Agent Back End - HTTPBasicAuth

If you want to implement a single sign on solution for all your agents, you can use HTTPBasic authentication (for all your systems) and use the HTTPBasicAuth module with OTRS. No login is needed with OTRS any more.

```
# This is an example configuration for an apache ($ENV{REMOTE_USER})
# auth. backend. Use it if you want to have a singe login through
# apache http-basic-auth.
#    $Self->{AuthModule} = 'Kernel::System::Auth::HTTPBasicAuth';
# In case there is a leading domain in the REMOTE_USER, you can
# replace it by the next config option.
#    $Self->{'AuthModule::HTTPBasicAuth::Replace'} = 'example_domain\\';
# In case you need to replace some part of the REMOTE_USER, you can
# use the following RegExp ($1 will be new login).
#    $Self->{'AuthModule::HTTPBasicAuth::ReplaceRegExp'} = '^(.+?)@.+?$';
# Defines a header name, that has to be present for agents to authenticate.
#    $Self->{'AuthModule::HTTPBasicAuth::RequiredLoginHeader'} = 'RequiredHeader';
# Defines a header value, that has to be present in the required header for agents to␣
↪authenticate.
#    $Self->{'AuthModule::HTTPBasicAuth::RequiredLoginHeaderValue'} =
↪'RequiredHeaderValue';
# Defines a header value regular expression, that has to be present in the required␣
↪header for agents to authenticate.
#    $Self->{'AuthModule::HTTPBasicAuth::RequiredLoginHeaderValueRegExp'} = '^
↪RequiredHeaderRegExp$';
# Note:
# If you use this module, you should use as fallback the following
# config settings if user isn't login through apache ($ENV{REMOTE_USER}).
#    $Self->{LoginURL} = 'http://host.example.com/not-authorised-for-otrs.html';
#    $Self->{LogoutURL} = 'http://host.example.com/thanks-for-using-otrs.html';
```

The configuration parameters shown in the example below can be used to synchronize the user data from the HTTP headers into the local OTRS database. To achieve this copy the following block from the `Defaults.pm` and paste it into the `Config.pm`.

```
$Self->{AuthSyncModule} = 'Kernel::System::Auth::Sync::HTTPHeader';

$Self->{'AuthSyncModule::HTTPHeader::UserSyncMap'} = {
    # DB -> Header
    UserFirstname => 'givenName',
    UserLastname  => 'lastName',
    UserEmail     => 'mail',
};

# AuthSyncModule::HTTPHeader::UserSyncInitialGroups
# Sync the following group(s) with rw permission after initial agent login.
$Self->{'AuthSyncModule::HTTPHeader::UserSyncInitialGroups'} = [
    'users',
];

# AuthSyncModule::HTTPHeader::UserSyncGroupsDefinition
# Sync groups based on headers (no value check, just existence of header is checked).
$Self->{'AuthSyncModule::HTTPHeader::UserSyncGroupsDefinition'} = {
    # header name
    'IsAgent' => {
        # otrs group
        'admin' => {
            # permission
            rw => 1,
            ro => 1,
        },
        'stats' => {
            rw => 0,
            ro => 1,
        },
    },
};

# AuthSyncModule::HTTPHeader::UserSyncAttributeGroupsDefinition
# Sync groups based on header values.
# Multiple values per header are possible if separated by ',' or ';' (e.g. 'OTRS_
↪Groups: admin, faq, users').
$Self->{'AuthSyncModule::HTTPHeader::UserSyncAttributeGroupsDefinition'} = {
    # header name
    'HTTPHeaderAttribute1' => {
        # header attribute
        'HTTPHeaderAttribute1Value1' => {
            # otrs group
            'admin' => {
                # permission
                rw => 1,
                ro => 1,
            },
            'stats' => {
                rw => 1,
                ro => 1,
            },
        },
    },
```

(continues on next page)

(continued from previous page)

```
    'HTTPHeaderAttribute2' => {
        'HTTPHeaderAttribute1Value2' => {
            'users' => {
                rw => 1,
                ro => 1,
            },
        },
    },
};

# AuthSyncModule::HTTPHeader::UserSyncRolesDefinition
# Sync roles based on headers (no value check, just existence of header is checked).
$Self->{'AuthSyncModule::HTTPHeader::UserSyncRolesDefinition'} = {
    # header name
    'IsAgent' => {
        # otrs role
        'role1' => 1,
        'role2' => 0,
    },
};

# AuthSyncModule::HTTPHeader::UserSyncAttributeRolesDefinition
# Sync roles based on header values.
# Multiple values per header are possible if separated by ',' or ';' (e.g. 'OTRS_
↪Roles: 1st_level, 2nd_level, admin').
$Self->{'AuthSyncModule::HTTPHeader::UserSyncAttributeRolesDefinition'} = {
    # header name
    'HTTPHeaderAttribute1' => {
        # header value
        'HTTPHeaderAttribute1Value1' => {
            # otrs role
            'role1' => 1,
            'role2' => 1,
        },
    },
    'HTTPHeaderAttribute2' => {
        'HTTPHeaderAttribute1Value2' => {
            'role3' => 1,
        },
    },
};
```

Note: The synchronization relies on the existence of the specified HTTP headers. The configuration of those headers is outside the scope of this documentation.

Agent Back End - Radius

The settings shown in example below can be used to authenticate your agents against a Radius server.

```
# This is example configuration to auth. agents against a radius server.
#    $Self->{'AuthModule'} = 'Kernel::System::Auth::Radius';
#    $Self->{'AuthModule::Radius::Host'} = 'radiushost';
#    $Self->{'AuthModule::Radius::Password'} = 'radiussecret';

# Die if backend can't work, e. g. can't connect to server.
#    $Self->{'AuthModule::Radius::Die'} = 1;
```

4.2 Agents   Calendar Teams

Once you have set up a team in *Calendar Teams* screen, you must define which agents belong to which teams. To use this function, at least one agent and one team need to have been added to the system. The management screen is available in the *Agents   Calendar Teams* module of the *Users, Groups & Roles* group. Additionally, it is also accessible from the *Calendar Teams* screen.

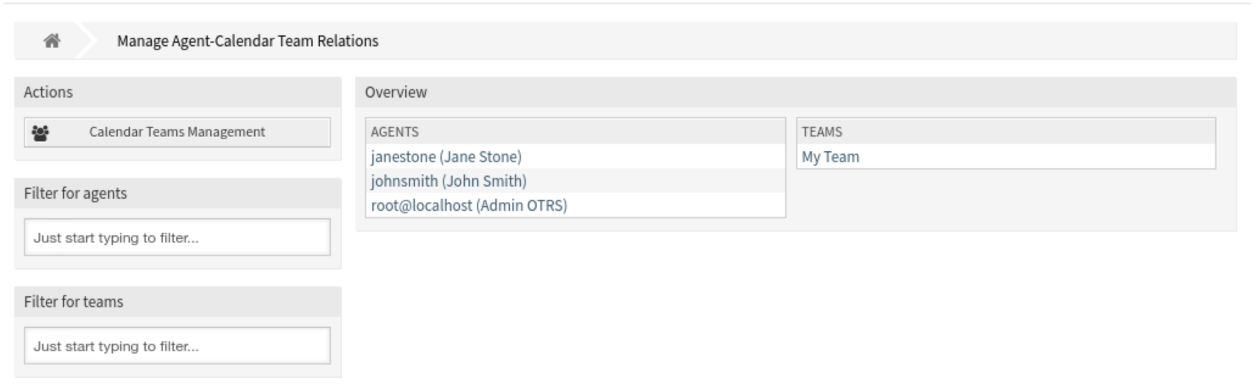

Manage Agent-Calendar Team Relations
Actions
Calendar Teams Management
Filter for agents
Just start typing to filter...
Filter for teams
Just start typing to filter...
Overview
AGENTS
janestone (Jane Stone)
johnsmith (John Smith)
root@localhost (Admin OTRS)
TEAMS
My Team

Fig. 5: Manage Agent-Calendar Team Relations Screen

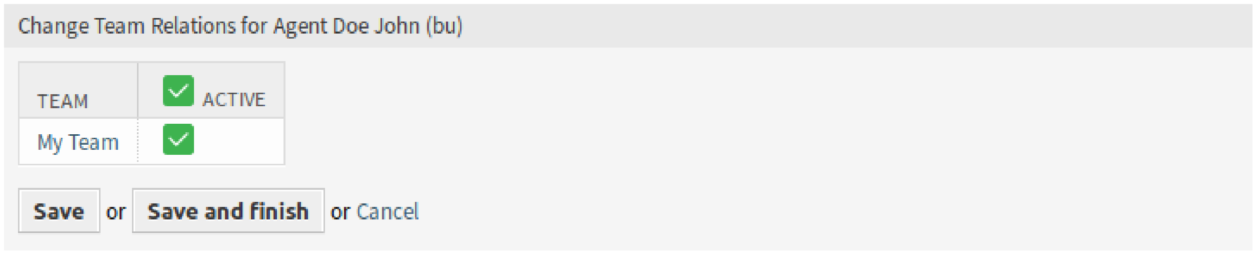
4.2.1 Manage Agents   Calendar Teams Relations

To assign some teams to an agent:

1. Click on an agent in the *Agents* column.
2. Select the teams you would like to assign the agent to.
3. Click on the *Save* or *Save and finish* button.

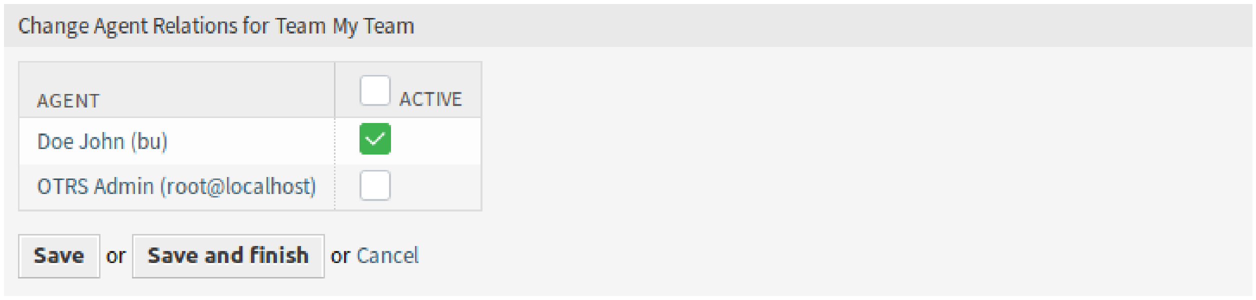
To assign some agents to a team:

1. Click on a team in the *Teams* column.
2. Select the agents you would like to assign to the team.
3. Click on the *Save* or *Save and finish* button.


Change Team Relations for Agent Doe John (bu)

TEAM	✓ ACTIVE
My Team	✓

Fig. 6: Change Team Relations for Agent


Change Agent Relations for Team My Team

AGENT	☐ ACTIVE
Doe John (bu)	✓
OTRS Admin (root@localhost)	☐

Fig. 7: Change Agent Relations for Team

Note: If several agents or teams are added to the system, use the filter box to find a particular agent or team by just typing the name to filter.

Multiple agents or teams can be assigned in both screens at the same time. Additionally clicking on an agent or clicking on a team in the relations screen will open the *Edit Agent* screen or the *Edit Team* screen accordingly.

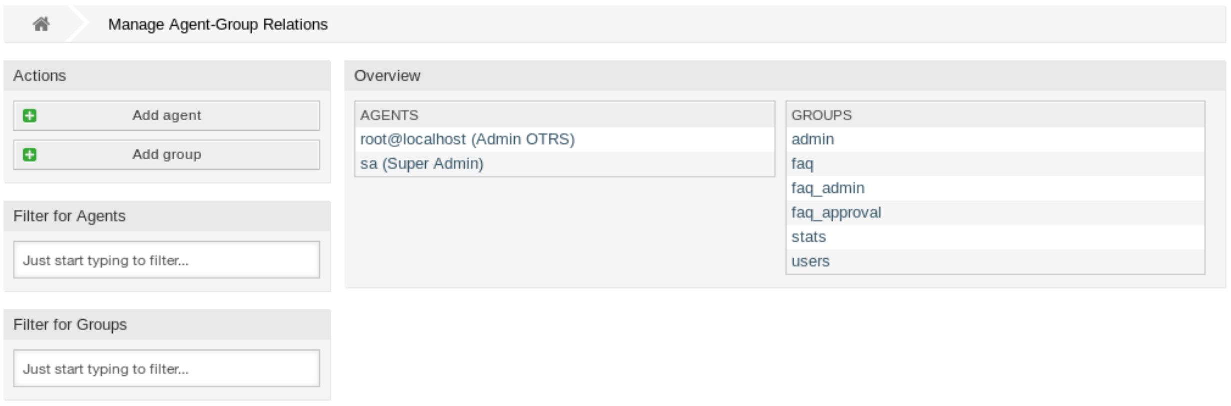
Warning: Accessing an agent or a team provides no back link to the relations screen.

4.3 Agents Groups

Efficient and straightforward management of permissions is essential in a growing business. Easy assignment of a particular user to a group for quick access, or to remove access, to resources is a must in every case.

The OTRS interface provides you both with the possibility to manage an agent' s access to one or more particular groups. As well, you can change multiple users access to any one group, efficiently and elegantly.

Use this screen to add one or more agents to one or more groups. To use this function, at least one agent and one group need to have been added to the system. The management screen is available in the *Agents Groups* module of the *Users, Groups & Roles* group.



Manage Agent-Group Relations

Actions

Filter for Agents

Just start typing to filter...

Filter for Groups

Just start typing to filter...

Overview

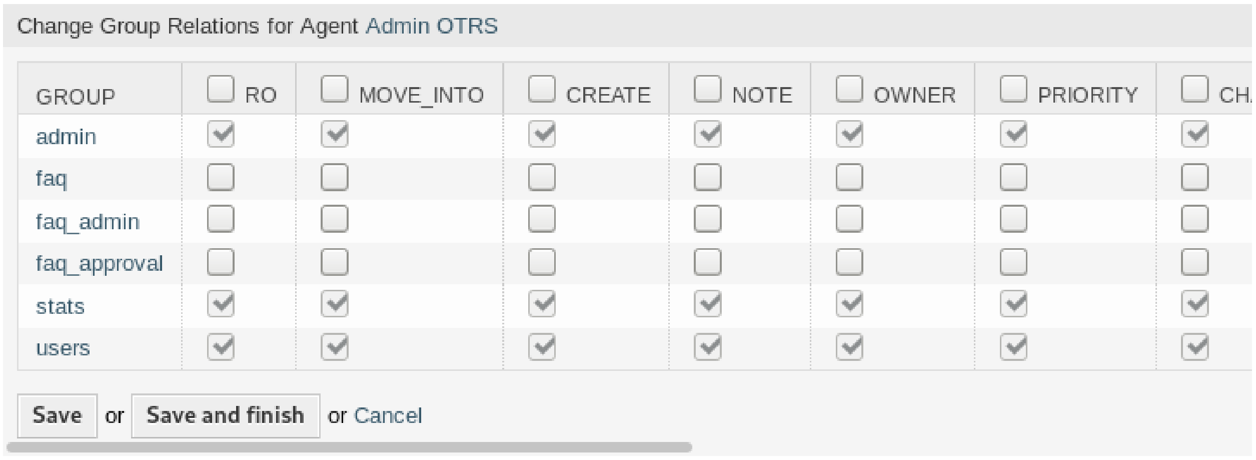
AGENTS	GROUPS
root@localhost (Admin OTRS)	admin
sa (Super Admin)	faq
	faq_admin
	faq_approval
	stats
	users

Fig. 8: Manage Agent-Group Relations

4.3.1 Manage Agents Groups Relations

To assign some groups to an agent:

1. Click on an agent in the *Agents* column.
2. Select the permissions you would like to connect the agent to groups with.
3. Click on the *Save* or *Save and finish* button.



Change Group Relations for Agent Admin OTRS

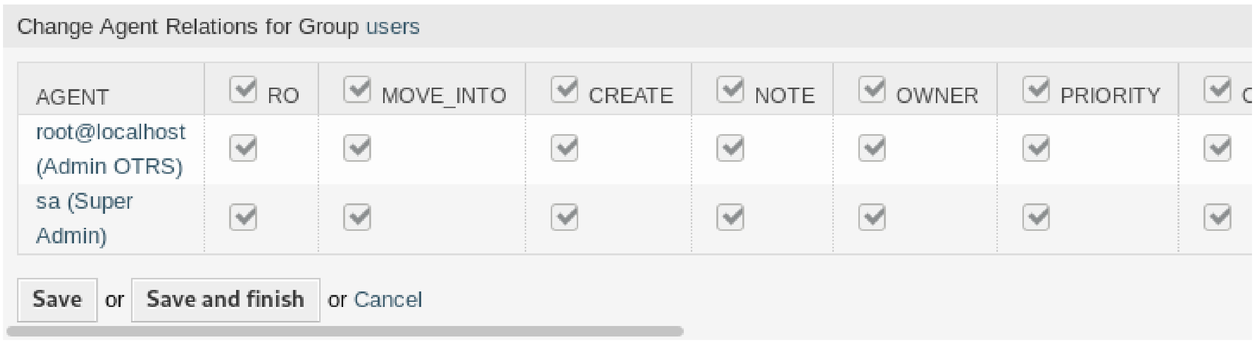
GROUP	RO	MOVE_INTO	CREATE	NOTE	OWNER	PRIORITY	CH
admin	✓	✓	✓	✓	✓	✓	✓
faq	☐	☐	☐	☐	☐	☐	☐
faq_admin	☐	☐	☐	☐	☐	☐	☐
faq_approval	☐	☐	☐	☐	☐	☐	☐
stats	✓	✓	✓	✓	✓	✓	✓
users	✓	✓	✓	✓	✓	✓	✓

Fig. 9: Change Group Relations for Agent

To assign some agents to a group:

1. Click on a group in the *Groups* column.
2. Select the permissions you would like to connect the group to agents with.
3. Click on the *Save* or *Save and finish* button.

Note: If several agents or groups are added to the system, use the filter box to find a particular agent or group by just typing the name to filter.


Change Agent Relations for Group users

AGENT	✓ RO	✓ MOVE_INTO	✓ CREATE	✓ NOTE	✓ OWNER	✓ PRIORITY	✓ C
root@localhost (Admin OTRS)	✓	✓	✓	✓	✓	✓	✓
sa (Super Admin)	✓	✓	✓	✓	✓	✓	✓

Fig. 10: Change Agent Relations for Group

Multiple agents or groups can be assigned in both screens at the same time. Additionally clicking on an agent or clicking on a group in the relations screen will open the *Edit Agent* screen or the *Edit Group* screen accordingly.

Warning: Accessing an agent or a group provides no back link to the relations screen.

4.3.2 Agents Groups Relations Reference

When assigning an agent to a group or vice versa, several permissions can be set as connection between an agent and a group. The following permissions are available by default:

ro
Read-only access to tickets in this group/queue.

move_into
Permission to move tickets in this group/queue and move existing tickets into this group/queue.

create
Permission to create tickets in this group/queue.

note
Permission to add notes to tickets and inform agents in this group/queue.

owner
Permission to set the owner of new tickets or change the owner of existing tickets in this group/queue.

priority
Permission to change the ticket priority in this group/queue.

chat_observer
Users with this permission type will only be able to observe chats in a channel after they have been invited.

chat_participant
Users with this permission type will be able to take part in a chat, but only after they get invited to it.

chat_owner
Users with this permission type will be able to accept chat customer/public requests and do all kinds of observer and participant actions on a chat.

rw
Full read and write access to the tickets in this group/queue.

See also:

Not all available permissions are shown by default. See System::Permission setting for permissions that can be added. These additional permissions can be added:

bounce
Permission to redirect an email.

close
Permission to close a ticket.

compose
Permission to compose an answer for a ticket.

customer
Permission to change the customer of a ticket.

forward
Permission to forward an article.

pending
Permission to set a ticket to pending.

phone
Permission to add a phone call to a ticket.

responsible
Permission to set the responsible agent of new tickets or change the responsible agent of existing tickets in this group/queue.

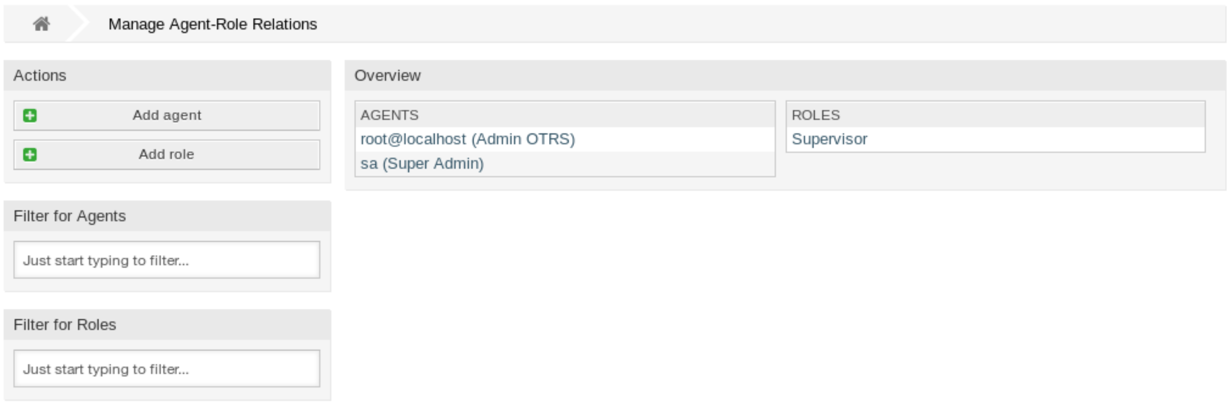
Note: By setting a checkbox in the header of a column will set all the checkboxes in the selected column. By setting the checkbox in the last *rw* column will set all the checkboxes in the selected row.

4.4 Agents Roles

As an organization grows, groups cannot be the denominator used for processing access rights. Roles become more and more a need because a role has a special set of permissions assigned. One mustn't give individual permissions, but the role carries the permissions built-in.

OTRS allows easy access to a predefined set of permissions via one or more roles defined. These roles are assigned easily to one or more agents, or one or more agent to a role.

Use this screen to add one or more agents to one or more roles. To use this function, at least one agent and one role need to have been added to the system. The management screen is available in the *Agents Roles* module of the *Users, Groups & Roles* group.


Manage Agent-Role Relations
Actions
Add agent
Add role
Filter for Agents
Just start typing to filter...
Filter for Roles
Just start typing to filter...
Overview

AGENTS
root@localhost (Admin OTRS)
sa (Super Admin)

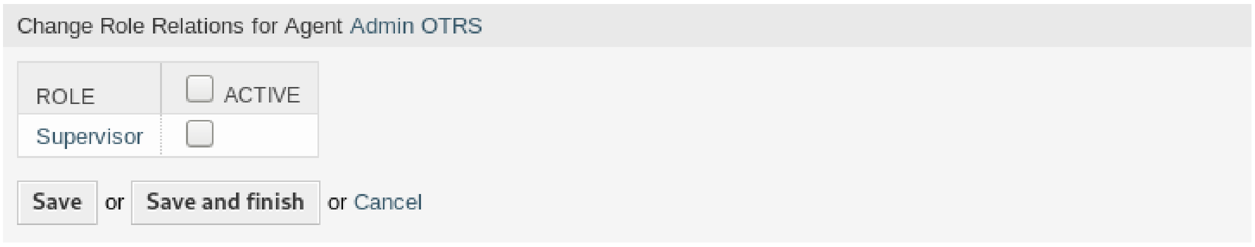
ROLES
Supervisor

Fig. 11: Manage Agent-Role Relations

4.4.1 Manage Agents Roles Relations

To assign some roles to an agent:

1. Click on an agent in the *Agents* column.
2. Select the roles you would like to the agent belongs to.
3. Click on the *Save* or *Save and finish* button.

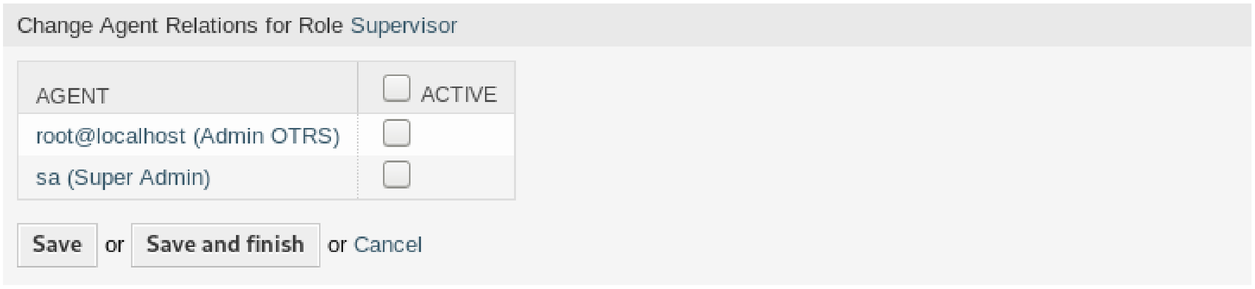

Change Role Relations for Agent Admin OTRS

ROLE	☐ ACTIVE
Supervisor	☐

Fig. 12: Change Role Relations for Agent

To assign some agents to a role:

1. Click on a role in the *Roles* column.
2. Select the agents you would like to add to the role.
3. Click on the *Save* or *Save and finish* button.


Change Agent Relations for Role Supervisor

AGENT	☐ ACTIVE
root@localhost (Admin OTRS)	☐
sa (Super Admin)	☐

Fig. 13: Change Agent Relations for Role

Note: If several agents or roles are added to the system, use the filter box to find a particular agent or role by just typing the name to filter.

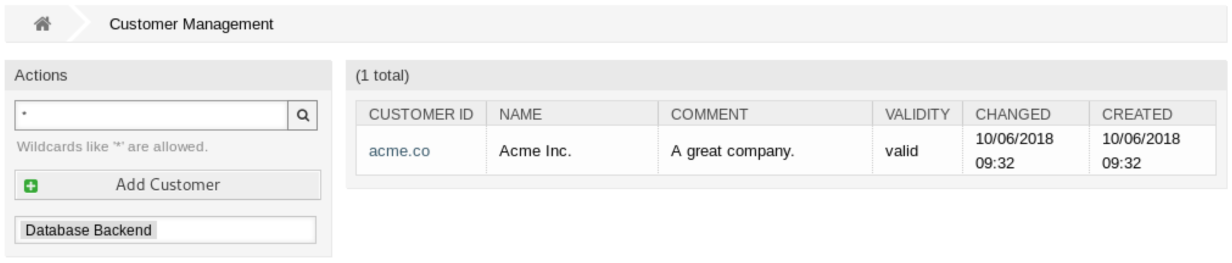
Multiple agents or roles can be assigned in both screens at the same time. Additionally clicking on an agent or clicking on a role in the relations screen will open the *Edit Agent* screen or the *Edit Role* screen accordingly.

Warning: Accessing an agent or a role provides no back link to the relations screen.

Note: By setting the checkbox in the header of the column *Active* will set all the checkboxes in the column.

4.5 Customers

Use this screen to add customer companies to the system. A fresh OTRS installation contains no customers by default. The customer management screen is available in the *Customers* module of the *Users, Groups & Roles* group.



CUSTOMER ID	NAME	COMMENT	VALIDITY	CHANGED	CREATED
acme.co	Acme Inc.	A great company.	valid	10/06/2018 09:32	10/06/2018 09:32

Fig. 14: Customer Management Screen

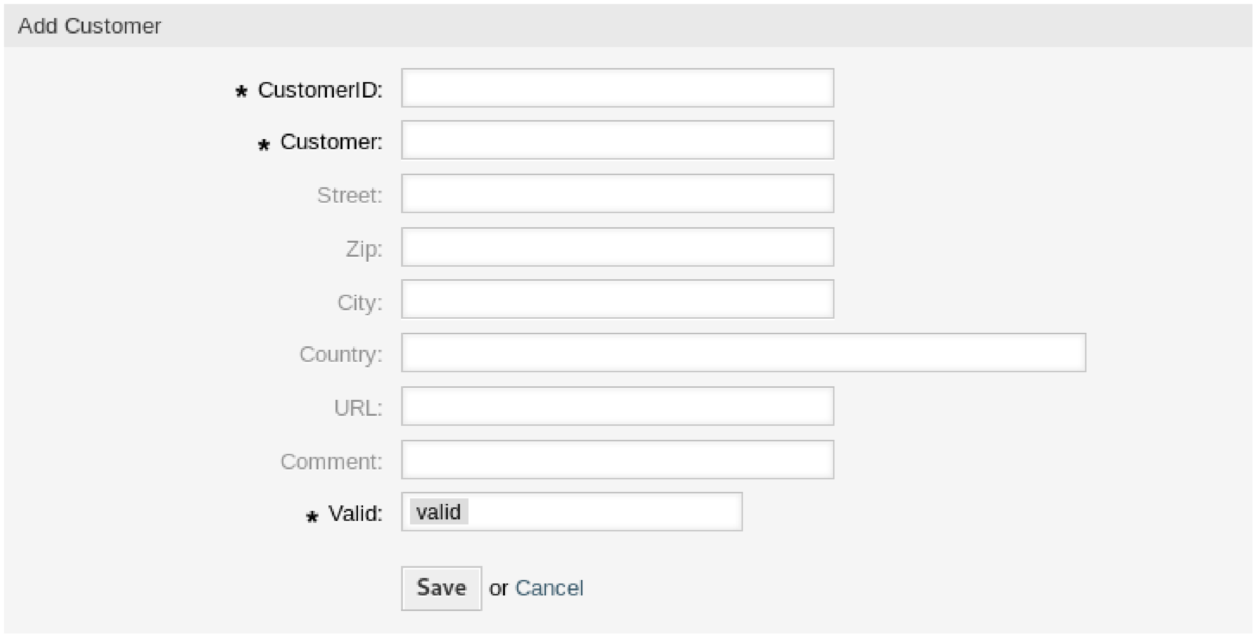
4.5.1 Manage Customers

Note: Adding or editing a customer is possible only by using database back end. Using external directory services like LDAP will disable the customer management functionality.

To add a customer:

1. Click on the *Add Customer* button in the left sidebar.
2. Fill in the required fields.
3. Click on the *Save* button.

Warning: Customers can not be deleted from the system. They can only be deactivated by setting the *Validity* option to *invalid* or *invalid-temporarily*.


Add Customer

* CustomerID:	
* Customer:	
Street:	
Zip:	
City:	
Country:	
URL:	
Comment:	
* Valid:	valid

Fig. 15: Add Customer Screen

To edit a customer:

1. Click on a customer in the list of customers.
2. Modify the fields.
3. Click on the *Save* or *Save and finish* button.

To find a customer:

1. Enter a search term to the search box in the left sidebar.
2. Click on the magnifying glass icon in the right part of the field or hit an `Enter`.

Note: If several customers are added to the system, use the search box to find a particular customer. Only the first 1000 customers are listed by default.

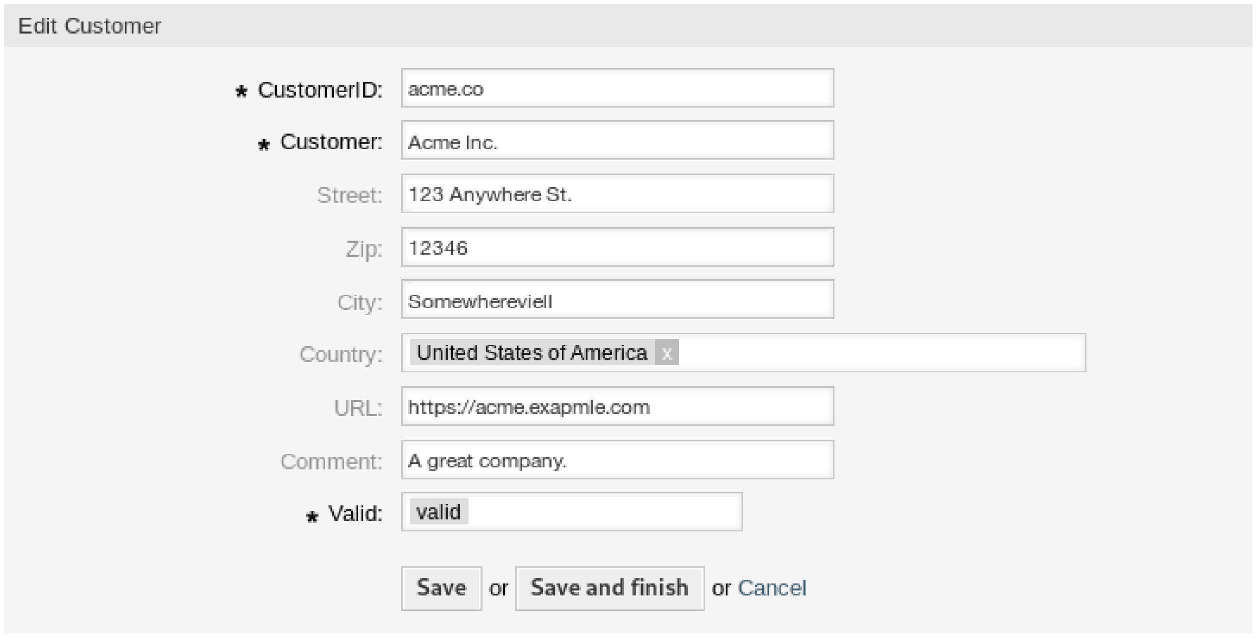
4.5.2 Customer Settings

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

Customer ID *
: The internal name of the customer. Should contain only letters, numbers and some special characters.

Customer *
: The name of this resource. Any type of characters can be entered to this field including uppercase letters and spaces. The name will be displayed in the overview table.

Street
: The street name of the customer.


Edit Customer

* CustomerID:	acme.co
* Customer:	Acme Inc.
Street:	123 Anywhere St.
Zip:	12346
City:	Somewhereviell
Country:	United States of America
URL:	https://acme.exapmle.com
Comment:	A great company.
* Valid:	valid

Fig. 16: Edit Customer Screen

Zip
The zip code of the customer.

City
The headquarter city of the customer.

Country
The country of the customer. Choose a country from the list.

URL
The web page or other URL of the customer.

Comment
Add additional information to this resource. It is recommended to always fill this field as a description of the resource with a full sentence for better clarity, because the comment will be also displayed in the overview table.

Validity *
Set the validity of this resource. Each resource can be used in OTRS only, if this field is set to *valid*. Setting this field to *invalid* or *invalid-temporarily* will disable the use of the resource.

4.5.3 Customer Back Ends

The system works with many customer data attributes such as customer ID, customer name, location information etc. These attributes are displayed in the agent interface and can be displayed in the external interface.

Customer data used or displayed within the system is highly customizable. The customer ID is always needed.

The administrator interface does not support the configuration of external back ends. Administrators need to edit the file `Kernel/Config.pm` by copying and pasting code snippets from `Kernel/Config/Defaults.pm` manually in case of using *On-Premise* system.

If you already have another customer back end (e.g. SAP), it is possible to write a module that uses it.

Warning: Do not modify the file `Kernel/Config/Defaults.pm`, it will be overwritten after upgrading the system! Copy and paste the code snippets into `Kernel/Config.pm` instead.

Note: This feature is only available to *On-Premise* customers. If you are a *Managed* customer, this feature is taken care of by the *Customer Solutions Team* in **OTRS**. Please contact us via support@otrs.com or in the OTRS Portal.

Customer Back End - Database

The default back end for customers is the OTRS database. With this back end, all customer data can be edited via the administrator interface.

```
    $Self->{CustomerCompany} = {
        Name   => Translatable('Database Backend'),
        Module => 'Kernel::System::CustomerCompany::DB',
        Params => {
            # if you want to use an external database, add the
            # required settings
#            DSN  => 'DBI:odbc:yourdsn',
#            Type => 'mssql', # only for ODBC connections
#            DSN => 'DBI:mysql:database=customerdb;host=customerdbhost',
#            User => '',
#            Password => '',
            Table => 'customer_company',
#            ForeignDB => 0,    # set this to 1 if your table does not have create_
↪time, create_by, change_time and change_by fields

            # CaseSensitive defines if the data storage of your DBMS is case␣
↪sensitive and will be
            # preconfigured within the database driver by default.
            # If the collation of your data storage differs from the default settings,
            # you can set the current behavior ( either 1 = CaseSensitive or 0 =␣
↪CaseINSensitive )
            # to fit your environment.
            #
#            CaseSensitive => 0,

            # SearchCaseSensitive will control if the searches within the data␣
↪storage are performed
            # case sensitively (if possible) or not. Change this option to 1, if you␣
↪want to search case sensitive.
            # This can improve the performance dramatically on large databases.
            SearchCaseSensitive => 0,
        },

        # company unique id
        CustomerCompanyKey             => 'customer_id',
        CustomerCompanyValid           => 'valid_id',
        CustomerCompanyListFields      => [ 'customer_id', 'name' ],
        CustomerCompanySearchFields    => [ 'customer_id', 'name' ],
```

(continues on next page)

(continued from previous page)

```
        CustomerCompanySearchPrefix    => '*',
        CustomerCompanySearchSuffix    => '*',
        CustomerCompanySearchListLimit => 250,
        CacheTTL                       => 60 * 60 * 24, # use 0 to turn off cache

        Map => [
            # Info about dynamic fields:
            #
            # Dynamic Fields of type CustomerCompany can be used within the mapping␣
↪(see example below).
            # The given storage (third column) then can also be used within the␣
↪following configurations (see above):
            # CustomerCompanySearchFields, CustomerCompanyListFields
            #
            # Note that the columns 'frontend' and 'readonly' will be ignored for␣
↪dynamic fields.

            # var, frontend, storage, shown (1=always,2=lite), required, storage-type,
↪ http-link, readonly
            [ 'CustomerID',             Translatable('CustomerID'), 'customer_id', 0,␣
↪1, 'var', '', 0 ],
            [ 'CustomerCompanyName',    Translatable('Customer'),   'name',        1,␣
↪1, 'var', '', 0 ],
            [ 'CustomerCompanyStreet',  Translatable('Street'),     'street',      1,␣
↪0, 'var', '', 0 ],
            [ 'CustomerCompanyZIP',     Translatable('Zip'),        'zip',         1,␣
↪0, 'var', '', 0 ],
            [ 'CustomerCompanyCity',    Translatable('City'),       'city',        1,␣
↪0, 'var', '', 0 ],
            [ 'CustomerCompanyCountry', Translatable('Country'),    'country',     1,␣
↪0, 'var', '', 0 ],
            [ 'CustomerCompanyURL',     Translatable('URL'),        'url',         1,␣
↪0, 'var', '[% Data.CustomerCompanyURL | html %]', 0 ],
            [ 'CustomerCompanyComment', Translatable('Comment'),    'comments',    1,␣
↪0, 'var', '', 0 ],
            [ 'ValidID',                Translatable('Valid'),      'valid_id',    0,␣
↪1, 'int', '', 0 ],

            # Dynamic field example
#             [ 'DynamicField_Name_Y', undef, 'Name_Y', 0, 0, 'dynamic_field', undef,␣
↪0 ],
        ],
    };
```

If you want to customize the customer data, change the columns or add new ones to the `customer_company` table in the database.

For example, to add a new field for VAT number:

1. Add a new column `vat` to table `customer_company`.

   MySQL or MariaDB:

   ```
   root> mysql -u root -p -e 'ALTER TABLE otrs.customer_company ADD vat VARCHAR (50)'
   ```

   PostgreSQL (from the `/opt/otrs` directory):

```
otrs> psql -c 'ALTER TABLE customer_company ADD COLUMN vat varchar(50)'
```

2. Copy the `$Self->{CustomerCompany}` section from `Kernel/Config/Defaults.pm` into `Kernel/Config.pm`.
3. Add the new column to the `Map` array.

```
[ 'CustomerCompanyVAT', 'VAT Number', 'vat', 0, 1, 'var', '', 0 ],
```

4. Add the new field to the customer create and update screen.

   Relevant system configurations:

   - `Forms###AgentFrontend::CustomerCompanyCreate::Properties`
   - `Forms###AgentFrontend::CustomerCompanyUpdate::Properties`

Note: It is recommended to always use English words for names.

See also:

Names can be translated into other languages with custom language files. For more information, see the *Custom Language File* chapter.

4.6 Customers ↔ Groups

Your organization grows, and it's not practical at some point to assign permissions to individual users, you need to assign the permissions to all customer users of a customer.

OTRS allows you to assign *group* permissions to a *customer*. Access works just the same as for agents, preventing a customer from modifying and viewing a request. Thus allowing the customer to focus on the results of the original communication and funneling the discussion through one ticket.

See also:

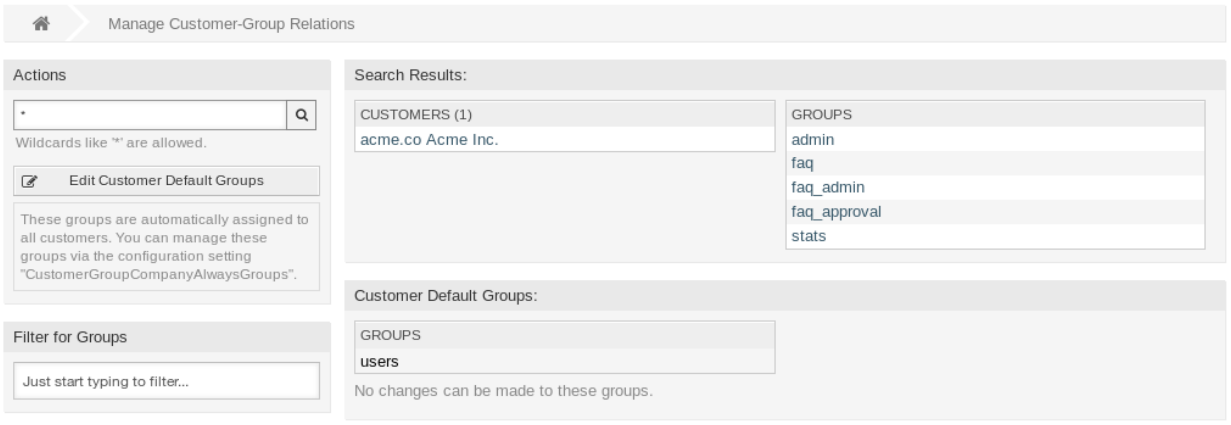
Assign a single customer user to a group using *Customer Users ↔ Groups*.

Use this screen to add one or more customers to one or more groups. To use this function, at least one customer and one group need to have been added to the system. The management screen is available in the *Customers ↔ Groups* module of the *Users, Groups & Roles* group.

Customer group support needs to be enabled in at least one customer user *back end* to use this function. For the default OTRS *back end*, this can be enabled in the system configuration by clicking on the *Enable it here!* button.

Note: To enable this feature in systems using a directory server or multiple non-default back ends, a custom configuration file needs to be placed in `Kernel/Config/Files` (for example named `ZZZ_CustomerBackend.pm`). Once activated, all customer users from this back end will require group assignment.

Warning: After making changes to the back end, the server cache will be deleted, which may cause a temporary drop in performance.



Manage Customer-Group Relations

Actions

Wildcards like '*' are allowed.

Edit Customer Default Groups

These groups are automatically assigned to all customers. You can manage these groups via the configuration setting "CustomerGroupCompanyAlwaysGroups".

Filter for Groups

Just start typing to filter...

Search Results:

CUSTOMERS (1)
acme.co Acme Inc.

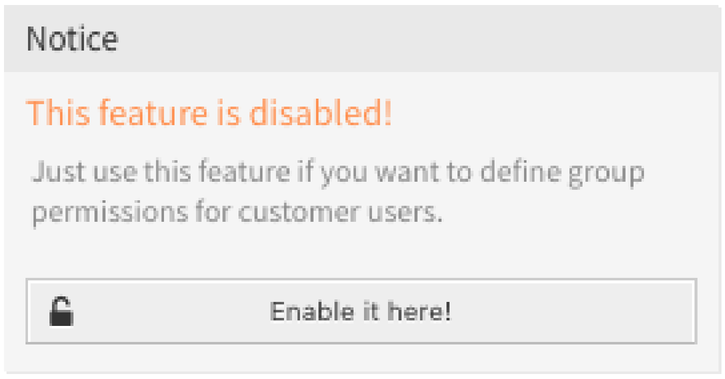
GROUPS
admin
faq
faq_admin
faq_approval
stats

Customer Default Groups:

GROUPS
users

No changes can be made to these groups.

Fig. 17: Manage Customer-Group Relations



Notice

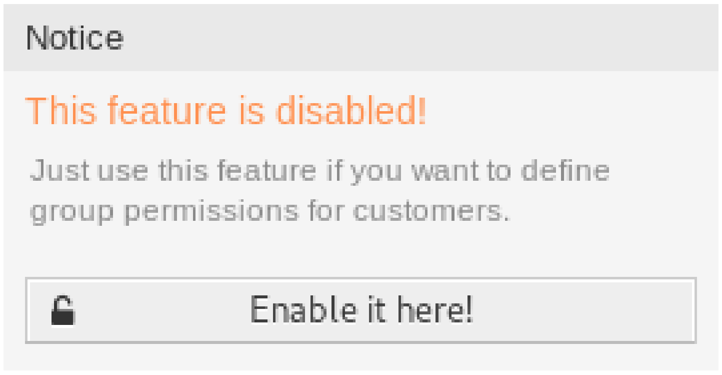
This feature is disabled!

Just use this feature if you want to define group permissions for customer users.

Fig. 18: Enable Customer Group Feature

4.6.1 Manage Customers Groups Relations

Note: To be able to use this feature, you have to activate the `CustomerGroupSupport` setting.



Notice

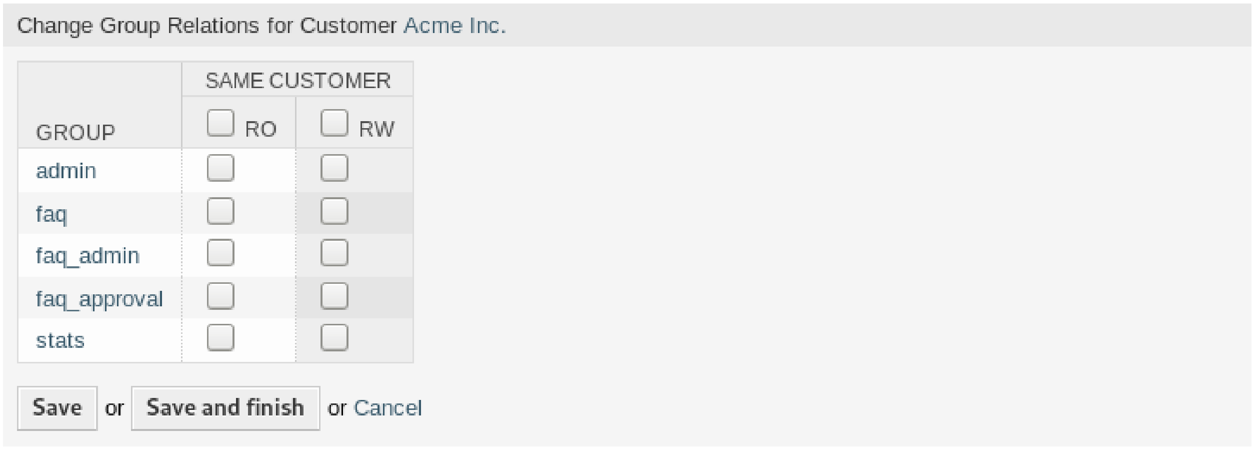
This feature is disabled!

Just use this feature if you want to define group permissions for customers.

Fig. 19: Enable Customer-Group Support

To assign some groups to a customer:

1. Click on a customer in the *Customers* column.
2. Select the permissions you would like to connect the customer to groups with.
3. Click on the *Save* or *Save and finish* button.



Change Group Relations for Customer Acme Inc.

	SAME CUSTOMER	
GROUP	☐ RO	☐ RW
admin	☐	☐
faq	☐	☐
faq_admin	☐	☐
faq_approval	☐	☐
stats	☐	☐

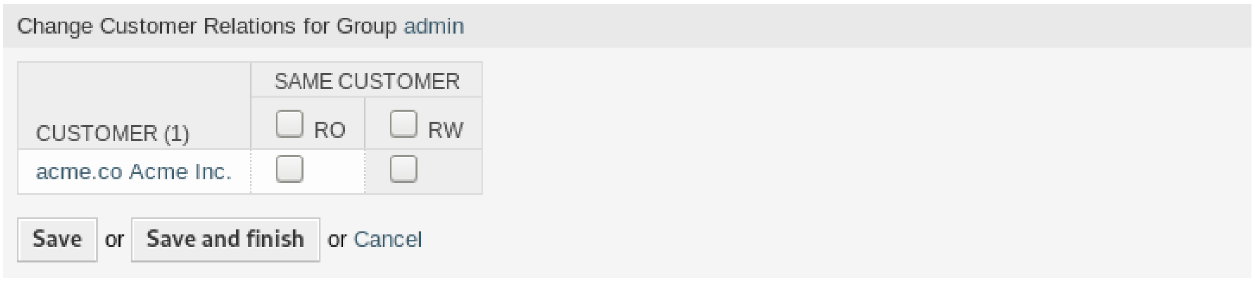
Fig. 20: Change Group Relations for Customer

To assign some customers to a group:

1. Click on a group in the *Groups* column.
2. Select the permissions you would like to connect the group to customers with.
3. Click on the *Save* or *Save and finish* button.

To change customer default groups:

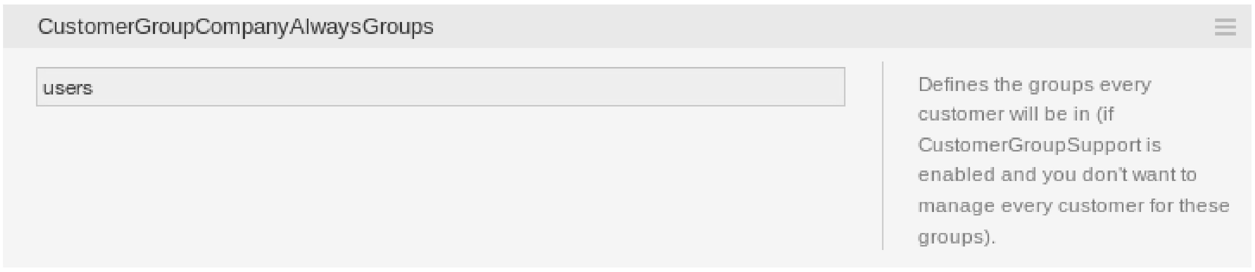
1. Click on the *Edit Customer Default Groups* button in the left sidebar.


Change Customer Relations for Group admin

	SAME CUSTOMER	
CUSTOMER (1)	☐ RO	☐ RW
acme.co Acme Inc.	☐	☐

Fig. 21: Change Customer Relations for Group

2. Add or modify groups in setting CustomerGroupCompanyAlwaysGroups.
3. Deploy the modified system configurations.


CustomerGroupCompanyAlwaysGroups

users

Defines the groups every customer will be in (if CustomerGroupSupport is enabled and you don't want to manage every customer for these groups).

Fig. 22: `CustomerGroupCompanyAlwaysGroups` System Configuration Screen

These groups are automatically assigned to all customers.

Note: If several customers or groups are added to the system, use the search box to find a particular customer or use the filter box to find a particular group by just typing the name to filter.

Multiple customers or groups can be assigned in both screens at the same time. Additionally clicking on a customer or clicking on a group in the relations will open the *Edit Customer* screen or the *Edit Group* screen accordingly.

Warning: Accessing a customer or a group provides no back link to the relations screen.

4.6.2 Customers Groups Relations Reference

When assigning a customer to a group or vice versa, several permissions can be set as connection between a customer and a group. Group permissions will be inherited by all customer users of the customer. Different contexts of permission assignment are available, which will determine how the permissions are inherited by customer users.

The following contexts are available:

Same Customer
Gives customer users group based access to tickets from customer users of the same customer (ticket `CustomerID` is a `CustomerID` of the customer user).

Note: This feature is enabled by default. You can disable it via the `CustomerGroupPermissionContext###001-CustomerID-same` setting.

Other Customers
Provides customer users access to tickets even if the tickets are not assigned to a customer user of the same customer ID(s), based on permission groups.

Note: To be able to use this feature, you have to activate the `CustomerGroupPermissionContext###100-CustomerID-other` setting.

The following permissions are available by default:

ro
Read only access to the resource.

rw
Full read and write access to the resource.

See also:

Not all available permissions are shown by default. See System::Customer::Permission setting for permissions that can be added. This additional permission can be added:

create
Permission to create a ticket.

Note: By setting a checkbox in the header of a column will set all the checkboxes in the selected column. By setting the checkbox in the last *rw* column will set all the checkboxes in the selected row.

4.6.3 Permission Functionality Example

Access to tickets on the external interface with enabled group support is mostly evaluated by a combination of group and individual (customer/customer user based) permission. Only if both criteria are met, specific access is granted.

If the resulting access is *rw*, a customer user can view and modify a ticket. If the access is *ro* only viewing is possible.

For ticket creation only the group permissions are used and a customer user can create tickets for all queues with *rw* permissions.

Group permissions are additive (meaning that only one method needs to grant permissions) and the following possibilities are taken into account:

- Customer user default groups via system configuration setting.
- Groups assigned to the customer user via the *Customer Users Groups* screen.
- Customer default groups via system configuration setting.
- Groups assigned to the customer via the *Customers Groups* screen.

For the methods above, all customers related to a customer user are used. This includes the *primary* customer (selected in the *Customer Users* screen), additional customers (added in *Customer Users Customers* screen) and other customer that might exist in the back end.

Individual permission checks require one of the following conditions to be met:

- Ticket is assigned to the customer user.
- Ticket is assigned to a customer that the customer user is related to (as explained above).
- Ticket is assigned to a customer with group permissions for the ticket queue while a customer related to the customer user has *Other Customers* permission to the same group.

An example for the last item to clarify the functionality:

- Ticket is assigned to customer user *Arvid Karlsson* with related customer *Ericsson AB*.
- Ticket is located in queue *Support Sweden*.
- Queue *Support Sweden* is in group *support-se*.
- Customer *Ericsson AB* has *Same Customer* context with *rw* permission to group *support-se*.
- Logged in customer user is *Barry Smith* which is related to customer *Farmers Inc.*.
- Customer *Farmers Inc.* has *Same Customer* context with *ro* permission to group *support-se*.
- Now, if customer *Farmers Inc.* is given *Other Customers* context with *ro* permission to group *support-se*, *Barry Smith* will be able to view the ticket.
- In order for *Barry* to modify the ticket, *rw* permission is required for both *Same Customer* and *Other Customers* contexts.

4.6.4 Multi-tier Customer Relationship

In this example we will create a multi-tier customer structure with resulting ticket permissions. To get the same results you will need a relatively clean system without many customizations.

1. Create the following customers in the *Customers* screen:

Customer ID	Customer
`de`	Graubrot AG
`mx`	Hernandez SA
`se`	Ericsson AB
`us`	Farmers Inc.

2. Create the following customer users in the *Customer Users* screen and assign them to the already created customers. Use any valid email address for the email field.

Firstname	Lastname	Username	Customer ID
Arvid	Karlsson	`ak`	Ericsson AB
Barry	Smith	`bs`	Farmers Inc.
Christian	Müller	`cm`	Graubrot AG
Diego	Garcia	`dg`	Hernandez SA

3. Create the following groups in the *Groups* screen:

   - `faq-amer`
   - `faq-emea`
   - `support-de`
   - `support-mx`

- `support-se`
- `support-us`

4. Go to the *Queues* screen and add corresponding queues which will use the previously created groups. In the *System address* field you can use any available address.

Name	Group
FAQ Germany	`faq-emea`
FAQ Mexico	`faq-amer`
FAQ Sweden	`faq-emea`
FAQ USA	`faq-amer`
Support Germany	`support-de`
Support Mexico	`support-mx`
Support Sweden	`support-se`
Support USA	`support-us`

5. Go to the *Customer Users   Customers* screen and assign the select customer users to other customers.

Customer User	Customers	Active
Arvid Karlsson	`de` Graubrot AG	yes {1}
Diego Garcia	`se` Ericsson AB `us` Farmers Inc.	yes {2}

6. Go to the *Customer Users   Groups* screen and assign a single customer user direct access to a group.

Customer User	Group	Permission
Diego Garcia	`faq-emea`	rw {3}

7. Go to the *Customers   Groups* screen and assign customers to groups according to the matrix below. Be sure to select proper permission level for each group and company.

Customer	Same Customer	Other Customers
`de` Graubrot AG	`faq-amer` → ro {4} `faq-emea` → ro `support-de` → rw `support-mx` → ro	
`mx` Hernandex SA	`faq-amer` → ro {5} `faq-emea` → ro `support-de` → ro `support-mx` → rw	`support-de` → rw {6} `support-mx` → rw
`se` Ericsson AB	`faq-amer` → ro {7} `faq-emea` → ro `support-se` → rw	
`us` Farmers Inc.	`faq-amer` → ro {8} `faq-emea` → ro `support-us` → rw	`faq-amer` → ro {9}

The {6} is intentional to demonstrate limitation to base permissions.

For reference, please consult the image below where all relationships are drawn as lines:

8. Create some tickets. Go to *New Phone Ticket* screen and create tickets, one each per customer user and queue (32 in total). By the way, this is possible in the agent interface as the customer group restrictions are only active on the external interface.

For checking resulting access to the tickets, you can easily switch between the customer users by activating `SwitchToCustomer` option in the system configuration. Then just go to the *Customer Users* and click on corresponding *Switch to customer* link next to the customer user's name.

You will be immediately logged in as that customer user and you can visit the *Company Tickets* screen using the *Ticket* menu item for checking the ticket access. It should conform to the matrix below. Click on a ticket to check if corresponding permission level is honored: for *ro* permission level you should not see the *Reply* button.

This is the expected result for each customer user. The marker `{N}` refers to the location above where the corresponding setting was taken (this shows why the access is granted).

Resulting access for customer user *Arvid Karlsson*:

- Queue FAQ Germany: ro (via {7}) + Christian's tickets ro (via {1})
- Queue FAQ Mexico: ro (via {7}) + Christian's tickets ro (via {1})
- Queue FAQ Sweden: ro (via {7}) + Christian's tickets ro (via {1})
- Queue FAQ USA: ro (via {7}) + Christian's tickets ro (via {1})

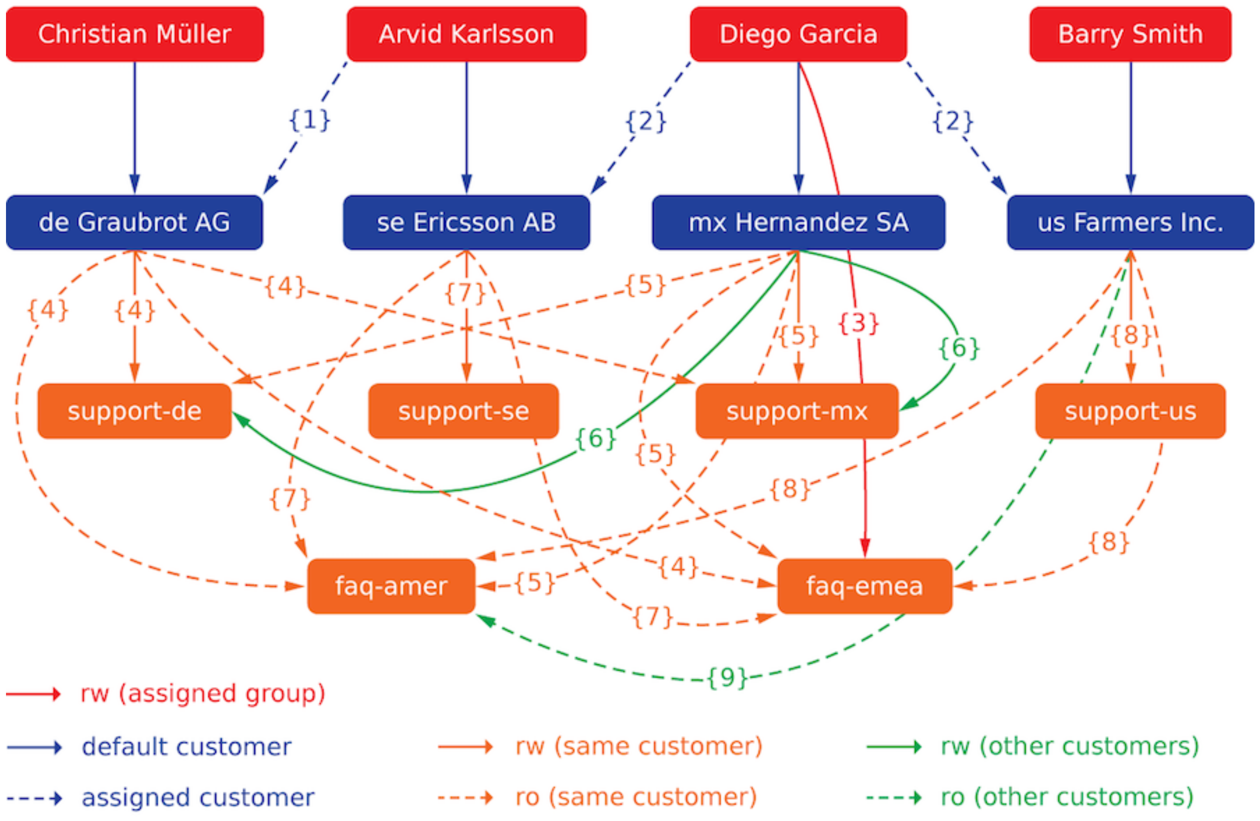


Fig. 23: Multi-tier Customer Relationship

- Queue Support Germany: rw (via {1 → 6}) + Christian' s tickets rw (via {1})
- Queue Support Mexico: -
- Queue Support Sweden: rw (via {7}) + Christian' s tickets rw (via {1})
- Queue Support USA: -

Resulting access for customer user *Barry Smith*:

- Queue FAQ Germany: ro (via {8})
- Queue FAQ Mexico: ro (via {8}) + Arvid' s, Christian' s, Diego' s tickets ro (via {9})
- Queue FAQ Sweden: ro (via {8})
- Queue FAQ USA: ro (via {8}) + Arvid' s, Christian' s, Diego' s tickets ro (via {9})
- Queue Support Germany: -
- Queue Support Mexico: -
- Queue Support Sweden: -
- Queue Support USA: rw (via {8})

Resulting access for customer user *Christian Müller*:

- Queue FAQ Germany: ro (via {4})
- Queue FAQ Mexico: ro (via {4})
- Queue FAQ Sweden: ro (via {4})
- Queue FAQ USA: ro (via {4})
- Queue Support Germany: rw (via {4})
- Queue Support Mexico: ro (via {4})
- Queue Support Sweden: -
- Queue Support USA: -

Resulting access for customer user *Diego Garcia*:

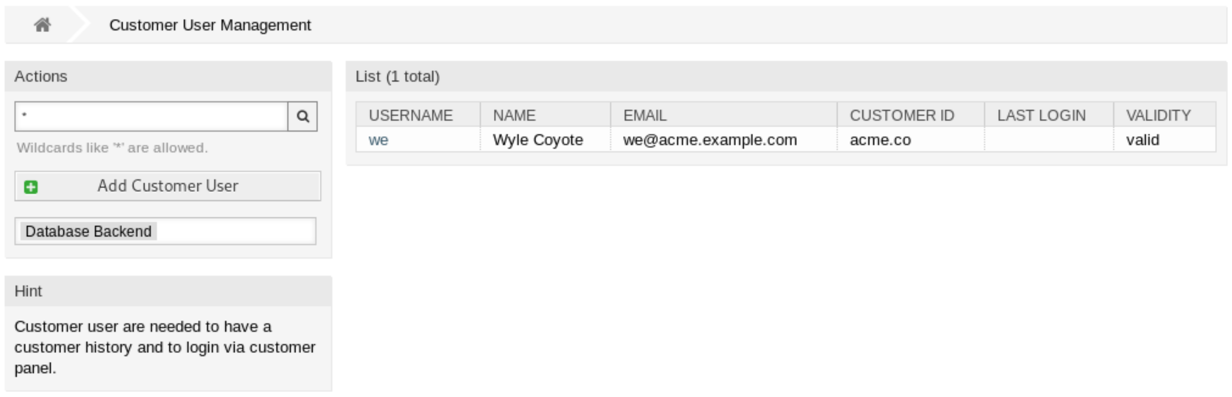
- Queue FAQ Germany: rw (via {3}) + Arvid' s, Barry' s tickets rw (via {2})
- Queue FAQ Mexico: ro (via {5}) + Arvid' s, Barry' s tickets ro (via {2}) + Christian' s tickets ro (via {2 → 9})
- Queue FAQ Sweden: rw (via {3}) + Arvid' s, Barry' s tickets rw (via {2})
- Queue FAQ USA: ro (via {5}) + Arvid' s, Barry' s tickets ro (via {2}) + Christian' s tickets ro (via {2 → 9})
- Queue Support Germany: ro (via {5}) + Arvid' s, Barry' s tickets ro (via {2}) + Christian' s tickets ro (via {6})
- Queue Support Mexico: rw (via {5}) + Arvid' s, Barry' s tickets rw (via {2}) + Christian' s tickets rw (via {6})
- Queue Support Sweden: rw (via {2 → 4}) + Arvid' s, Barry' s tickets rw (via {2})
- Queue Support USA: rw (via {2 → 5}) + Arvid' s, Barry' s tickets rw (via {2})

4.7 Customer Users

A record of who your company deals with requires more information about that individual: physical location for shipping and billing purposes, as well as contact information for email and phone contact.

OTRS offers a great way to save individual information about contacts within organizations which your company serves. You can add as many personal connections into OTRS as needed.

Use this screen to add a *customer user* to the system. A fresh OTRS installation contains no customer users by default. The customer user management screen is available in the *Customer Users* module of the *Users, Groups & Roles* group.



USERNAME	NAME	EMAIL	CUSTOMER ID	LAST LOGIN	VALIDITY
we	Wyle Coyote	we@acme.example.com	acme.co		valid

Customer user are needed to have a customer history and to login via customer panel.

Fig. 24: Customer User Management Screen

4.7.1 Manage Customer Users

Warning: A customer user can only be added to the system when at least one *customer* exists. Create one or more *Customers* first.

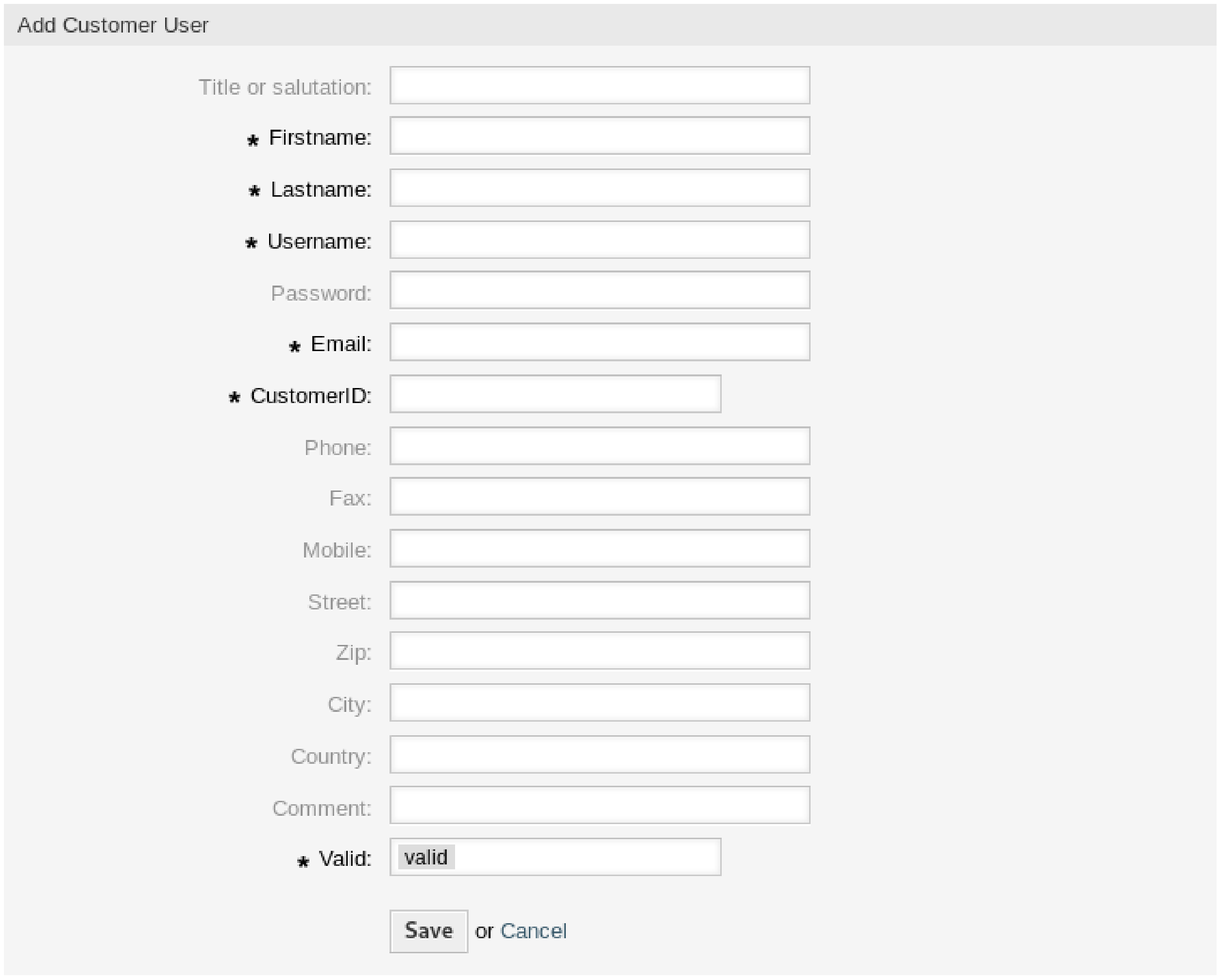
Note: Adding or editing a customer user is possible only by using database back end. Using external directory services like LDAP will disable the customer user management functionality.

To add a customer user:

1. Click on the *Add Customer User* button in the left sidebar.
2. Fill in the required fields.
3. Click on the *Save* button.

Warning: Customer users can not be deleted from the system. They can only be deactivated by setting the *Validity* option to *invalid* or *invalid-temporarily*.

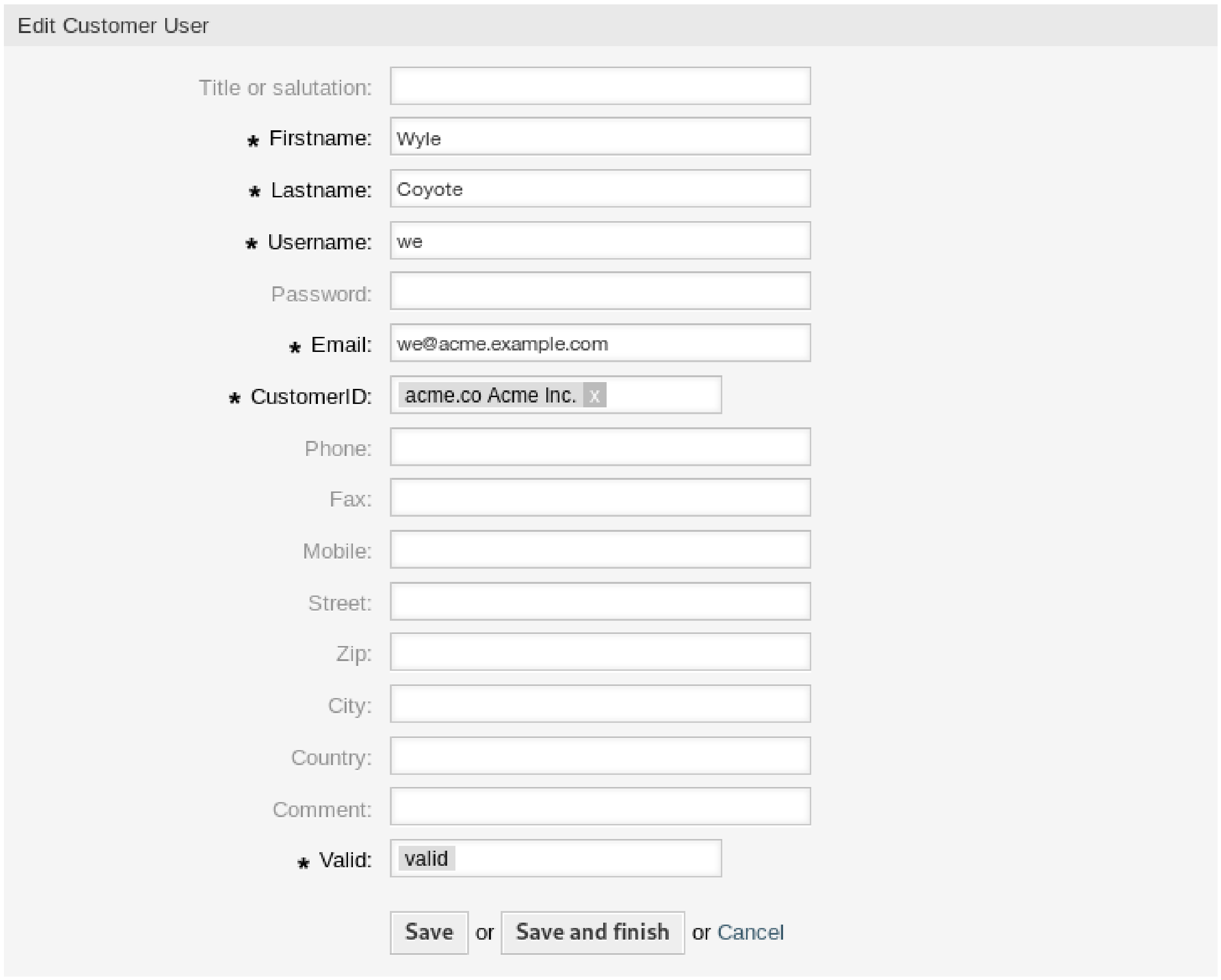
To edit a customer user:


Add Customer User

Title or salutation:	
* Firstname:	
* Lastname:	
* Username:	
Password:	
* Email:	
* CustomerID:	
Phone:	
Fax:	
Mobile:	
Street:	
Zip:	
City:	
Country:	
Comment:	
* Valid:	valid

Fig. 25: Add Customer User Screen

1. Click on a customer user in the list of customer users.
2. Modify the fields.
3. Click on the *Save* or *Save and finish* button.


Edit Customer User

Title or salutation:	
⁎ Firstname:	Wyle
⁎ Lastname:	Coyote
⁎ Username:	we
Password:	
⁎ Email:	we@acme.example.com
⁎ CustomerID:	acme.co Acme Inc. x
Phone:	
Fax:	
Mobile:	
Street:	
Zip:	
City:	
Country:	
Comment:	
⁎ Valid:	valid

Fig. 26: Edit Customer User Screen

To find a customer user:

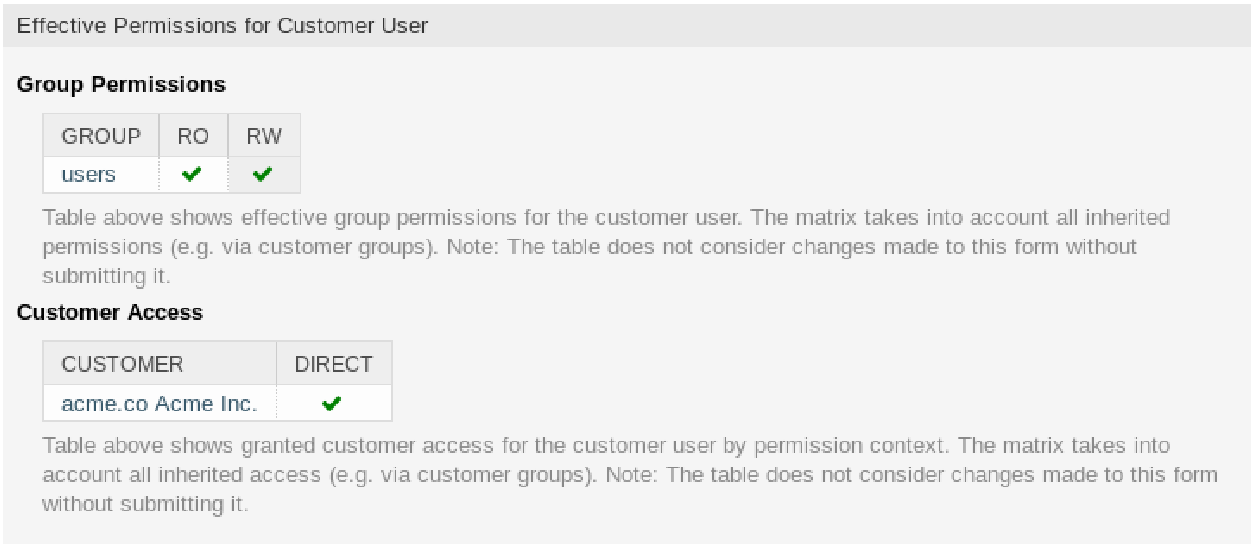
1. Enter a search term to the search box in the left sidebar.
2. Click on the magnifying glass icon in the right part of the field or hit an `Enter`.

Note: If several customer users are added to the system, use the search box to find a particular customer user. Only the first 1000 customer users are listed by default.

The customer user permissions can be controlled by adding a customer or a customer user to *Groups*. This can result a complex matrix of permissions. The effective permissions for a customer user can be checked in the bottom of the *Edit Customer User* screen.

See also:

Customer Users *Groups* needs to be enabled to use this feature.



Effective Permissions for Customer User

Group Permissions

GROUP	RO	RW
users	✓	✓

Table above shows effective group permissions for the customer user. The matrix takes into account all inherited permissions (e.g. via customer groups). Note: The table does not consider changes made to this form without submitting it.

Customer Access

CUSTOMER	DIRECT
acme.co Acme Inc.	✓

Table above shows granted customer access for the customer user by permission context. The matrix takes into account all inherited access (e.g. via customer groups). Note: The table does not consider changes made to this form without submitting it.

Fig. 27: Effective Permissions for Customer User Widget

4.7.2 Customer User Settings

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

Note: These are the default fields available for the internal database table.

Title or salutation
Some name prefix can be added here like *Mr.*, *Dr.*, *Jr.*, etc.

Firstname *
The first name of the customer user.

Lastname *
The last name of the customer user.

Username *
The username of the customer user to login to the system.

Password
The password of the customer user. This will be auto-generated if left empty.

Email *
The email address of the customer user.

Customer *
The customer company the customer user belongs to. Select a customer from the list of *Customers*.

Phone
The phone number of the customer user.

Fax
The fax number of the customer user.

Mobile
The cellphone number of the customer user.

Street
The street name of the customer.

Zip
The zip code of the customer.

City
The headquarter city of the customer.

Country
The country of the customer user.

Comment
Add additional information to this resource. It is recommended to always fill this field as a description of the resource with a full sentence for better clarity, because the comment will be also displayed in the overview table.

Validity *
Set the validity of this resource. Each resource can be used in OTRS only, if this field is set to *valid*. Setting this field to *invalid* or *invalid-temporarily* will disable the use of the resource.

See also:

It is possible to assign multiple customers to customer users via the *Customer Users Customers* screen.

4.7.3 Customer User Back Ends

The system works with many customer user data attributes such as username, email address, phone number, etc. These attributes are displayed in both the agent and the external interface, and also used for the authentication of customer users.

Customer data used or displayed within the system is highly customizable. The user login and the email address are always needed for customer authentication.

The administrator interface does not support the configuration of external back ends. Administrators need to edit the file `Kernel/Config.pm` by copying and pasting code snippets from `Kernel/Config/Defaults.pm` manually in case of using *On-Premise* system.

If you already have another customer back end (e.g. SAP), it is possible to write a module that uses it.

Warning: Do not modify the file `Kernel/Config/Defaults.pm`, it will be overwritten after upgrading the system! Copy and paste the code snippets into `Kernel/Config.pm` instead.

Note: This feature is only available to *On-Premise* customers. If you are a *Managed* customer, this feature is taken care of by the *Customer Solutions Team* in **OTRS**. Please contact us via support@otrs.com or in the OTRS Portal.

Customer User Back End - Database

The default user authentication back end for customer users is the OTRS database. With this back end, all customer user data can be edited via the administrator interface.

```
# This is the auth. module for the otrs db
# you can also configure it using a remote database
$Self->{'Customer::AuthModule'}                      =
↪'Kernel::System::CustomerAuth::DB';
$Self->{'Customer::AuthModule::DB::Table'}           = 'customer_user';
$Self->{'Customer::AuthModule::DB::CustomerKey'}     = 'login';
$Self->{'Customer::AuthModule::DB::CustomerPassword'} = 'pw';

#    $Self->{'Customer::AuthModule::DB::DSN'} = "DBI:mysql:database=customerdb;
↪host=customerdbhost";
#    $Self->{'Customer::AuthModule::DB::User'} = "some_user";
#    $Self->{'Customer::AuthModule::DB::Password'} = "some_password";

# if you use odbc or you want to define a database type (without autodetection)
#    $Self->{'Customer::AuthModule::DB::Type'} = 'mysql';

# password crypt type (bcrypt|sha2|sha1|md5|apr1|crypt|plain)
#    $Self->{'Customer::AuthModule::DB::CryptType'} = 'sha2';
```

The example below shows the configuration of a database customer back end, which uses customer user data stored in the database table `customer_user`.

```
    # CustomerUser
    # (customer user database backend and settings)
    $Self->{CustomerUser} = {
        Name   => Translatable('Database Backend'),
        Module => 'Kernel::System::CustomerUser::DB',
        Params => {

            # if you want to use an external database, add the
            # required settings
#            DSN  => 'DBI:odbc:yourdsn',
#            Type => 'mssql', # only for ODBC connections
#            DSN => 'DBI:mysql:database=customerdb;host=customerdbhost',
#            User => '',
#            Password => '',
            Table => 'customer_user',
#            ForeignDB => 0,    # set this to 1 if your table does not have create_
↪time, create_by, change_time and change_by fields

            # CaseSensitive defines if the data storage of your DBMS is case␣
↪sensitive and will be
            # preconfigured within the database driver by default.
            # If the collation of your data storage differs from the default settings,
            # you can set the current behavior ( either 1 = CaseSensitive or 0 =␣
↪CaseINSensitive )
            # to fit your environment.
            #
#            CaseSensitive => 0,

            # SearchCaseSensitive will control if the searches within the data␣
↪storage are performed
```

(continues on next page)

(continued from previous page)

```
            # case sensitively (if possible) or not. Change this option to 1, if you␣
↪want to search case sensitive.
            # This can improve the performance dramatically on large databases.
            SearchCaseSensitive => 0,
        },

        # customer unique id
        CustomerKey => 'login',

        # customer #
        CustomerID    => 'customer_id',
        CustomerValid => 'valid_id',

        # The last field must always be the email address so that a valid
        #   email address like "John Doe" <john.doe@domain.com> can be constructed␣
↪from the fields.
        CustomerUserListFields => [ 'first_name', 'last_name', 'email' ],

#        CustomerUserListFields => ['login', 'first_name', 'last_name', 'customer_id',
↪ 'email'],
        CustomerUserSearchFields           => [ 'login', 'first_name', 'last_name',
↪'customer_id' ],
        CustomerUserSearchPrefix           => '*',
        CustomerUserSearchSuffix           => '*',
        CustomerUserSearchListLimit        => 250,
        CustomerUserPostMasterSearchFields => ['email'],
        CustomerUserNameFields             => [ 'title', 'first_name', 'last_name' ],
        CustomerUserEmailUniqCheck         => 1,

#        # Configures the character for joining customer user name parts. Join single␣
↪space if it is not defined.
#        # CustomerUserNameFieldsJoin => '',

#        # show now own tickets in customer panel, CompanyTickets
#        CustomerUserExcludePrimaryCustomerID => 0,
#        # generate auto logins
#        AutoLoginCreation => 0,
#        # generate auto login prefix
#        AutoLoginCreationPrefix => 'auto',
#        # admin can change customer preferences
#        AdminSetPreferences => 1,
        # use customer company support (reference to company, See CustomerCompany␣
↪settings)
        CustomerCompanySupport => 1,
        # cache time to live in sec. - cache any database queries
        CacheTTL => 60 * 60 * 24,
#        # Consider this source read only.
#        ReadOnly => 1,
        Map => [

            # Info about dynamic fields:
            #
            # Dynamic Fields of type CustomerUser can be used within the mapping (see␣
↪example below).
            # The given storage (third column) then can also be used within the␣
↪following configurations (see above):
```

(continues on next page)

(continued from previous page)

```
            # CustomerUserSearchFields, CustomerUserPostMasterSearchFields,␣
↪CustomerUserListFields, CustomerUserNameFields
            #
            # Note that the columns 'frontend' and 'readonly' will be ignored for␣
↪dynamic fields.

            # note: Login, Email and CustomerID needed!
            # var, frontend, storage, shown (1=always,2=lite), required, storage-type,
↪ http-link, readonly, http-link-target, link class(es)
            [ 'UserTitle',        Translatable('Title or salutation'), 'title',       ␣
↪   1, 0, 'var', '', 0, undef, undef ],
            [ 'UserFirstname',    Translatable('Firstname'),          'first_name', ␣
↪   1, 1, 'var', '', 0, undef, undef ],
            [ 'UserLastname',     Translatable('Lastname'),           'last_name',  ␣
↪   1, 1, 'var', '', 0, undef, undef ],
            [ 'UserLogin',        Translatable('Username'),           'login',      ␣
↪   1, 1, 'var', '', 0, undef, undef ],
            [ 'UserPassword',     Translatable('Password'),           'pw',         ␣
↪   0, 0, 'var', '', 0, undef, undef ],
            [ 'UserEmail',        Translatable('Email'),              'email',      ␣
↪   1, 1, 'var', '', 0, undef, undef ],
#             [ 'UserEmail',        Translatable('Email'),              'email',      ␣
↪    1, 1, 'var', '[% Env("CGIHandle") %]?Action=AgentTicketCompose;ResponseID=1;
↪TicketID=[% Data.TicketID | uri %];ArticleID=[% Data.ArticleID | uri %]', 0, '',
↪'AsPopup OTRSPopup_TicketAction' ],
            [ 'UserCustomerID',   Translatable('CustomerID'),         'customer_id',␣
↪   0, 1, 'var', '', 0, undef, undef ],
#             [ 'UserCustomerIDs',  Translatable('CustomerIDs'),         'customer_ids
↪',    1, 0, 'var', '', 0, undef, undef ],
            [ 'UserPhone',        Translatable('Phone'),              'phone',      ␣
↪   1, 0, 'var', '', 0, undef, undef ],
            [ 'UserFax',          Translatable('Fax'),                'fax',        ␣
↪   1, 0, 'var', '', 0, undef, undef ],
            [ 'UserMobile',       Translatable('Mobile'),             'mobile',     ␣
↪   1, 0, 'var', '', 0, undef, undef ],
            [ 'UserStreet',       Translatable('Street'),             'street',     ␣
↪   1, 0, 'var', '', 0, undef, undef ],
            [ 'UserZip',          Translatable('Zip'),                'zip',        ␣
↪   1, 0, 'var', '', 0, undef, undef ],
            [ 'UserCity',         Translatable('City'),               'city',       ␣
↪   1, 0, 'var', '', 0, undef, undef ],
            [ 'UserCountry',      Translatable('Country'),            'country',    ␣
↪   1, 0, 'var', '', 0, undef, undef ],
            [ 'UserComment',      Translatable('Comment'),            'comments',   ␣
↪   1, 0, 'var', '', 0, undef, undef ],
            [ 'ValidID',          Translatable('Valid'),              'valid_id',   ␣
↪   0, 1, 'int', '', 0, undef, undef ],

            # Dynamic field example
#             [ 'DynamicField_Name_X', undef, 'Name_X', 0, 0, 'dynamic_field', undef,␣
↪0, undef, undef ],
        ],

        # default selections
        Selections => {
```

(continues on next page)

(continued from previous page)

```
#             UserTitle => {
#                 'Mr.' => Translatable('Mr.'),
#                 'Mrs.' => Translatable('Mrs.'),
#             },
        },
    };
```

If you want to customize the customer user data, change the columns or add new ones to the `customer_user` table in the database.

For example, to add a new field for room number:

1. Add a new column `room` to table `customer_user`.

   MySQL or MariaDB:

   ```
   root> mysql -u root -p -e 'ALTER TABLE otrs.customer_user ADD room VARCHAR (250)'
   ```

   PostgreSQL (from the `/opt/otrs` directory):

   ```
   otrs> psql -c 'ALTER TABLE customer_user ADD COLUMN room varchar(250)'
   ```

2. Copy the `$Self->{CustomerUser}` section from `Kernel/Config/Defaults.pm` into `Kernel/Config.pm`.

3. Add the new column to the `Map` array.

   ```
   [ 'UserRoom', 'Room', 'room', 0, 1, 'var', '', 0, undef, undef ],
   ```

   You can set the HTTP link target and link class (the last two keys) to `undef` in map array elements, if they are not to be used. These keys add `target=""` and `class=""` attributes to the HTTP link element, respectively. They are ignored if HTTP link is not set (it is `''` in this example).

4. Add the new field to the customer user create and update screen.

   Relevant system configurations:

   - `Forms###AgentFrontend::CustomerUserCreate::Properties`
   - `Forms###AgentFrontend::CustomerUserUpdate::Properties`

Note: It is recommended to always use English words for names.

See also:

Names can be translated into other languages with custom language files. For more information, see the *Custom Language File* chapter.

Customer User Back End - LDAP

If you have an LDAP directory with all your customer user data, you can use the LDAP module to authenticate your customer users. Because this module has only read-access to the LDAP back end, it is not possible to edit the customer user data via the administrator interface.

```
# This is an example configuration for an LDAP auth. backend.
# (take care that Net::LDAP is installed!)
#    $Self->{'Customer::AuthModule'} = 'Kernel::System::CustomerAuth::LDAP';
#    $Self->{'Customer::AuthModule::LDAP::Host'} = 'ldap.example.com';
#    $Self->{'Customer::AuthModule::LDAP::BaseDN'} = 'dc=example,dc=com';
#    $Self->{'Customer::AuthModule::LDAP::UID'} = 'uid';

# Check if the user is allowed to auth in a posixGroup
# (e. g. user needs to be in a group xyz to use otrs)
#    $Self->{'Customer::AuthModule::LDAP::GroupDN'} = 'cn=otrsallow,ou=posixGroups,
↪dc=example,dc=com';
#    $Self->{'Customer::AuthModule::LDAP::AccessAttr'} = 'memberUid';
# for ldap posixGroups objectclass (just uid)
#    $Self->{'Customer::AuthModule::LDAP::UserAttr'} = 'UID';
# for non ldap posixGroups objectclass (full user dn)
#    $Self->{'Customer::AuthModule::LDAP::UserAttr'} = 'DN';

# The following is valid but would only be necessary if the
# anonymous user do NOT have permission to read from the LDAP tree
#    $Self->{'Customer::AuthModule::LDAP::SearchUserDN'} = '';
#    $Self->{'Customer::AuthModule::LDAP::SearchUserPw'} = '';

# in case you want to add always one filter to each ldap query, use
# this option. e. g. AlwaysFilter => '(mail=*)' or AlwaysFilter => '(objectclass=user)
↪'
#   $Self->{'Customer::AuthModule::LDAP::AlwaysFilter'} = '';

# in case you want to add a suffix to each customer login name, then
# you can use this option. e. g. user just want to use user but
# in your ldap directory exists user@domain.
#    $Self->{'Customer::AuthModule::LDAP::UserSuffix'} = '@domain.com';

# Net::LDAP new params (if needed - for more info see perldoc Net::LDAP)
#    $Self->{'Customer::AuthModule::LDAP::Params'} = {
#        port    => 389,
#        timeout => 120,
#        async   => 0,
#        version => 3,
#    };

# Die if backend can't work, e. g. can't connect to server.
#    $Self->{'Customer::AuthModule::LDAP::Die'} = 1;
```

The example below shows the configuration of a LDAP customer user back end.

```
# CustomerUser
# (customer user ldap backend and settings)
    $Self->{CustomerUser} = {
        Name => 'LDAP Backend',
        Module => 'Kernel::System::CustomerUser::LDAP',
        Params => {
```

(continues on next page)

(continued from previous page)

```
            # ldap host
            Host => 'bay.csuhayward.edu',
            # ldap base dn
            BaseDN => 'ou=seas,o=csuh',
            # search scope (one|sub)
            SSCOPE => 'sub',
            # The following is valid but would only be necessary if the
            # anonymous user does NOT have permission to read from the LDAP tree
            UserDN => '',
            UserPw => '',
            # in case you want to add always one filter to each ldap query, use
            # this option. e. g. AlwaysFilter => '(mail=*)' or AlwaysFilter =>
↪'(objectclass=user)'
            AlwaysFilter => '',
            # if the charset of your ldap server is iso-8859-1, use this:
#            # SourceCharset => 'iso-8859-1',
            # die if backend can't work, e. g. can't connect to server
            Die => 0,
            # Net::LDAP new params (if needed - for more info see perldoc Net::LDAP)
            Params => {
                port    => 389,
                timeout => 120,
                async   => 0,
                version => 3,
            },
        },
        # customer unique id
        CustomerKey => 'uid',
        # customer #
        CustomerID => 'mail',
        CustomerUserListFields => ['cn', 'mail'],
        CustomerUserSearchFields => ['uid', 'cn', 'mail'],
        CustomerUserSearchPrefix => '',
        CustomerUserSearchSuffix => '*',
        CustomerUserSearchListLimit => 250,
        CustomerUserPostMasterSearchFields => ['mail'],
        CustomerUserNameFields => ['givenname', 'sn'],
        # Configures the character for joining customer user name parts. Join single␣
↪space if it is not defined.
        CustomerUserNameFieldsJoin => '',
        # show customer user and customer tickets in the external interface
        CustomerUserExcludePrimaryCustomerID => 0,
        # add a ldap filter for valid users (expert setting)
#        # CustomerUserValidFilter => '(!(description=gesperrt))',
        # admin can't change customer preferences
        AdminSetPreferences => 0,
        # cache time to live in sec. - cache any ldap queries
#        CacheTTL => 0,
        Map => [
            # note: Login, Email and CustomerID needed!
            # var, frontend, storage, shown (1=always,2=lite), required, storage-type,
↪ http-link, readonly, http-link-target, link class(es)
            [ 'UserTitle',      Translatable('Title or salutation'), 'title',      ␣
↪      1, 0, 'var', '', 1, undef, undef ],
            [ 'UserFirstname',  Translatable('Firstname'),          'givenname',  ␣
↪      1, 1, 'var', '', 1, undef, undef ],
```

(continues on next page)

(continued from previous page)

```
            [ 'UserLastname',     Translatable('Lastname'),          'sn',          ␣
↪       1, 1, 'var', '', 1, undef, undef ],
            [ 'UserLogin',        Translatable('Username'),          'uid',         ␣
↪       1, 1, 'var', '', 1, undef, undef ],
            [ 'UserEmail',        Translatable('Email'),             'mail',        ␣
↪       1, 1, 'var', '', 1, undef, undef ],
            [ 'UserCustomerID',   Translatable('CustomerID'),        'mail',        ␣
↪       0, 1, 'var', '', 1, undef, undef ],
            # [ 'UserCustomerIDs', Translatable('CustomerIDs'),        'second_
↪customer_ids', 1, 0, 'var', '', 1, undef, undef ],
            [ 'UserPhone',        Translatable('Phone'),             'telephonenumber
↪',     1, 0, 'var', '', 1, undef, undef ],
            [ 'UserAddress',      Translatable('Address'),           'postaladdress',
↪       1, 0, 'var', '', 1, undef, undef ],
            [ 'UserComment',      Translatable('Comment'),           'description', ␣
↪       1, 0, 'var', '', 1, undef, undef ],

            # this is needed, if "SMIME::FetchFromCustomer" is active
            # [ 'UserSMIMECertificate', 'SMIMECertificate', 'userSMIMECertificate', 0,
↪ 1, 'var', '', 1, undef, undef ],

            # Dynamic field example
            # [ 'DynamicField_Name_X', undef, 'Name_X', 0, 0, 'dynamic_field', undef,␣
↪0, undef, undef ],
        ],
    };
```

To activate and configure the LDAP back end:

1. Copy the `$Self->{CustomerUser}` section from `Kernel/Config/Defaults.pm` into `Kernel/Config.pm`.
2. Remove the comments (`#` characters) from the beginning of the lines.

If additional customer user attributes are stored in your LDAP directory, such as a manager name, a mobile phone number, or a department, this information can be displayed in OTRS.

To display additional customer user attributes from LDAP directory:

1. Expand the `Map` array in `Kernel/Config.pm` with the entries for these attributes.

```
[ 'UserMobilePhone', 'Mobile Phone', 'mobilephone', 1, 0, 'var', '', 1, undef,␣
↪undef ],
```

Note: It is recommended to always use English words for names.

See also:

Names can be translated into other languages with custom language files. For more information, see the *Custom Language File* chapter.

Customer User Back End - HTTPBasicAuth

If you want to implement a single sign on solution for all your customer users, you can use HTTPBasic authentication (for all your systems) and use the HTTPBasicAuth module with OTRS. No login is needed with OTRS any more.

```
# This is an example configuration for an apache ($ENV{REMOTE_USER})
# auth. backend. Use it if you want to have a singe login through
# apache http-basic-auth
#   $Self->{'Customer::AuthModule'} = 'Kernel::System::CustomerAuth::HTTPBasicAuth';

# In case there is a leading domain in the REMOTE_USER, you can
# replace it by the next config option.
#   $Self->{'Customer::AuthModule::HTTPBasicAuth::Replace'} = 'example_domain\\';
# Note:
# In case you need to replace some part of the REMOTE_USER, you can
# use the following RegExp ($1 will be new login).
#    $Self->{'Customer::AuthModule::HTTPBasicAuth::ReplaceRegExp'} = '^(.+?)@.+?$';
# Defines a header name, that has to be present for customers to authenticate.
#    $Self->{'Customer::AuthModule::HTTPBasicAuth::RequiredLoginHeader'} =
↪'RequiredHeader';
# Defines a header value, that has to be present in the required header for customers␣
↪to authenticate.
#    $Self->{'Customer::AuthModule::HTTPBasicAuth::RequiredLoginHeaderValue'} =
↪'RequiredHeaderValue';
# Defines a header value regular expression, that has to be present in the required␣
↪header for customers to authenticate.
#    $Self->{'Customer::AuthModule::HTTPBasicAuth::RequiredLoginHeaderValueRegExp'} =
↪'^RequiredHeaderRegExp$';
# If you use this module, you should use as fallback the following
# config settings if user isn't login through apache ($ENV{REMOTE_USER})
#    $Self->{CustomerPanelLoginURL} = 'http://host.example.com/not-authorised-for-
↪otrs.html';
#    $Self->{CustomerPanelLogoutURL} = 'http://host.example.com/thanks-for-using-otrs.
↪html';
```

Customer User Back End - Radius

The settings shown in example below can be used to authenticate your customer users against a Radius server.

```
# This is example configuration to auth. agents against a radius server
#    $Self->{'Customer::AuthModule'} = 'Kernel::System::Auth::Radius';
#    $Self->{'Customer::AuthModule::Radius::Host'} = 'radiushost';
#    $Self->{'Customer::AuthModule::Radius::Password'} = 'radiussecret';
```

4.7.4 Multiple Customer User Back Ends

If you want to use more than one customer user data source, the `CustomerUser` configuration parameter should be expanded with a number, like `CustomerUser1` and `CustomerUser2`.

The following configuration example shows usage of both a database and an LDAP customer user back end.

```
# Data source 1: customer user database back end and settings.
    $Self->{CustomerUser1} = {
        Name   => 'Database Backend',
        Module => 'Kernel::System::CustomerUser::DB',
        Params => {
            DSN => 'DBI:odbc:yourdsn',
            DSN => 'DBI:mysql:database=customerdb;host=customerdbhost',
            User => '',
            Password => '',
            Table => 'customer_user',
        },
        # Other setting here.
    };

# Data source 2: customer user LDAP back end and settings.
    $Self->{CustomerUser2} = {
        Name => 'LDAP Backend',
        Module => 'Kernel::System::CustomerUser::LDAP',
        Params => {
            Host => 'bay.csuhayward.edu',
            BaseDN => 'ou=seas,o=csuh',
            SSCOPE => 'sub',
            UserDN => '',
            UserPw => '',
            AlwaysFilter => '',
            Die => 0,
            Params => {
                port    => 389,
                timeout => 120,
                async   => 0,
                version => 3,
            },
        },
        # Other setting here.
    };
```

It is possible to integrate up to 10 different customer back ends. Use the *Customer Users* screen to view or edit (assuming write access is enabled) all customer user data.

4.7.5 Customer User Data in Dynamic Fields

Sometimes it can be useful to also store customer user data directly in dynamic fields of a ticket, for example to include this data in special statistics later.

The dynamic field values will be set when a ticket is created or when the customer user of a ticket is changed. The values of the dynamic fields are taken from the customer user data. This works for all back ends, but is especially useful for LDAP back ends.

To activate this optional feature:

1. Activate the setting `Ticket::EventModulePost###4100-DynamicFieldFromCustomerUser`.
2. Activate the setting `DynamicFieldFromCustomerUser::Mapping`, and modify its value. This setting should contain the map between customer user field names and names of dynamic fields which will inherit their values.
3. Create the dynamic fields, if the dynamic fields are not present in the system yet.
4. Enable display of the dynamic fields in the *Properties* widget, so you can check their current values easily. You can do this via *following instructions*.

Note: The dynamic fields in question must not be enabled in the following action forms:

- `Forms###AgentFrontend::TicketCreate::Email::CreateProperties`
- `Forms###AgentFrontend::TicketCreate::Phone::CreateProperties`
- `Forms###AgentFrontend::TicketCreate::SMS::CreateProperties`
- `Forms###AgentFrontend::Ticket::Action::Customer`

If they were, the field values from the screen would have precedence over the automatically set values.

4.8 Customer Users   Customers

In an organization, for example, which views its customers as it's departments and teams, a customer user may have to have access to multiple customers tickets for controlling purposes. Maybe you have partners who represent several different companies, or a corporation wants to have a look at all the requests of their subsidiaries.

For all situations, OTRS provides the means. Aside from a primary customer, your customer users can gain access to multiple customer tickets as defined by you.

Use this screen to add one or more customer users to one or more customers. To use this function, at least one customer user and one customer need to have been added to the system. The management screen is available in the *Customer Users   Customers* module of the *Users, Groups & Roles* group.

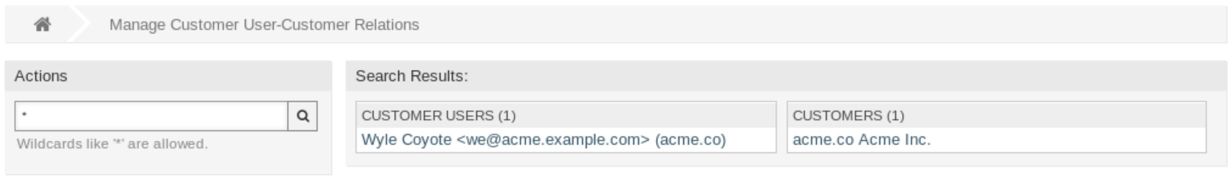

Manage Customer User-Customer Relations
Actions
*
Wildcards like '*' are allowed.
Search Results:
CUSTOMER USERS (1)
Wyle Coyote <we@acme.example.com> (acme.co)
CUSTOMERS (1)
acme.co Acme Inc.

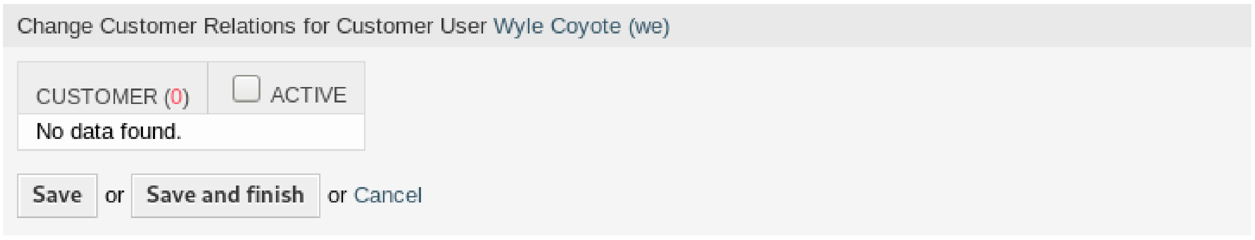
Fig. 28: Manage Customer User-Customer Relations

4.8.1 Manage Customer Users   Customers Relations

Note: This module is for assigning a *customer user* to additional *customer* records. The primary customer is assigned via the *Customer User Settings*.

To assign some customers to a customer user:

1. Click on a customer user in the *Customer Users* column.
2. Select the customers you would like to assign to the customer user.
3. Click on the *Save* or *Save and finish* button.

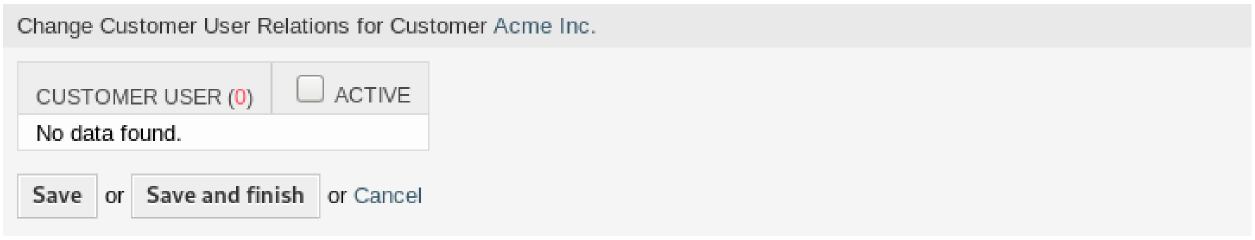

Change Customer Relations for Customer User Wyle Coyote (we)

CUSTOMER (0)	☐ ACTIVE
No data found.	

Fig. 29: Change Customer Relations for Customer User

To assign some customer users to a customer:

1. Click on a customer in the *Customers* column.
2. Select the customer users you would like to assign to the customer.
3. Click on the *Save* or *Save and finish* button.


Change Customer User Relations for Customer Acme Inc.

CUSTOMER USER (0)	☐ ACTIVE
No data found.	

Fig. 30: Change Customer User Relations for Customer

In the previous versions of **OTRS** it was only possible to enable or disable access to company tickets for customer users for every customer user back end via `CustomerUserExcludePrimaryCustomerID` parameter. It was not possible to select the primary customer ID, because it was assumed to be always there anyways, and for the same reason it is still not possible if `CustomerUserExcludePrimaryCustomerID` is disabled for that customer user back end.

In order to allow privilege separation for customer users of a common company, it should be possible to remove access to tickets of the same company and then individually reassign access to specific customer users. Then these customer users have company ticket access while all others do not.

To allow privilege separation for the customer users of the same company:

1. Go to the *System Configuration* screen.
2. Search for the setting `CustomerDisableCompanyTicketAccess` and enable it to make sure not all customer users get access to company tickets until the configuration is finished.

3. Copy the `$Self->{CustomerUser}` section from `Kernel/Config/Defaults.pm` into `Kernel/Config.pm`.
4. Remove the comment (# character) from the beginning of the line contained `CustomerUserExcludePrimaryCustomerID` and set the value to `1`.

```
CustomerUserExcludePrimaryCustomerID => 1,
```

5. Set *Customer Users   Customers* relations for customer users who need to have access to company tickets.
6. Go to the *System Configuration* screen.
7. Search for the setting `CustomerDisableCompanyTicketAccess` and disable it to allow access to company tickets only for customer users configured in step 5.

Note: If several customer users or customers are added to the system, use the search box to find a particular customer user or customer. Only the first 1000 customer users and customers are listed by default.

Multiple customer users or customers can be assigned in both screens at the same time. Additionally clicking on a customer user or clicking on a customer in the relations screen will open the *Edit Customer User* screen or the *Edit Customer* screen accordingly.

Warning: Accessing a customer user or a customer provides no back link to the relations screen.

Note: By setting a checkbox in the header of a column will set all the checkboxes in the selected column.

4.9 Customer Users   Groups

Customer users shouldn't need to be bothered with the internal workings of your service desk. A single point of contact request can trigger several processes within your organization, all of which having the customer user information attached and are visible to the customer.

OTRS allows you to assign *group* permissions to customer users. Access works just the same as for agents, preventing a customer from modifying and viewing a request. Thus allowing the customer to focus on the results of the original communication and funneling the discussion through one ticket.

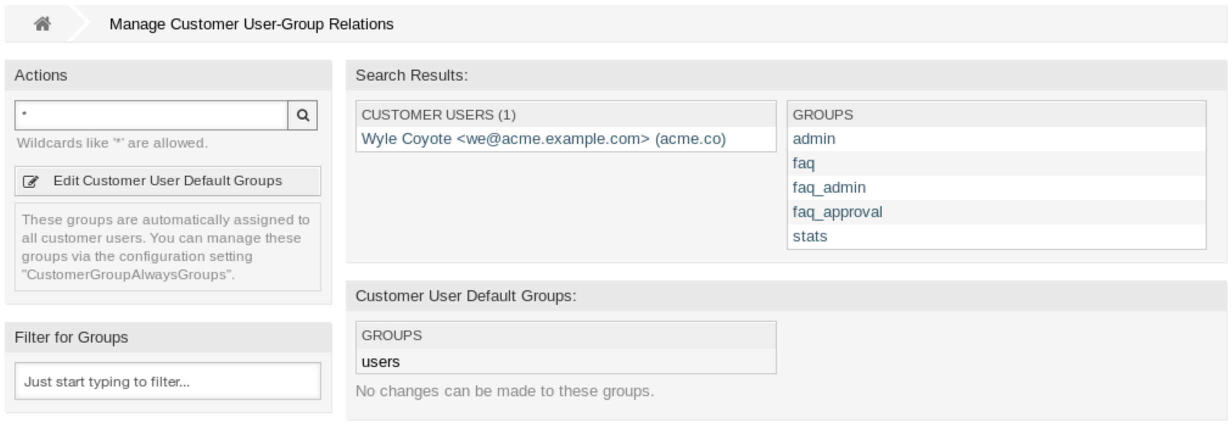
See also:

Assign a group to an entire customer using *Customers   Groups*.

Use this screen to add one or more customer users to one or more groups. To use this function, at least one customer user and one group need to have been added to the system. The management screen is available in the *Customers Users   Groups* module of the *Users, Groups & Roles* group.

Customer group support needs to be enabled in at least one customer user *back end* to use this function. For the default OTRS *back end*, this can be enabled in the system configuration by clicking on the *Enable it here!* button.

Note: To enable this feature in systems using a directory server or multiple non-default back ends, a custom configuration file needs to be placed in `Kernel/Config/Files` (for example named


Manage Customer User-Group Relations
Actions
Wildcards like '*' are allowed.
Edit Customer User Default Groups
Filter for Groups
Just start typing to filter...
Search Results:
Customer User Default Groups:

These groups are automatically assigned to all customer users. You can manage these groups via the configuration setting "CustomerGroupAlwaysGroups".

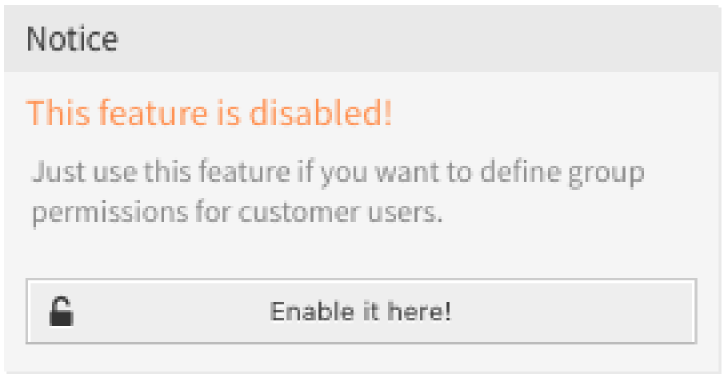
CUSTOMER USERS (1)
Wyle Coyote <we@acme.example.com> (acme.co)

GROUPS
admin
faq
faq_admin
faq_approval
stats

GROUPS
users

No changes can be made to these groups.

Fig. 31: Manage Customer User-Group Relations



This feature is disabled!

Just use this feature if you want to define group permissions for customer users.

Fig. 32: Enable Customer Group Feature

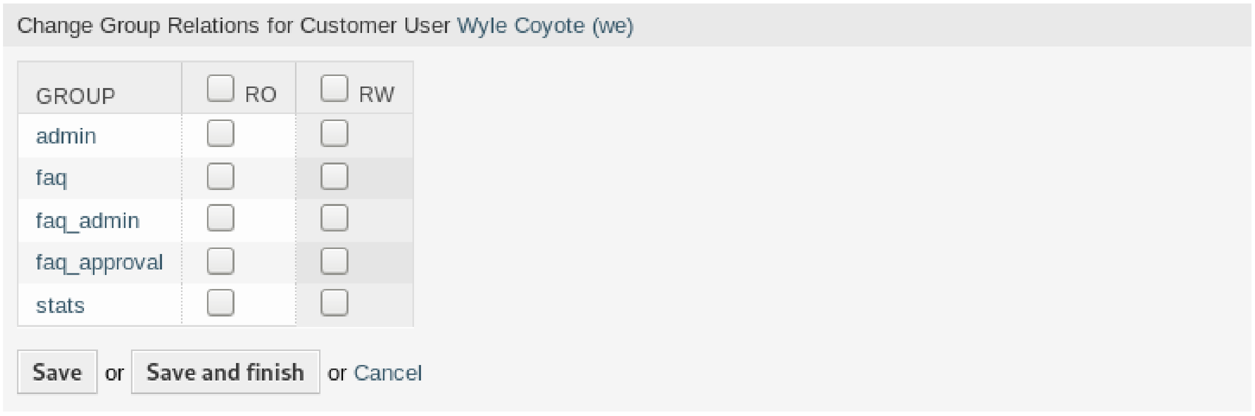
ZZZ_CustomerBackend.pm). Once activated, all customer users from this back end will require group assignment.

Warning: After making changes to the back end, the server cache will be deleted, which may cause a temporary drop in performance.

4.9.1 Manage Customer Users Groups Relations

To assign some groups to a customer user:

1. Click on a customer user in the *Customer Users* column.
2. Select the permissions you would like to connect the customer user to groups with.
3. Click on the *Save* or *Save and finish* button.



Change Group Relations for Customer User Wyle Coyote (we)

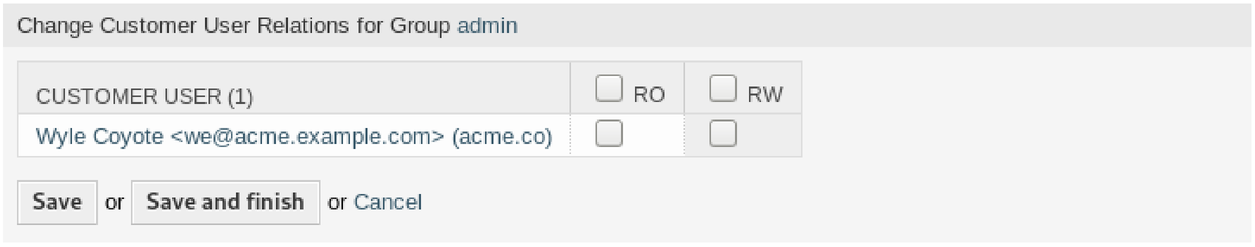
GROUP	☐ RO	☐ RW
admin	☐	☐
faq	☐	☐
faq_admin	☐	☐
faq_approval	☐	☐
stats	☐	☐

Save or Save and finish or Cancel

Fig. 33: Change Group Relations for Customer User

To assign some customer users to a group:

1. Click on a group in the *Groups* column.
2. Select the permissions you would like to connect the group to customer users with.
3. Click on the *Save* or *Save and finish* button.



Change Customer User Relations for Group admin

CUSTOMER USER (1)	☐ RO	☐ RW
Wyle Coyote <we@acme.example.com> (acme.co)	☐	☐

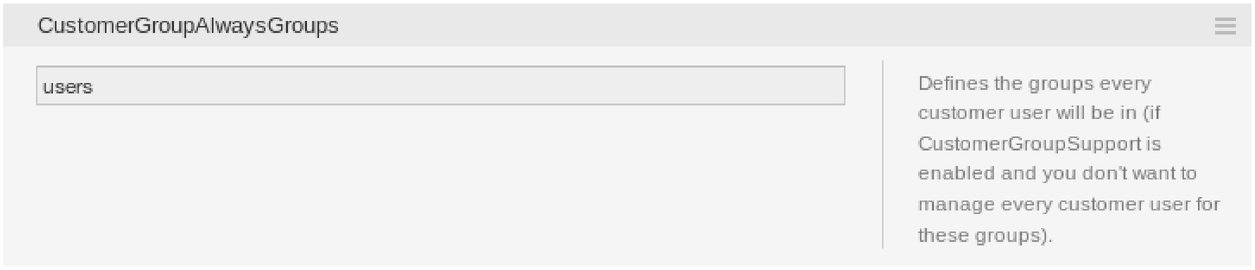
Save or Save and finish or Cancel

Fig. 34: Change Customer User Relations for Group

To change customer user default groups:

1. Click on the *Edit Customer User Default Groups* button in the left sidebar.

2. Add or modify groups in setting CustomerGroupAlwaysGroups.
3. Deploy the modified system configurations.



CustomerGroupAlwaysGroups

users

Defines the groups every customer user will be in (if CustomerGroupSupport is enabled and you don't want to manage every customer user for these groups).

Fig. 35: `CustomerGroupAlwaysGroups` System Configuration Screen

These groups are automatically assigned to all customer users.

Note: If several customer users or groups are added to the system, use the search box to find a particular customer user or use the filter box to find a particular group by just typing the name to filter.

Multiple customer users or groups can be assigned in both screens at the same time. Additionally clicking on a customer user or clicking on a group in the relations screen will open the *Edit Customer User* screen or the *Edit Group* screen accordingly.

Warning: Accessing a customer user or a group provides no back link to the relations screen.

4.9.2 Customer Users   Groups Relations Reference

When assigning a customer user to a group or vice versa, several permissions can be set as connection between a customer user and a group. The following permissions are available by default:

ro
Read only access to the resource.

rw
Full read and write access to the resource.

See also:

Not all available permissions are shown by default. See System::Customer::Permission setting for permissions that can be added. This additional permission can be added:

create
Permission to create a ticket.

Note: By setting a checkbox in the header of a column will set all the checkboxes in the selected column. By setting the checkbox in the last *rw* column will set all the checkboxes in the selected row.

4.10 Customer Users   Services

Use this screen to add one or more customer users to one or more services. To use this function, at least one customer user and one service need to have been added to the system. The management screen is available in the *Customers Users   Services* module of the *Users, Groups & Roles* group.

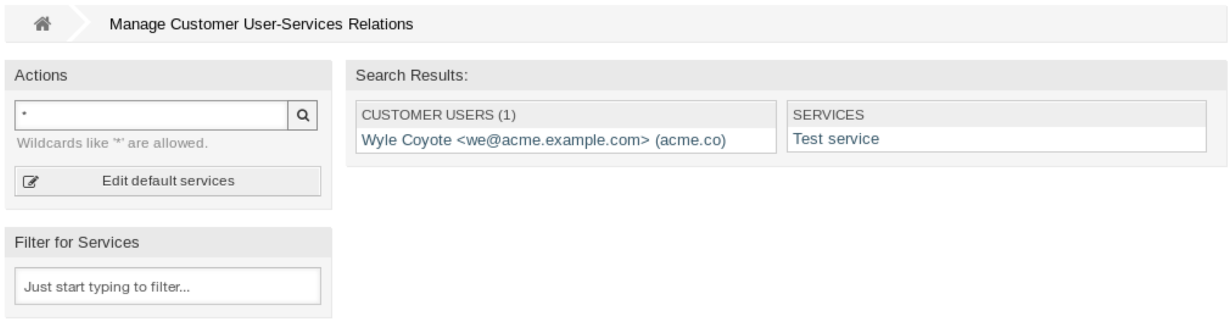

Manage Customer User-Services Relations
Actions
Wildcards like '*' are allowed.
Edit default services
Filter for Services
Just start typing to filter...
Search Results:
CUSTOMER USERS (1)
Wyle Coyote <we@acme.example.com> (acme.co)
SERVICES
Test service

Fig. 36: Manage Customer User-Service Relations

4.10.1 Manage Customers Users   Services Relations

To allocate some services to a customer user:

1. Click on a customer user in the *Customer Users* column.
2. Select the services you would like to allocate to the customer user.
3. Click on the *Save* or *Save and finish* button.

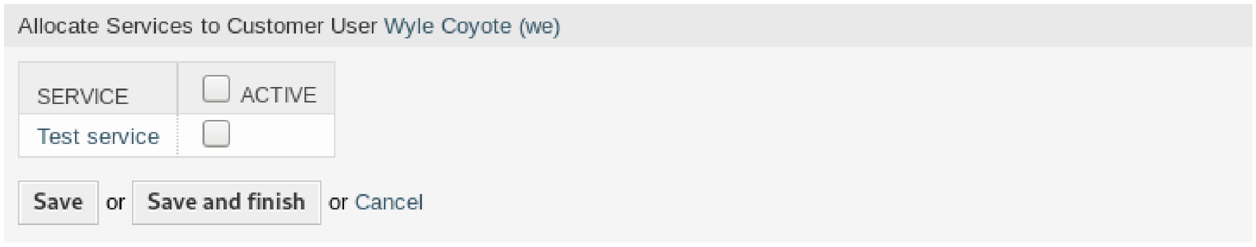
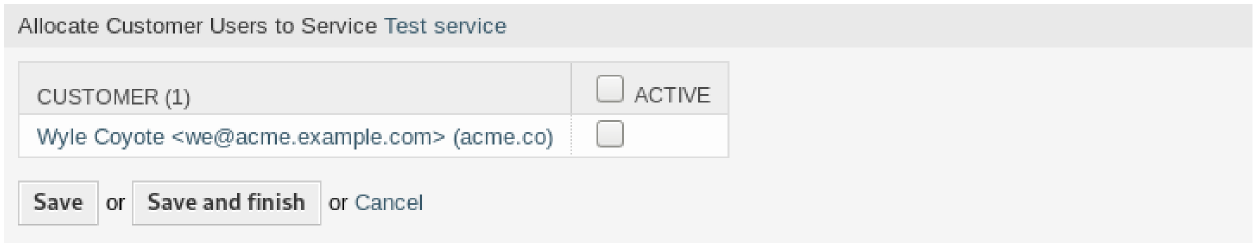


Fig. 37: Change Service Relations for Customer User

To allocate some customer users to a service:

1. Click on a service in the *Services* column.
2. Select the customer users you would like to allocate to the service.
3. Click on the *Save* or *Save and finish* button.

Note: If several customer users or services are added to the system, use the search box to find a particular customer user or use the filter box to find a particular service by just typing the name to filter.


Allocate Customer Users to Service Test service

CUSTOMER (1)	ACTIVE
Wyle Coyote <we@acme.example.com> (acme.co)	☐

Fig. 38: Change Customer User Relations for Service

Multiple customer users or services can be assigned in both screens at the same time. Additionally clicking on a customer user or clicking on a service in the relations screen will open the *Edit Customer User* screen or the *Edit Service* screen accordingly.

Warning: Accessing a customer user or a service provides no back link to the relations screen.

Note: By setting a checkbox in the header of a column will set all the checkboxes in the selected column.

4.10.2 Manage Default Services

It is possible to add default services, so that all customer users may access them. This prevents having to add each service to each customer user.

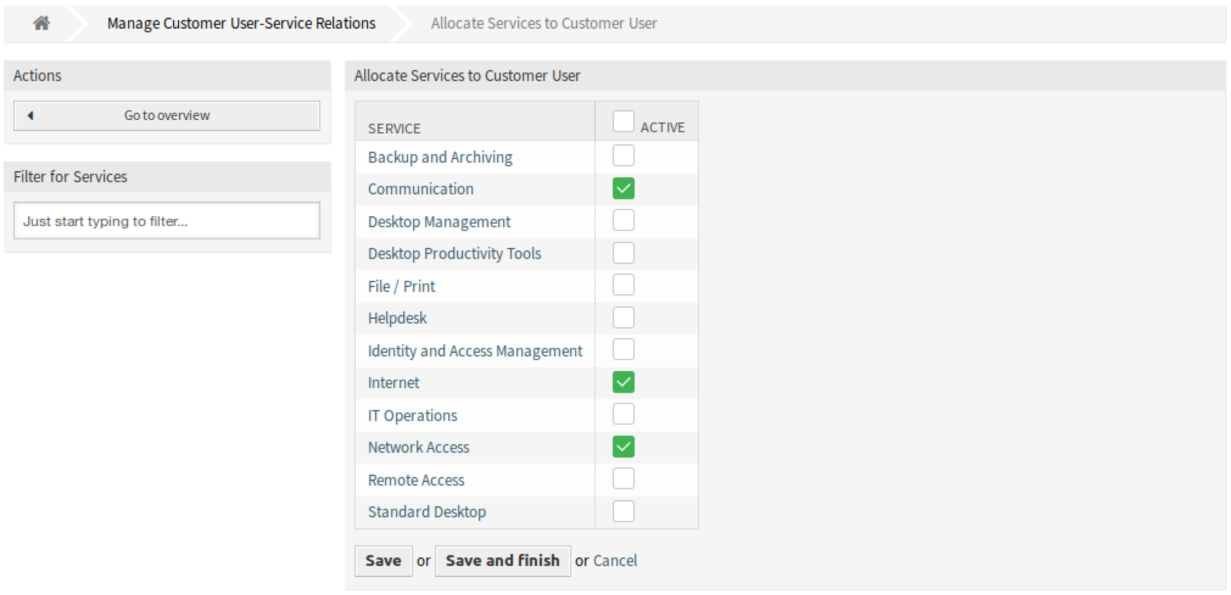
To edit the default services:

1. Click on the *Edit default services* button in the left sidebar.
2. Select the services which should be selectable for all customer users.
3. Click on the *Save* or *Save and finish* button.

Warning: Mixing default services and customer specific services can be confusing. If a customer user has specific services assigned, then the default services will be not applied.

4.11 Dynamic Ticket Templates   Groups

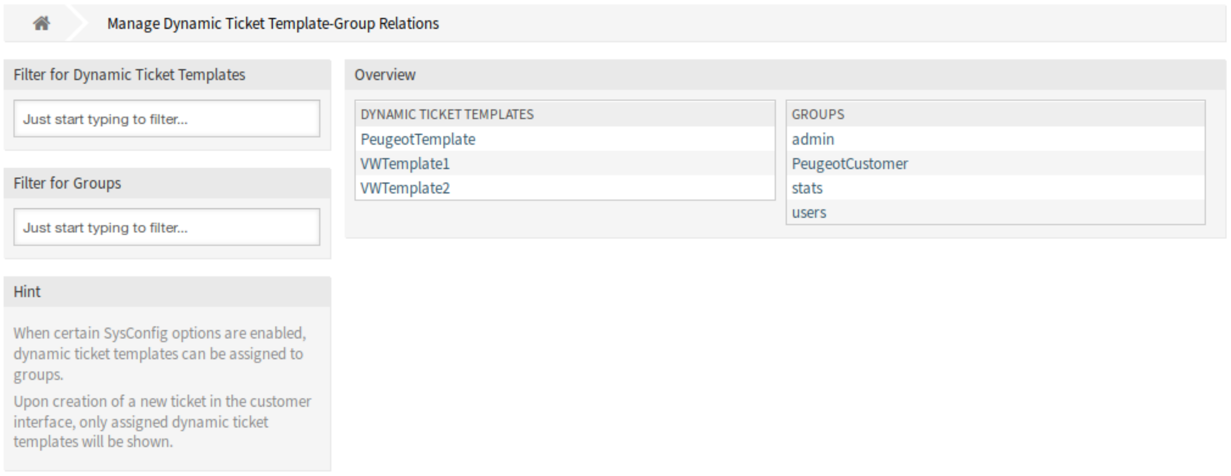
Use this screen to add one or more dynamic ticket templates to one or more groups. To use this function, at least one dynamic ticket template and one group need to have been added to the system. The management screen is available in the *Dynamic Ticket Templates   Groups* module of the *Users, Groups & Roles* group.


Manage Customer User-Service Relations
Allocate Services to Customer User
Actions
Go to overview
Filter for Services
Just start typing to filter...
Save
or
Save and finish
or Cancel

Allocate Services to Customer User

SERVICE	ACTIVE
Backup and Archiving	
Communication	✓
Desktop Management	
Desktop Productivity Tools	
File / Print	
Helpdesk	
Identity and Access Management	
Internet	✓
IT Operations	
Network Access	✓
Remote Access	
Standard Desktop	

Fig. 39: Allocate Services to Customer User Screen


Manage Dynamic Ticket Template-Group Relations
Filter for Dynamic Ticket Templates
Just start typing to filter...
Filter for Groups
Just start typing to filter...
Overview

Hint

When certain SysConfig options are enabled, dynamic ticket templates can be assigned to groups.

Upon creation of a new ticket in the customer interface, only assigned dynamic ticket templates will be shown.

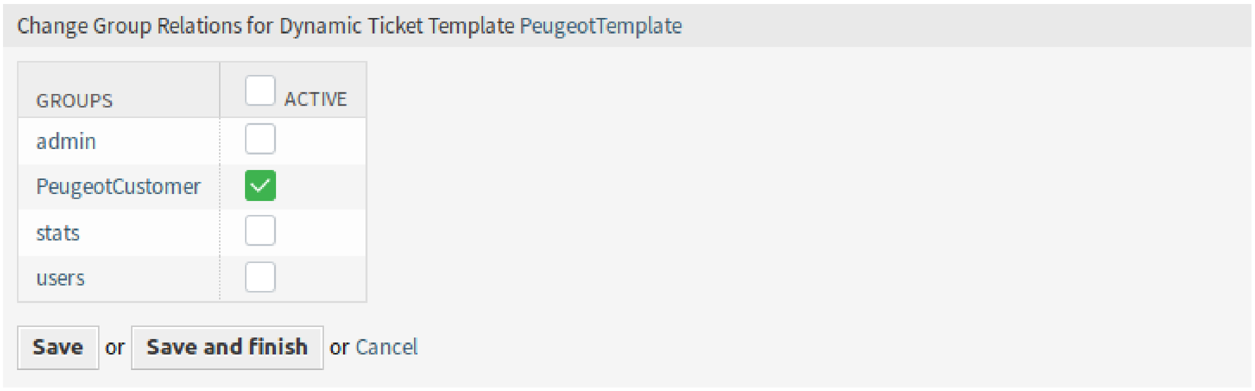
DYNAMIC TICKET TEMPLATES	GROUPS
PeugeotTemplate	admin
VWTemplate1	PeugeotCustomer
VWTemplate2	stats
	users

Fig. 40: Manage Dynamic Ticket Template-Group Relations

4.11.1 Manage Dynamic Ticket Templates Groups Relations

To assign some groups to a dynamic ticket template:

1. Click on a dynamic ticket template in the *Dynamic Ticket Templates* column.
2. Select the groups you would like to assign the dynamic ticket template to.
3. Click on the *Save* or *Save and finish* button.

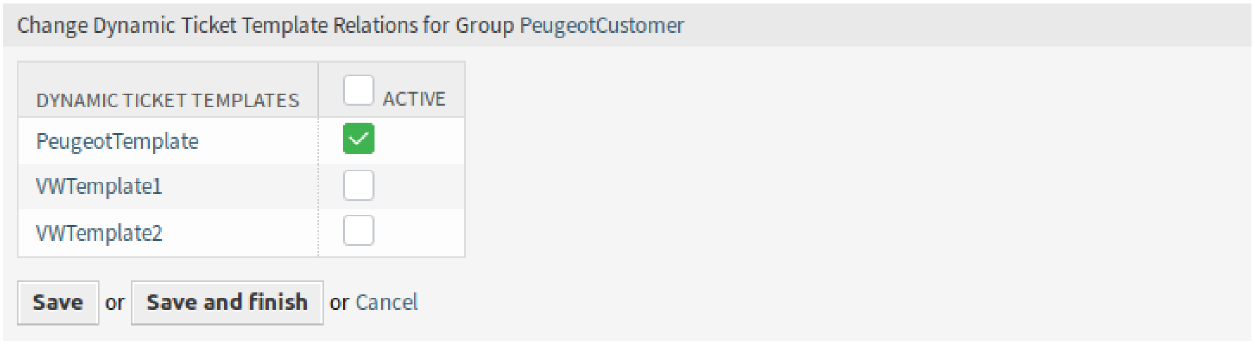

Change Group Relations for Dynamic Ticket Template PeugeotTemplate

GROUPS	☐ ACTIVE
admin	☐
PeugeotCustomer	✓
stats	☐
users	☐

Fig. 41: Change Group Relations for Dynamic Ticket Template

To assign some dynamic ticket templates to a group:

1. Click on a group in the *Groups* column.
2. Select the dynamic ticket templates you would like to assign the group to.
3. Click on the *Save* or *Save and finish* button.


Change Dynamic Ticket Template Relations for Group PeugeotCustomer

DYNAMIC TICKET TEMPLATES	☐ ACTIVE
PeugeotTemplate	✓
VWTemplate1	☐
VWTemplate2	☐

Fig. 42: Change Dynamic Ticket Template Relations for Group

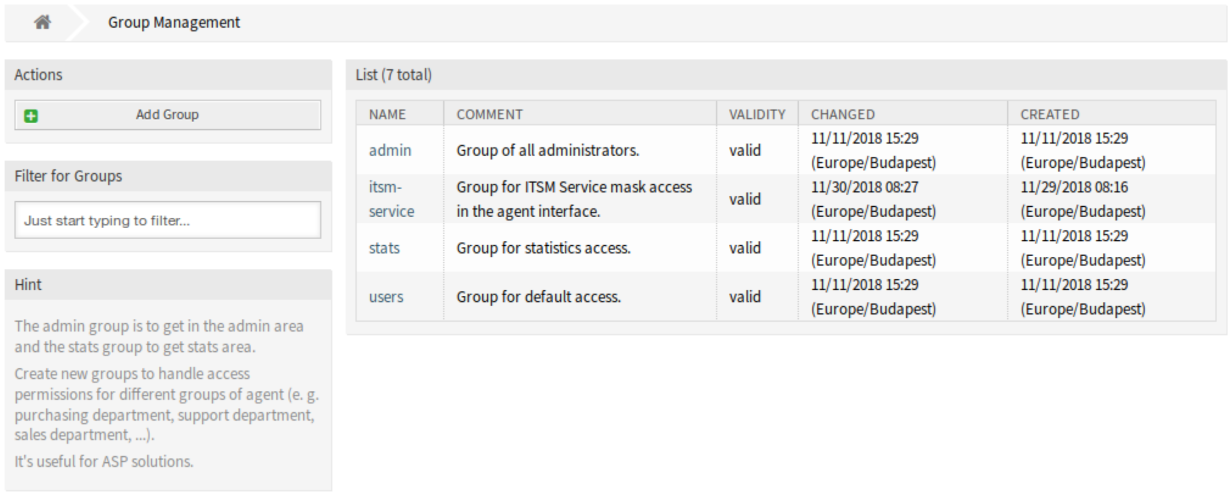
Note: If several dynamic ticket templates or groups are added to the system, use the filter box to find a particular dynamic ticket template or group by just typing the name to filter.

Multiple dynamic ticket templates or groups can be assigned in both screens at the same time. Additionally clicking on a dynamic ticket template or clicking on a group in the relations screen will open the *Edit Template* screen or the *Edit Group* screen accordingly.

Warning: Accessing a dynamic ticket template or a group provides no back link to the relations screen.

4.12 Groups

Use this screen to add groups to the system. A fresh OTRS installation contains some default groups. The group management screen is available in the *Groups* module of the *Users, Groups & Roles* group.



Group Management

Actions

Add Group

Filter for Groups

Just start typing to filter...

Hint

The admin group is to get in the admin area and the stats group to get stats area.

Create new groups to handle access permissions for different groups of agent (e. g. purchasing department, support department, sales department, ...).

It's useful for ASP solutions.

List (7 total)

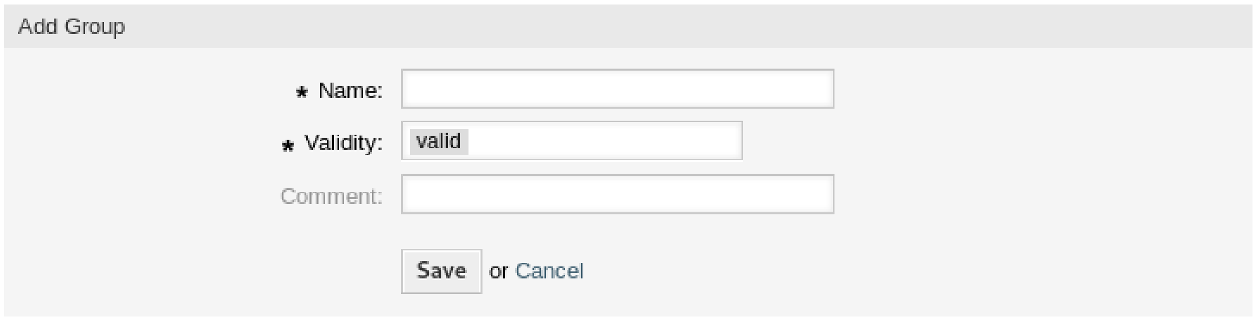
NAME	COMMENT	VALIDITY	CHANGED	CREATED
admin	Group of all administrators.	valid	11/11/2018 15:29 (Europe/Budapest)	11/11/2018 15:29 (Europe/Budapest)
itsm-service	Group for ITSM Service mask access in the agent interface.	valid	11/30/2018 08:27 (Europe/Budapest)	11/29/2018 08:16 (Europe/Budapest)
stats	Group for statistics access.	valid	11/11/2018 15:29 (Europe/Budapest)	11/11/2018 15:29 (Europe/Budapest)
users	Group for default access.	valid	11/11/2018 15:29 (Europe/Budapest)	11/11/2018 15:29 (Europe/Budapest)

Fig. 43: Group Management Screen

4.12.1 Manage Groups

To add a group:

1. Click on the *Add Group* button in the left sidebar.
2. Fill in the required fields.
3. Click on the *Save* button.



Add Group

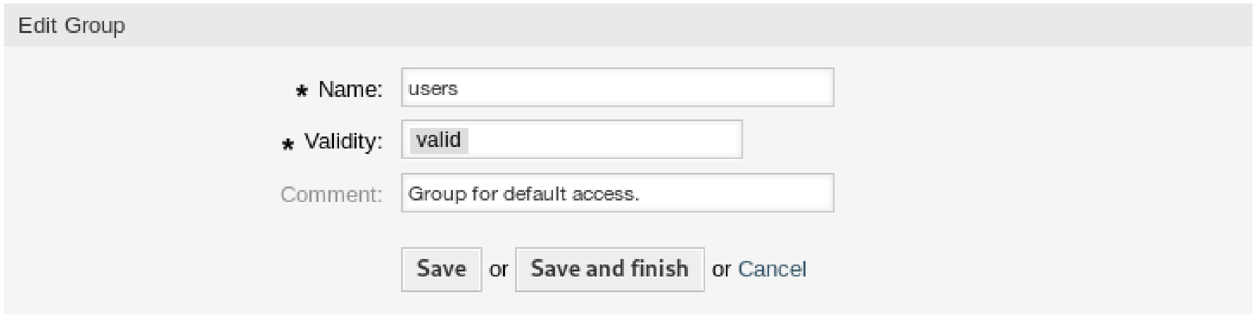
* Name:	
* Validity:	valid
Comment:	

Fig. 44: Add Group Screen

Warning: Groups can not be deleted from the system. They can only be deactivated by setting the *Validity* option to *invalid* or *invalid-temporarily*.

To edit a group:

1. Click on a group in the list of groups.
2. Modify the fields.
3. Click on the *Save* or *Save and finish* button.


Edit Group

* Name:	users
* Validity:	valid
Comment:	Group for default access.

Fig. 45: Edit Group Screen

Note: If several groups are added to the system, use the filter box to find a particular group by just typing the name to filter.

4.12.2 Group Settings

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

Name *
The name of this resource. Any type of characters can be entered to this field including uppercase letters and spaces. The name will be displayed in the overview table.

Note: Renaming a group does not affect permissions previously given. When *group1* is now called *group2*, then all the permissions are the same for the users which used to be assigned to *group1*. This result is because OTRS uses IDs for the relationship, and not the name.

Validity *
Set the validity of this resource. Each resource can be used in OTRS only, if this field is set to *valid*. Setting this field to *invalid* or *invalid-temporarily* will disable the use of the resource.

Note: Invalidating a group does not remove the permissions from the user, but only makes them invalid. If you reactivate this group, even with a new name, the permissions take effect.

Comment
Add additional information to this resource. It is recommended to always fill this field as a description

of the resource with a full sentence for better clarity, because the comment will be also displayed in the overview table.

4.12.3 Default Groups

Every agent' s account should belong to at least one group or role. In a fresh installation, there are some pre-defined groups available:

`admin`
Allowed to perform administrative tasks in the system.

`itsm-service`
Group for accessing the *Service Management* screens of the agent interface.

`stats`
Qualified to access the stats module of OTRS and generate statistics.

`users`
Agents should belong to this group, with read and write permissions. They can then access all functions of the ticket system.

Note: The primary administrator user (root@localhost) is added to the groups with permission *rw* by default.

See also:

To set the correct permissions for other users, check the following relations:

- *Agents Groups*
- *Customers Groups*
- *Customer Users Groups*
- *Roles Groups*

4.13 Roles

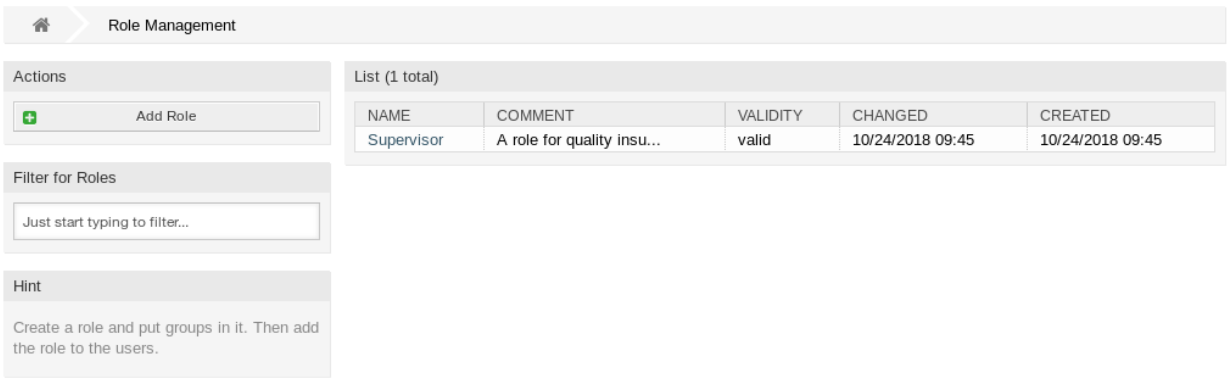
Use this screen to add roles to the system. A fresh OTRS installation contains no roles by default. The role management screen is available in the *Roles* module of the *Users, Groups & Roles* group.

4.13.1 Manage Roles

To add a role:

1. Click on the *Add Role* button in the left sidebar.
2. Fill in the required fields.
3. Click on the *Save* button.

Warning: Roles can not be deleted from the system. They can only be deactivated by setting the *Validity* option to *invalid* or *invalid-temporarily*.

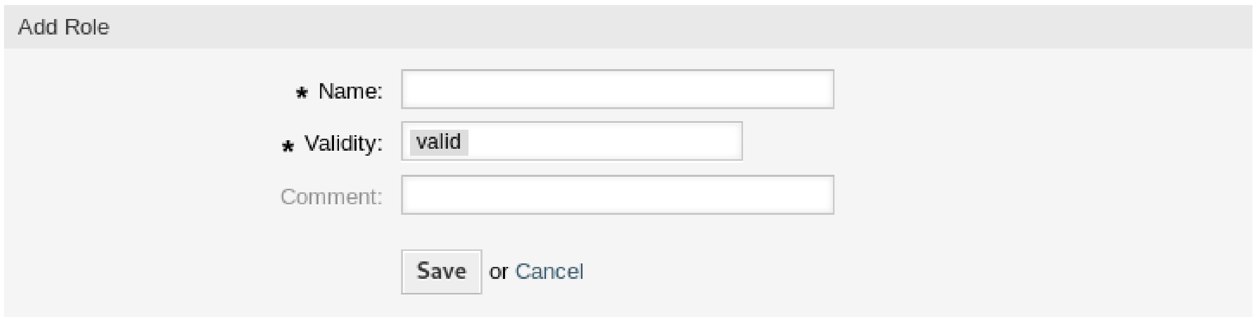


Create a role and put groups in it. Then add the role to the users.

List (1 total)

NAME	COMMENT	VALIDITY	CHANGED	CREATED
Supervisor	A role for quality insu...	valid	10/24/2018 09:45	10/24/2018 09:45

Fig. 46: Role Management Screen

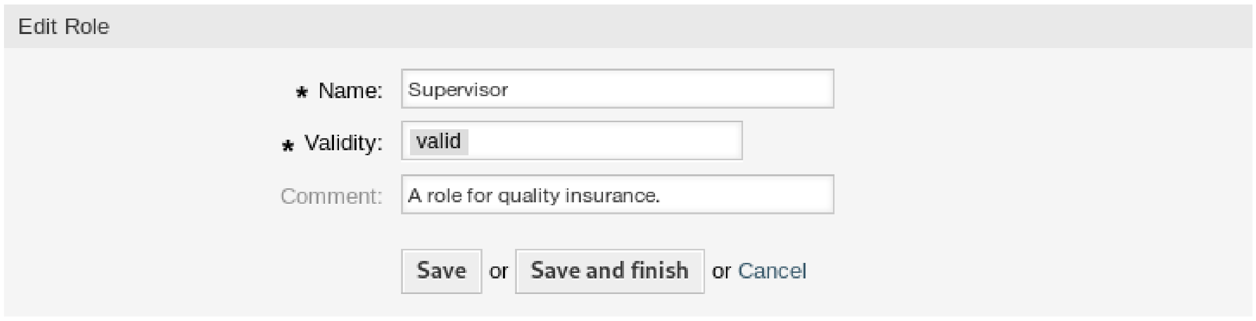


* Name:	
* Validity:	valid
Comment:	

Fig. 47: Add Role Screen

To edit a role:

1. Click on a role in the list of roles.
2. Modify the fields.
3. Click on the *Save* or *Save and finish* button.


Edit Role

* Name:	Supervisor
* Validity:	valid
Comment:	A role for quality insurance.

Fig. 48: Edit Role Screen

Note: If several roles are added to the system, use the filter box to find a particular role by just typing the name to filter.

4.13.2 Role Settings

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

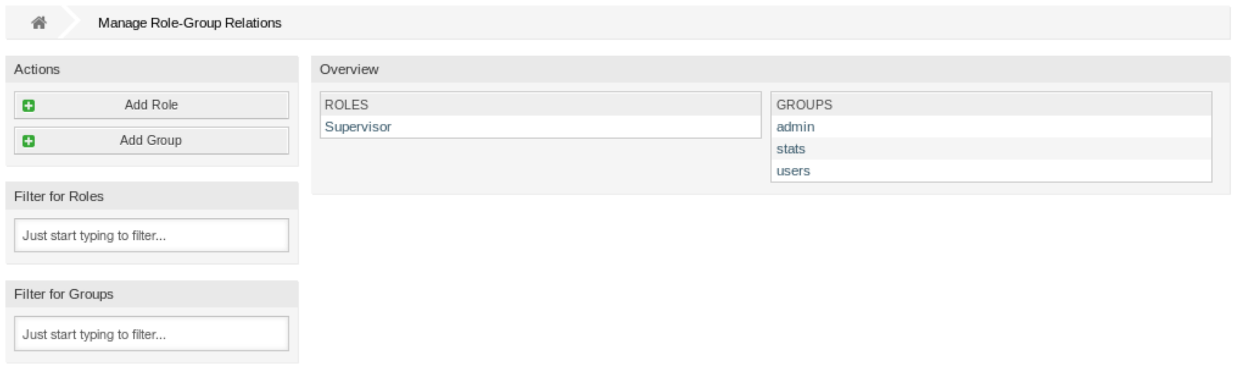
Name *
: The name of this resource. Any type of characters can be entered to this field including uppercase letters and spaces. The name will be displayed in the overview table.

Validity *
: Set the validity of this resource. Each resource can be used in OTRS only, if this field is set to *valid*. Setting this field to *invalid* or *invalid-temporarily* will disable the use of the resource.

Comment
: Add additional information to this resource. It is recommended to always fill this field as a description of the resource with a full sentence for better clarity, because the comment will be also displayed in the overview table.

4.14 Roles Groups

Use this screen to add one or more roles to one or more groups. To use this function, at least one role and one group need to have been added to the system. The management screen is available in the *Roles Groups* module of the *Users, Groups & Roles* group.


Manage Role-Group Relations
Actions
Add Role
Add Group
Filter for Roles
Just start typing to filter...
Filter for Groups
Just start typing to filter...
Overview

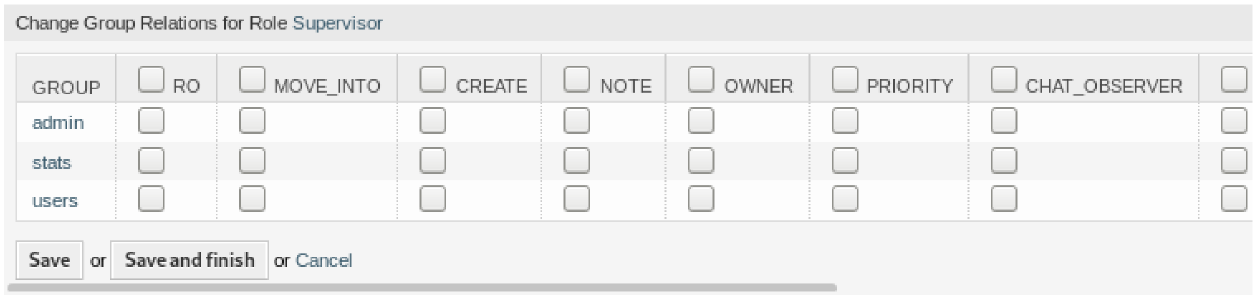
ROLES	GROUPS
Supervisor	admin
	stats
	users

Fig. 49: Manage Role-Group Relations

4.14.1 Manage Roles Groups Relations

To assign some groups to a role:

1. Click on a role in the *Roles* column.
2. Select the permissions you would like to connect the role to groups with.
3. Click on the *Save* or *Save and finish* button.



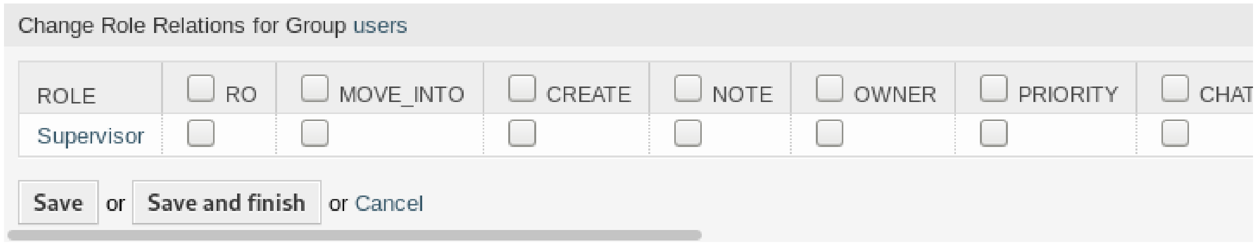
Change Group Relations for Role Supervisor

GROUP	☐ RO	☐ MOVE_INTO	☐ CREATE	☐ NOTE	☐ OWNER	☐ PRIORITY	☐ CHAT_OBSERVER	☐
admin	☐	☐	☐	☐	☐	☐	☐	☐
stats	☐	☐	☐	☐	☐	☐	☐	☐
users	☐	☐	☐	☐	☐	☐	☐	☐

Fig. 50: Change Group Relations for Role

To assign some roles to a group:

1. Click on a group in the *Groups* column.
2. Select the permissions you would like to connect the group to roles with.
3. Click on the *Save* or *Save and finish* button.



Change Role Relations for Group users

ROLE	☐ RO	☐ MOVE_INTO	☐ CREATE	☐ NOTE	☐ OWNER	☐ PRIORITY	☐ CHAT
Supervisor	☐	☐	☐	☐	☐	☐	☐

Fig. 51: Change Role Relations for Group

Note: If several roles or groups are added to the system, use the filter box to find a particular role or group by just typing the name to filter.

Multiple roles or groups can be assigned in both screens at the same time. Additionally clicking on a role or clicking on a group in the relations screen will open the *Edit Role* screen or the *Edit Group* screen accordingly.

Warning: Accessing a role or a group provides no back link to the relations screen.

4.14.2 Roles Groups Relations Reference

When assigning a role to a group or vice versa, several permissions can be set as connection between a role and a group. The following permissions are available by default:

ro
Read-only access to tickets in this group/queue.

move_into
Permission to move tickets in this group/queue and move existing tickets into this group/queue.

create
Permission to create tickets in this group/queue.

note
Permission to add notes to tickets and inform agents in this group/queue.

owner
Permission to set the owner of new tickets or change the owner of existing tickets in this group/queue.

priority
Permission to change the ticket priority in this group/queue.

chat_observer
Users with this permission type will only be able to observe chats in a channel after they have been invited.

chat_participant
Users with this permission type will be able to take part in a chat, but only after they get invited to it.

chat_owner
Users with this permission type will be able to accept chat customer/public requests and do all kinds of observer and participant actions on a chat.

rw
Full read and write access to the tickets in this group/queue.

See also:

Not all available permissions are shown by default. See System::Permission setting for permissions that can be added. These additional permissions can be added:

bounce
Permission to redirect an email.

close
Permission to close a ticket.

compose
Permission to compose an answer for a ticket.

customer
Permission to change the customer of a ticket.

forward
Permission to forward an article.

pending
Permission to set a ticket to pending.

phone
Permission to add a phone call to a ticket.

responsible
Permission to set the responsible agent of new tickets or change the responsible agent of existing tickets in this group/queue.

Note: By setting a checkbox in the header of a column will set all the checkboxes in the selected column. By setting the checkbox in the last *rw* column will set all the checkboxes in the selected row.

CHAPTER

FIVE

PROCESSES & AUTOMATION

Working effectively with tickets requires more than just the possibility to manually change their state, add information, communicate with other persons and finally close the tickets.

Automation frees agents from reoccurring and time-consuming tasks and allows them to focus on activities, where their interaction is required.

Process management guides customer users and agents through ticket creation to closure ensuring that tickets take defined workflows at any time.

OTRS offers many options to automate tasks based on events, time, external systems and defined processes. OTRS also includes the possibility to add individual information types to tickets and help agents to lower their error rate when working with tickets by allowing only defined activities for tickets in specific states.

5.1 Access Control Lists (ACL)

Working with tickets can become a bewildering task. Many options are given to process, or close tickets, even if they are not needed in the current state of a ticket or due to the role of the current agent. Hiding unneeded entries cleans up the menu bar and gets it easier to work with, hiding values from dynamic fields or next queues lowers chance of human error.

OTRS uses access control lists (ACL) to restrict agents and customer users on ticket options, allowing only correct and meaningful activities with a ticket. OTRS administrators can easily generate ACLs in the graphical interface to prevent ticket closure until meeting specific requirements, prevent tickets from being moved to queues before adding the defined information and much more.

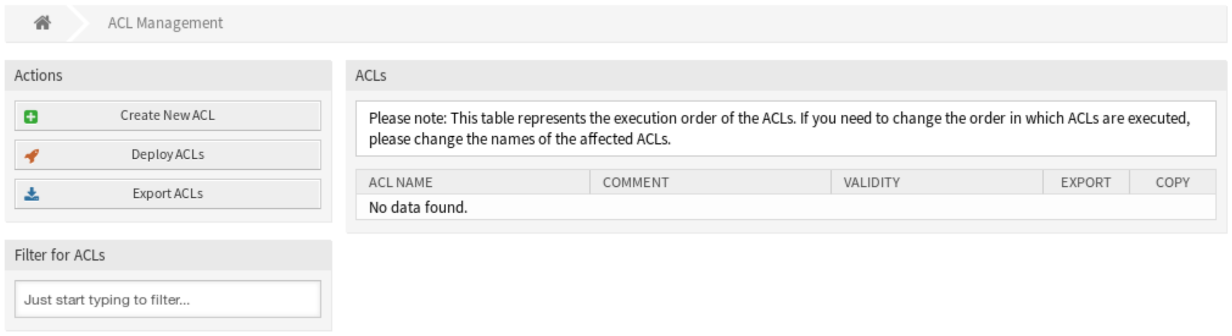
Use this screen to manage access control lists in the system. A fresh OTRS installation contains no access control lists by default. The access control lists management screen is available in the *Access Control Lists (ACL)* module of the *Processes & Automation* group.

5.1.1 Manage Access Control Lists

Note: When creating some access control lists, please keep in mind that they are executed alphabetically as displayed in the access control lists overview.

Warning: ACL restrictions will be ignored for the superuser account (UserID 1).

To create a new ACL:



Please note: This table represents the execution order of the ACLs. If you need to change the order in which ACLs are executed, please change the names of the affected ACLs.

ACL NAME	COMMENT	VALIDITY	EXPORT	COPY
No data found.				

Fig. 1: ACL Management Screen

1. Click on the *Create New ACL* button in the left sidebar.
2. Fill in the required fields.
3. Click on the *Save* button.
4. You will be redirected to *Edit ACL* screen to edit the ACL structure.

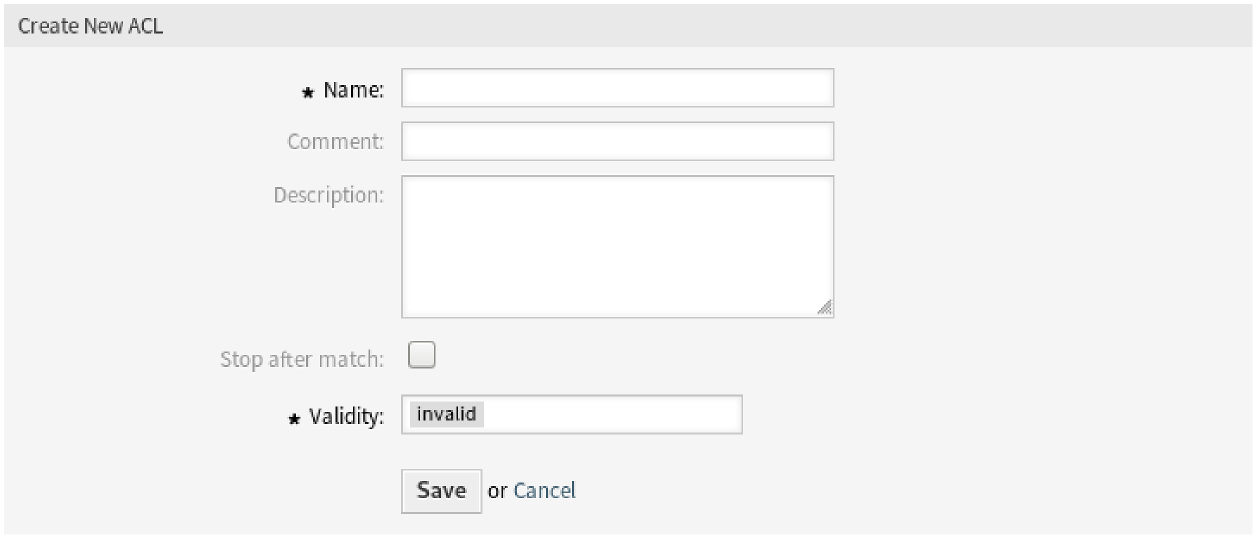


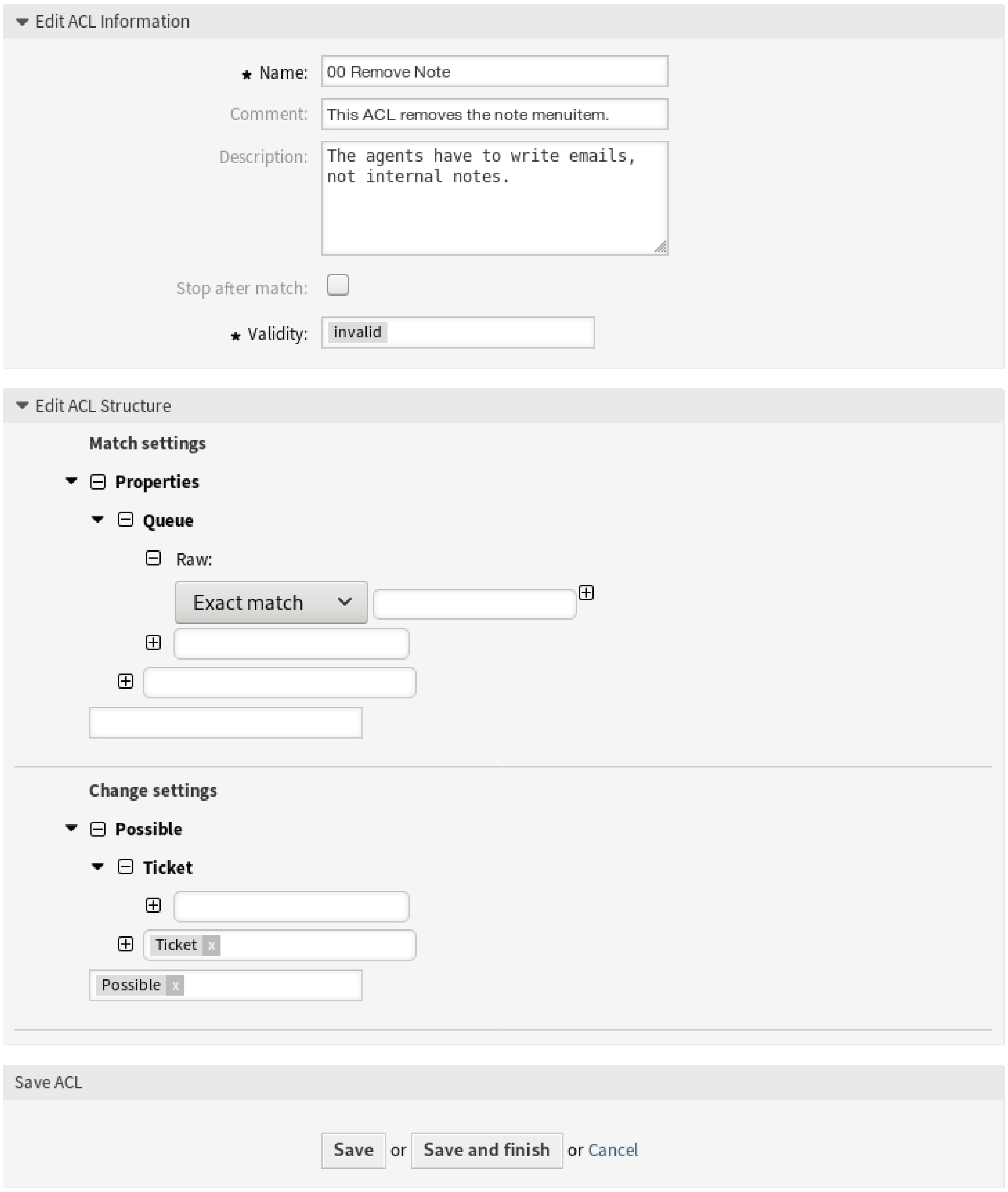
Fig. 2: Create New ACL Screen

To edit an ACL:

1. Click on an ACL in the list of ACLs or you are already redirected here from *Create New ACL* screen.
2. Modify the fields and the ACL structure.
3. Click on the *Save* or *Save and finish* button.
4. Deploy all ACLs.

To delete an ACL:

1. Click on an ACL in the list of ACLs.
2. Set the *Validity* option to *invalid* or *invalid-temporarily*.
3. Click on the *Save* button. A new *Delete Invalid ACL* button will appear in the left sidebar.



Name: 00 Remove Note

Comment: This ACL removes the note menuitem.

Description: The agents have to write emails, not internal notes.

Stop after match: ☐

Validity: invalid

Match settings

Properties

Queue

Raw:

Exact match

Change settings

Possible

Ticket

Ticket

Possible

Fig. 3: Edit ACL Structure Screen

4. Click on the *Delete Invalid ACL* button.
5. Click on the *Delete* button in the confirmation screen.
6. Deploy all ACLs.

Warning: ACLs are written into `ZZZACL.pm` file in Perl format. Without deploying, all ACLs are still in this cache file even if they are deleted or the *Validity* option is set to *invalid* or *invalid-temporarily*. Don't forget to deploy all ACLs after modifications!

To deploy all ACLs:

1. Click on the *Deploy ACLs* button in the left sidebar.

Note: New or modified ACLs have to deploy in order to make affect the behavior of the system. Setting the *Validity* option to *valid* just indicates, which ACLs should be deployed.

To export all ACLs:

1. Click on the *Export ACLs* button in the left sidebar.
2. Choose a location in your computer to save the `Export_ACL.yml` file.

To import ACLs:

1. Click on the *Browse…* button in the left sidebar.
2. Select a previously exported `.yml` file.
3. Click on the *Overwrite existing ACLs?* checkbox, if you would like to overwrite the existing ACLs.
4. Click on the *Import ACL configuration(s)* button.
5. Deploy the imported ACLs with *Deploy ACLs* button.

Note: If several ACLs are added to the system, use the filter box to find a particular ACL by just typing the name to filter.

Warning: The maximum number of 80 *valid* ACLs should not be exceeded. Exceeding this limit may affect the system performance.

Warning: Changing the name of this object should be done with care, the check only provides verification for certain settings and ignores things where the name can't be verified. Some examples are dashboard filters, access control lists (ACLs), and processes (sequence flow actions) to name a few. Documentation of your setup is key to surviving a name change.

5.1.2 Possible Data Loss

Warning: If a drop-down field has a value that is forbidden by ACL, the stored value in a ticket will be changed or removed after the form is submitted. This can cause possible data loss!

Here is an example to explain the possible problems:

A drop-down dynamic field is created with four possible values: *BRONZE*, *SILVER*, *GOLD* and *VIP*. Empty value also allowed. The agent can select *BRONZE*, *SILVER* and *GOLD* only. The *VIP* value can be set only by the generic agent. This is restricted by an ACL. The dynamic field is added to some ticket screens. In a screen the field is set as mandatory but in another screen the field is not mandatory and empty value is allowed.

1. The agent creates a new ticket. The agent can select only the allowed values, the *VIP* value is not displayed. *SILVER* is selected and the ticket is created.
2. The generic agent changes the value to *VIP*.
3. The agent opens a ticket action where the field is added as mandatory. Since the field is mandatory the agent has to select an other value instead of *VIP* which is not visible to the agent due to an ACL rule. The form is submitted and the dynamic field value is changed. This can be an **unintended change**.
4. The generic agent changes the value to *VIP* again.
5. The agent opens a ticket action where the field is added as optional. The field shows an empty value because the current *VIP* value is not visible to the agent. Since the field is not mandatory the agent does not change the value and leaves it empty. The form is submitted and the dynamic field value is changed to empty value. This can be a **possible data loss**.

Be careful of unintended data change! The same situation can happen with dynamic fields, priorities, queues, states, types and any other drop-down fields that are forbidden by ACLs.

5.1.3 ACL Settings

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

Name *
The name of this resource. This field can only contain the following characters: a-z, A-Z, 0-9, +, -, *, /. The name will be displayed in the overview table.

Comment
Add additional information to this resource. It is recommended to always fill this field as a description of the resource with a full sentence for better clarity, because the comment will be also displayed in the overview table.

Description
Like comment, but longer text can be added here.

Stop after match
ACLs are evaluated in alphabetical order. This setting disables the evaluation of the subsequent ACLs.

Validity *
Set the validity of this resource. Each resource can be used in OTRS only, if this field is set to *valid*. Setting this field to *invalid* or *invalid-temporarily* will disable the use of the resource.

5.1.4 Edit ACL Structure

The ACL definition can be split into two big parts, *Match settings* and *Change settings*.

Match Settings

In the match settings the ACLs contain attributes that has to be met in order to use the ACL. If an ACL contains more than one attribute then all attributes have to be met. If the attributes defined in the ACL do not match with the attributes that are sent, then the ACL does not take any affect, but any other match ACL will.

`Properties`
This section contains matching options that can be changed on the fly. For example on a ticket creation time the data of the ticket changes dynamically as the agent sets the information. If an ACL is set to match a ticket attribute then only when the matching attribute is selected the ACL will be active and might reduce other ticket attributes, but as soon as another value is selected the ACL will not take any affect.

`PropertiesDatabase`
This section is similar to `Properties` but does not take changes in ticket attributes that are not saved into the database, this means that changing an attribute without submit will not make any effect. This section is not use for ticket creation screens (as tickets are not yet created in the database).

Change Settings

The change settings contain the rules to reduce the possible options for a ticket.

`Possible`
This section is used to reset the data to be reduce to only the elements that are set in this section.

`PossibleAdd`
This section is used to add missing elements that were reduced in other ACLs. This section is only used in together with other ACLs that have `Possible` or `PossibleNot` sections.

`PossibleNot`
This section is used to remove specific elements from the current data. It could be used stand alone or together with other ACLs with a `Possible` or `PossibleAdd` sections.

Modifiers

In order to make the development of ACLs easier and more powerful there is a set of so called modifiers for the attributes on each section. This modifiers are explained below:

`[Not]`
This modifier is used to negate a value, for example `[Not]2 low`. Talking about priorities will be the same as to have: *1 very low*, *3 normal*, *4 high*, *5 very high*.

`[RegExp]`
It is used to define a regular expression for matching several values, for example `[RegExp]low`. In this case talking about priorities is the same as *1 very low*, *2 low*.

`[regexp]`
It is very similar to `[RegExp]` but it is case insensitive.

[NotRegExp]
Negated regular expressions, for example `[NotRegExp]low`. Talking about priorities is the same as *3 normal*, *4 high*, *5 very high*.

[Notregexp]
It is very similar to `[NotRegExp]` but it is case insensitive.

5.1.5 ACL Examples

Limit available queues based on current queue and priority
This example shows how to restrict the available queues to the *Alert* queue if the ticket is currently in the queue *Raw* and has the priority *5 very high*.

In this example `PropertiesDatabase` is used. This means that the ticket must already be in the queue *Raw* and must have the priority *5 very high* in order to apply this ACL.

This ACL is not applied if a user only selects the queue *Raw* and the priority *5 very high* in a dialog. To achieve this, `Properties` has to be used.

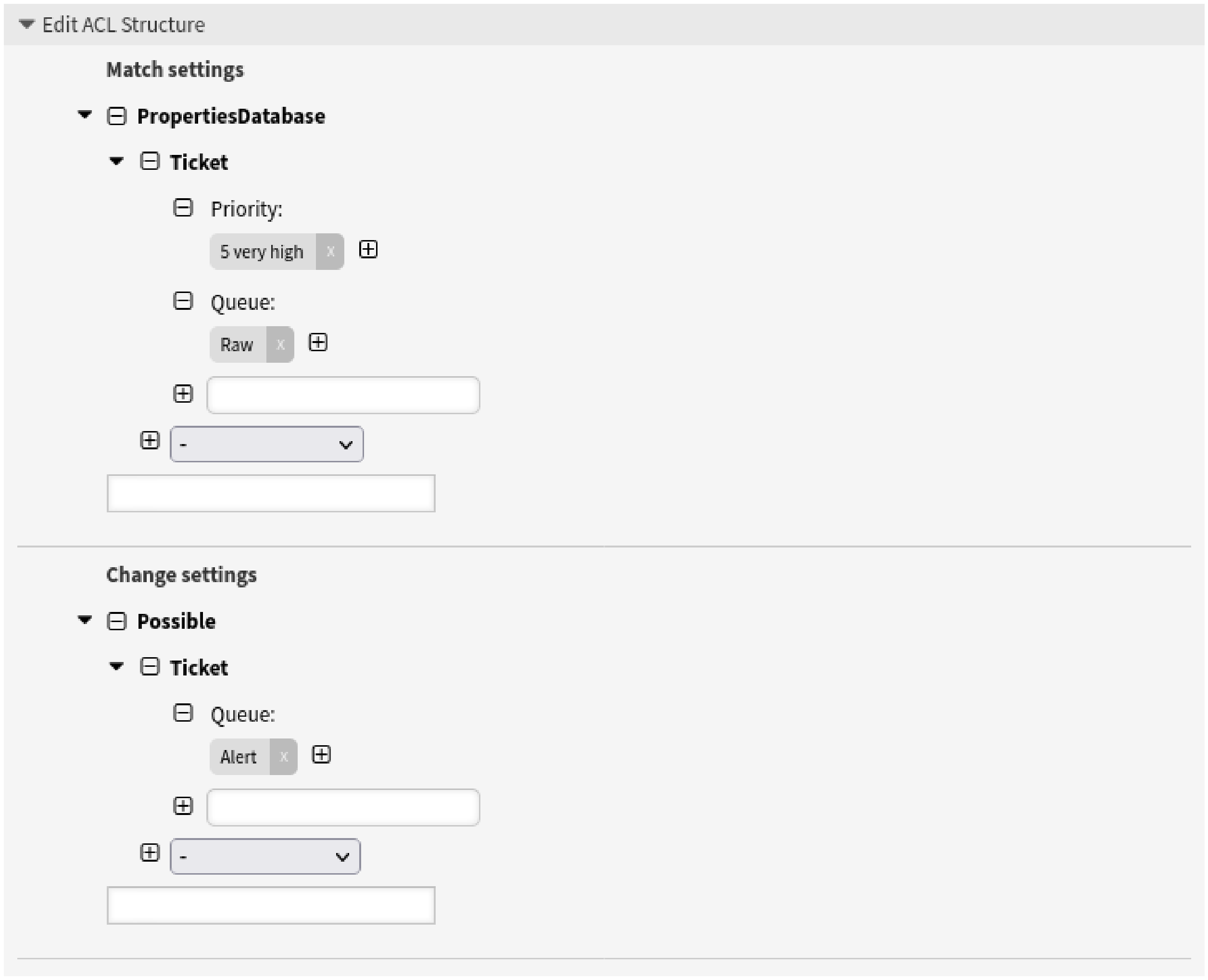
Using this ACL, tickets that are in the *Raw* queue and have the priority *5 very high* can only be moved to the *Alert* queue.

```
---
- Comment: Limit available queues based on current queue and priority
  ConfigChange:
    Possible:
      Ticket:
        Queue:
        - Alert
  ConfigMatch:
    PropertiesDatabase:
      Ticket:
        Priority:
        - 5 very high
        Queue:
        - Raw
  Description: Restrict the available queues to the “Alert” queue if the ticket␣
↪is
    currently in the queue “Raw” and has the priority “5 very high”.
  Name: 010 Example ACL
```

Make certain ticket type not selectable
This example shows how to make the ticket type *Unclassified* not selectable for users.

In this example, the match settings are empty. This means that the ACL is always applied because the filter is empty.

```
---
- Comment: Make certain ticket type not selectable
  ConfigChange:
    PossibleNot:
      Ticket:
        Type:
        - Unclassified
  ConfigMatch: ''
  Description: Make the ticket type “Unclassified” not selectable.
  Name: 020 Example ACL
```



Edit ACL Structure

Match settings

PropertiesDatabase

Ticket

Priority:

5 very high

Queue:

Raw

-

Change settings

Possible

Ticket

Queue:

Alert

-

Fig. 4: Limit available queues based on current queue and priority

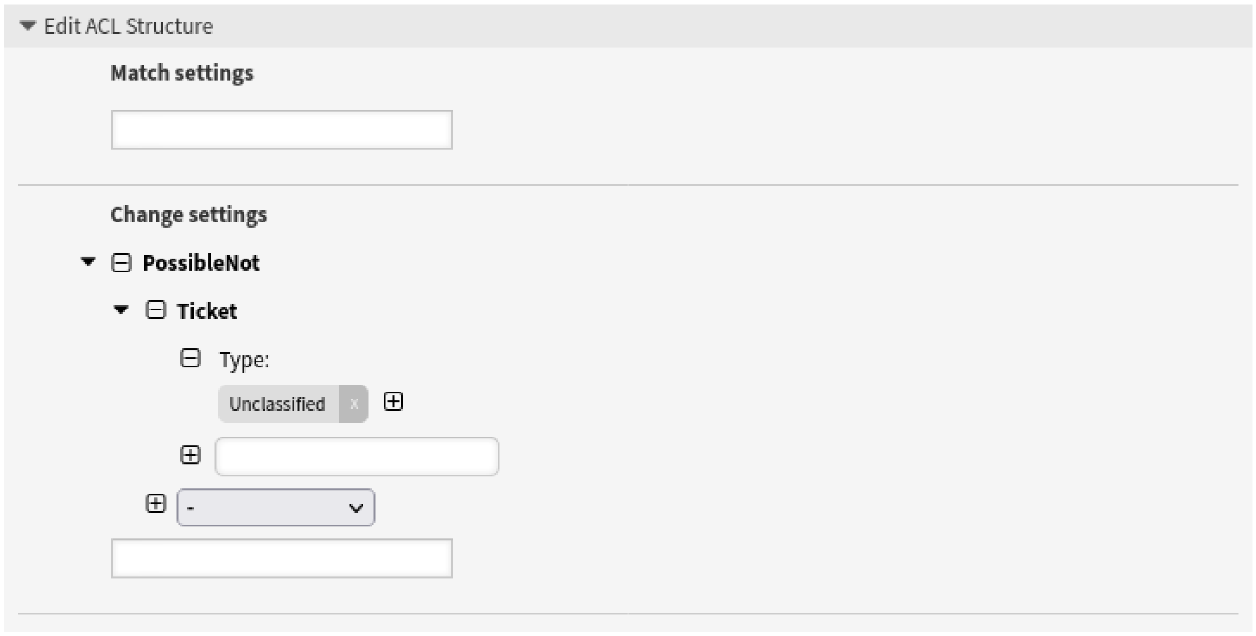


Fig. 5: Make certain ticket type not selectable

Note that you still may find tickets with the ticket type *Unclassified*. This is for example the case of incoming emails which are converted into tickets.

Ticket closing not possible for certain ticket type

This example shows how to prevent tickets with the ticket type *Unclassified* from being closed.

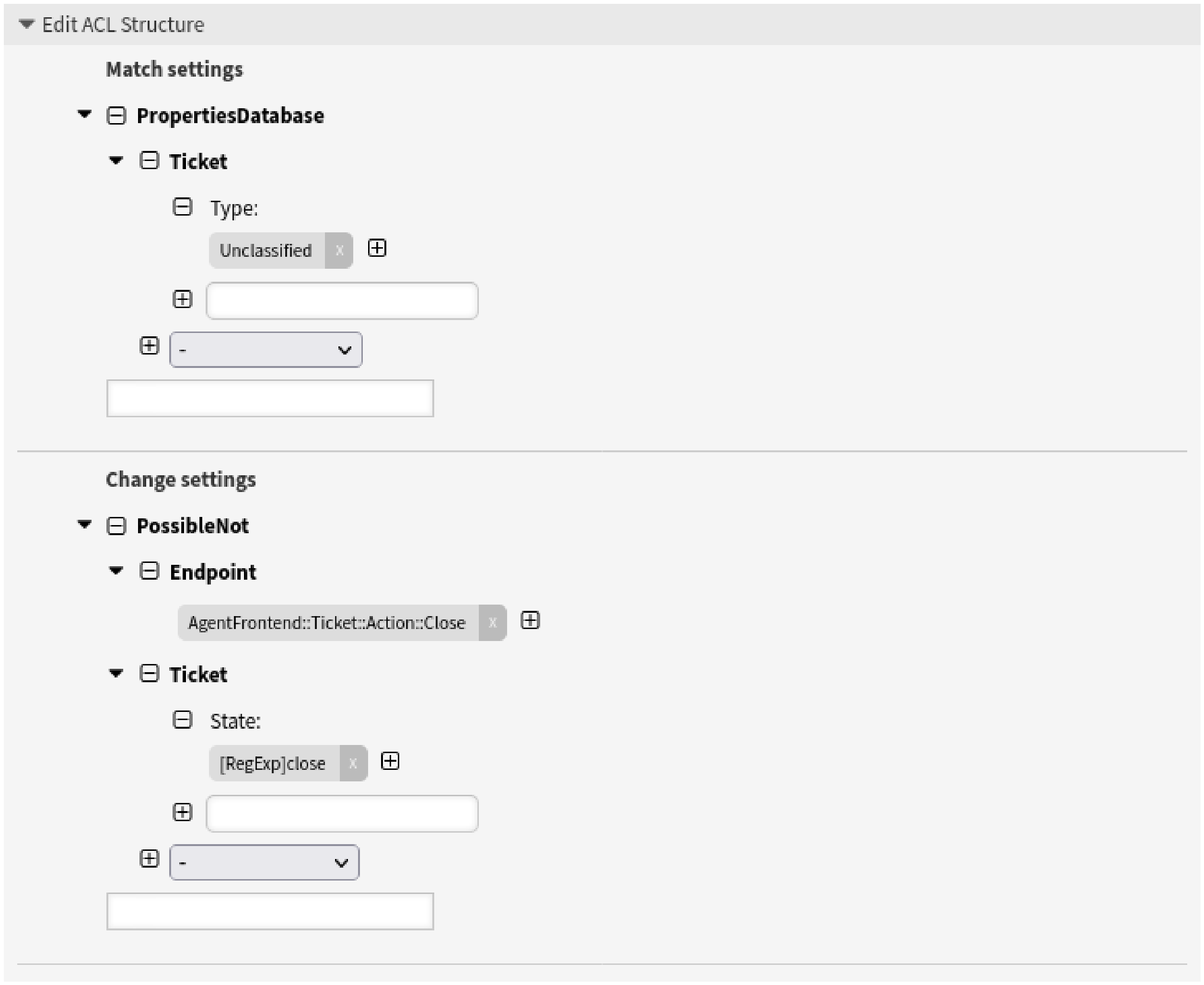
```
---
- Comment: Ticket closing not possible for cerain ticket type
  ConfigChange:
    PossibleNot:
      Endpoint:
      - AgentFrontend::Ticket::Action::Close
      Ticket:
        State:
        - '[RegExp]close'
  ConfigMatch:
    PropertiesDatabase:
      Ticket:
        Type:
        - Unclassified
  Description: Prevent tickets with the ticket type "Unclassified" from being␣
↪closed.
  Name: 030 Example ACL
```

This example ACL can be used in combination with the previously described example ACL.

Note: Pay attention to the order of the ACLs. The previously described example ACL must be executed before the ACL described here.

Using regular expressions

This example shows how to use regular expressions for matching tickets and for filtering the available selection options.



Edit ACL Structure

Match settings

PropertiesDatabase

Ticket

Type:

Unclassified

-

Change settings

PossibleNot

Endpoint

AgentFrontend::Ticket::Action::Close

Ticket

State:

[RegExp]close

-

Fig. 6: Ticket closing not possible for certain ticket type

With this ACL, only services that contain `Hardware` in their name are displayed in agent interface (and if configured as well in the external interface) if the ticket is stored in a queue whose name starts with `HW`, or if a queue whose name starts with `HW` is selected in a user dialog.

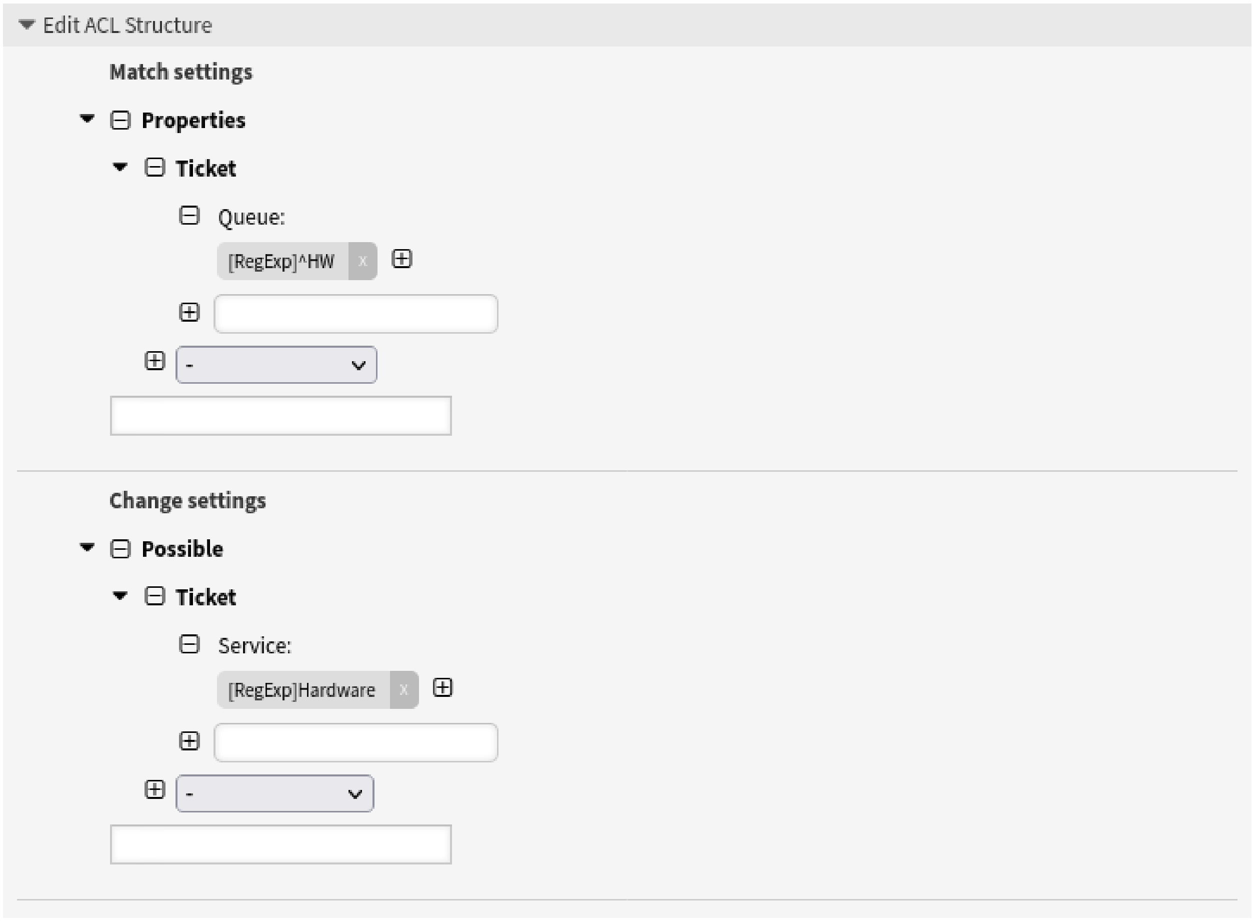


Fig. 7: Using regular expressions

```
---
- Comment: Using regular expressions
  ConfigChange:
    Possible:
      Ticket:
        Service:
        - '[RegExp]Hardware'
  ConfigMatch:
    Properties:
      Ticket:
        Queue:
        - '[RegExp]^HW'
  Description: Use regular expressions for matching tickets and for filtering the
    available selection options.
  Name: 040 Example ACL
```

Restrict one dynamic field based on another dynamic field

This example shows how to restrict the selectable values of the dynamic field `CarModel` to *Polo*, *Golf* and *Passat*, if a user has selected the value *VW* in the dynamic field `CarManufacturer` in the same user dialog.

Make sure that you use the name of the dynamic field in format `DynamicField_Name` and make sure that you use the data keys of the possible values and not the data values shown to the user. Both is specified in the definition of the dynamic field.

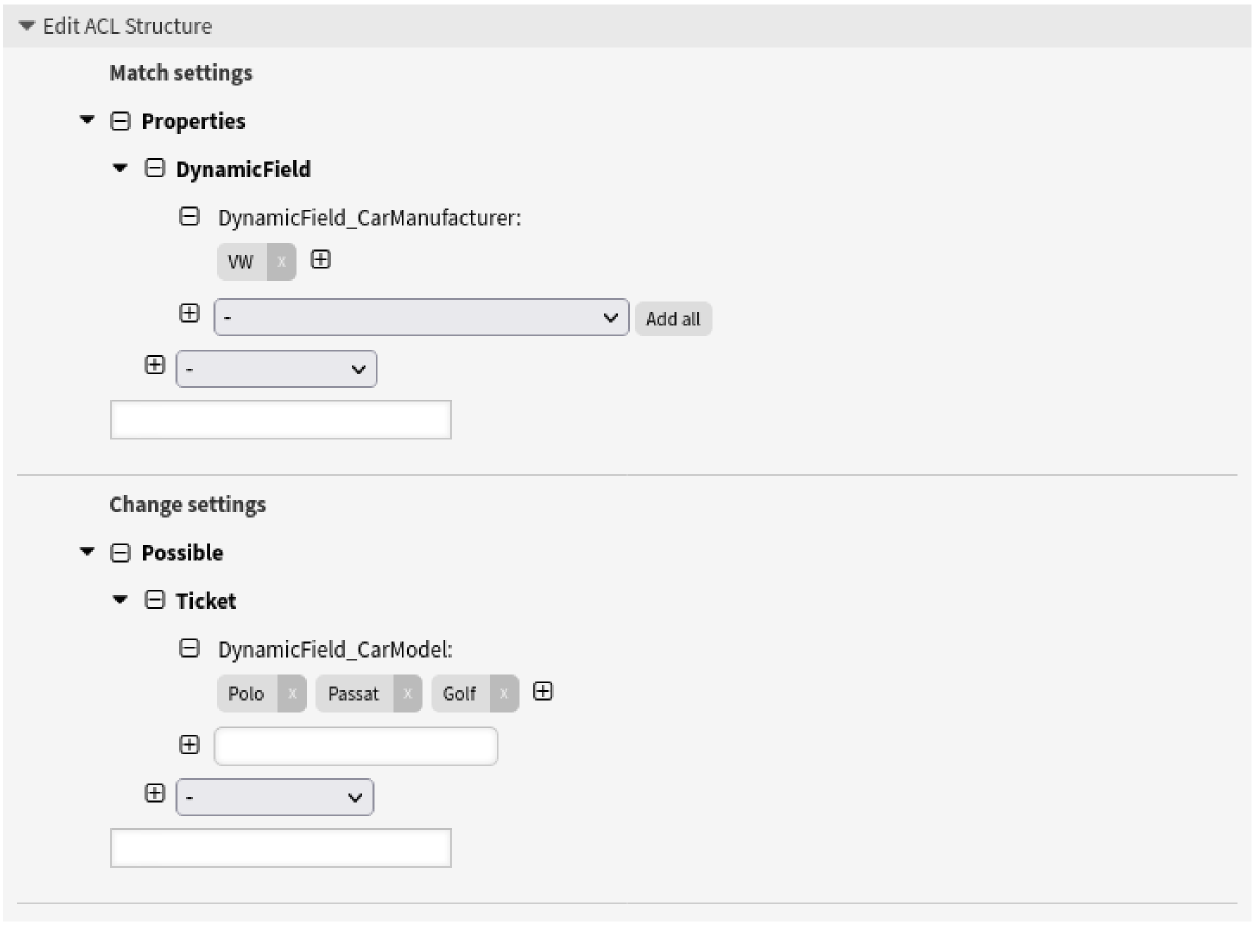


Fig. 8: Restrict one dynamic field based on another dynamic field

```
---
- Comment: Restrict one dynamic field based on another dynamic field
  ConfigChange:
    Possible:
      Ticket:
        DynamicField_CarModel:
        - Polo
        - Passat
        - Golf
  ConfigMatch:
    Properties:
      DynamicField:
        DynamicField_CarManufacturer:
        - VW
  Description: Restrict the selectable values of the dynamic field “CarModel” to
“Polo” ,
     “Golf” and “Passat” , if a user has selected the value “VW” in the dynamic␣
↪field
     “CarManufacturer” in the same user dialog.
```

(continues on next page)

(continued from previous page)

```
Name: 050 Example ACL
```

In the *Match settings* the `Ticket` object can be used as an alternative to the `DynamicField` object.

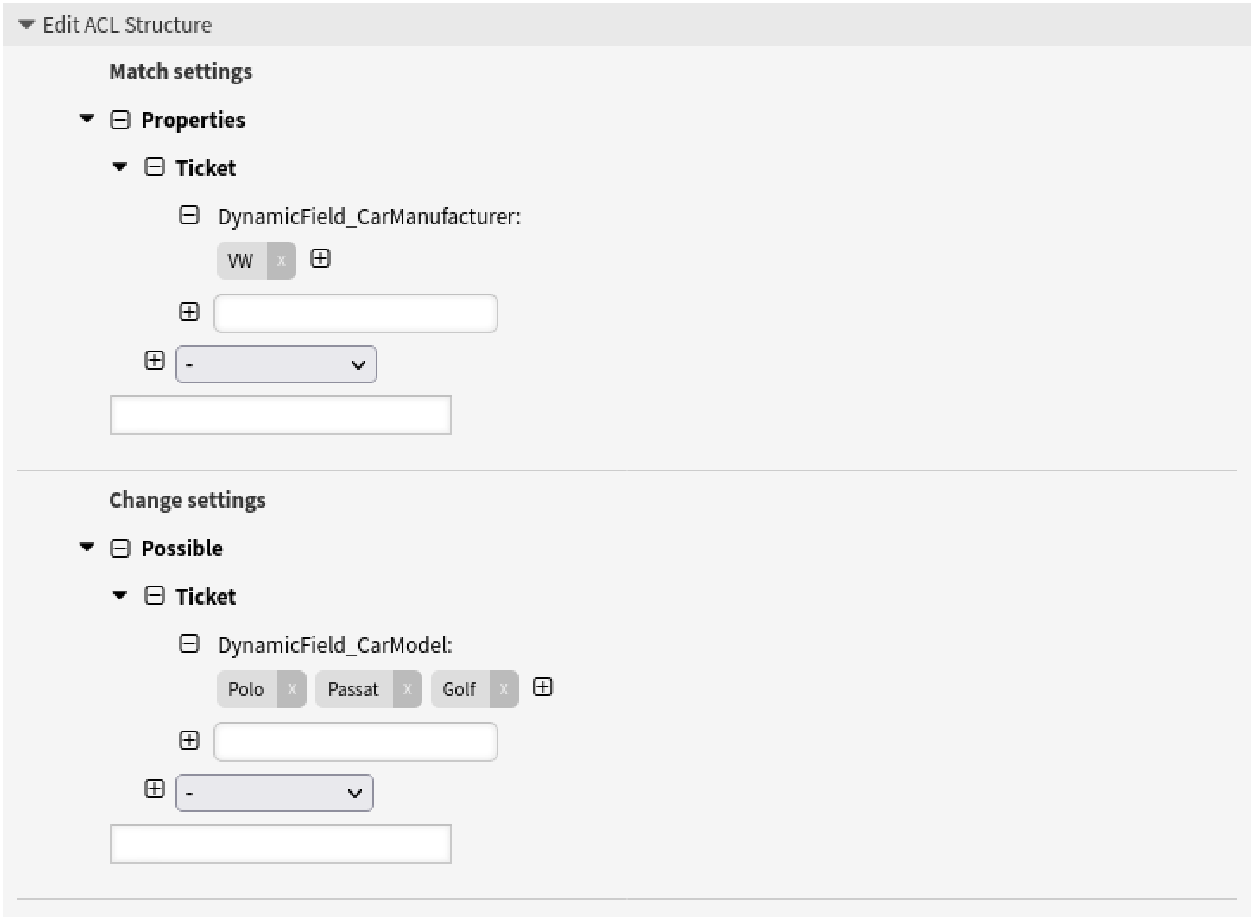


Fig. 9: Restrict one dynamic field based on another dynamic field

```
---
- Comment: Restrict one dynamic field based on another dynamic field
  ConfigChange:
    Possible:
      Ticket:
        DynamicField_CarModel:
        - Polo
        - Passat
        - Golf
  ConfigMatch:
    Properties:
      Ticket:
        DynamicField_CarManufacturer:
        - VW
  Description: Restrict the selectable values of the dynamic field “CarModel” to
“Polo” ,
     “Golf” and “Passat” , if a user has selected the value “VW” in the dynamic␣
↪field
```

(continues on next page)

(continued from previous page)

```
    “CarManufacturer” in the same user dialog.
  Name: 051 Example ACL
```

Restrict one dynamic field based on another dynamic field in an action

The previous example can be extended by the restriction that it should only apply to the ticket action *Change Free Fields*. For this purpose, the condition that the `FreeText` endpoint must be affected, has to be added in the *Match settings*.

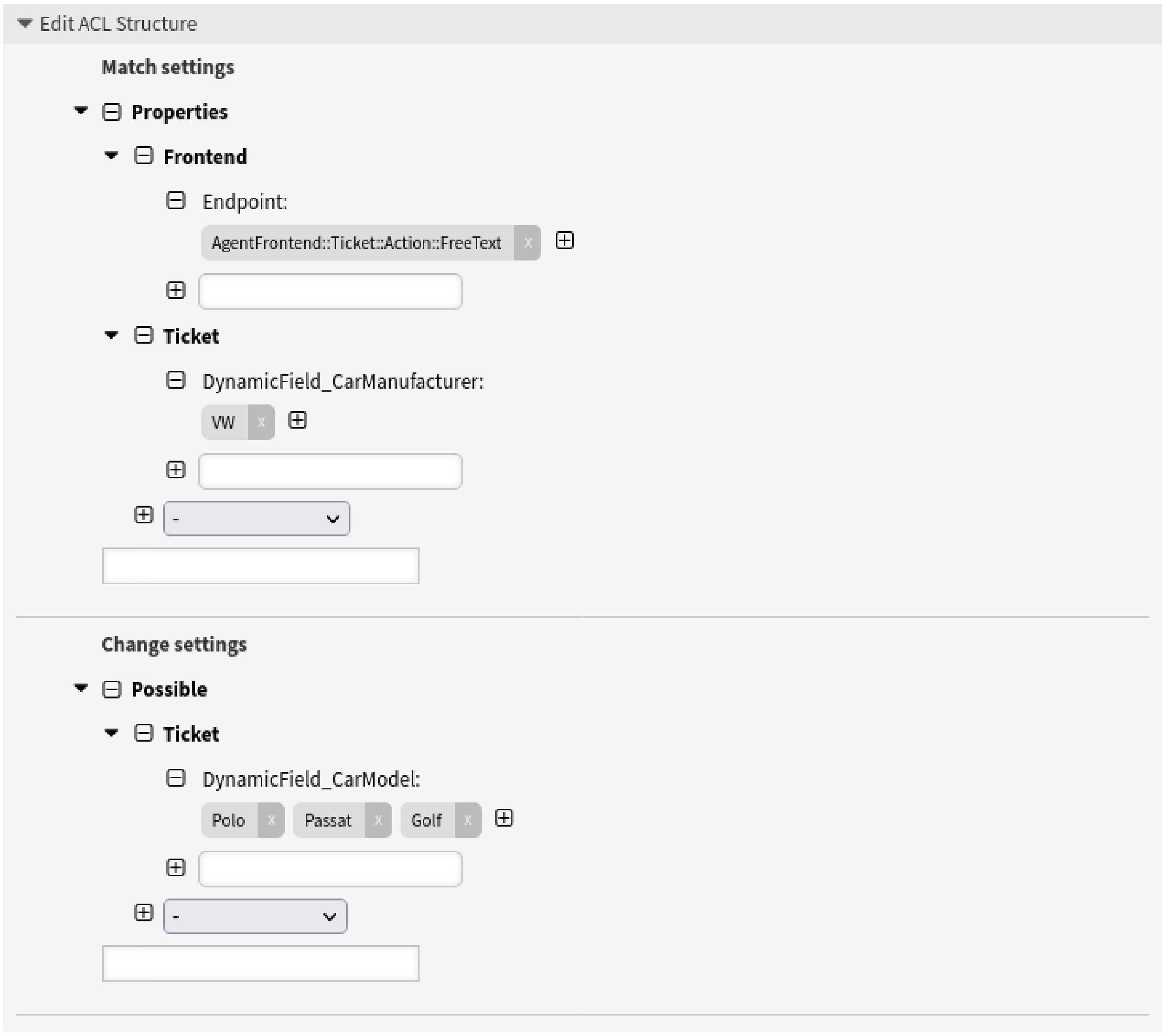


Fig. 10: Restrict one dynamic field based on another dynamic field in an action

```
---
- Comment: Restrict one dynamic field based on another dynamic field in an action
  ConfigChange:
    Possible:
      Ticket:
        DynamicField_CarModel:
        - Polo
```

(continues on next page)

(continued from previous page)

```
        - Passat
        - Golf
  ConfigMatch:
    Properties:
      Frontend:
        Endpoint:
        - AgentFrontend::Ticket::Action::FreeText
      Ticket:
        DynamicField_CarManufacturer:
        - VW
  Description: Restrict the selectable values of the dynamic field "CarModel" to
"Polo",
     "Golf" and "Passat", if a user has selected the value "VW" in the dynamic␣
↪field
     "CarManufacturer" in the "Free Fields" action.
  Name: 052 Example ACL
```

Disallow a process for a customer ID

This ACL prevents the usage of a certain process for all customer users assigned to a certain customer ID (otrs.com in this example).

Since customer users only use the external interface, the ACL is applied there.

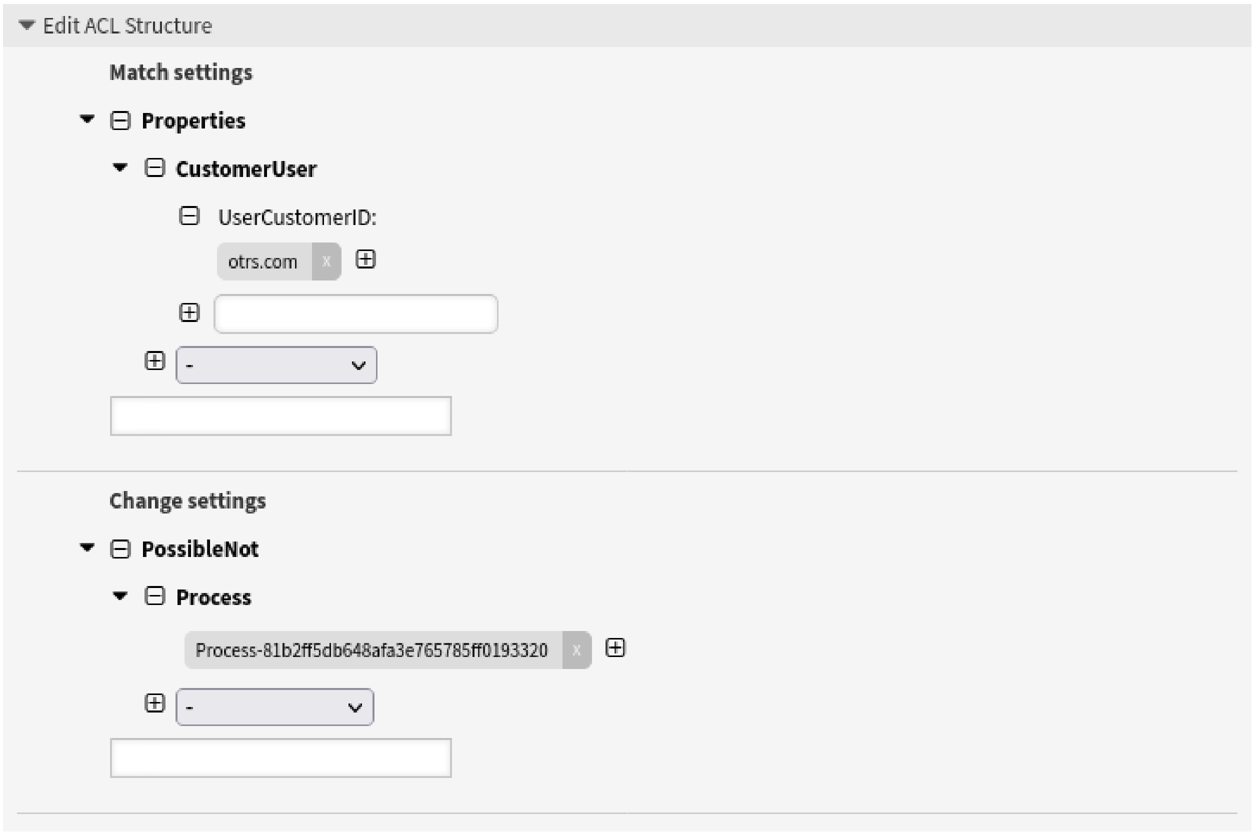


Fig. 11: Disallow a process for a customer ID

```
---
- Comment: Disallow a process for a customer ID
```

(continues on next page)

(continued from previous page)

```
  ConfigChange:
    PossibleNot:
      Process:
      - Process-81b2ff5db648afa3e765785ff0193320
  ConfigMatch:
    Properties:
      CustomerUser:
        UserCustomerID:
        - otrs.com
  Description: Prevents the use of a certain process for all customer users␣
↪assigned
    to a certain customer ID.
  Name: 060 Example ACL
```

Disallow the ticket actions close and move for a certain process

This ACL example shows how to stop the ticket actions *Move Ticket* and *Close Ticket* from displaying when a specific process is selected.

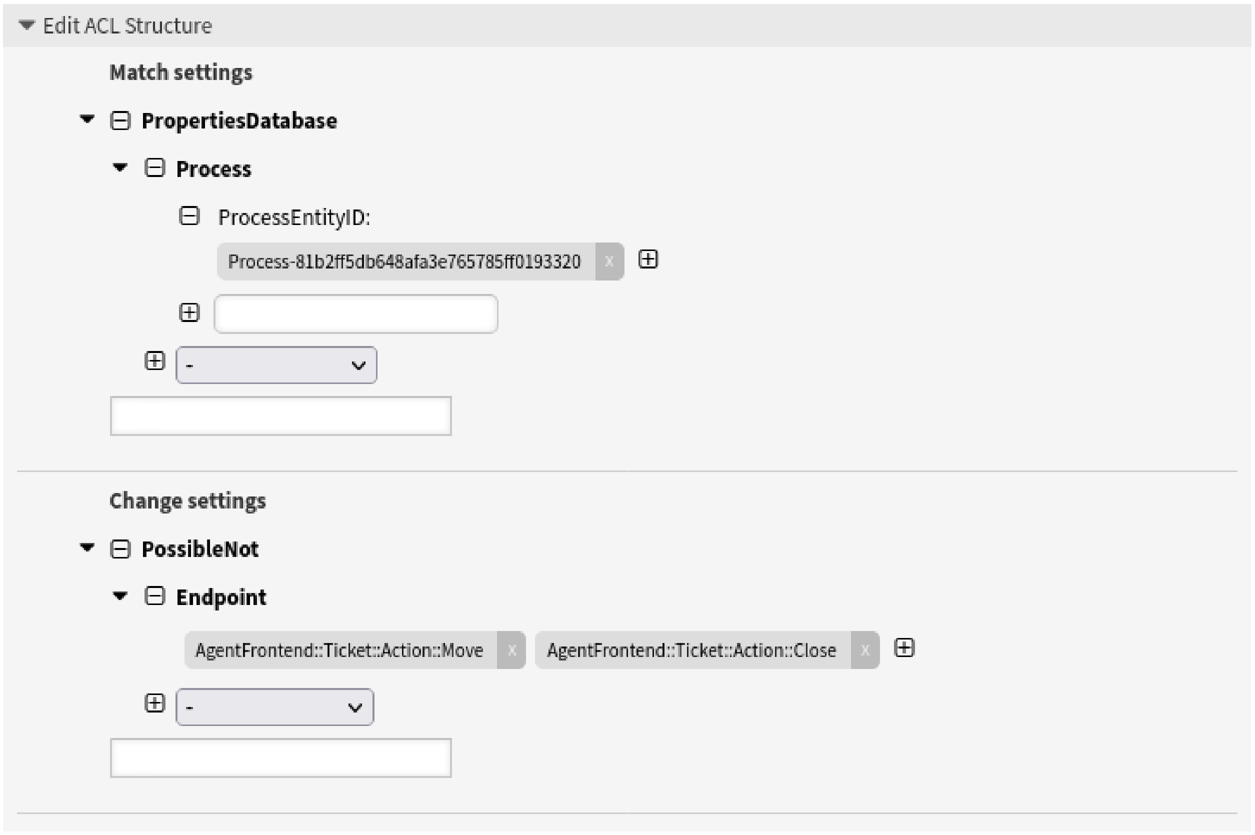

Edit ACL Structure
Match settings
PropertiesDatabase
Process
ProcessEntityID:
Process-81b2ff5db648afa3e765785ff0193320
Change settings
PossibleNot
Endpoint
AgentFrontend::Ticket::Action::Move
AgentFrontend::Ticket::Action::Close

Fig. 12: Disallow the ticket actions close and move for a certain process

```
---
- Comment: Disallow the ticket actions close and move for a certain process
  ConfigChange:
    PossibleNot:
      Endpoint:
      - AgentFrontend::Ticket::Action::Move
```

(continues on next page)

(continued from previous page)

```
      - AgentFrontend::Ticket::Action::Close
  ConfigMatch:
    PropertiesDatabase:
      Process:
        ProcessEntityID:
        - Process-81b2ff5db648afa3e765785ff0193320
  Description: Stop the ticket actions “Move Ticket” and “Close Ticket” from␣
↪displaying
    when a specific process is selected.
  Name: 070 Example ACL
```

Hide user task activity dialogs in a process based on the agent role

Let’ s assume that a process contains a user task activity that has two user task activity dialogs assigned to it and that these user task activity dialogs shall be executed only by agents who have a specific role.

The goal is to display the correct user task activity dialogs only to the agents who have the correct role.

User task activity dialog	Agent role
Approval from Sales	Head of Sales
Approval from Marketing	Head of Marketing

For this purpose three ACLs are required.

Note: Process and user task activity dialogs are referenced by ID.

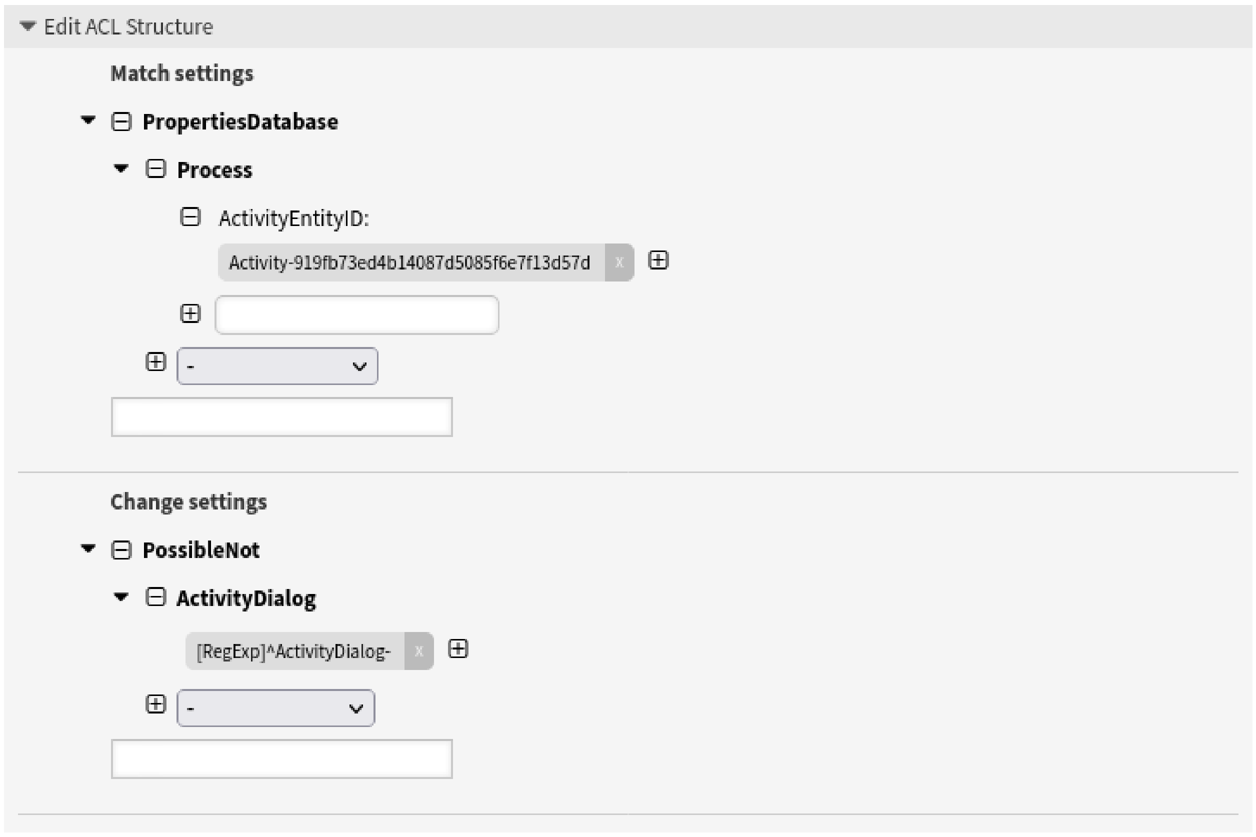
The first ACL prevents the display of all user task activity dialogs within this user task activity.

```
---
- Comment: Hide user task activity dialogs in a process based on the agent role
  ConfigChange:
    PossibleNot:
      ActivityDialog:
      - '[RegExp]^ActivityDialog-'
  ConfigMatch:
    PropertiesDatabase:
      Process:
        ActivityEntityID:
        - Activity-919fb73ed4b14087d5085f6e7f13d57d
  Description: Prevents the display of all user task activity dialogs within this
    user task activity.
  Name: 080 Example ACL
```

By means of the next ACL it is achieved that all agents who have the role *Head of Sales* get the user task activity dialog *Approval from Sales* displayed.

```
---
- Comment: Hide user task activity dialogs in a process based on the agent role
  ConfigChange:
    PossibleAdd:
      ActivityDialog:
      - ActivityDialog-8ff3a399b5939d1a11e87ad35f72f576
  ConfigMatch:
    PropertiesDatabase:
```

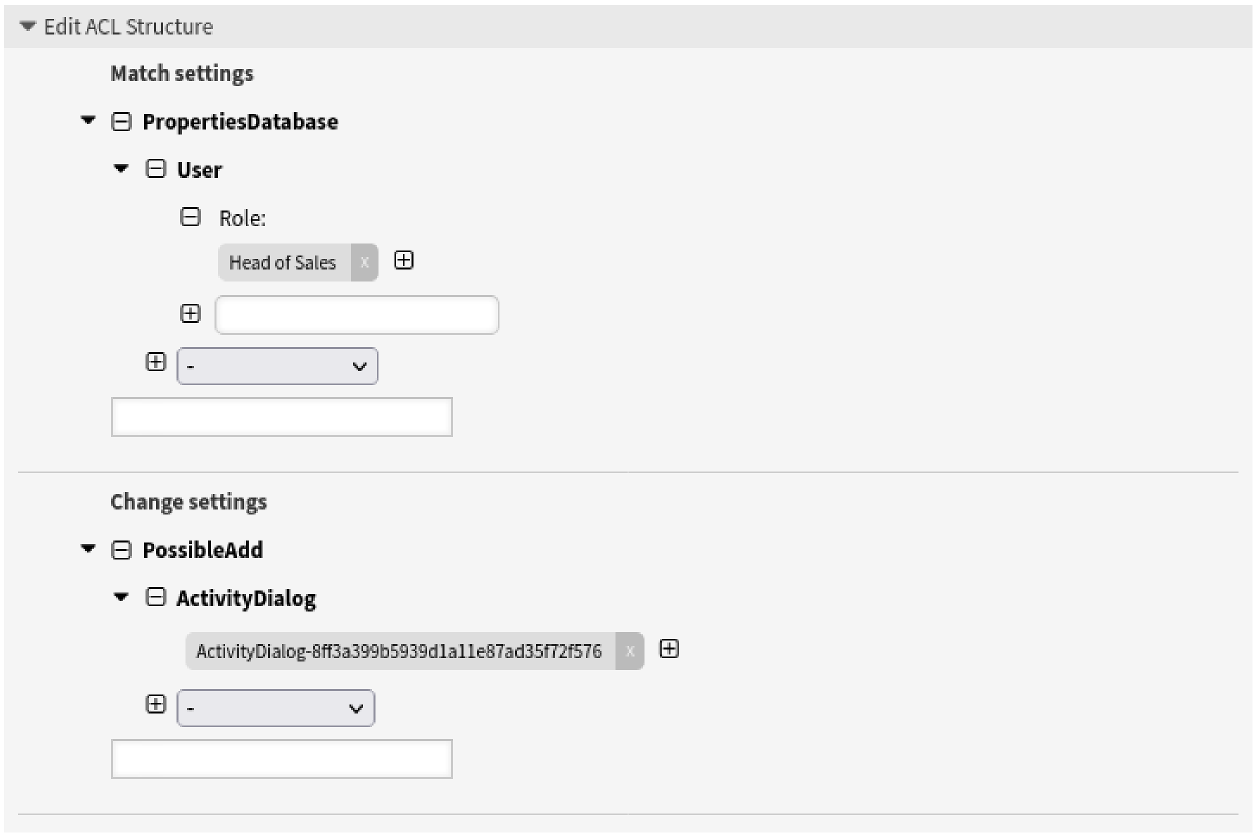
(continues on next page)



ActivityEntityID:
Activity-919fb73ed4b14087d5085f6e7f13d57d

[RegExp]^ActivityDialog-

Fig. 13: Hide user task activity dialogs in a process based on the agent role



Edit ACL Structure

Match settings

PropertiesDatabase

User

Role:

Head of Sales

Change settings

PossibleAdd

ActivityDialog

ActivityDialog-8ff3a399b5939d1a11e87ad35f72f576

Fig. 14: Hide user task activity dialogs in a process based on the agent role

(continued from previous page)

```
      User:
        Role:
        - Head of Sales
  Description: All agents who have the role "Head of Sales" get the user task␣
↪activity
    dialog "Approval from Sales" displayed.
  Name: 081 Example ACL
```

By means of the last ACL it is achieved that all agents who have the role *Head of Marketing* get the user task activity dialog *Approval from Marketing* displayed.

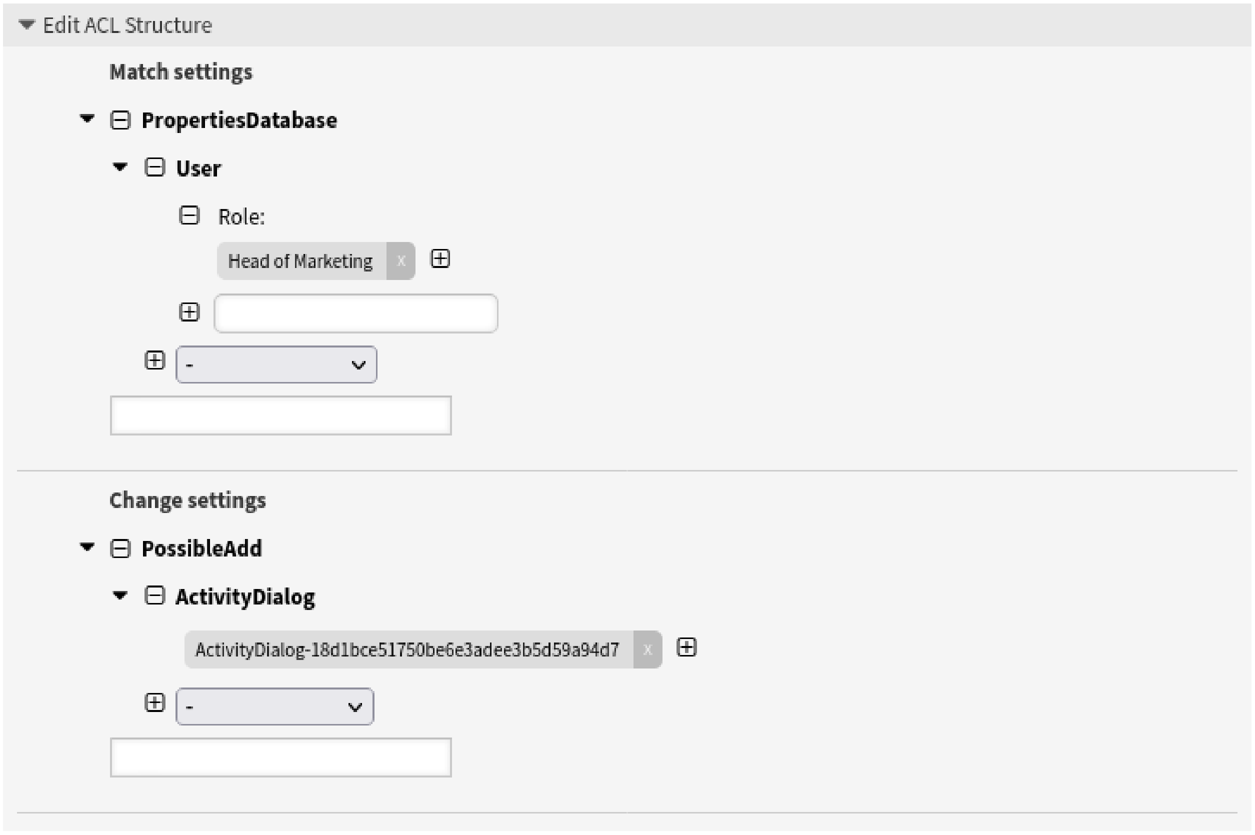


Fig. 15: Hide user task activity dialogs in a process based on the agent role

```
---
- Comment: Hide user task activity dialogs in a process based on the agent role
  ConfigChange:
    PossibleAdd:
      ActivityDialog:
      - ActivityDialog-18d1bce51750be6e3adee3b5d59a94d7
  ConfigMatch:
    PropertiesDatabase:
      User:
        Role:
        - Head of Marketing
  Description: All agents who have the role "Head of Marketing" get the user task
    activity dialog "Approval from Marketing" displayed.
  Name: 082 Example ACL
```

Note: Pay attention to the order of the ACLs. The ACL described first in this example must be executed before the last two ACLs.

Pitfalls using properties

If ticket attributes are used in both *Match settings* and *Change settings* at the same time, a logical problem occurs when using `Properties`, which may prevent ticket actions from being used in a proper way.

In this example you can see in the *Match settings* the queue *Raw* and the priority *5 very high* and in the *Change settings* the queue *Alert* is possible.

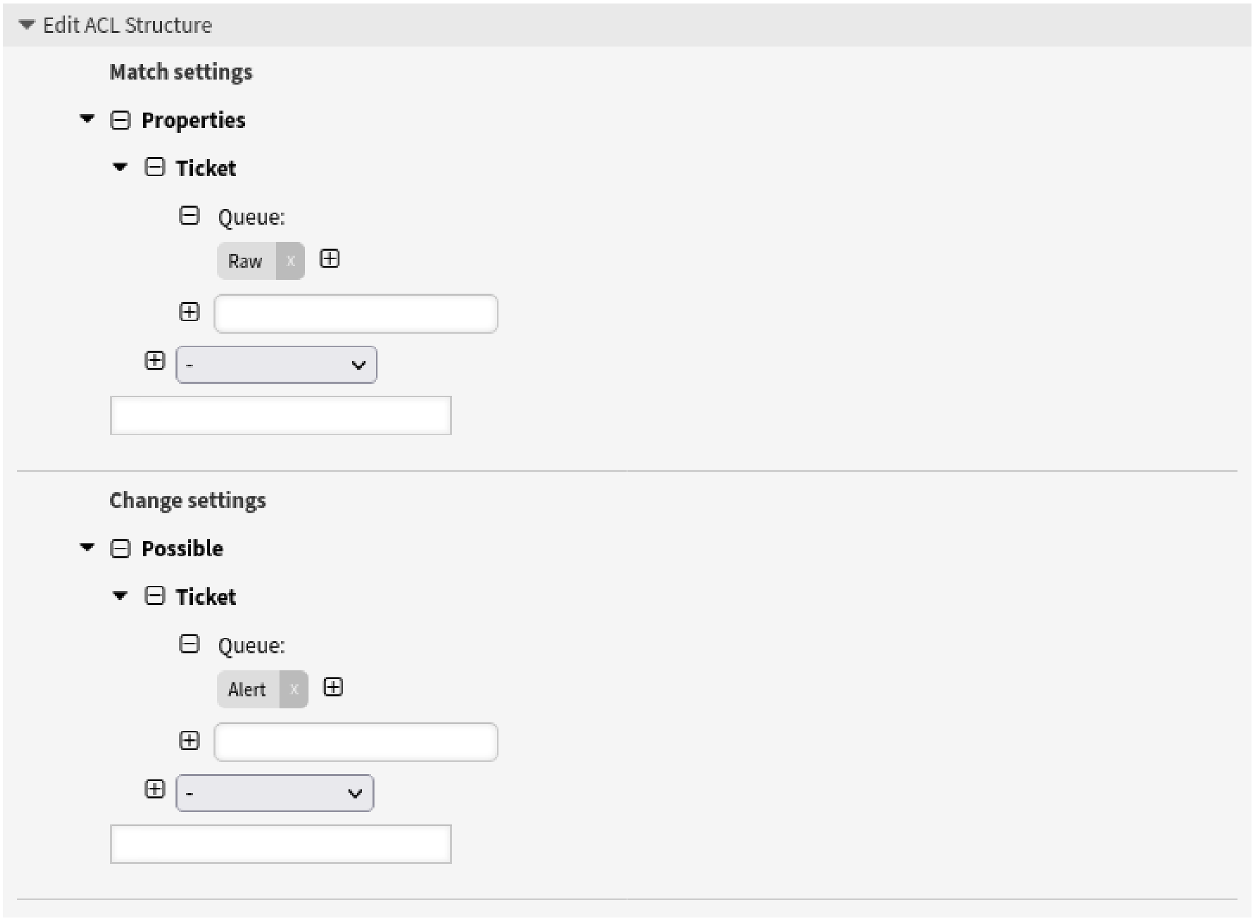


Fig. 16: Pitfalls using properties

```
---
- Comment: Pitfalls Using Properties
  ConfigChange:
    Possible:
      Ticket:
        Queue:
        - Alert
  ConfigMatch:
    Properties:
      Ticket:
```

(continues on next page)

(continued from previous page)

```
        Queue:
        - Raw
  Description: If ticket attributes are used in both Match settings and Change␣
↪settings
    at the same time, a logical problem occurs.
  Name: 090 Pitfalls
```

If you deploy this ACL and then run the *Move Ticket* ticket action in a ticket, you will see that you cannot click the *Send* button if you select the *Raw* queue because the change setting will be applied immediately (due to `Properties`) and this will result in your selection being deleted because the *Raw* queue is no longer a valid selection.

Typically this logical problem can be avoided by using `PropertiesDatabase` instead of `Properties`. Please compare with the very first given example.

5.1.6 ACL Reference

Properties, keys and values that can be used in ACLs are highly depend on the OTRS installation. For example the possibilities can be extended by installing extension modules, as well as it can be depend on the customer user mapping set in `Config.pm`. Therefore it is not possible to provide a full ACL reference, that contains all settings.

For properties, keys and values that can be used in ACLs, see the following example ACL in YAML format. The example ACL is complete for a default OTRS installation. The ellipsis denotes any options available from installed features or customer user mappings.

```
---
- ChangeBy: root@localhost
  ChangeTime: 2020-04-15 16:46:23
  Comment: ACL Reference.
  ConfigMatch:
    Properties:
      # Match properties (current values from the form).
      CustomerUser:
        UserLogin:
        - some login
        UserCustomerID:
        - some customer ID
        Group_rw:
        - some group
      DynamicField:
        # Names must be in DynamicField_<field_name> format.
        # Values for dynamic fields must always be the untranslated internal
        #   data keys specified in the dynamic field definition and not the
        #   data values shown to the user.
        # Using the key is also mandatory for dynamic field of type database
        #   and dynamic field of type web service.
        DynamicField_Field1:
        - some value
        DynamicField_OtherField:
        - some value
        DynamicField_TicketFreeText2:
        - some value
        # more dynamic fields
      Frontend:
```

(continues on next page)

(continued from previous page)

```
        Endpoint:
        - AgentFrontend::PersonalPreferences
        - AgentFrontend::ProcessTicketNextStep
        - AgentFrontend::Ticket::Action::Close
        - AgentFrontend::Ticket::Action::Customer
        - AgentFrontend::Ticket::Action::EmailOutbound
        - AgentFrontend::Ticket::Action::FreeText
        - AgentFrontend::Ticket::Action::Link
        - AgentFrontend::Ticket::Action::Lock
        - AgentFrontend::Ticket::Action::Merge
        - AgentFrontend::Ticket::Action::Move
        - AgentFrontend::Ticket::Action::Note
        - AgentFrontend::Ticket::Action::Owner
        - AgentFrontend::Ticket::Action::Pending
        - AgentFrontend::Ticket::Action::PhoneCallInbound
        - AgentFrontend::Ticket::Action::PhoneCallOutbound
        - AgentFrontend::Ticket::Action::Print
        - AgentFrontend::Ticket::Action::Priority
        - AgentFrontend::Ticket::Action::Process
        - AgentFrontend::Ticket::Action::Redirect
        - AgentFrontend::Ticket::Action::Responsible
        - AgentFrontend::Ticket::Action::SmsOutbound
        - AgentFrontend::Ticket::Action::TicketHistory
        - AgentFrontend::Ticket::Action::Unlock
        - AgentFrontend::Ticket::Action::Unwatch
        - AgentFrontend::Ticket::Action::Watch
        - AgentFrontend::Ticket::InlineEditing::Property::CustomerUserID
        - AgentFrontend::Ticket::InlineEditing::Property::Lock
        - AgentFrontend::Ticket::InlineEditing::Property::Owner
        - AgentFrontend::Ticket::InlineEditing::Property::Priority
        - AgentFrontend::Ticket::InlineEditing::Property::Queue
        - AgentFrontend::Ticket::InlineEditing::Property::Responsible
        - AgentFrontend::Ticket::InlineEditing::Property::Service
        - AgentFrontend::Ticket::InlineEditing::Property::State
        - AgentFrontend::Ticket::InlineEditing::Property::Type
        - AgentFrontend::Ticket::InlineEditing::Property::Watch
        - AgentFrontend::TicketArticle::Action::CopyLink
        - AgentFrontend::TicketArticle::Action::Forward
        - AgentFrontend::TicketArticle::Action::MarkAsImportant
        - AgentFrontend::TicketArticle::Action::MessageLog
        - AgentFrontend::TicketArticle::Action::Plain
        - AgentFrontend::TicketArticle::Action::Print
        - AgentFrontend::TicketArticle::Action::Redirect
        - AgentFrontend::TicketArticle::Action::Reply
        - AgentFrontend::TicketArticle::Action::ReplyAll
        - AgentFrontend::TicketArticle::Action::ReplyToNote
        - AgentFrontend::TicketArticle::Action::ReplyViaSms
        - AgentFrontend::TicketArticle::Action::Split
        - AgentFrontend::TicketArticle::Action::UnmarkAsImportant
        - AgentFrontend::TicketCreate::Email
        - AgentFrontend::TicketCreate::Phone
        - AgentFrontend::TicketCreate::Process
        - AgentFrontend::TicketCreate::SMS
        - AgentFrontend::TicketDetailView
        - AgentFrontend::TicketDetailView::Property
        - AgentFrontend::TicketList::Bulk
```

(continues on next page)

(continued from previous page)

```
        - AgentFrontend::TicketList::Filters
        - ...
        - ExternalFrontend::PersonalPreferences
        - ExternalFrontend::ProcessTicketCreate
        - ExternalFrontend::ProcessTicketNextStep
        - ExternalFrontend::Ticket::ExportList     # used for technical purpose, not␣
↪for ACLs
        - ExternalFrontend::Ticket::List           # used for technical purpose, not␣
↪for ACLs
        - ExternalFrontend::Ticket::Print
        - ExternalFrontend::TicketCreate
        - ExternalFrontend::TicketDetailView
        - ...
      Owner:
        UserLogin:
        - some login
        Group_rw:
        - some group
        Role:
        - admin
        # more owner attributes
      Priority:
        ID:
        - some ID
        Name:
        - some name
        # more priority attributes
      Process:
        ProcessEntityID:
        # the process that the current ticket is part of
        - Process-9c378d7cc59f0fce4cee7bb9995ee3eb
        ActivityEntityID:
        # the current activity of the ticket
        - Activity-f8b2fdebe54eeb7b147a5f8e1da5e35c
        ActivityDialogEntityID:
        # the current activity dialog that the agent/customer is using
        - ActivityDialog-aff0ae05fe6803f38de8fff6cf33b7ce
      Queue:
        Name:
        - Raw
        QueueID:
        - some ID
        GroupID:
        - some ID
        Email:
        - some email
        RealName:
        - OTRS System
        # more queue attributes
      Responsible:
        UserLogin:
        - some login
        Group_rw:
        - some group
        Role:
        - admin
```

(continues on next page)

(continued from previous page)

```
        # more responsible attributes
      Service:
        ServiceID:
        - some ID
        Name:
        - some name
        ParentID:
        - some ID
        # more service attributes
      SLA:
        SLAID:
        - some ID
        Name:
        - some name
        Calendar:
        - some calendar
        # more SLA attributes
      State:
        ID:
        - some ID
        Name:
        - some name
        TypeName:
        - some state type name
        TypeID:
        - some state type ID
        # more state attributes
      Ticket:
        Queue:
        - Raw
        State:
        - new
        - open
        Priority:
        - some priority
        Lock:
        - lock
        CustomerID:
        - some ID
        CustomerUserID:
        - some ID
        Owner:
        - some owner
        DynamicField_Field1:
        - some value
        DynamicField_MyField:
        - some value
        # more ticket attributes
      Type:
        ID:
        - some ID
        Name:
        - some name
        # more type attributes
      User:
        UserLogin:
```

(continues on next page)

(continued from previous page)

```
          - some_login
          Group_rw:
          - some group
          Role:
          - admin
      PropertiesDatabase:
        # Match properties (existing values from the database).
        # Please note that Frontend is not in the database, but in the framework.
        # See section "Properties", the same configuration can be used here.
    ConfigChange:
      Possible:
        # Reset possible options (white list).
        Action:
        # Possible action options (white list).
        - ...
        ActivityDialog:
        # Limit the number of possible activity dialogs the agent/customer can use in a␣
↪process ticket.
        - ActivityDialog-aff0ae05fe6803f38de8fff6cf33b7ce
        - ActivityDialog-429d61180a593414789a8087cc4b3c6f
        - ...
        Endpoint:
        # Limit the functions on agent interface.
        - AgentFrontend::PersonalPreferences
        - AgentFrontend::ProcessTicketNextStep
        - AgentFrontend::Ticket::Action::Close
        - AgentFrontend::Ticket::Action::Customer
        - AgentFrontend::Ticket::Action::EmailOutbound
        - AgentFrontend::Ticket::Action::FreeText
        - AgentFrontend::Ticket::Action::Link
        - AgentFrontend::Ticket::Action::Lock
        - AgentFrontend::Ticket::Action::Merge
        - AgentFrontend::Ticket::Action::Move
        - AgentFrontend::Ticket::Action::Note
        - AgentFrontend::Ticket::Action::Owner
        - AgentFrontend::Ticket::Action::Pending
        - AgentFrontend::Ticket::Action::PhoneCallInbound
        - AgentFrontend::Ticket::Action::PhoneCallOutbound
        - AgentFrontend::Ticket::Action::Print
        - AgentFrontend::Ticket::Action::Priority
        - AgentFrontend::Ticket::Action::Process
        - AgentFrontend::Ticket::Action::Redirect
        - AgentFrontend::Ticket::Action::Responsible
        - AgentFrontend::Ticket::Action::SmsOutbound
        - AgentFrontend::Ticket::Action::TicketHistory
        - AgentFrontend::Ticket::Action::Unlock
        - AgentFrontend::Ticket::Action::Unwatch
        - AgentFrontend::Ticket::Action::Watch
        - AgentFrontend::Ticket::InlineEditing::Property::CustomerUserID
        - AgentFrontend::Ticket::InlineEditing::Property::Lock
        - AgentFrontend::Ticket::InlineEditing::Property::Owner
        - AgentFrontend::Ticket::InlineEditing::Property::Priority
        - AgentFrontend::Ticket::InlineEditing::Property::Queue
        - AgentFrontend::Ticket::InlineEditing::Property::Responsible
        - AgentFrontend::Ticket::InlineEditing::Property::Service
        - AgentFrontend::Ticket::InlineEditing::Property::State
```

(continues on next page)

(continued from previous page)

```
      - AgentFrontend::Ticket::InlineEditing::Property::Type
      - AgentFrontend::Ticket::InlineEditing::Property::Watch
      - AgentFrontend::TicketArticle::Action::CopyLink
      - AgentFrontend::TicketArticle::Action::Forward
      - AgentFrontend::TicketArticle::Action::MarkAsImportant
      - AgentFrontend::TicketArticle::Action::MessageLog
      - AgentFrontend::TicketArticle::Action::Plain
      - AgentFrontend::TicketArticle::Action::Print
      - AgentFrontend::TicketArticle::Action::Redirect
      - AgentFrontend::TicketArticle::Action::Reply
      - AgentFrontend::TicketArticle::Action::ReplyAll
      - AgentFrontend::TicketArticle::Action::ReplyToNote
      - AgentFrontend::TicketArticle::Action::ReplyViaSms
      - AgentFrontend::TicketArticle::Action::Split
      - AgentFrontend::TicketArticle::Action::UnmarkAsImportant
      - AgentFrontend::TicketCreate::Email
      - AgentFrontend::TicketCreate::Phone
      - AgentFrontend::TicketCreate::Process
      - AgentFrontend::TicketCreate::SMS
      - AgentFrontend::TicketDetailView
      - AgentFrontend::TicketDetailView::Property
      - AgentFrontend::TicketList::Bulk
      - AgentFrontend::TicketList::Filters
      - ...
      # Limit the functions on external interface.
      - ExternalFrontend::PersonalPreferences
      - ExternalFrontend::ProcessTicketCreate
      - ExternalFrontend::ProcessTicketNextStep
      - ExternalFrontend::Ticket::ExportList     # used for technical purpose, not␣
↪for ACLs
      - ExternalFrontend::Ticket::List           # used for technical purpose, not␣
↪for ACLs
      - ExternalFrontend::Ticket::Print
      - ExternalFrontend::TicketCreate
      - ExternalFrontend::TicketDetailView
      - ...
      Form:
      # This holds the configuration for the visibility of dynamic fields.
      - list of dynamic field names
      Process:
      # Limit the number of possible processes that can be started.
      - Process-9c378d7cc59f0fce4cee7bb9995ee3eb
      - Process-12345678901234567890123456789012
      - ...
      Ticket:
      # Possible ticket options (white list).
        DynamicField_Field1:
        - some value
        DynamicField_MyField:
        - some value
        # more dynamic fields
        NewOwner:
        # For ticket action screens, where the Owner is already set.
        - some owner
        OldOwner:
        # For ticket action screens, where the Owner is already set.
```

(continues on next page)

(continued from previous page)

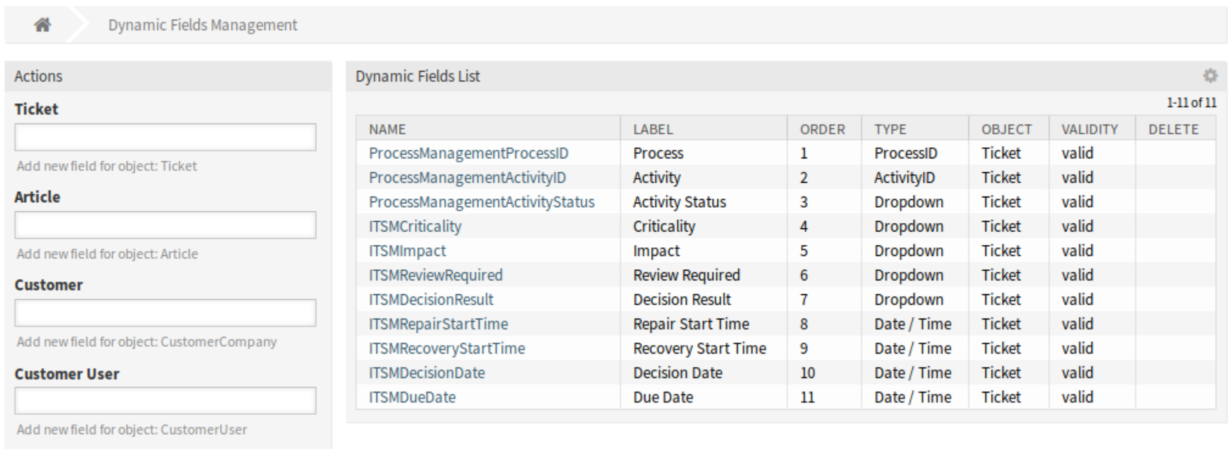
```
        - some owner
        Owner:
        # For ticket create screens, because Owner is not set yet.
        - some owner
        Priority:
        - 5 very high
        Queue:
        - Raw
        - some other queue
        Service:
        - some service
        ServiceID:
        - some service ID
        SLA:
        - some SLA
        SLAID:
        - some SLA ID
        State:
        - some state
        StateID:
        - some state ID
        # more ticket attributes
    PossibleAdd:
       # Add options (white list).
       # See section "Possible", the same configuration can be used here.
    PossibleNot:
       # Remove options (black list).
       # See section "Possible", the same configuration can be used here.
  CreateBy: root@localhost
  CreateTime: 2020-04-15 16:46:23
  Description: This is the long description of the ACL to explain its usage.
  ID: 1
  Name: 200-ACL-Reference
  StopAfterMatch: 0
  ValidID: 3
```

5.2 Dynamic Fields

Beside general information that required for all tickets, organizations have individual needs to add specific details to tickets. This needed information takes various formats like texts, integers, date-time and more.

OTRS supports adding a so-called *dynamic field* to handle texts, integers, drop-down lists, multi-select fields, date-time, checkboxes and more. OTRS administrators can define where those fields should be visible or editable, and of course, the dynamic fields are also available in statistics and reports.

Use this screen to manage dynamic fields in the system. A fresh OTRS installation contains three dynamic fields by default. The dynamic field management screen is available in the *Dynamic Fields* module of the *Processes & Automation* group.



Dynamic Fields Management

Actions

Ticket

Add new field for object: Ticket

Article

Add new field for object: Article

Customer

Add new field for object: CustomerCompany

Customer User

Add new field for object: CustomerUser

Dynamic Fields List

1-11 of 11

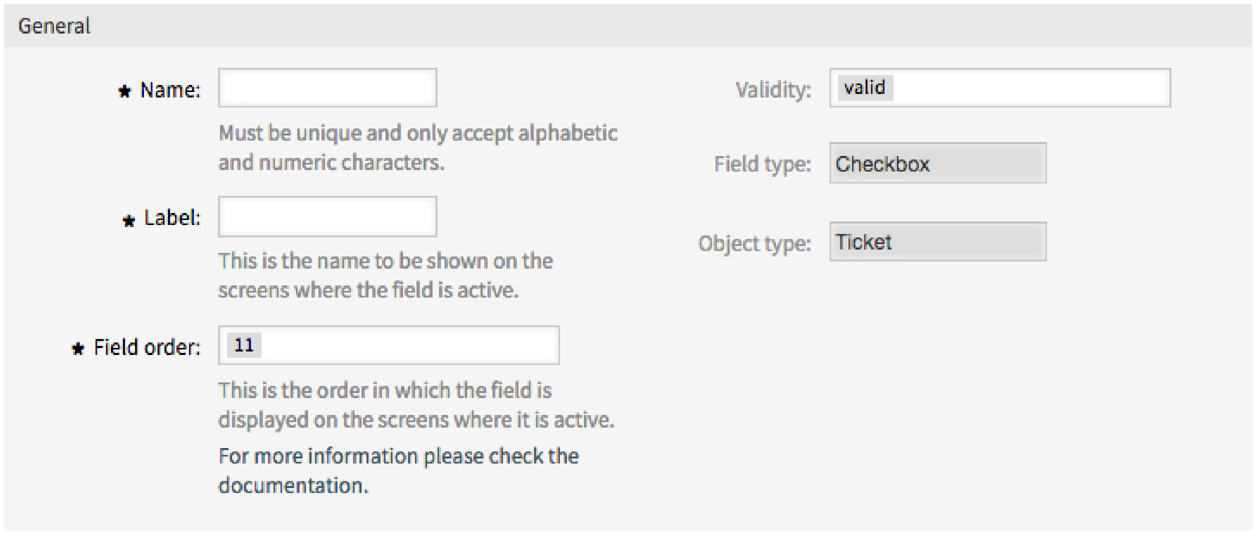
NAME	LABEL	ORDER	TYPE	OBJECT	VALIDITY	DELETE
ProcessManagementProcessID	Process	1	ProcessID	Ticket	valid	
ProcessManagementActivityID	Activity	2	ActivityID	Ticket	valid	
ProcessManagementActivityStatus	Activity Status	3	Dropdown	Ticket	valid	
ITSMCriticality	Criticality	4	Dropdown	Ticket	valid	
ITSMImpact	Impact	5	Dropdown	Ticket	valid	
ITSMReviewRequired	Review Required	6	Dropdown	Ticket	valid	
ITSMDecisionResult	Decision Result	7	Dropdown	Ticket	valid	
ITSMRepairStartTime	Repair Start Time	8	Date / Time	Ticket	valid	
ITSMRecoveryStartTime	Recovery Start Time	9	Date / Time	Ticket	valid	
ITSMDecisionDate	Decision Date	10	Date / Time	Ticket	valid	
ITSMDueDate	Due Date	11	Date / Time	Ticket	valid	

Fig. 17: Dynamic Field Management Screen

5.2.1 Manage Dynamic Fields

To create a new dynamic field:

1. Choose an object in the left sidebar and select a dynamic field type from its drop-down.
2. Fill in the required fields.
3. Click on the *Save* button.



General

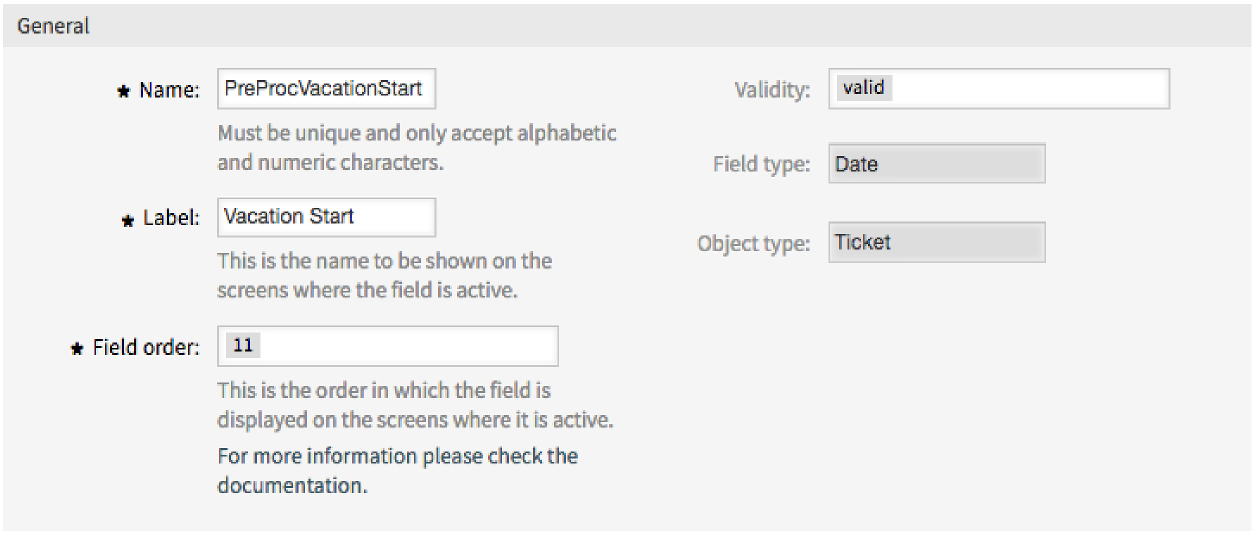
★ Name:	
	Must be unique and only accept alphabetic and numeric characters.
★ Label:	
	This is the name to be shown on the screens where the field is active.
★ Field order:	11
	This is the order in which the field is displayed on the screens where it is active.
	For more information please check the documentation.
Validity:	valid
Field type:	Checkbox
Object type:	Ticket

Fig. 18: Create New Dynamic Field Screen

To edit a dynamic field:

1. Click on a dynamic field in the list of dynamic fields.
2. Modify the fields.
3. Click on the *Save* or *Save and finish* button.

To delete a dynamic field:

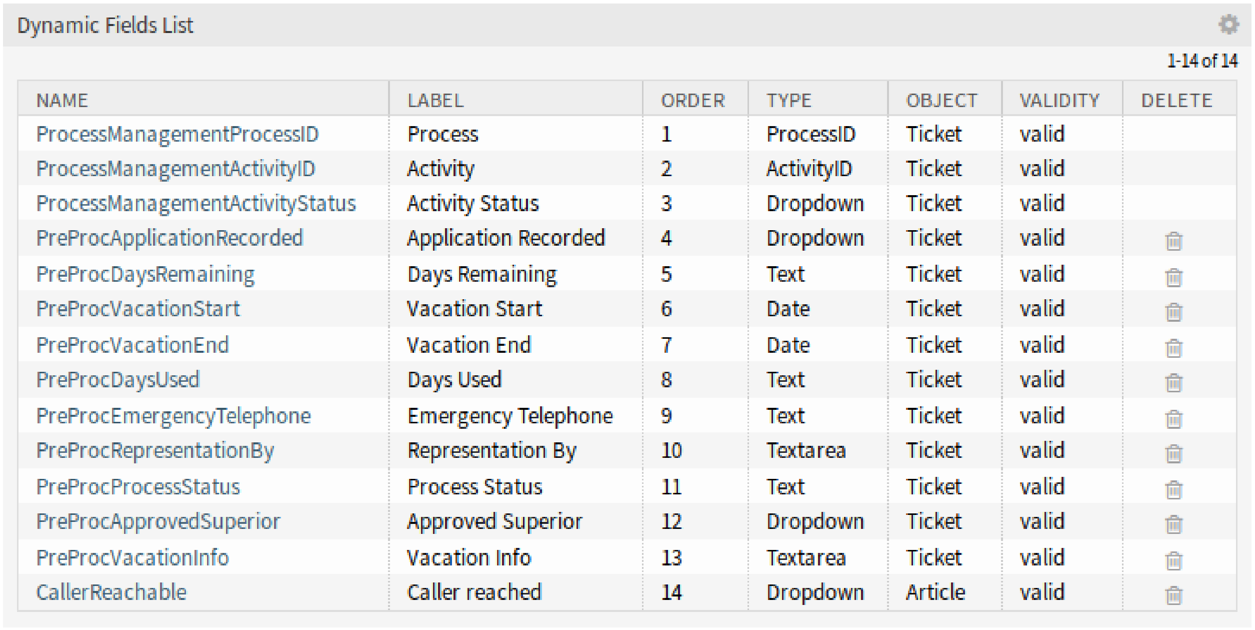


General

★ Name:	PreProcVacationStart	Validity:	valid
	Must be unique and only accept alphabetic and numeric characters.	Field type:	Date
★ Label:	Vacation Start	Object type:	Ticket
	This is the name to be shown on the screens where the field is active.		
★ Field order:	11		
	This is the order in which the field is displayed on the screens where it is active. For more information please check the documentation.		

Fig. 19: Edit Dynamic Field Screen

1. Click on the trash icon in the last column of the overview table.
2. Click on the *Confirm* button.



Dynamic Fields List

1-14 of 14

NAME	LABEL	ORDER	TYPE	OBJECT	VALIDITY	DELETE
ProcessManagementProcessID	Process	1	ProcessID	Ticket	valid	
ProcessManagementActivityID	Activity	2	ActivityID	Ticket	valid	
ProcessManagementActivityStatus	Activity Status	3	Dropdown	Ticket	valid	
PreProcApplicationRecorded	Application Recorded	4	Dropdown	Ticket	valid	🗑
PreProcDaysRemaining	Days Remaining	5	Text	Ticket	valid	🗑
PreProcVacationStart	Vacation Start	6	Date	Ticket	valid	🗑
PreProcVacationEnd	Vacation End	7	Date	Ticket	valid	🗑
PreProcDaysUsed	Days Used	8	Text	Ticket	valid	🗑
PreProcEmergencyTelephone	Emergency Telephone	9	Text	Ticket	valid	🗑
PreProcRepresentationBy	Representation By	10	Textarea	Ticket	valid	🗑
PreProcProcessStatus	Process Status	11	Text	Ticket	valid	🗑
PreProcApprovedSuperior	Approved Superior	12	Dropdown	Ticket	valid	🗑
PreProcVacationInfo	Vacation Info	13	Textarea	Ticket	valid	🗑
CallerReachable	Caller reached	14	Dropdown	Article	valid	🗑

Fig. 20: Delete Dynamic Field Screen

Note: If several dynamic fields are added to the system, use the filter box to find a particular dynamic field by just typing the name to filter.

Warning: The maximum number of 300 *valid* dynamic fields should not be exceeded. Exceeding this limit may affect the system performance.

5.2.2 Built-in Dynamic Fields

OTRS comes with pre-defined dynamic fields.

Warning: Do not delete these dynamic fields. They are required to work the process management and the system monitoring properly.

`ProcessManagementProcessID`
This dynamic field will store the process ID from the *Process Management* when a process is started.

`ProcessManagementActivityID`
This dynamic field will store the activity ID from the *Process Management* while a process is running.

`ProcessManagementActivityStatus`
This dynamic field will store the activity status from the *Process Management* while a process is running.

`SystemMonitoringHost`
This dynamic field will store the host name comes from the *System Monitoring* suite.

`SystemMonitoringService`
This dynamic field will store the service name comes from the *System Monitoring* suite.

`SystemMonitoringState`
This dynamic field will store the state name comes from the *System Monitoring* suite.

`ITSMCriticality`
This is a drop-down dynamic field that contains criticality levels from *1 very low* to *5 very high*.

`ITSMImpact`
This is a drop-down dynamic field that contains impact levels from *1 very low* to *5 very high*.

`ITSMReviewRequired`
This is a drop-down dynamic field that contains *Yes* and *No* to indicate if a review is required.

`ITSMDecisionResult`
This is a drop-down dynamic field that contains some possible results for decisions.

`ITSMRepairStartTime`
This is a date/time dynamic field for holding the repair start time.

`ITSMRecoveryStartTime`
This is a date/time dynamic field for holding the recovery start time.

`ITSMDecisionDate`
This is a date/time dynamic field for holding the decision time.

`ITSMDueDate`
This is a date/time dynamic field for holding the due date.

The dynamic fields are activated in many screens by default.

To see the complete list of screens:

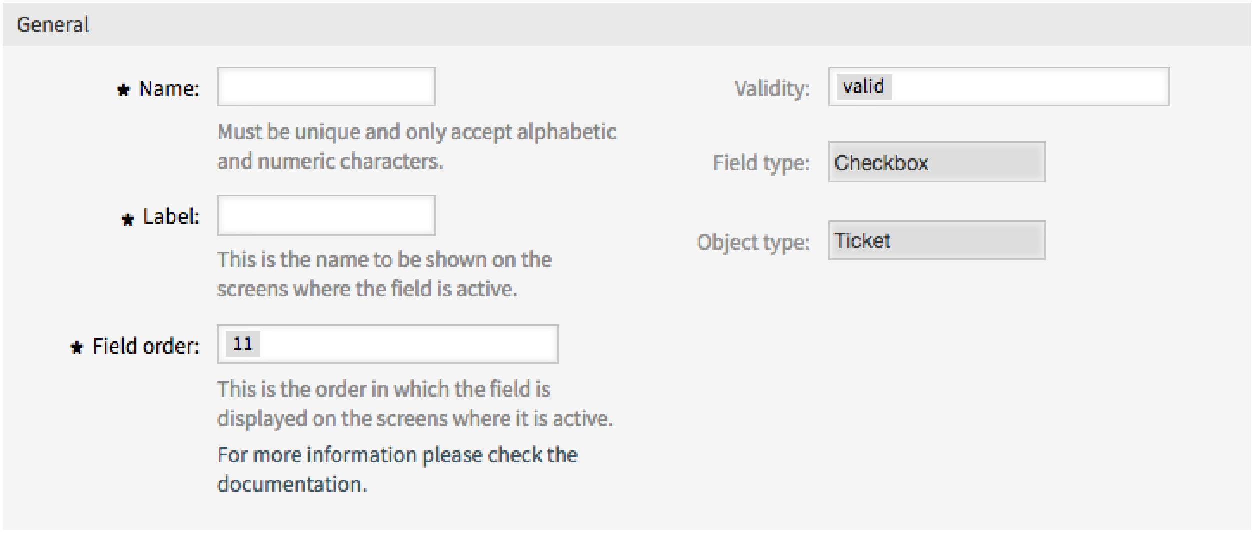
1. Go to the *System Configuration* screen.
2. Navigate to `Frontend` → `Agent` → `View` or `Frontend` → `External` → `View` to see the screens.

5.2.3 Dynamic Field Settings

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

General Dynamic Field Settings

These settings are the same for all types of dynamic fields.



General

★ Name:		Validity:	valid
	Must be unique and only accept alphabetic and numeric characters.	Field type:	Checkbox
★ Label:		Object type:	Ticket
	This is the name to be shown on the screens where the field is active.		
★ Field order:	11		
	This is the order in which the field is displayed on the screens where it is active. For more information please check the documentation.		

Fig. 21: Dynamic Field General Screen

Name *
The name of this resource. Any type of characters can be entered to this field including uppercase letters and spaces. The name will be displayed in the overview table.

Label *
This is the name to be shown on the screens where the field is active.

See also:

It is possible to add translations for a dynamic field label. Label translations have to be added manually to language translation files.

Field order *
This is the order in which the field is displayed on the screens where it is active.

Note: The configured value for the field order is considered in the following screens:

- all screens in the administrator interface
- all screens in the external interface
- *Business Process Information* widget in the agent interface

In other screens the order can be configured in other ways, e.g. via *a specific order of fields in the form configuration*.

Validity *
: Set the validity of this resource. Each resource can be used in OTRS only, if this field is set to *valid*. Setting this field to *invalid* or *invalid-temporarily* will disable the use of the resource.

Field type
: This type have been selected in the previous page and can not be changed here anymore. This is a read-only field.

Object type
: This type have been selected in the previous page and can not be changed here anymore. This is a read-only field.

Note: The object type determines where the dynamic field can be used. For example dynamic field with object type *Ticket* can be used only in tickets, and can not be used in articles.

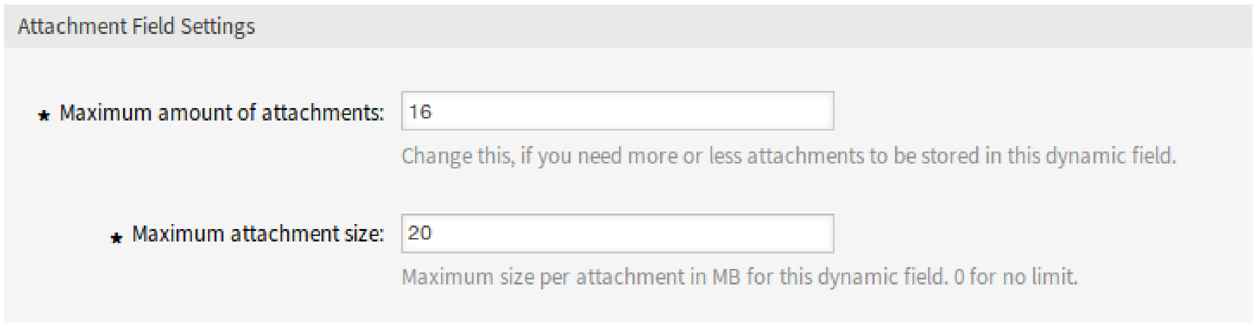
The following settings are relevant only for the particular type of dynamic fields.

Attachment Dynamic Field Settings

Attachment dynamic field is used to store attachments for tickets and articles.

Article dynamic fields can hold attachments for each ticket article. Attachments stored in an article dynamic field of type attachment can just be downloaded, but not deleted or changed.

Ticket dynamic fields can hold attachments for each ticket. Attachments stored in a ticket dynamic field of type attachment are stored once for each ticket and can be downloaded and deleted.


Attachment Field Settings

* Maximum amount of attachments: 16
Change this, if you need more or less attachments to be stored in this dynamic field.

* Maximum attachment size: 20
Maximum size per attachment in MB for this dynamic field. 0 for no limit.

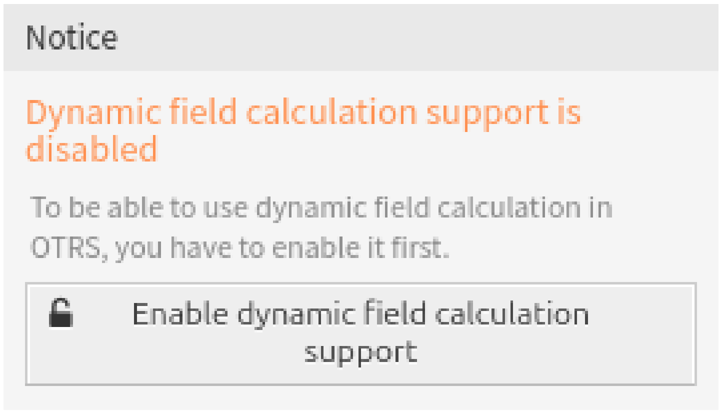
Fig. 22: Attachment Dynamic Field Settings

Maximum amount of attachments *
: This setting holds the amount of files that can be stored in the dynamic field of type attachment per ticket or article. Increasing this value enables the agents to add more files. Reducing this amount does not delete attachments already stored in dynamic fields of type attachment, but reduces the possibility for adding files up to the configured maximum.

Maximum attachment size *
: This setting holds the maximum file size in megabyte each uploaded file can have. If an agent is uploading a file greater than the here configured file size, the file is accepted initially. As soon as the upload is done, the size is checked and the file gets deleted as well as the user informed that the file was not stored because it exceeded the maximum attachment size limit.

Calculation Dynamic Field Settings

Note: To be able to use dynamic field calculation in OTRS, you have to enable it first.



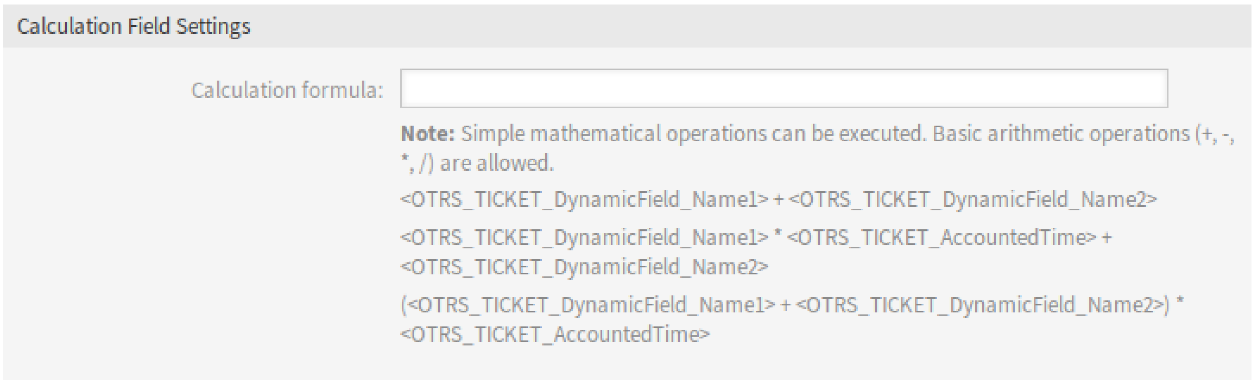
Notice

Dynamic field calculation support is disabled

To be able to use dynamic field calculation in OTRS, you have to enable it first.

Fig. 23: Enable Dynamic Field Calculation Support

Calculation dynamic field is used to perform simple mathematical operations.



Calculation Field Settings

Calculation formula:

Note: Simple mathematical operations can be executed. Basic arithmetic operations (+, -, *, /) are allowed.

<OTRS_TICKET_DynamicField_Name1> + <OTRS_TICKET_DynamicField_Name2>

<OTRS_TICKET_DynamicField_Name1> * <OTRS_TICKET_AccountedTime> + <OTRS_TICKET_DynamicField_Name2>

(<OTRS_TICKET_DynamicField_Name1> + <OTRS_TICKET_DynamicField_Name2>) * <OTRS_TICKET_AccountedTime>

Fig. 24: Calculation Dynamic Field Settings

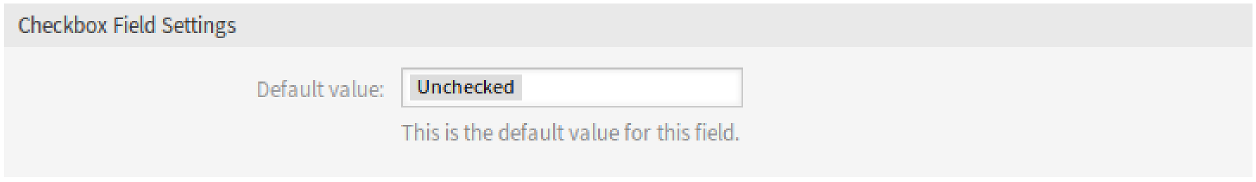
Calculation formula

This setting is used to add a calculation formula. Simple mathematical operations are possible. The default value is zero.

Numbers in the formula should have dot as decimal separator and should be used without thousands separator like `123456.78`.

Checkbox Dynamic Field Settings

Checkbox dynamic field is used to store true or false value.



Checkbox Field Settings

Default value: Unchecked

This is the default value for this field.

Fig. 25: Checkbox Dynamic Field Settings

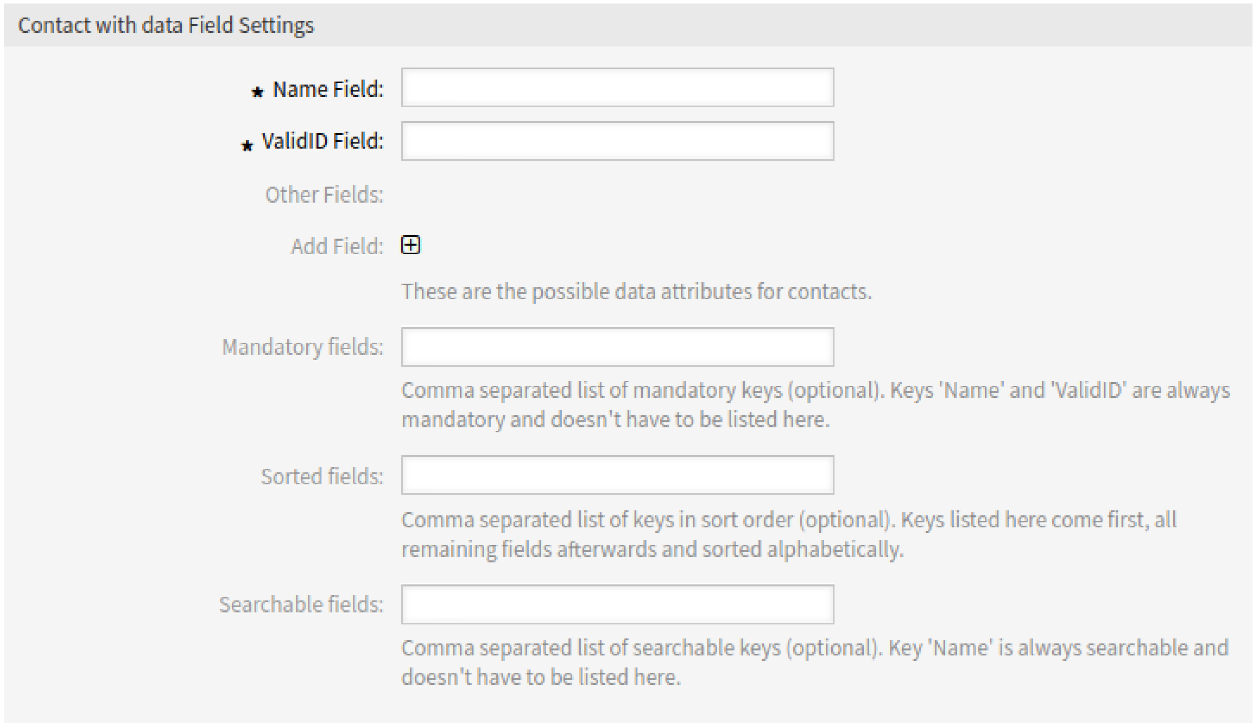
Default value *
The default value for the checkbox.

Checked
The checkbox is checked by default.

Unchecked
The checkbox is unchecked by default.

Contact With Data Dynamic Field Settings

This dynamic field allows to add contacts with data to tickets.



Contact with data Field Settings

★ Name Field:

★ ValidID Field:

Other Fields:

Add Field:

These are the possible data attributes for contacts.

Mandatory fields:

Comma separated list of mandatory keys (optional). Keys 'Name' and 'ValidID' are always mandatory and doesn't have to be listed here.

Sorted fields:

Comma separated list of keys in sort order (optional). Keys listed here come first, all remaining fields afterwards and sorted alphabetically.

Searchable fields:

Comma separated list of searchable keys (optional). Key 'Name' is always searchable and doesn't have to be listed here.

Fig. 26: Contact With Data Dynamic Field Settings

Name Field *
The attribute `Name` is always mandatory and it is not automatically added, so for each new data source

this attribute must be added manually. Within the data source definition (or dynamic field configuration) this must be represented by the key `Name` and the value could be *Name* for example.

ValidID Field *
The attribute `ValidID` is always mandatory and it is not automatically added, so for each new data source this attribute must be added manually. Within the data source definition (or dynamic field configuration) this must be represented by the key `ValidID` and the value could be *Validity* for example.

Other Fields
These are the possible data attributes for contacts. Clicking on the button of the *Add Fields* will add two new fields, where a *Key* (internal value) and a *Value* (displayed value) can be set. With the button you can add multiple key-value pairs.

Warning: The keys `Name` and `ValidID` are already used by *Name Field* and *ValidID Field*. Do not use these keys again!

Add Fields
Use this button to add more fields to dynamic field.

Mandatory fields
Comma separated list of mandatory keys.

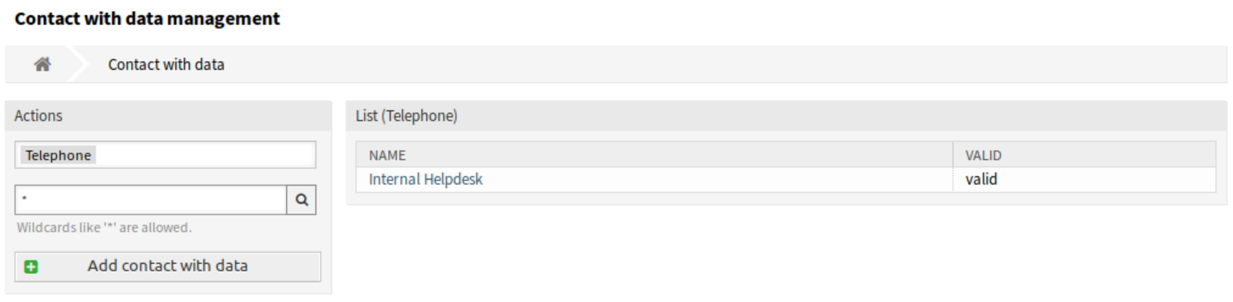
Note: Keys `Name` and `ValidID` are always mandatory and doesn' t have to be listed here.

Sorted fields
Comma separated list of keys in sort order. Keys listed here come first, all remaining fields afterwards and sorted alphabetically.

Searchable fields
Comma separated list of searchable keys.

Note: Key `Name` is always searchable and doesn' t have to be listed here.

When the dynamic field was saved, click on the name of the newly created dynamic field in the overview table. The *Edit Dynamic Field* screen will open. There is a button *Add or edit contacts*, that points to *Tickets* → *Edit contacts with data* to add some data.

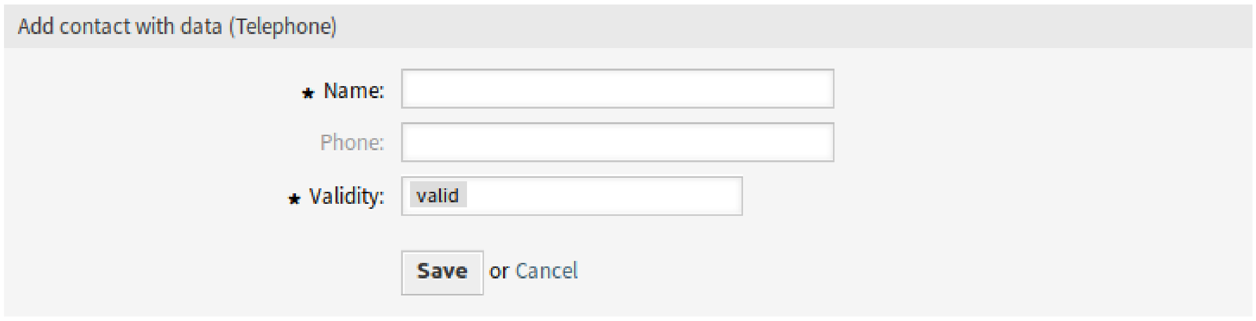


NAME	VALID
Internal Helpdesk	valid

Fig. 27: Contact With Data Management Screen

To add a new contact with data:

1. Select a dynamic field from the drop-down list in *Actions* widget of the left sidebar.
2. Click on the *Add contact with data* button in the left sidebar.
3. Fill in the required fields.
4. Click on the *Save* button.

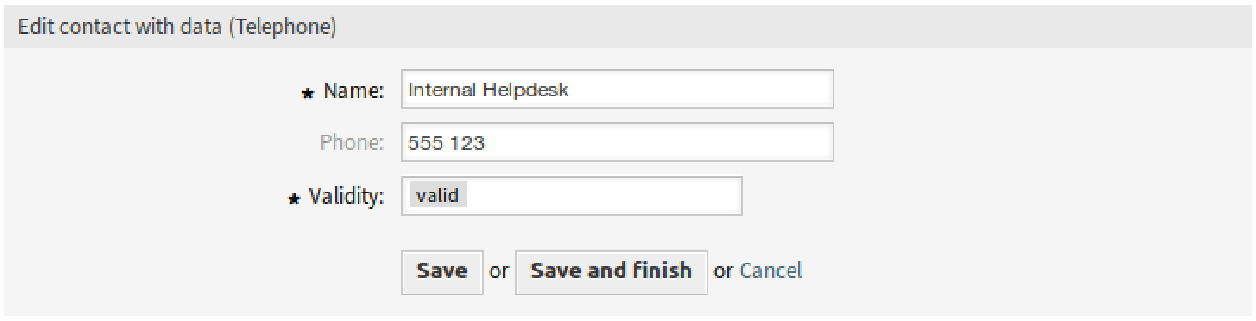

Add contact with data (Telephone)

★ Name:	
Phone:	
★ Validity:	valid

Fig. 28: Add Contact With Data Screen

To edit a contact with data:

1. Click on a contact with data in the list of contact with data entries.
2. Modify the fields.
3. Click on the *Save* or *Save and finish* button.


Edit contact with data (Telephone)

★ Name:	Internal Helpdesk
Phone:	555 123
★ Validity:	valid

Fig. 29: Edit Contact With Data Screen

The usage of this type of dynamic field is more complex then the others. An exemplary usage of contacts with data is as follows:

1. Create a new dynamic field of type contact with data.
2. Set the possible contact attributes (possible values).
   - Type *Name* into *Name Field*.
   - Type *Validity* into *ValidID Field*.
   - Add any other attribute with *Add Fields* button such as `Telephone` attribute (key: `Telephone`, value: *Phone*).
3. Add the list of mandatory attribute keys comma separated (`Name` and `ValidID` are not needed).
4. Set the attribute key order list comma separated as: `Name,Telephone,ValidID`.
5. Add the list of searchable attribute keys comma separated (`Name` is not needed).

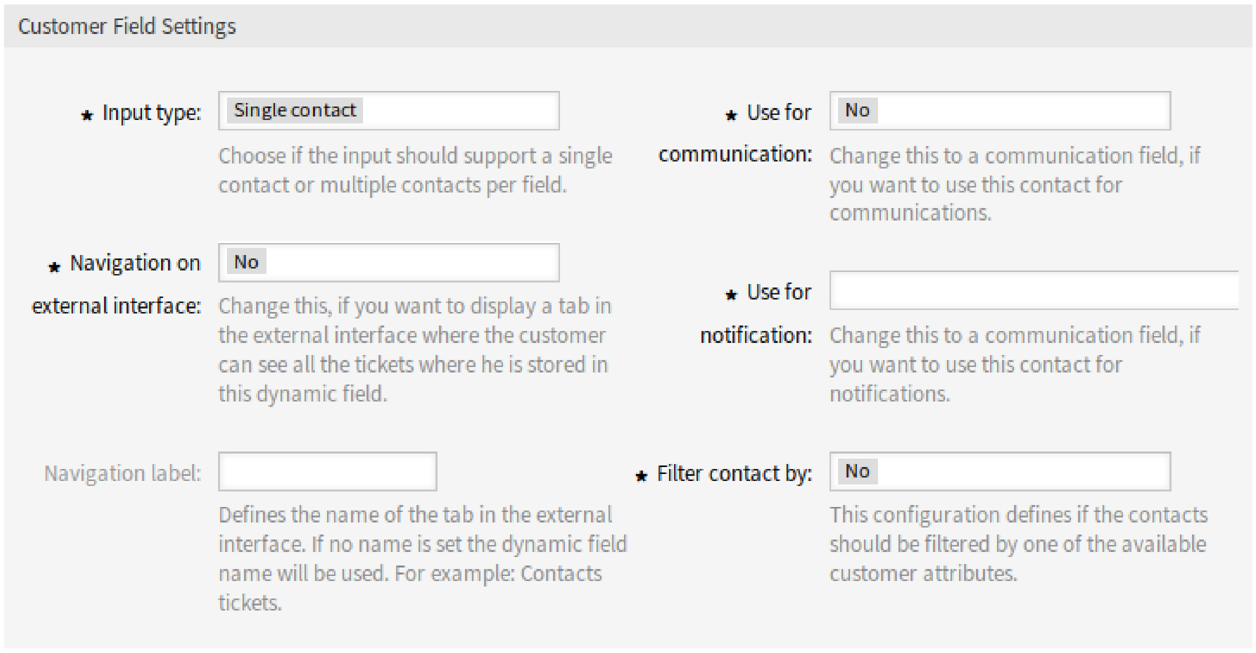
6. Populate the data source by adding at least one contact in the newly created data source by using *Manage Contacts with Data* screen from the main menu of the agent interface.
7. Add the new dynamic field to the screen's configuration where it should be shown. For example in *New Phone Ticket* screen by updating the system configuration setting `Forms###AgentFrontend::TicketCreate::Phone::CreateProperties` and do the same for `AgentFrontend::TicketDetailView::WidgetType###Properties`.

   See also:

   See *Display Dynamic Fields on Screens* for more information.
8. Go to *New Phone Ticket* screen, and notice that the new field is there. Add all needed information to the ticket.
9. Select an existing contact using autocomplete and choosing a contact.
10. Click on the *Create* button to create the ticket and go to the ticket detail view.
11. Open the widget configuration of the *Properties* widget in the ticket detail view, and enable the dynamic field in the *Hide/Show Properties* list.
12. The assigned contact and its attributes will be shown in the ticket detail view.
13. It is possible to update the attributes of the contact by clicking the *Edit contact* button that appears in the right side of dynamic field property card (if the current user is a member of the groups defined in system configuration setting `Frontend::Module###AdminDynamicFieldContactWithData`).
14. If is necessary to change the contact for this ticket, it can be done via any other ticket action where the dynamic field is configured for display.

Customer Dynamic Field Settings

Customer dynamic field is used to store customer data from integrated external customer databases as contacts in a ticket.


Customer Field Settings

⋆ Input type:	Single contact Choose if the input should support a single contact or multiple contacts per field.	⋆ Use for communication:	No Change this to a communication field, if you want to use this contact for communications.
⋆ Navigation on external interface:	No Change this, if you want to display a tab in the external interface where the customer can see all the tickets where he is stored in this dynamic field.	⋆ Use for notification:	 Change this to a communication field, if you want to use this contact for notifications.
Navigation label:	 Defines the name of the tab in the external interface. If no name is set the dynamic field name will be used. For example: Contacts tickets.	⋆ Filter contact by:	No This configuration defines if the contacts should be filtered by one of the available customer attributes.

Fig. 30: Customer Dynamic Field Settings

Input type *
: Defines whether the dynamic field should be able to store a *Single contact* or *Multiple contacts*.

Navigation on external interface *
: Defines whether or not a new navigation entry in the external interface should be shown. This navigation entry will only be shown to customer users that are stored in one of the *Customer* fields. The label of this entry can be defined in the setting *Navigation label* if this setting is saved with *yes*.

Navigation label
: This configuration makes it possible to define a custom label for the navigation entry displayed in the external interface. If no value is configured, a default combination of the field label and *Tickets* will be used, for example *Contacts Tickets*.

Use for communication *
: If customer users stored in a dynamic field of type *Customer* should get used for communication, this configuration has to be used. It is possible to use customer users for the *To*, *Cc* and *Bcc* field. If this configuration is active, the customer user will be added to the configured field. If the field is already containing the address of the customer user, it will not get added a second time. The agent still has the opportunity to remove the address from the field when composing a new message.

Use for notification *
: If customer users stored in a dynamic field of type *Customer* should get used for notification, this configuration has to be used. It is possible to use customer users for the *To*, *Cc* and *Bcc* recipient. If this configuration is active, the dynamic field will be shown in the recipient block of the ticket notifications.

  Note: If the configuration option `UserForNotification` is not set in the dynamic field table, the fallback recipient will be used. The fallback recipient is the *To* recipient.

Filter contact by *
: In some scenarios it is necessary to filter possible contacts by their attributes. A filter for example could be the customer number, the city or a custom attribute. To activate the filter functionality, select the needed attribute from the drop-down list. If the customer number is selected, the contact will be filtered by the customer number of the customer of the ticket. For each other attribute the filter can be defined as a text value. After selecting the filter attribute from the drop-down list, a second input field will be displayed where the filter can be defined.

Filter criteria *
: In this field the filter of the attributes for the customer result is defined. For example, if only customer users with the first name *Theo* should be selectable, the filter *Fistname* from the *Filter contact by* has to be selected. After that the *Filter criteria* field will be displayed. Now we can insert *Theo* into the filter field. Currently it is not possible to define a regular expression or a placeholder for filtering customer users.

After the dynamic field of the type *Customer* was configured and added to the different views, the functionality can get used. The contact fields will be displayed below the recipients for new tickets or otherwise in the dynamic fields block of the page. The exact position and label of the field depends on the field configuration.

If a customer number filter was configured the dynamic field will be locked and unchangeable as long as no customer is added to the ticket.

After inserting at least one character into the text field of the dynamic field a search over the configured customer databases starts. At this point the configured filter takes it place and removes unmatched customer users from the result set. To start a search for all available customer users for this field you can use the `*` wildcard.

If a field has been configured to use the contacts for communication and the *Reply via Email* or *Reply to All via Email* actions are used in the ticket detail view, all customer users stored in the respective field will be added to the configured recipient list (to, cc or bcc) unless they exist as a recipient already.

Note: The field must not be configured in the dynamic field section of the reply actions.

To allow searching of the *Customer* dynamic field values after the data was stored to the ticket the search view has to be configured accordingly.

The search works equals to the search for regular customer users. The login name of the customer user has to be used as search criteria.

Database Dynamic Field Settings

Database dynamic field is used to store data from external databases in tickets.

Before an external database can be searched and the results be saved at the ticket through the dynamic field, the credentials have to be stored in the configuration of the dynamic field.

Possible values
The possible values will fill up the *Identifier* field below automatically, which defines the value that will be stored in the dynamic field. Possible values can be created as much as needed (or at least as many table columns as the database table has). The possible values defines the database columns to search in. It is possible to set the column name, a description (label) the field should have, the needed data type and if the field should be a search or list field.

Name *
The exact name of the database column which will be requested through the database queries.

Label *
The label of the field which will be displayed in the detailed search.

Datatype *
The data type which will be stored in the dynamic field. Possible values: *Date*, *Integer* or *Text*.

Filter
With the filter field, it is possible to choose a ticket attribute or a dynamic field as a filter for the related column. If the dynamic field is bound to a related ticket, the attributes will be used for the filter mechanism, otherwise the filters will be ignored. If the filter will be configured to a table column, only search results matching to the search term and the related ticket attribute on exactly the configured column will be displayed.

Searchfield
Indicates if a field should be included in the search requests.

Listfield
Indicates if a field should be displayed in the results.

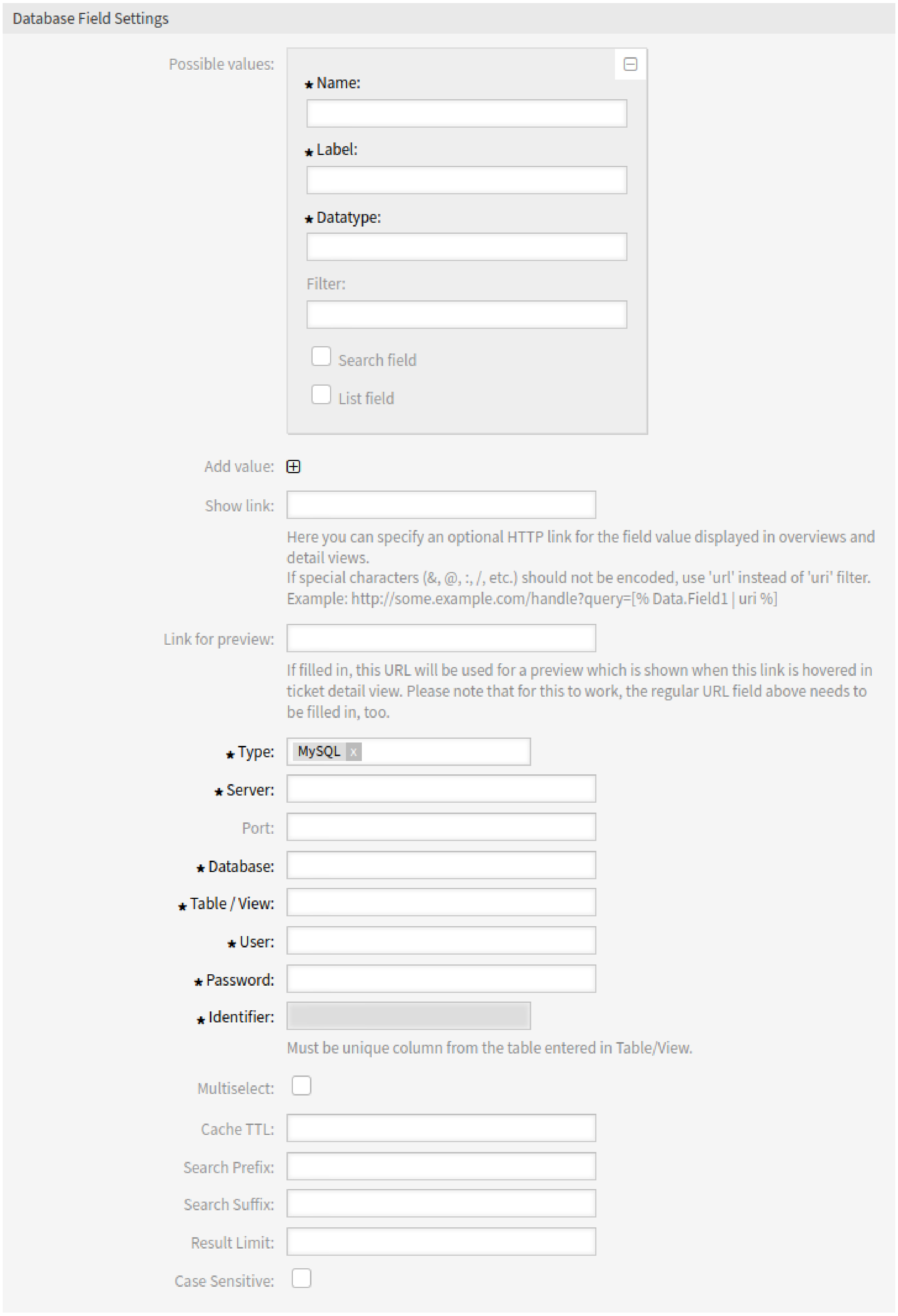
Add value
Click on the + icon to add a new section to *Possible values*.

Show link
Here you can specify an optional HTTP link for the field value displayed in overviews and detail views. Examples:

- The value of dynamic field named `Field1` is part of the link:

```
https://some.example.com/handle?query=[% Data.Field1 | uri %]
```

- The value of dynamic field named `Field1` is the complete link with and without protocol:


Database Field Settings

Possible values:

★ Name:

★ Label:

★ Datatype:

Filter:

☐ Search field

☐ List field

Add value:

Show link:

Here you can specify an optional HTTP link for the field value displayed in overviews and detail views.
If special characters (&, @, :, /, etc.) should not be encoded, use 'url' instead of 'uri' filter.
Example: http://some.example.com/handle?query=[% Data.Field1 | uri %]

Link for preview:

If filled in, this URL will be used for a preview which is shown when this link is hovered in ticket detail view. Please note that for this to work, the regular URL field above needs to be filled in, too.

★ Type: MySQL

★ Server:

Port:

★ Database:

★ Table / View:

★ User:

★ Password:

★ Identifier:

Must be unique column from the table entered in Table/View.

Multiselect: ☐

Cache TTL:

Search Prefix:

Search Suffix:

Result Limit:

Case Sensitive: ☐

Fig. 31: Database Dynamic Field Settings

```
[% Data.Field1 %]
https://[% Data.Field1 %]
```

- The value of dynamic field named `100Field` needs special handling, because the name of the dynamic field starts with a number:

```
https://some.example.com/handle?query=[% Data.item("100Field") | uri %]
```

If special characters (&, @, :, /, etc.) should not be encoded, use `url` instead of `uri` filter.

Link for preview
If filled in, this URL will be used for a preview which is shown when this link is hovered in ticket detail view. Please note that for this to work, the regular URL field above needs to be filled in, too.

Type *
The type of the desired database can be selected here. The following database types are supported: MySQL, Oracle, PostgreSQL or MSSQL.

SID
This option is only available for Oracle connections and will be shown or hidden automatically. Within this option you have to enter the SID of your Oracle connection.

Driver
This option is only available for ODBC connections and will be shown or hidden automatically. Within this option you have to enter in the host system previously configured ODBC driver to connect to the desired MSSQL database.

Server *
The database host (host name or IP address).

Port
The port of the database server.

Database *
Defines the desired target database of the DBMS. This database will be used for queries.

Table/View *
This table or view will be used for the queries.

User *
The username for the database connection.

Password *
The user password for the database connection.

Identifier *
This select box will be automatically filled through *Possible values*. This field represents the value which will be stored in the dynamic field. Must be unique column from the table entered in *Table/View*.

Multiselect
If this field is selected, it will be possible to store more than one value to the dynamic field. Those values will be stored comma separated.

Cache TTL
This value defines the period of validity of the database cache in seconds. Equal queries to the database will be answered through the cache (local file system) within this period instead of asking the database again.

Search Prefix
This value will be put in the front of every search term while using the auto-completion to search the

database. Wildcard characters are supported as well. The search prefix will be ignored during the detailed search, but it is still possible to use wildcard characters in those masks.

Search Suffix
This value will be put in the end of every search term while using the auto-completion to search the database. Wildcard characters are supported as well. The search suffix will be ignored during the detailed search, but it is still possible to use wildcard characters in those masks.

Result Limit
The entered integer value defines the maximum amount of allowed results during a database search. This includes the auto-completion search as well as the detailed search.

Case Sensitive
If this field is selected, case-sensitivity will take effect on searches.

Storage of Historical Data

To store historical data, it is necessary to activate and set the settings in the system configuration.

In the configuration option for the `SourceDynamicField` it is needed to fill in the already created dynamic field name, which will be used to gather the historical data. In the related option `TargetDynamicField` the key have to be filled with the table columns of the connected external database, which will be readout. For every column the related target dynamic field has to be configured in the field content. The gathered data will be saved in these dynamic fields.

If the configuration is ready and active, the configured fields will be readout from the external database, since the source field gets a new value via the configured masks. The data will be searched by its stored identifier via an event module and the found values will be stored in the target dynamic fields.

Note: The mapping of multiselect fields to historical data is not supported.

Searching and Saving Data Sets

After this type of dynamic field is added to screens it is possible to input search terms and therefore execute a search over all configured database fields. Otherwise click on the *Detailed search* link and start a detailed search in which the fields to search in are selected explicitly. Wildcard character * is allowed in every single field.

Since search terms are typed in into the text field, a database search will be started over the configured columns and the result will displayed via an auto-completion below the text field. The more exact the search term is, the more exact will be the result (less result entries).

If the wished value will be displayed in the results, it can be selected via a mouse click or via the keyboard and therefore be added to the dynamic field results.

Via the link *Details* a popup screen can be accessed, which offers detailed information about the whole result row. This information includes the line headers and the data. This information can be used to get an overview about the rest (of the not configured) columns or to compare data. The added result entries can be removed via the minus button.

The link *Detailed search* opens a new modal dialog to start a new database search. In this mask it is possible to select the fields to search on explicitly.

By default the first available field is activated, but it is also possible to remove available fields or add additional ones. Only activated and filled fields are considered for the search. Wildcard character * is allowed in every single field.

The database search will be executed via the *Start search* button and the results will be tabular displayed. If the search was successful, the results will be listed and one of the entries can be selected via a mouse click. The value will be added to the list of saved values afterwards.

Independent of using the auto-completion or the detailed search, every single result can just selected ones. If an agent tries to select a value multiple times, a related warning message is displayed.

Database Autofill Dynamic Field Settings

Database autofill dynamic field is used to provide data from external databases and use them in tickets as template.

Before an external database can be searched and the results be saved at the ticket through the dynamic field, the credentials have to be stored in the configuration of the dynamic field.

This dynamic field operates identical to database dynamic field, but with these differences:

- Always has a single option select field.
- Available only for ticket object.

The settings of this dynamic field are very similar to database dynamic field, but with these differences:

- There is no *Show link* setting.
- A template has to be provided to be used in ticket create screens or in ticket actions.

Template *
: The text entered here will be injected to the article text. You can use the possible values names as a placeholder like this: `<name>`.

In the agent interface, if a ticket is created or a ticket action is selected, the template of the dynamic field will be appended to the end of the body of the form.

Date Dynamic Field Settings

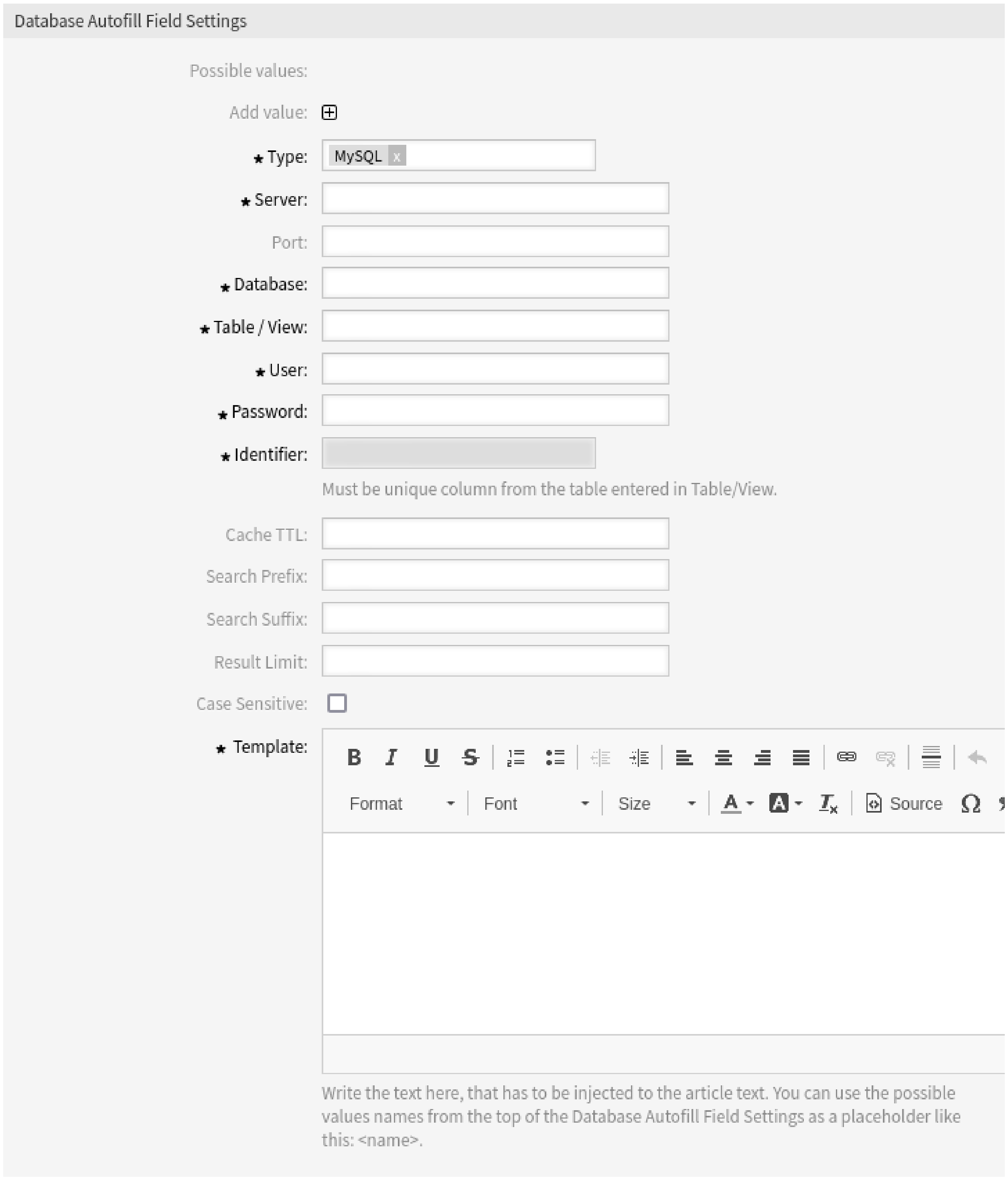
Date dynamic field is used to store a date value.

Default date difference
: The difference from **now** (in seconds) to calculate the field default value (e.g. 3600 or -60).

Define years period
: Activate this feature to define a fixed range of years (in the future and in the past) to be displayed on the year part of the field. If set to *Yes* the following options will be available:

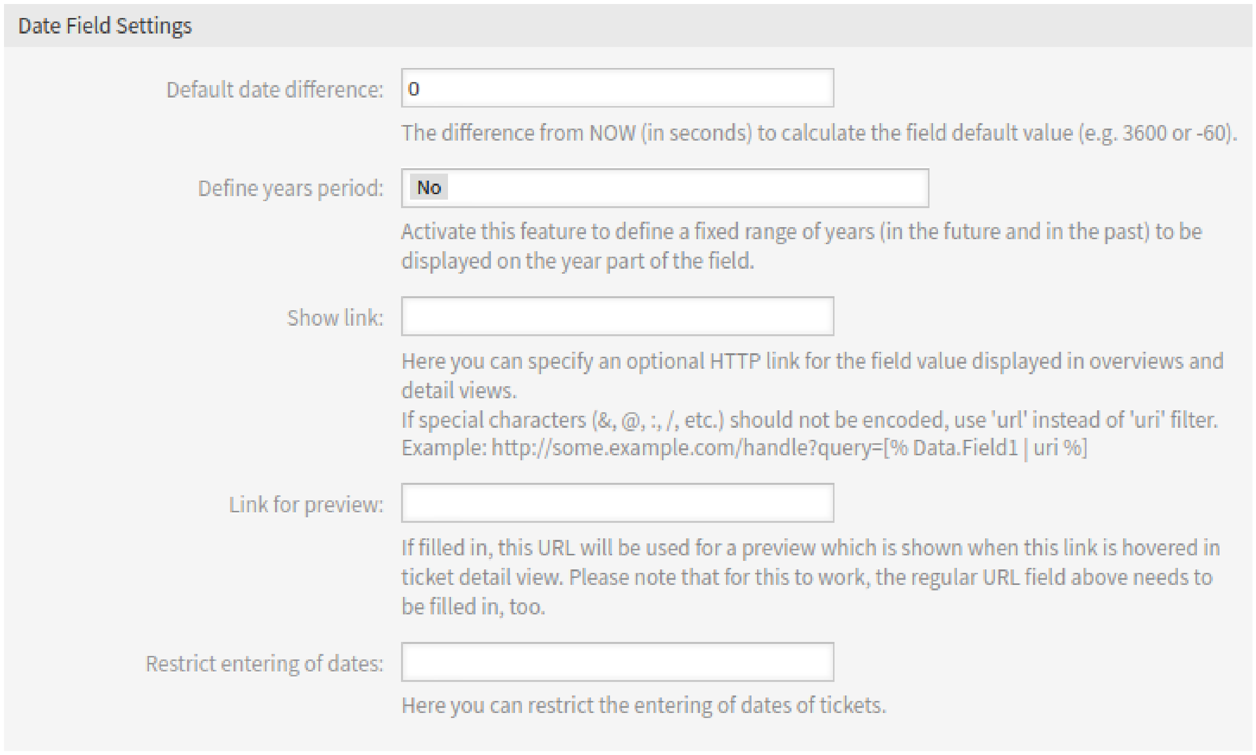
  Years in the past
  : Define the number of years in the past from the current day to display in the year selection for this dynamic field in edit screens.

  Years in the future
  : Define the number of years in the future from the current day to display in the year selection for this dynamic field in edit screens.


Database Autofill Field Settings

Possible values:	
Add value:	⊞
★ Type:	MySQL x
★ Server:	
Port:	
★ Database:	
★ Table / View:	
★ User:	
★ Password:	
★ Identifier:	Must be unique column from the table entered in Table/View.
Cache TTL:	
Search Prefix:	
Search Suffix:	
Result Limit:	
Case Sensitive:	☐
★ Template:	Write the text here, that has to be injected to the article text. You can use the possible values names from the top of the Database Autofill Field Settings as a placeholder like this: <name>.

Fig. 32: Database Autofill Dynamic Field Settings



Date Field Settings

Default date difference: 0

The difference from NOW (in seconds) to calculate the field default value (e.g. 3600 or -60).

Define years period: No

Activate this feature to define a fixed range of years (in the future and in the past) to be displayed on the year part of the field.

Show link:

Here you can specify an optional HTTP link for the field value displayed in overviews and detail views.
If special characters (&, @, :, /, etc.) should not be encoded, use 'url' instead of 'uri' filter.
Example: http://some.example.com/handle?query=[% Data.Field1 | uri %]

Link for preview:

If filled in, this URL will be used for a preview which is shown when this link is hovered in ticket detail view. Please note that for this to work, the regular URL field above needs to be filled in, too.

Restrict entering of dates:

Here you can restrict the entering of dates of tickets.

Fig. 33: Date Dynamic Field Settings

Note: If no years period is specified, the system uses the default value for the period: 10 years (5 years in the past and 5 years in the future).

Show link

Here you can specify an optional HTTP link for the field value displayed in overviews and detail views. Examples:

- The value of dynamic field named `Field1` is part of the link:

```
https://some.example.com/handle?query=[% Data.Field1 | uri %]
```

- The value of dynamic field named `Field1` is the complete link with and without protocol:

```
[% Data.Field1 | uri %]
https://[% Data.Field1 | uri %]
```

- The value of dynamic field named `100Field` needs special handling, because the name of the dynamic field starts with a number:

```
https://some.example.com/handle?query=[% Data.item("100Field") | uri %]
```

- The value of dynamic field named `Field1` contains the link, the value of dynamic field named `Field2` contains an important value for a parameter:

```
https://[% Data.Field1 | uri %]?query=[% Data.DynamicField_Field2 |␣
↪uri %]
```

If special characters (&, @, :, /, etc.) should not be encoded, use `url` instead of `uri` filter.

Link for preview
If filled in, this URL will be used for a preview which is shown when this link is hovered in ticket detail view. Please note that for this to work, the regular URL field above needs to be filled in, too.

See also:

The URL has to be added to the allowed origins. Read the *External Link Previews* chapter for more information.

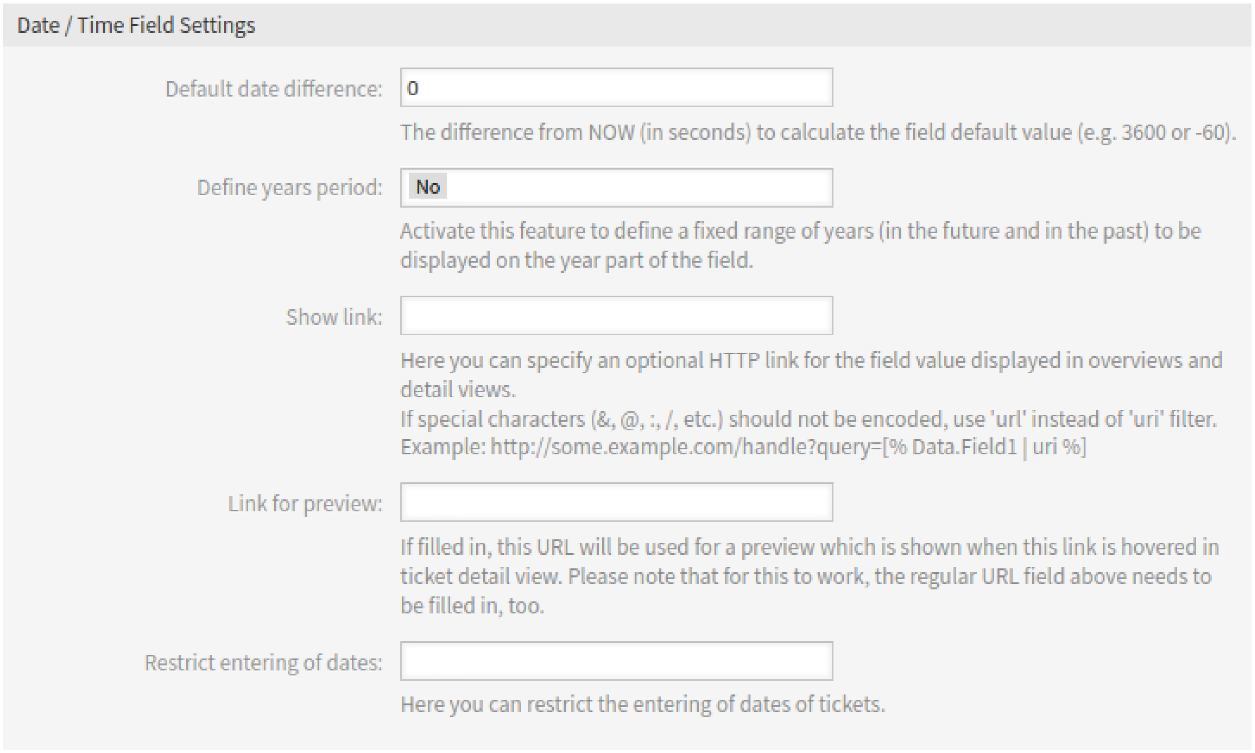
Restrict entering of dates
Here you can restrict the entering of dates of tickets.

Prevent entry of dates in the future
Selecting this option will prevent entering a date that is after the current date.

Prevent entry of dates in the past
Selecting this option will prevent entering a date that is before the current date.

Date / Time Dynamic Field Settings

Date / time dynamic field is used to store a date time value.



Date / Time Field Settings

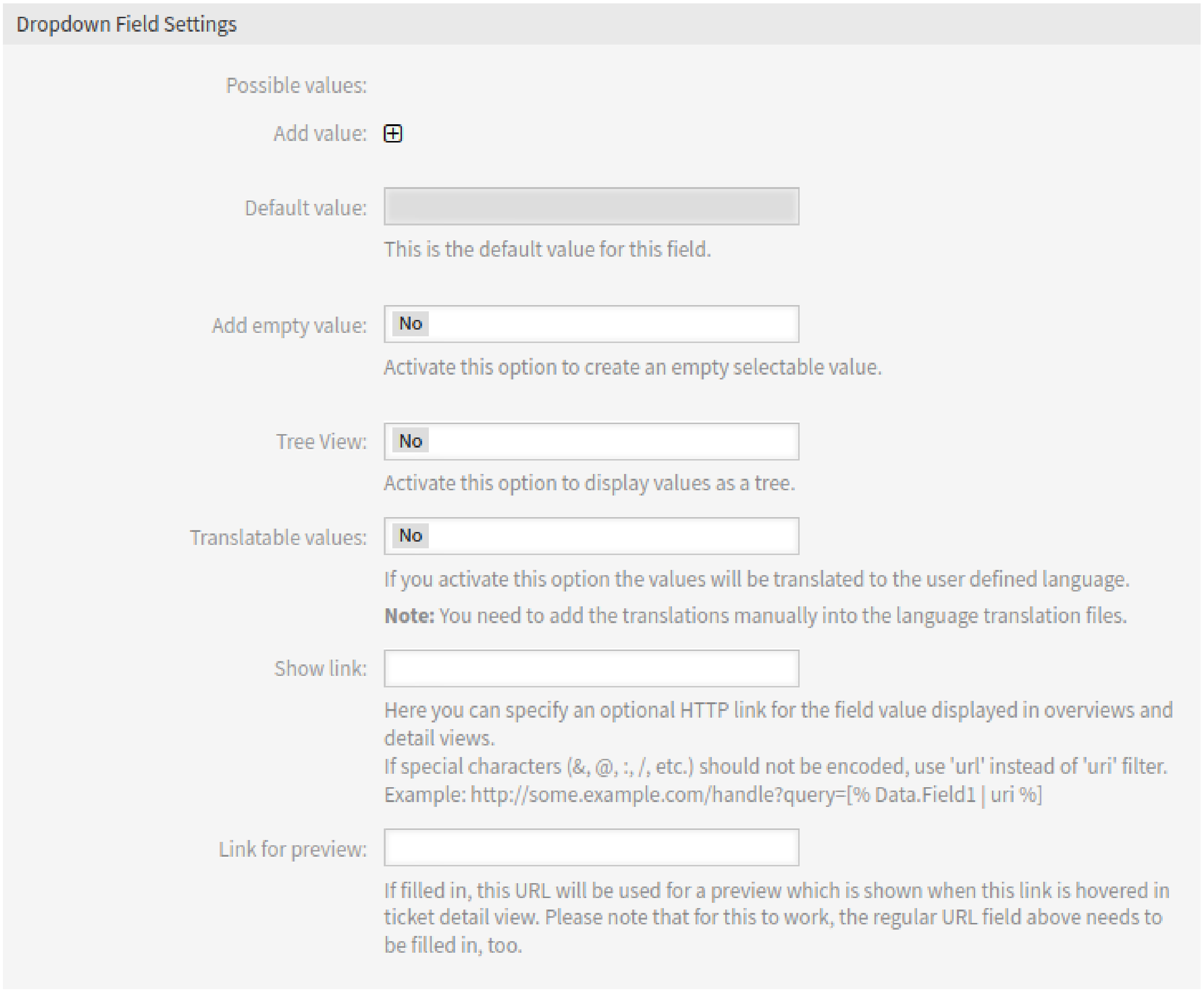
Default date difference:	0
	The difference from NOW (in seconds) to calculate the field default value (e.g. 3600 or -60).
Define years period:	No
	Activate this feature to define a fixed range of years (in the future and in the past) to be displayed on the year part of the field.
Show link:	
	Here you can specify an optional HTTP link for the field value displayed in overviews and detail views. If special characters (&, @, :, /, etc.) should not be encoded, use 'url' instead of 'uri' filter. Example: http://some.example.com/handle?query=[% Data.Field1 | uri %]
Link for preview:	
	If filled in, this URL will be used for a preview which is shown when this link is hovered in ticket detail view. Please note that for this to work, the regular URL field above needs to be filled in, too.
Restrict entering of dates:	
	Here you can restrict the entering of dates of tickets.

Fig. 34: Date / Time Dynamic Field Settings

The settings for this type of dynamic field is the same as for date dynamic field.

Dropdown Dynamic Field Settings

Drop-down dynamic field is used to store a single value, from a closed list.


Dropdown Field Settings

Possible values:

Add value: ⊞

Default value:

This is the default value for this field.

Add empty value: No

Activate this option to create an empty selectable value.

Tree View: No

Activate this option to display values as a tree.

Translatable values: No

If you activate this option the values will be translated to the user defined language.

Note: You need to add the translations manually into the language translation files.

Show link:

Here you can specify an optional HTTP link for the field value displayed in overviews and detail views.
If special characters (&, @, :, /, etc.) should not be encoded, use 'url' instead of 'uri' filter.
Example: http://some.example.com/handle?query=[% Data.Field1 | uri %]

Link for preview:

If filled in, this URL will be used for a preview which is shown when this link is hovered in ticket detail view. Please note that for this to work, the regular URL field above needs to be filled in, too.

Fig. 35: Dropdown Dynamic Field Settings

Possible values

These are the possible data attributes for contacts. Clicking on the button will add two new fields, where a key (internal value) and a value (displayed value) can be set. With the button you can add multiple key-value pairs.

> **Warning:** The maximum number of 100 values should not be exceeded. Exceeding this limit may affect the system performance.

Default value

This is the default value for this field and this will be shown on the edit screens.

Add empty value

If this option is activated an extra value is defined to show as a - in the list of possible values. This special value is empty internally.

Tree View
Activate this option to display values as a tree. If you use a sub-value, specify it as *Parent::Sub*.

Translatable values
If you activate this option the values will be translated to the user defined language.

Note: You need to add the translations manually into the language translation files.

Show link
Here you can specify an optional HTTP link for the field value displayed in overviews and detail views. Examples:

- The value of dynamic field named `Field1` is part of the link:

```
https://some.example.com/handle?query=[% Data.Field1 | uri %]
```

- The value of dynamic field named `Field1` is the complete link with and without protocol:

```
[% Data.Field1 | uri %]
https://[% Data.Field1 | uri %]
```

- The value of dynamic field named `100Field` needs special handling, because the name of the dynamic field starts with a number:

```
https://some.example.com/handle?query=[% Data.item("100Field") | uri %]
```

- The value of dynamic field named `Field1` contains the link, the value of dynamic field named `Field2` contains an important value for a parameter:

```
https://[% Data.Field1 | uri %]?query=[% Data.DynamicField_Field2 |␣
↪uri %]
```

If special characters (&, @, :, /, etc.) should not be encoded, use `url` instead of `uri` filter.

Link for preview
If filled in, this URL will be used for a preview which is shown when this link is hovered in ticket detail view. Please note that for this to work, the regular URL field above needs to be filled in, too.

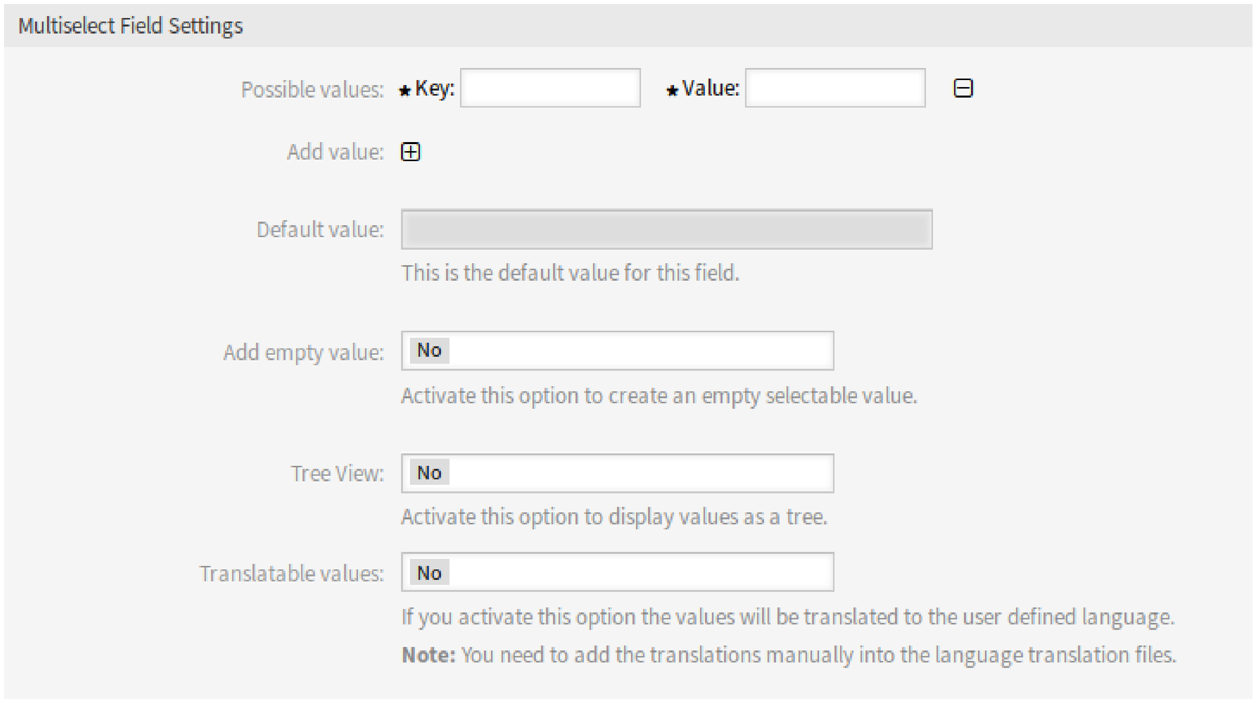
See also:

The URL has to be added to the allowed origins. Read the *External Link Previews* chapter for more information.

Multiselect Dynamic Field Settings

Possible values
These are the possible data attributes for contacts. Clicking on the button will add two new fields, where a key (internal value) and a value (displayed value) can be set. With the button you can add multiple key-value pairs.

Warning: The maximum number of 100 values should not be exceeded. Exceeding this limit may affect the system performance.


Multiselect Field Settings

Possible values: * Key: * Value:

Add value:

Default value:

This is the default value for this field.

Add empty value: No

Activate this option to create an empty selectable value.

Tree View: No

Activate this option to display values as a tree.

Translatable values: No

If you activate this option the values will be translated to the user defined language.

Note: You need to add the translations manually into the language translation files.

Fig. 36: Multiselect Dynamic Field Settings

Default value
This is the default value for this field and this will be shown on the edit screens.

Add empty value
If this option is activated an extra value is defined to show as a - in the list of possible values. This special value is empty internally.

Tree View
Activate this option to display values as a tree. If you use a sub-value, specify it as *Parent::Sub*.

Translatable values
If you activate this option the values will be translated to the user defined language.

Note: You need to add the translations manually into the language translation files.

Number Dynamic Field Settings

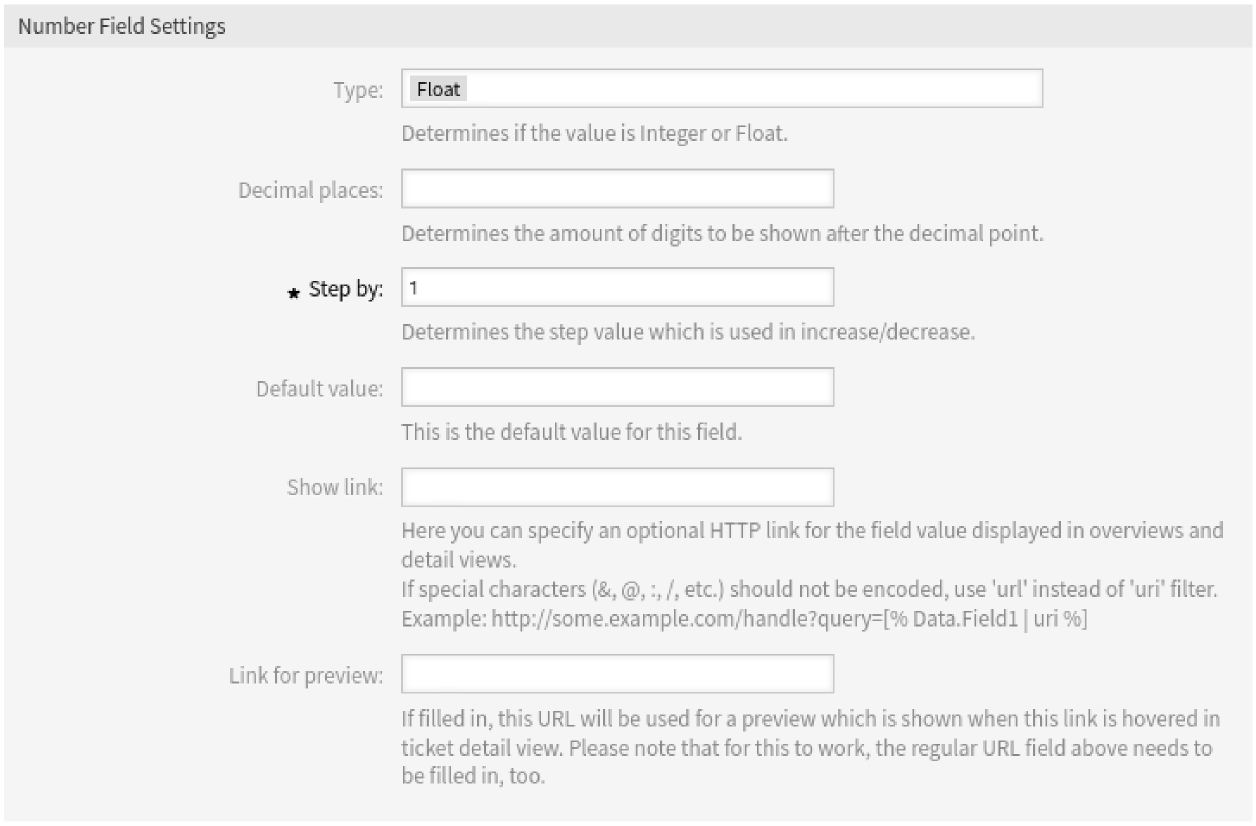
Dynamic fields of type number are used to store integers and floating point numbers (float).

Type
Select the type of the number. Possible values: *Float* or *Integer*.

Decimal places
Determines the amount of digits to be shown after the decimal separator.

This setting is only available if *Float* is selected in the *Type* field.

Step by *
Determines the step value which is used in increase or decrease.



Number Field Settings

Type: Float

Determines if the value is Integer or Float.

Decimal places:

Determines the amount of digits to be shown after the decimal point.

* Step by: 1

Determines the step value which is used in increase/decrease.

Default value:

This is the default value for this field.

Show link:

Here you can specify an optional HTTP link for the field value displayed in overviews and detail views.
If special characters (&, @, :, /, etc.) should not be encoded, use 'url' instead of 'uri' filter.
Example: http://some.example.com/handle?query=[% Data.Field1 | uri %]

Link for preview:

If filled in, this URL will be used for a preview which is shown when this link is hovered in ticket detail view. Please note that for this to work, the regular URL field above needs to be filled in, too.

Fig. 37: Number Dynamic Field Settings

Default value
This is the default value for this field and this will be shown on the edit screens.

Show link
Here you can specify an optional HTTP link for the field value displayed in overviews and detail views. Examples:

- The value of dynamic field named `Field1` is part of the link:

```
https://some.example.com/handle?query=[% Data.Field1 | uri %]
```

- The value of dynamic field named `Field1` is the complete link with and without protocol:

```
[% Data.Field1 | uri %]
https://[% Data.Field1 | uri %]
```

- The value of dynamic field named `100Field` needs special handling, because the name of the dynamic field starts with a number:

```
https://some.example.com/handle?query=[% Data.item("100Field") | uri %]
```

- The value of dynamic field named `Field1` contains the link, the value of dynamic field named `Field2` contains an important value for a parameter:

```
https://[% Data.Field1 | uri %]?query=[% Data.DynamicField_Field2 |␣
↪uri %]
```

If special characters (&, @, :, /, etc.) should not be encoded, use `url` instead of `uri` filter.

Link for preview
If filled in, this URL will be used for a preview which is shown when this link is hovered in ticket detail view. Please note that for this to work, the regular URL field above needs to be filled in, too.

See also:

The URL has to be added to the allowed origins. Read the *External Link Previews* chapter for more information.

Text Dynamic Field Settings

Text dynamic field is used to store a single line string.

Default value
This is the default value for this field and this will be shown on the edit screens.

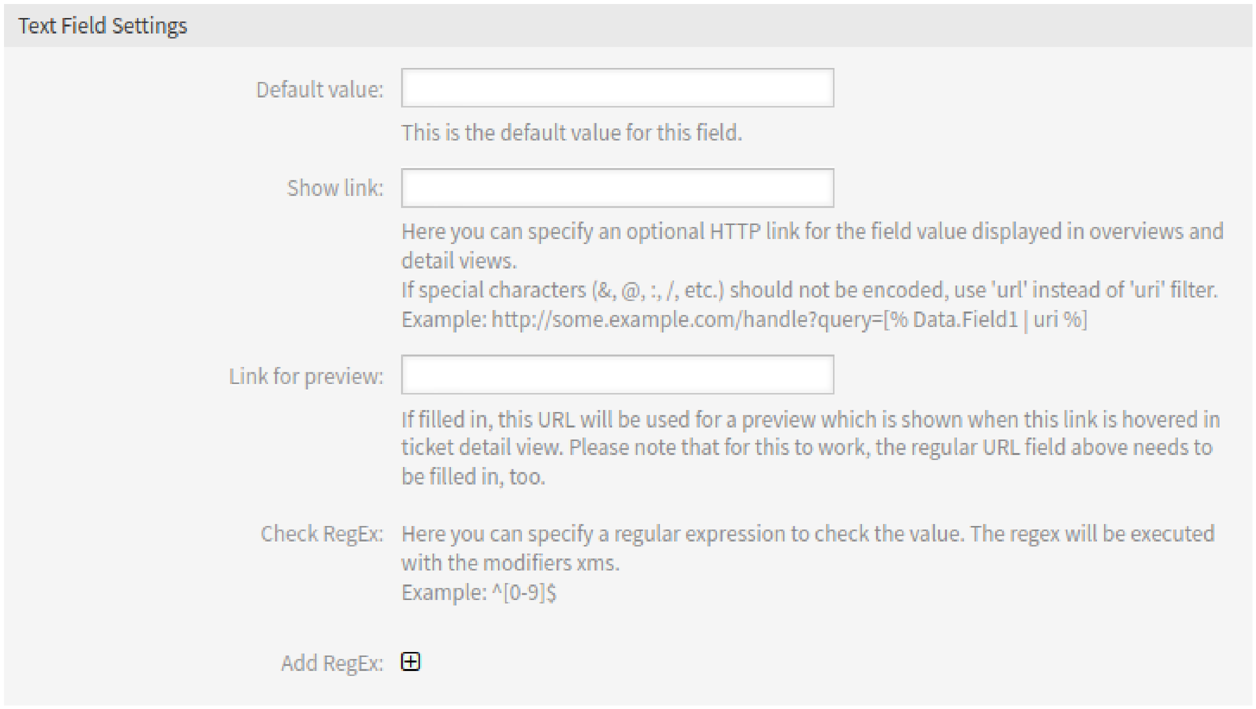
Show link
Here you can specify an optional HTTP link for the field value displayed in overviews and detail views. Examples:

- The value of dynamic field named `Field1` is part of the link:

```
https://some.example.com/handle?query=[% Data.Field1 | uri %]
```

- The value of dynamic field named `Field1` is the complete link with and without protocol:

```
[% Data.Field1 | uri %]
https://[% Data.Field1 | uri %]
```


Text Field Settings

Default value:	
	This is the default value for this field.
Show link:	
	Here you can specify an optional HTTP link for the field value displayed in overviews and detail views. If special characters (&, @, :, /, etc.) should not be encoded, use 'url' instead of 'uri' filter. Example: http://some.example.com/handle?query=[% Data.Field1 | uri %]
Link for preview:	
	If filled in, this URL will be used for a preview which is shown when this link is hovered in ticket detail view. Please note that for this to work, the regular URL field above needs to be filled in, too.
Check RegEx:	Here you can specify a regular expression to check the value. The regex will be executed with the modifiers xms. Example: ^[0-9]$
Add RegEx:	⊞

Fig. 38: Text Dynamic Field Settings

- The value of dynamic field named `100Field` needs special handling, because the name of the dynamic field starts with a number:

  ```
  https://some.example.com/handle?query=[% Data.item("100Field") |␣
  ↪uri %]
  ```

- The value of dynamic field named `Field1` contains the link, the value of dynamic field named `Field2` contains an important value for a parameter:

  ```
  https://[% Data.Field1 | uri %]?query=[% Data.DynamicField_Field2␣
  ↪| uri %]
  ```

If special characters (&, @, :, /, etc.) should not be encoded, use `url` instead of `uri` filter.

Link for preview
If filled in, this URL will be used for a preview which is shown when this link is hovered in ticket detail view. Please note that for this to work, the regular URL field above needs to be filled in, too.

See also:

The URL has to be added to the allowed origins. Read the *External Link Previews* chapter for more information.

Check RegEx
Here you can specify a regular expression to check the value. The regex will be executed with the modifiers `xms`. Example:

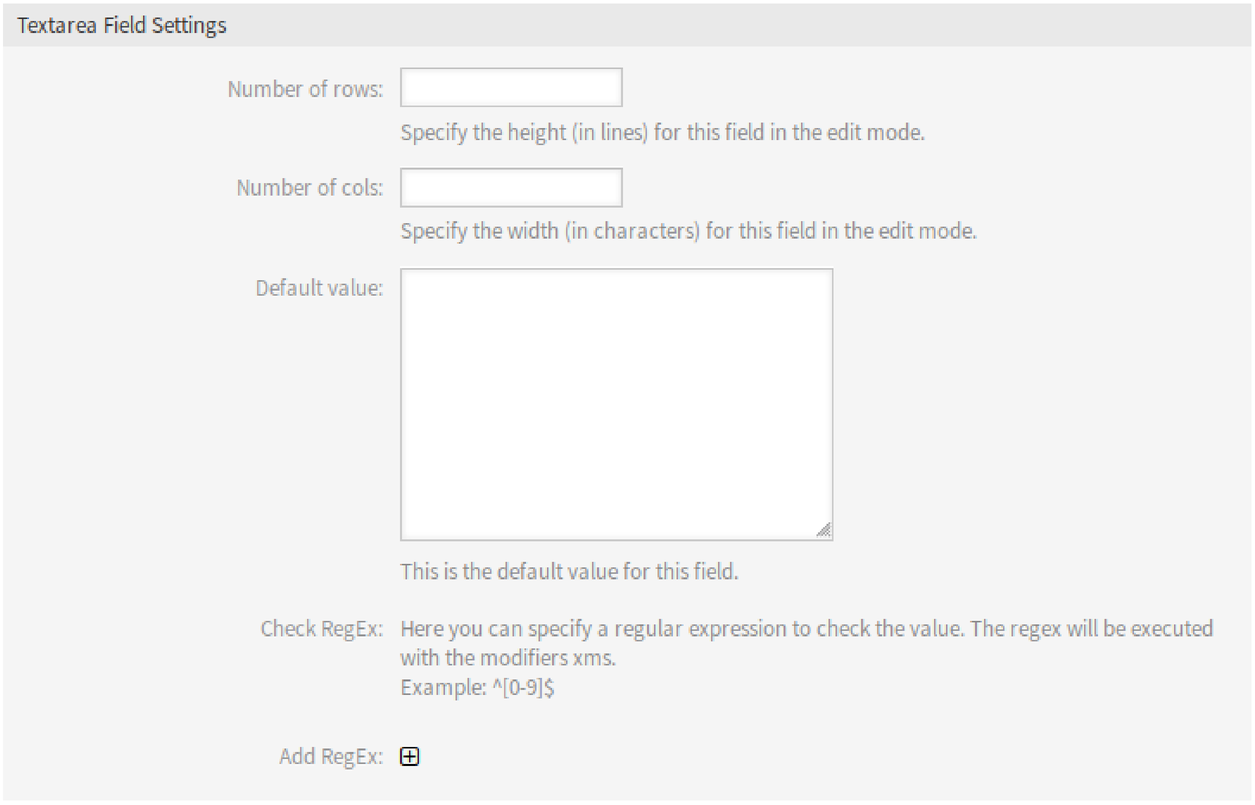
```
^[0-9]$
```

Add RegEx
Clicking on the button will add two new fields, where a regular expression and an error message can

be added.

Textarea Dynamic Field Settings

Textarea dynamic field is used to store a multiple line string.


Textarea Field Settings

Number of rows:

Specify the height (in lines) for this field in the edit mode.

Number of cols:

Specify the width (in characters) for this field in the edit mode.

Default value:

This is the default value for this field.

Check RegEx: Here you can specify a regular expression to check the value. The regex will be executed with the modifiers xms.
Example: ^[0-9]$

Add RegEx:

Fig. 39: Textarea Dynamic Field Settings

Number of rows
The height (in lines) for this field in the edit mode.

Number of cols
The width (in characters) for this field in the edit mode.

Default value
This is the default value for this field and this will be shown on the edit screens.

Check RegEx
Here you can specify a regular expression to check the value. The regex will be executed with the modifiers `xms`. Example:

```
^[0-9]$
```

Add RegEx
Clicking on the button will add two new fields, where a regular expression and an error message can be added.

Web Service Dynamic Field Settings

Dynamic field of type web service is used to store data from external systems for tickets.

It is necessary to have an *already working web service* before creating a new dynamic field. The dynamic field gathers its selectable options from an external system using a web service.

The response from the external system defines the possible options to be displayed, and they could vary depending on the data that is sent in the request. Normally when a field is changed in a screen (e. g. the ticket priority in the *New Phone Ticket* screen) the values of other fields could be updated. That is the case with this type of dynamic fields, as they could also include all screen field values in the request and the remote server could potentially return completely different values depending on input.

Additionally if the dynamic field source object already exists (e. g. a ticket, and the field is set in the *Change Free Fields* action), the details of the already created ticket are also included in the request.

Web service *
The configured web service whose invokers will be to triggered when a dynamic field is displayed.

Invoker *
The invoker that is used to send requests to external systems. Within this field, just invokers of type `Generic::PassThrough` will be displayed.

Multiselect
A drop-down menu to determine if the displayed dynamic field should act as a multiselect field instead of a drop-down field.

Cache TTL
A cache time to live value, that contains a value (in minutes). If the value is 0 or empty, no caching will be active. This cache is to prevent unnecessary requests to the remote server using the same values.

Add empty value
Defines if it is possible to save an empty value in the field.

Tree View
This option activates the tree view of possible values, if they are supplied in the correct format.

Translatable values
If you activate this option the values will be translated to the user defined language.

Note: You need to add the translations manually into the language translation files.

Show link
Here you can specify an optional HTTP link for the field value displayed in overviews and detail views. Optional HTTP link works only for single-select fields. Examples:

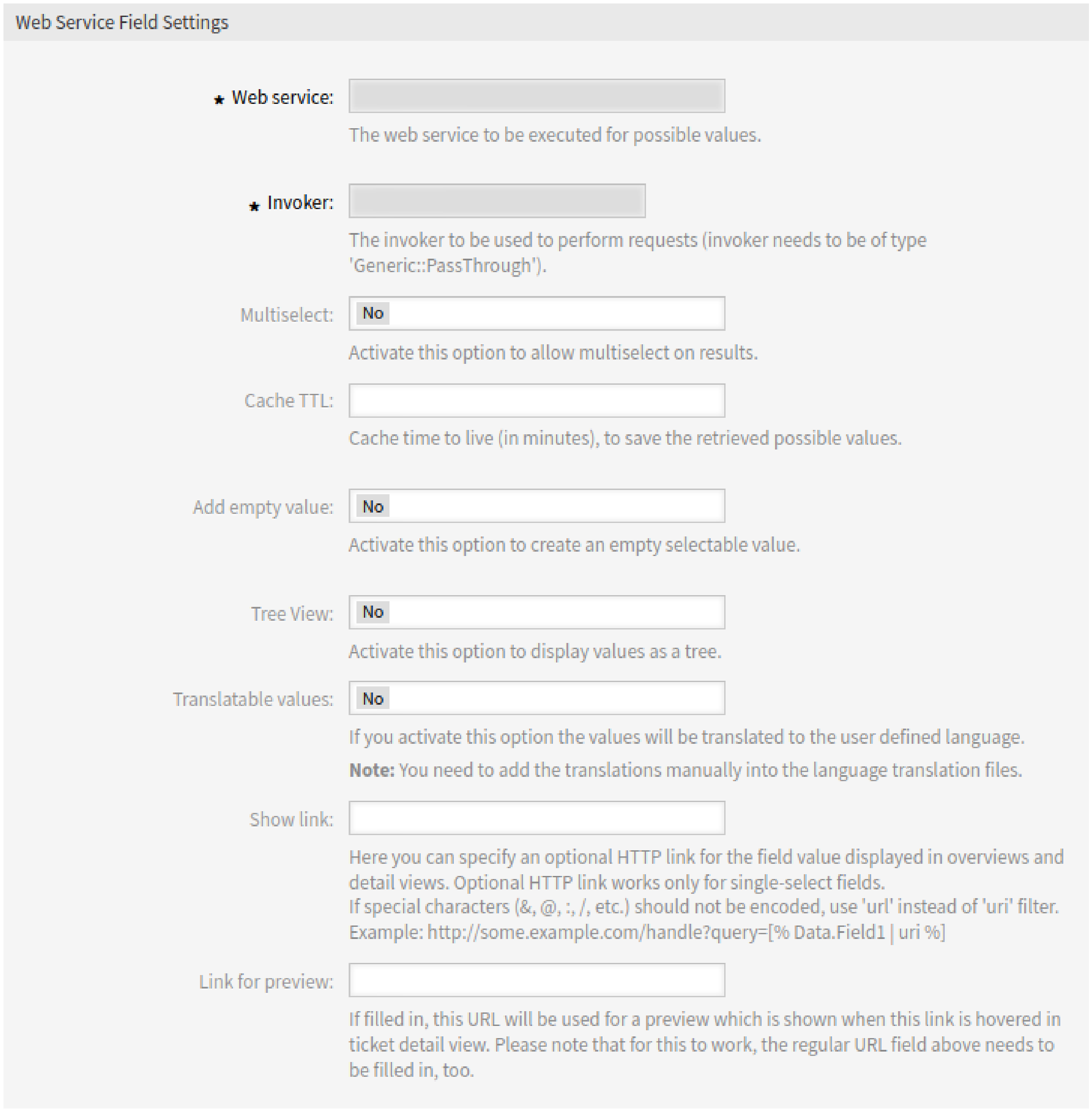
- The value of dynamic field named `Field1` is part of the link:

```
https://some.example.com/handle?query=[% Data.Field1 | uri %]
```

- The value of dynamic field named `Field1` is the complete link with and without protocol:

```
[% Data.Field1 %]
https://[% Data.Field1 %]
```

- The value of dynamic field named `100Field` needs special handling, because the name of the dynamic field starts with a number:



Web Service Field Settings

★ Web service:	The web service to be executed for possible values.
★ Invoker:	The invoker to be used to perform requests (invoker needs to be of type 'Generic::PassThrough').
Multiselect: No	Activate this option to allow multiselect on results.
Cache TTL:	Cache time to live (in minutes), to save the retrieved possible values.
Add empty value: No	Activate this option to create an empty selectable value.
Tree View: No	Activate this option to display values as a tree.
Translatable values: No	If you activate this option the values will be translated to the user defined language. **Note:** You need to add the translations manually into the language translation files.
Show link:	Here you can specify an optional HTTP link for the field value displayed in overviews and detail views. Optional HTTP link works only for single-select fields. If special characters (&, @, :, /, etc.) should not be encoded, use 'url' instead of 'uri' filter. Example: http://some.example.com/handle?query=[% Data.Field1 | uri %]
Link for preview:	If filled in, this URL will be used for a preview which is shown when this link is hovered in ticket detail view. Please note that for this to work, the regular URL field above needs to be filled in, too.

Fig. 40: Web Service Dynamic Field Settings

```
https://some.example.com/handle?query=[% Data.item("100Field") | uri %]
```

If special characters (&, @, :, /, etc.) should not be encoded, use `url` instead of `uri` filter.

Link for preview
If filled in, this URL will be used for a preview which is shown when this link is hovered in ticket detail view. Please note that for this to work, the regular URL field above needs to be filled in, too.

5.2.4 Dynamic Field Search Booster

When using the ticket search in agent or external interface, historical values will be gathered from the database for select-type dynamic fields (e.g. drop-down and multiselect). On systems with a large number of tickets and/or dynamic fields, looking through potentially millions of entries can take more time than desirable which leads to a noticeable delay until the form has opened.

The result is cached but as dynamic fields change constantly on heavily used systems, caches are outdated quite frequently and the database has to be queried again. This package keeps track of used values and their number of occurrence for all relevant dynamic fields and uses this data to present a quick response when historical values are requested.

The current state for all dynamic fields will be retrieved and afterwards it is automatically being updated. However, synchronization can be force triggered by the console command `Maint::Ticket::DynamicFieldSearchBoosterSync`.

Note: This feature is only available to *On-Premise* customers. If you are a *Managed* customer, this feature is taken care of by the *Customer Solutions Team* in **OTRS**. Please contact us via support@otrs.com or in the OTRS Portal.

The search booster is initialized via a daemon task. It might take a while for this task to complete. Until that time historical values retrieved in the search form may be incorrect.

The search booster is in effect automatically when the ticket search form is opened and should deliver the same results as before.

5.2.5 Display Dynamic Fields on Screens

The following sections describe how to add a dynamic field to certain screens. The following examples use a dynamic field named `Test1`. Please make sure that you replace it with the actual name of your dynamic field.

Note: Make sure that the *Validity* of the dynamic field is set to *valid*.

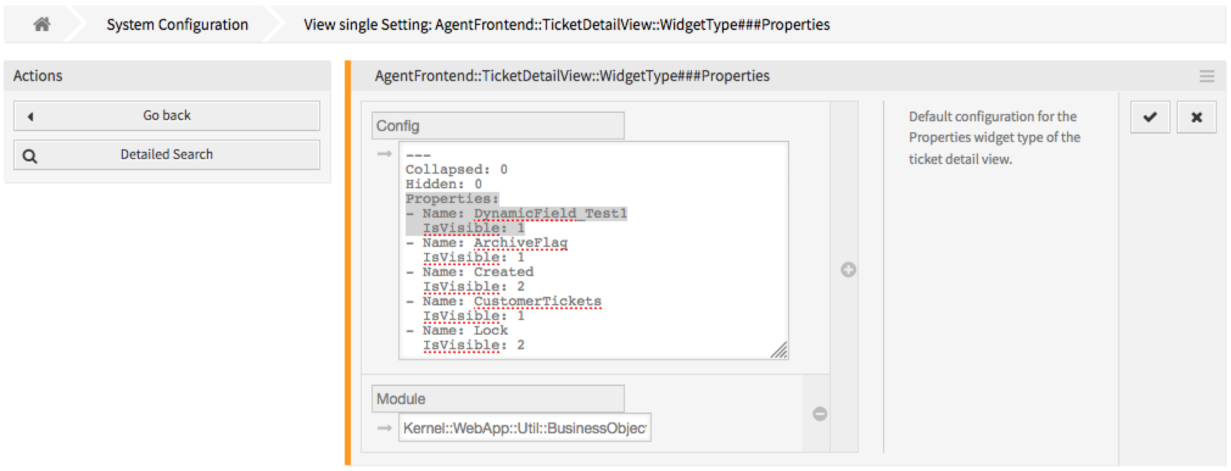
Dynamic Field in Business Object Detail View

Dynamic fields can be added to any widget of the business object detail view. The following examples show the possibilities.

To add a **ticket** dynamic field to the *Properties* widget:

1. Go to the *System Configuration* screen.
2. Search for the setting `AgentFrontend::TicketDetailView::WidgetType###Properties`.
3. Add the ticket dynamic field to the `Properties` section of the YAML configuration.

```
Properties:
- Name: DynamicField_Test1
  IsVisible: 1
```


System Configuration
View single Setting: AgentFrontend::TicketDetailView::WidgetType###Properties
Actions
Go back
Detailed Search
AgentFrontend::TicketDetailView::WidgetType###Properties
Config
Module

```
---
Collapsed: 0
Hidden: 0
Properties:
- Name: DynamicField_Test1
  IsVisible: 1
- Name: ArchiveFlag
  IsVisible: 1
- Name: Created
  IsVisible: 2
- Name: CustomerTickets
  IsVisible: 1
- Name: Lock
  IsVisible: 2
```

Module: Kernel::WebApp::Util::BusinessObjec

Default configuration for the Properties widget type of the ticket detail view.

Fig. 41: Add Dynamic Field to Properties Widget Configuration

See also:

Detailed information about the configuration of the property card and possible keys can be found under *the reference*.

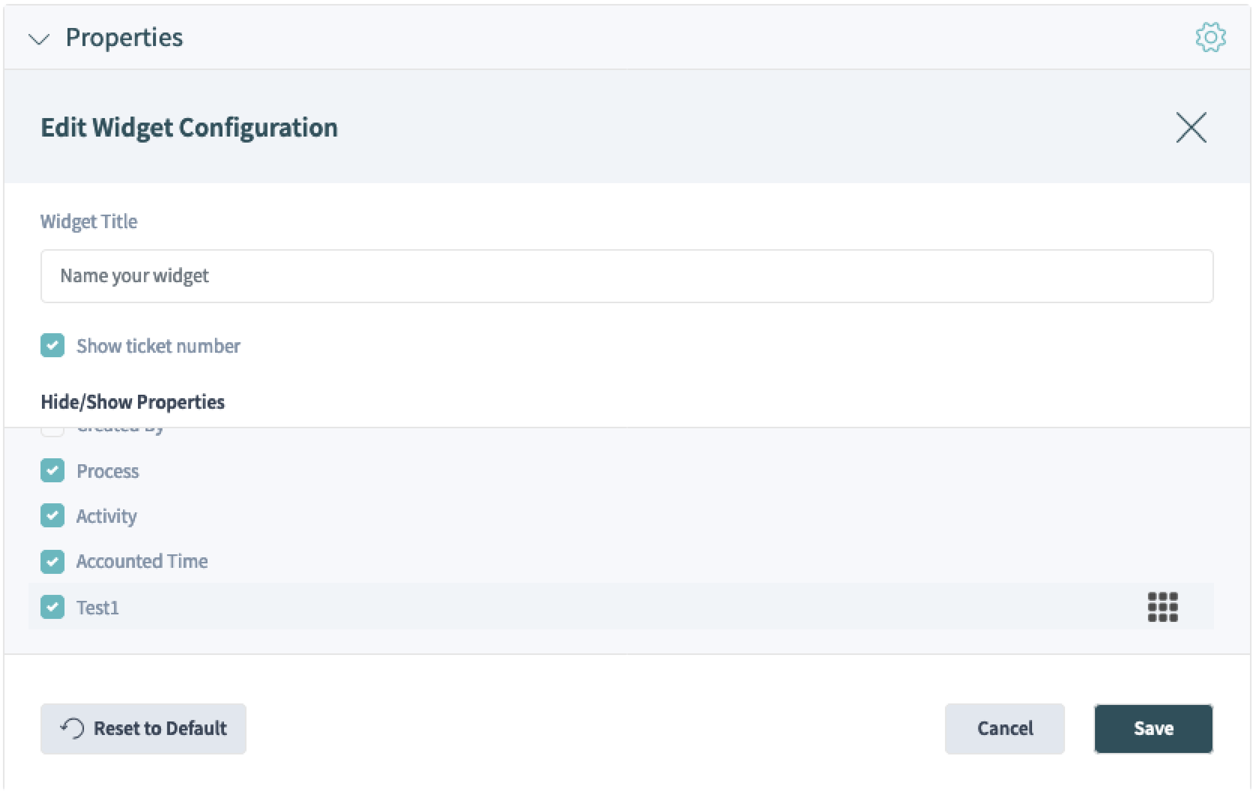
4. Deploy the modified system configuration.
5. Go to the ticket detail view screen in the agent interface.
6. Find the *Properties* widget and open its configuration via the gear icon. Identify the `Test1` dynamic field property in the *Hide/Show Properties* list. Activate it by clicking on the checkbox. Click on the *Save* button to apply the change and close the widget configuration.

If defined, the ticket dynamic field value will then be displayed as a separate property card.

To add an **article** dynamic field to the *Communication Stream* widget:

1. Go to the *System Configuration* screen.
2. Search for the setting `AgentFrontend::TicketDetailView::WidgetType###CommunicationStream`.
3. Add the article dynamic field name to the `ArticleDynamicFields` array of the YAML configuration.

```
ArticleDynamicFields:
- Test1
```



Properties

Edit Widget Configuration

Widget Title

Name your widget

✓ Show ticket number

Hide/Show Properties

Created By

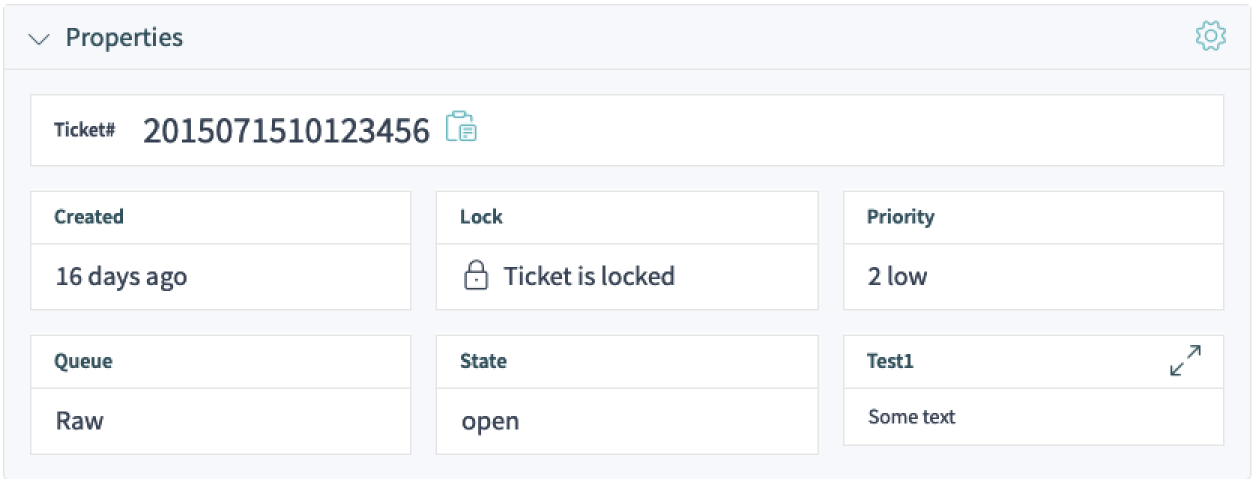
✓ Process

✓ Activity

✓ Accounted Time

✓ Test1

Fig. 42: Enable Dynamic Field Property Card in Properties Widget



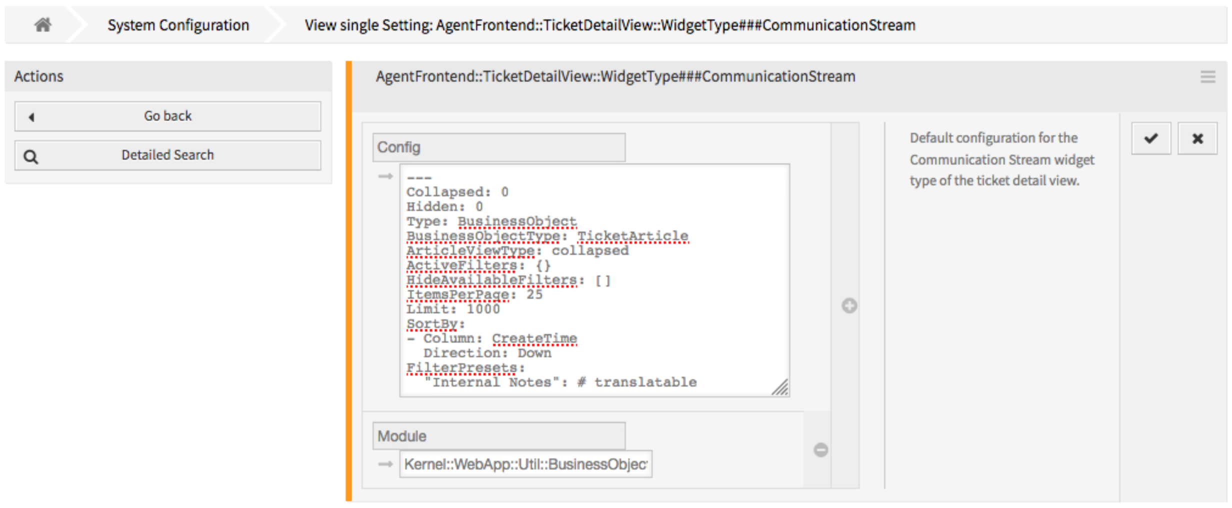
Properties

Ticket# 2015071510123456

Created	Lock	Priority
16 days ago	Ticket is locked	2 low

Queue	State	Test1
Raw	open	Some text

Fig. 43: Display Dynamic Field Property Card in Properties Widget


System Configuration
View single Setting: AgentFrontend::TicketDetailView::WidgetType###CommunicationStream
Actions
Go back
Detailed Search
AgentFrontend::TicketDetailView::WidgetType###CommunicationStream
Config
Module

```
---
Collapsed: 0
Hidden: 0
Type: BusinessObject
BusinessObjectType: TicketArticle
ArticleViewType: collapsed
ActiveFilters: {}
HideAvailableFilters: []
ItemsPerPage: 25
Limit: 1000
SortBy:
- Column: CreateTime
  Direction: Down
FilterPresets:
  "Internal Notes": # translatable
```

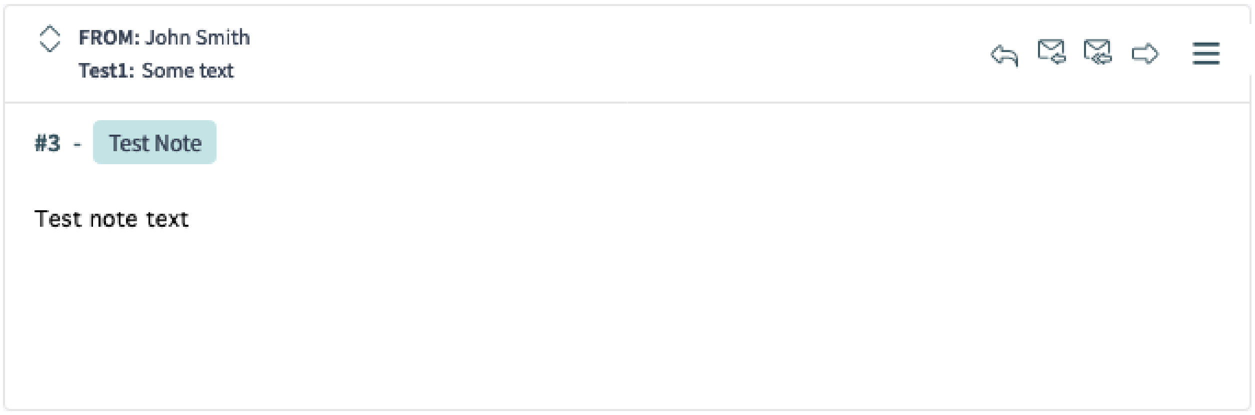
Module: Kernel::WebApp::Util::BusinessObjec

Default configuration for the Communication Stream widget type of the ticket detail view.

Fig. 44: Add Dynamic Field to Communication Stream Widget Configuration

4. Deploy the modified system configuration.

If defined, the article dynamic field value will then be displayed as a separate line in the article header in the *Communication Stream* widget. If not visible, make sure the header is expanded by clicking on it.


FROM: John Smith
Test1: Some text

#3 - Test Note

Test note text

Fig. 45: Display Dynamic Field Value in Communication Stream Widget

To add a **ticket** dynamic field to the *Business Process Information* widget:

1. Go to the *System Configuration* screen.
2. Search for the setting `AgentFrontend::TicketDetailView::Widget::BusinessProcessInformation###D`
3. Add the dynamic field to the list and enabled it.
4. Deploy the modified system configuration.

If defined, the ticket dynamic field value will then be displayed as a separate property card in the *Business Process Information* widget of a process ticket.

See also:

If you want to group the dynamic fields into groups, please check the setting AgentFrontend::TicketDetailView::Widget::BusinessProcessInformation###DynamicFieldGroups.

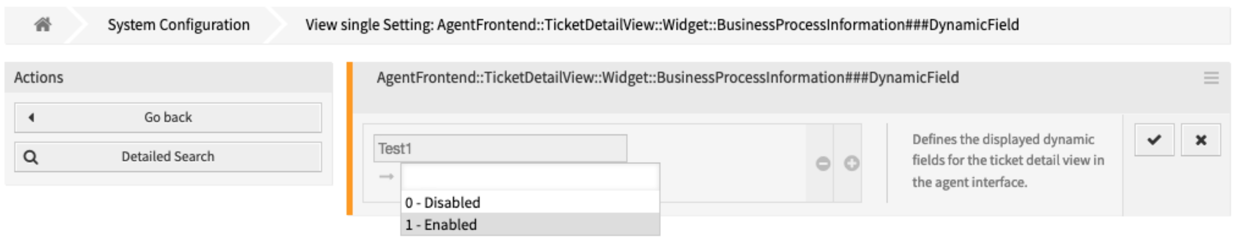

System Configuration
View single Setting: AgentFrontend::TicketDetailView::Widget::BusinessProcessInformation###DynamicField
Actions
Go back
Detailed Search
AgentFrontend::TicketDetailView::Widget::BusinessProcessInformation###DynamicField
Test1
0 - Disabled
1 - Enabled
Defines the displayed dynamic fields for the ticket detail view in the agent interface.

Fig. 46: Add Dynamic Field to Business Process Information Widget Configuration

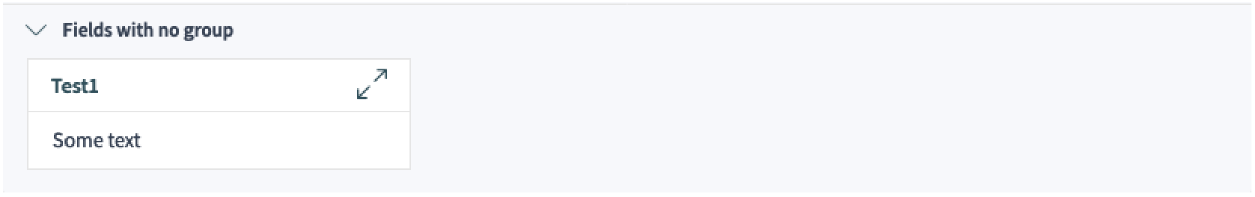


Fig. 47: Display Dynamic Field Property Card in Business Process Information Widget

Dynamic Field in Ticket and Article Action

The following example shows how to add a dynamic field to the *Close Ticket* action. The steps are identical for other actions, only the system configuration keys are different for each form.

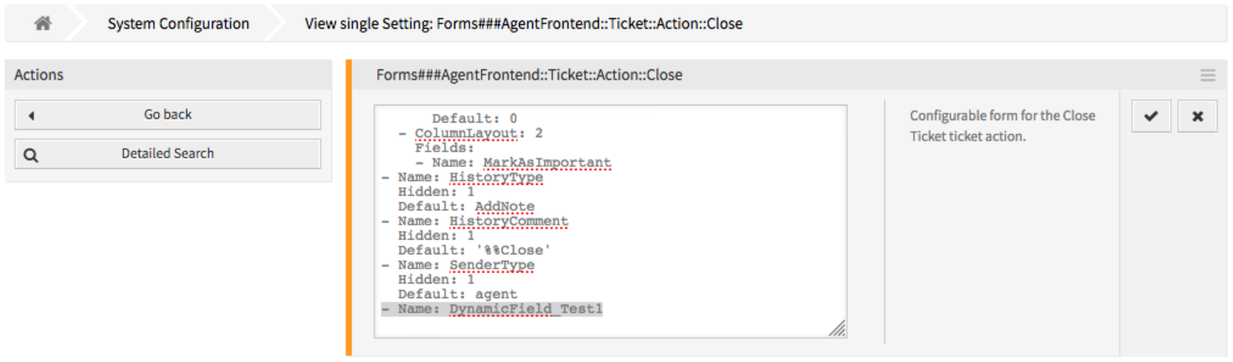
See also:

For the names and explanations of other forms, please consult the *Form Fields* chapter.

To add a **ticket** or an **article** dynamic field to the *Close Ticket* form:

1. Go to the *System Configuration* screen.
2. Search for the setting `Forms###AgentFrontend::Ticket::Action::Close`.
3. Add the dynamic field to the `Fields` section of the YAML configuration.

```
Fields:
- Name: DynamicField_Test1
```

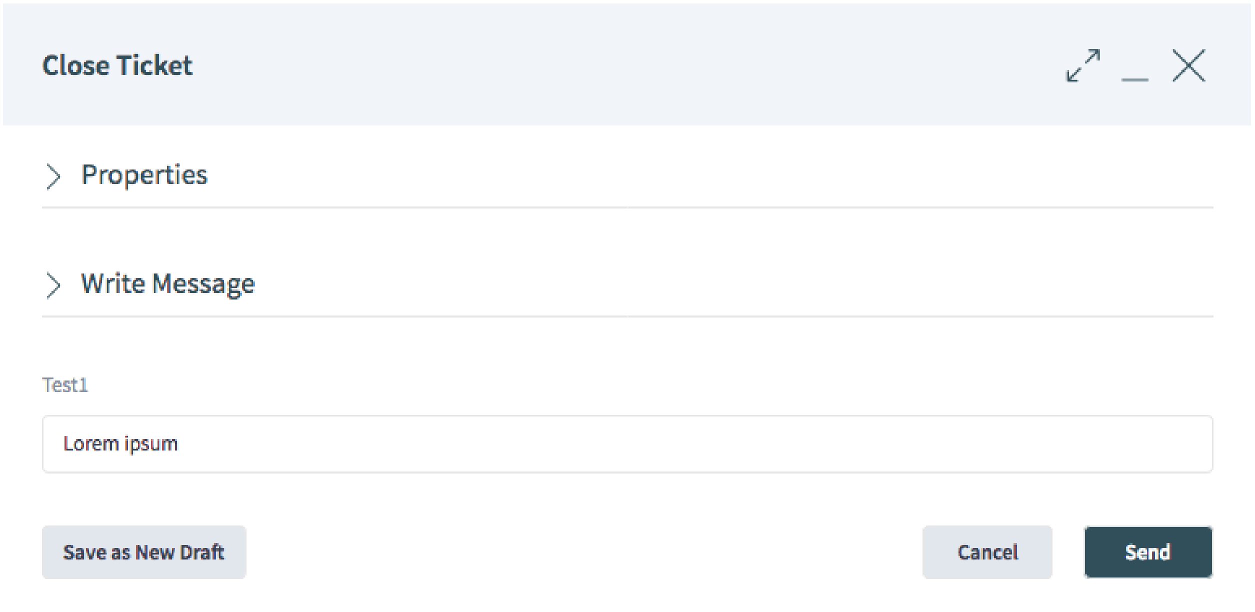

System Configuration
View single Setting: Forms###AgentFrontend::Ticket::Action::Close
Actions
Go back
Detailed Search
Forms###AgentFrontend::Ticket::Action::Close
Configurable form for the Close Ticket ticket action.

```
      Default: 0
  - ColumnLayout: 2
    Fields:
    - Name: MarkAsImportant
- Name: HistoryType
  Hidden: 1
  Default: AddNote
- Name: HistoryComment
  Hidden: 1
  Default: '%%Close'
- Name: SenderType
  Hidden: 1
  Default: agent
- Name: DynamicField_Test1
```

Fig. 48: Add Dynamic Field to Close Ticket Form Configuration

4. Deploy the modified system configuration.

The ticket dynamic field will be displayed in the relevant ticket or article action as a form field.



Test1

Lorem ipsum

Fig. 49: Display Dynamic Field in Close Ticket Action

Note: Article dynamic fields are only displayed if article related fields such as `Subject` and `Body` are present. If these fields do not exist, no article is created and no value is set for the dynamic field when the action is executed.

Dynamic Field in Business Object List

The following examples show how to add a dynamic field as a column or as a filter to the organizer item type *Ticket List*. This affects all organizer items of the type *Ticket List*. If only a specific organizer item needs to be affected, then please modify the system configuration key of the relevant organizer item.

See also:

For the names and explanations for other organizer items please read the chapter *Business Object Lists*.

To add a **ticket** dynamic field as a list column:

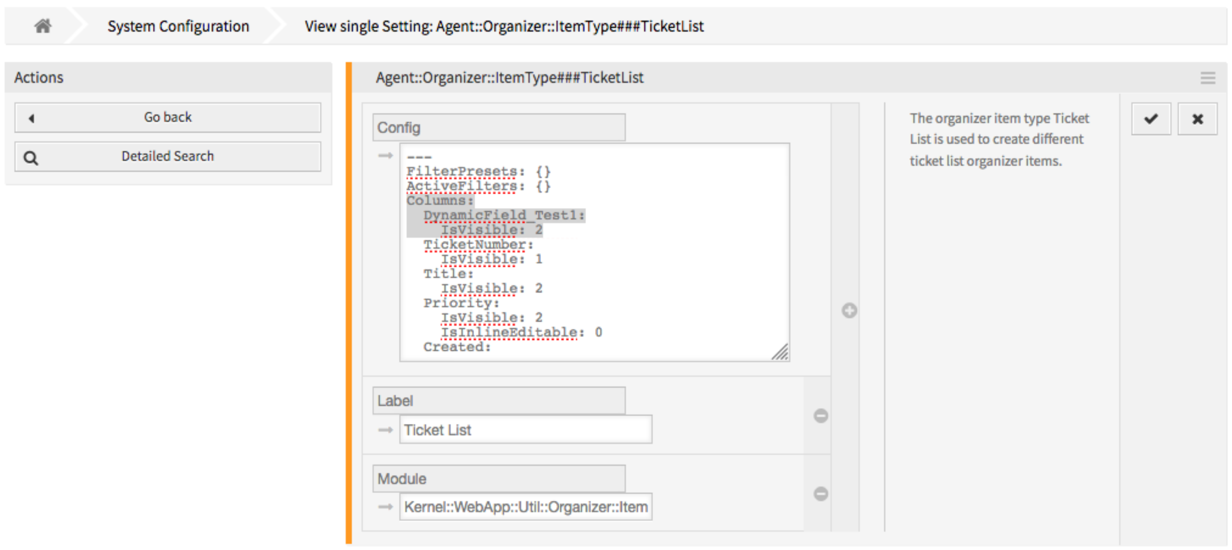
1. Go to the *System Configuration* screen.
2. Search for the setting `Agent::Organizer::ItemType###TicketList`.
3. Add the ticket dynamic field to the `Columns` section of the YAML configuration.

```
Columns:
  DynamicField_Test1:
    IsVisible: 2
```

   See also:

   For detailed information about the column configuration and possible keys, please check *the reference*.

4. Deploy the modified system configuration.


System Configuration
View single Setting: Agent::Organizer::ItemType###TicketList
Actions
Go back
Detailed Search
Agent::Organizer::ItemType###TicketList
Config
Label
Module

```
---
FilterPresets: {}
ActiveFilters: {}
Columns:
  DynamicField_Test1:
    IsVisible: 2
  TicketNumber:
    IsVisible: 1
  Title:
    IsVisible: 2
  Priority:
    IsVisible: 2
    IsInlineEditable: 0
  Created:
```

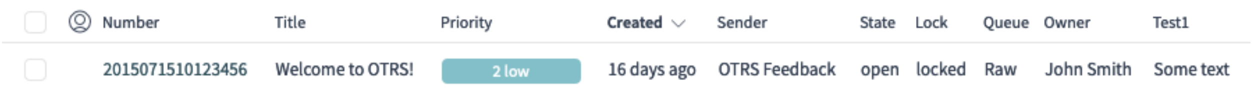
Label: Ticket List

Module: Kernel::WebApp::Util::Organizer::Item

The organizer item type Ticket List is used to create different ticket list organizer items.

Fig. 50: Add Dynamic Field to Ticket List Configuration

If defined, the ticket dynamic field value will then be displayed in a column cell, for any of the organizer items of the ticket list type.



Number	Title	Priority	Created	Sender	State	Lock	Queue	Owner	Test1
2015071510123456	Welcome to OTRS!	2 low	16 days ago	OTRS Feedback	open	locked	Raw	John Smith	Some text

Fig. 51: Display Dynamic Field Value in Ticket List

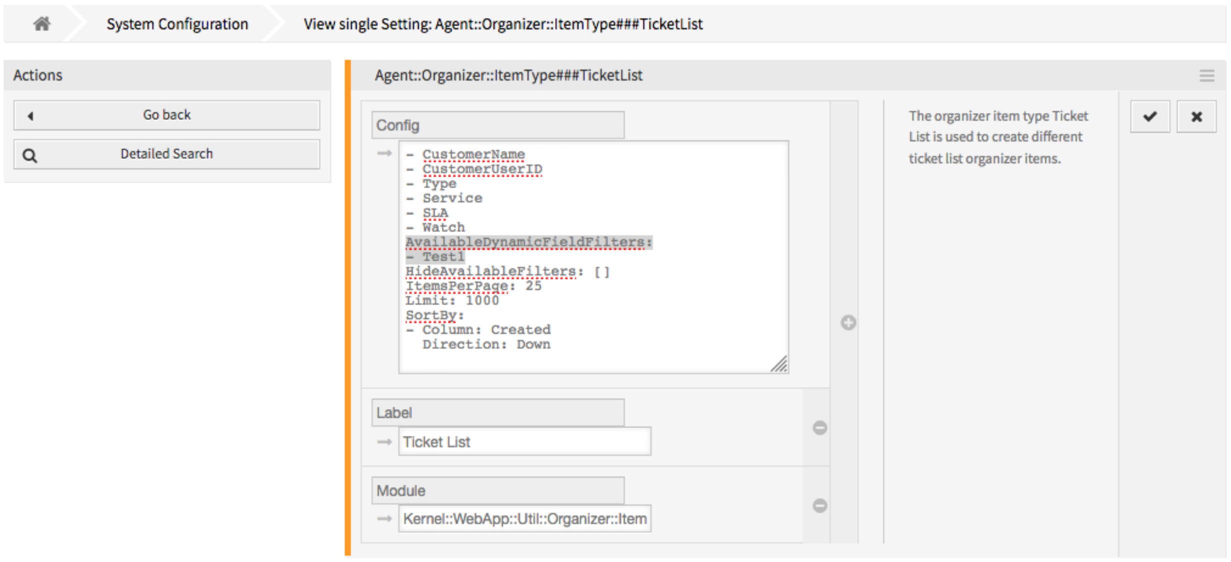
To add a **ticket** dynamic field as a list filter:

1. Go to the *System Configuration* screen.
2. Search for the setting `Agent::Organizer::ItemType###TicketList`.
3. Add the ticket dynamic field name to the `AvailableDynamicFieldFilters` array of the YAML configuration.

```
AvailableDynamicFieldFilters:
- Test1
```

4. Deploy the modified system configuration.

The ticket dynamic field can now be used as a filter in the business object list.


System Configuration
View single Setting: Agent::Organizer::ItemType###TicketList
Actions
Go back
Detailed Search
Agent::Organizer::ItemType###TicketList
Config
Label
Module

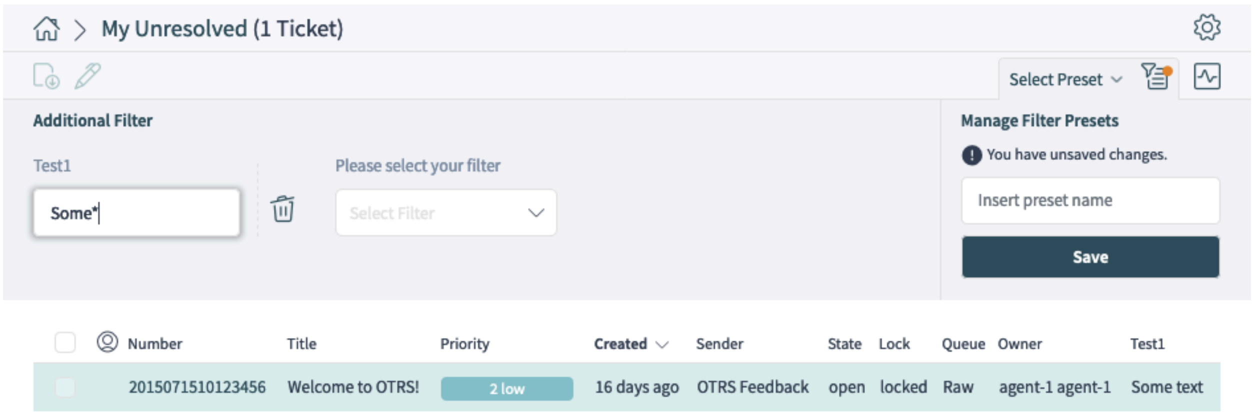
```
- CustomerName
- CustomerUserID
- Type
- Service
- SLA
- Watch
AvailableDynamicFieldFilters:
- Test1
HideAvailableFilters: []
ItemsPerPage: 25
Limit: 1000
SortBy:
- Column: Created
  Direction: Down
```

Label: Ticket List

Module: Kernel::WebApp::Util::Organizer::Item

The organizer item type Ticket List is used to create different ticket list organizer items.

Fig. 52: Add Dynamic Field Filter to Ticket List Configuration



Test1: Some*

You have unsaved changes.

Insert preset name

	Number	Title	Priority	Created	Sender	State	Lock	Queue	Owner	Test1
	2015071510123456	Welcome to OTRS!	2 low	16 days ago	OTRS Feedback	open	locked	Raw	agent-1 agent-1	Some text

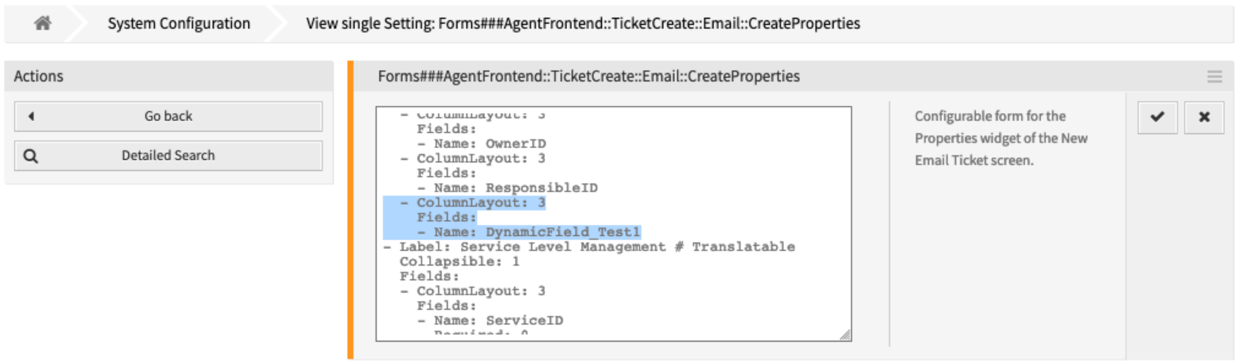
Fig. 53: Use Dynamic Field as Filter in Ticket List

Dynamic Field in Ticket Create Screen

To add a **ticket** or an **article** dynamic field to *New Email Ticket* screen:

1. Go to the *System Configuration* screen.
2. Search for the setting `Forms###AgentFrontend::TicketCreate::Email::CreateProperties`.
3. Add the ticket dynamic field to the `Fields` section of the YAML configuration.

   ```
   Fields:
   - Name: DynamicField_Test1
   ```


System Configuration
View single Setting: Forms###AgentFrontend::TicketCreate::Email::CreateProperties
Actions
Go back
Detailed Search
Forms###AgentFrontend::TicketCreate::Email::CreateProperties

```
    Fields:
    - Name: OwnerID
  - ColumnLayout: 3
    Fields:
    - Name: ResponsibleID
  - ColumnLayout: 3
    Fields:
    - Name: DynamicField_Test1
- Label: Service Level Management # Translatable
  Collapsible: 1
  Fields:
  - ColumnLayout: 3
    Fields:
    - Name: ServiceID
```

Configurable form for the Properties widget of the New Email Ticket screen.

Fig. 54: Add Dynamic Field to New Email Ticket Configuration

   See also:

   For more information about form fields and groups, please consult the *Forms* chapter.

4. Deploy the modified system configuration.

The ticket or article dynamic field can now be used as part of the *New Email Ticket* screen.

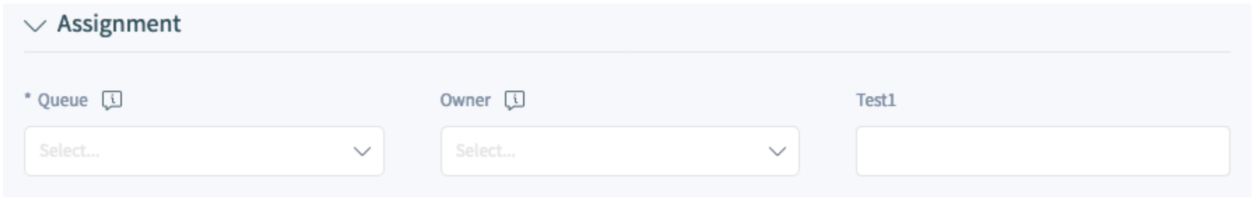


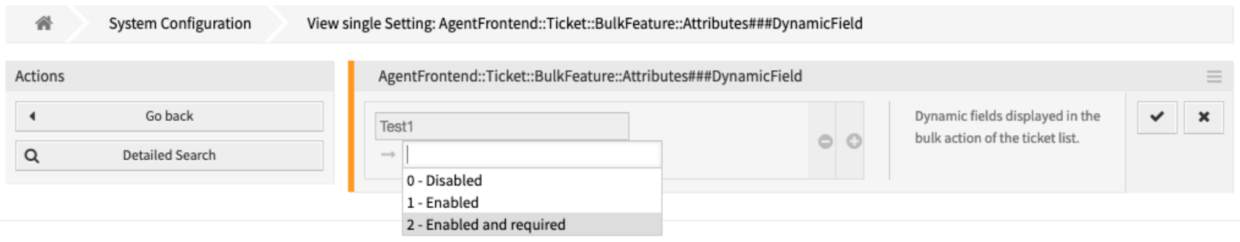
Fig. 55: Display Dynamic Field in New Email Ticket Screen

Dynamic Field in Ticket Bulk Action

To add a **ticket** or an **article** dynamic field to a ticket bulk action:

1. Go to the *System Configuration* screen.
2. Search for the setting `AgentFrontend::Ticket::BulkFeature::Attributes###DynamicField`.
3. Add the dynamic field to the list and enabled it.
4. Deploy the modified system configuration.

The ticket or article dynamic field is now displayed in the *Properties* section in the action form of the bulk action.



System Configuration

View single Setting: AgentFrontend::Ticket::BulkFeature::Attributes###DynamicField

Actions

Go back

Detailed Search

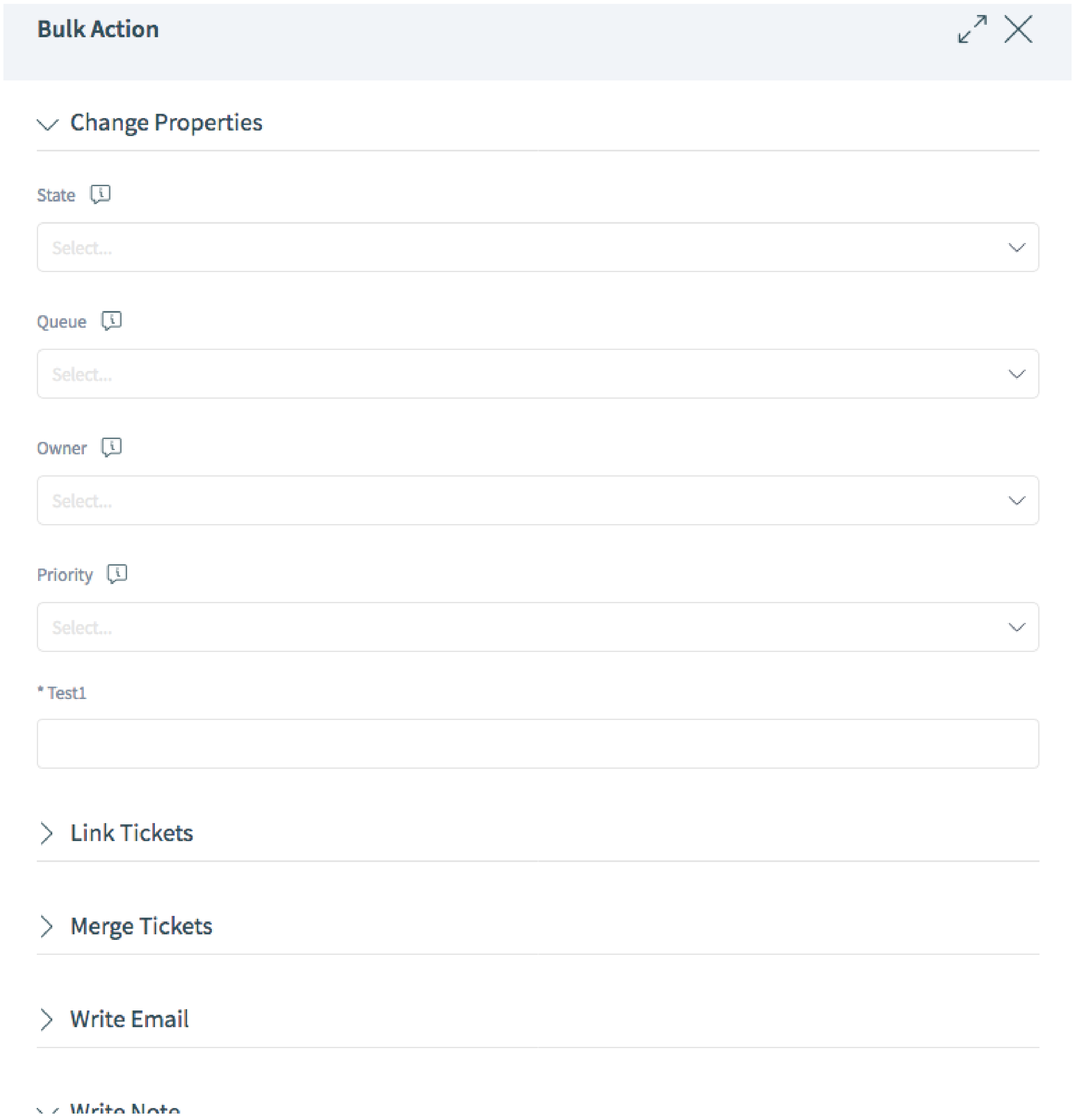
AgentFrontend::Ticket::BulkFeature::Attributes###DynamicField

Test1

0 - Disabled
1 - Enabled
2 - Enabled and required

Dynamic fields displayed in the bulk action of the ticket list.

Fig. 56: Add Dynamic Field to Ticket Bulk Configuration



Bulk Action

Change Properties

State

Select...

Queue

Select...

Owner

Select...

Priority

Select...

* Test1

Link Tickets

Merge Tickets

Write Email

Fig. 57: Display Dynamic Field in Ticket Bulk Action

Dynamic Field in Calendars

Once you have set up the dynamic fields for appointment, you have to activate the dynamic fields for calendars.

To activate dynamic fields for calendars:

1. Go to the *Calendars* screen.
2. Select a calendar or create a new one.
3. Find the *Dynamic Fields* section in the *Manage Settings* widget.
4. Define for each appointment dynamic field to *Hide*, *Show*, *Show as mandatory* or *Use form configuration*.

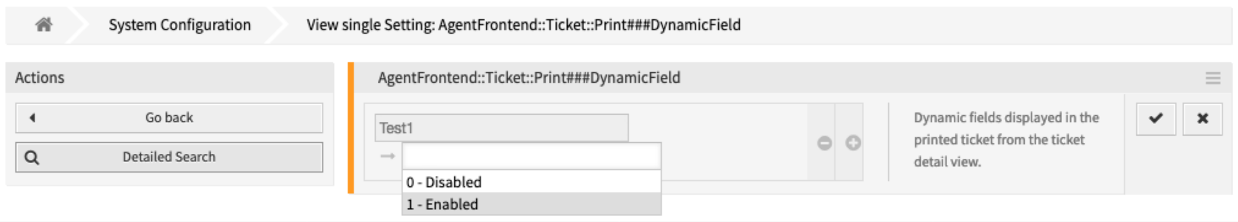
The dynamic fields can be added to the *Linked Appointments* widget of the detail views as well. The following settings have to be extended:

- `AgentFrontend::TicketDetailView::Widget###LinkedObjects::CalendarAppointment`
- `AgentFrontend::KnowledgeBaseArticleDetailView::Widget###KBALinkedObjects::CalendarAppo`

Dynamic Field in the Print Output

To add a **ticket** or an **article** dynamic field to the print output:

1. Go to the *System Configuration* screen.
2. Search for the setting `AgentFrontend::Ticket::Print###DynamicField`.
3. Add the dynamic field to the list and enabled it.



View single Setting: AgentFrontend::Ticket::Print###DynamicField

AgentFrontend::Ticket::Print###DynamicField

Test1

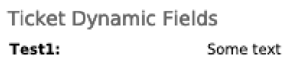
0 - Disabled
1 - Enabled

Dynamic fields displayed in the printed ticket from the ticket detail view.

Fig. 58: Add Dynamic Field to Print Ticket Configuration

4. Deploy the modified system configuration.

The ticket or article dynamic field is now displayed in the printed output of the ticket.



Ticket Dynamic Fields

Test1: Some text

Fig. 59: Display Dynamic Field in Printed Output

Dynamic Field in External Interface

To add a dynamic field to the external interface:

1. Go to the *System Configuration* screen.
2. Navigate to *Frontend* → *External* → *View* in the navigation tree.
3. Select the screen where the dynamic field should be displayed.
4. Search for the dynamic field setting of the screen and click on the *Edit this setting* button.

   The following settings are relevant to adding dynamic fields:

   ```
   - ExternalFrontend::KnowledgeBaseDetailView###DynamicField
   - ExternalFrontend::TicketCreate###DynamicField
   - ExternalFrontend::TicketDetailView###DynamicField
   - ExternalFrontend::TicketDetailView###FollowUpDynamicField
   - ExternalFrontend::TicketOverview###DynamicField
   ```

5. Click on the + button to add the dynamic field. The key is the name of the dynamic field, the value is *1 - Enabled*. This setting is used to display the content of dynamic field in the selected screen.
6. Search for setting `ExternalFrontend::TicketDetailView###FollowUpDynamicField` and click on the *Edit this setting* button.
7. Click on the + button to add the dynamic field. The key is the name of the dynamic field, the value is *1 - Enabled*. This setting is used to configure dynamic fields in the answer part of customers detail view.
8. Repeat the steps for other views, if needed.
9. Deploy the modified settings.

5.2.6 Add Dynamic Fields to Search Engine

By default, the content of the dynamic fields cannot be searched using the document search functionality. Each dynamic field has to be added manually.

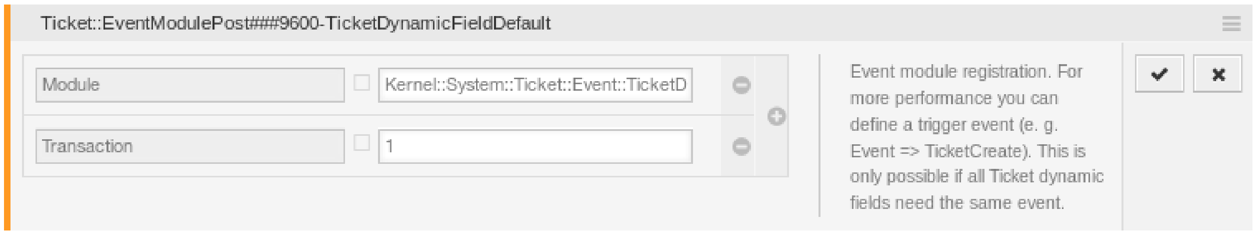
To add a dynamic field to the document search functionality:

1. Go to the *System Configuration* screen.
2. Navigate to *Frontend* → *Agent* → *DocumentSearch* and *Frontend* → *External* → *DocumentSearch* in the navigation tree.
3. Search for the setting `DocumentSearch::Agent::DynamicField` and `DocumentSearch::External::DynamicField` respectively.
4. Click on the *Edit this setting* button.
5. Click on the + button to add the dynamic field.
6. Enter the name of the dynamic field to the text box and click on the tick button.
7. Select *0 - Disabled* or *1 - Enabled*.
8. Click on the tick button on the right to save the setting.
9. Deploy the modified system configuration.

5.2.7 Set Default Value via Ticket Event Module

A ticket event (e.g. `TicketCreate`) can trigger a value set for a certain field, if the field does not have a value yet.

1. Go to the *System Configuration* screen.
2. Navigate to *Core* → *Event* → *Ticket* and search for the setting `Ticket::EventModulePost###9600-TicketDynamicFieldDefault`.
3. Click on the *Edit this setting* button to activate the setting.
4. Click on the tick button on the right to save the setting.
5. Deploy the modified system configuration.


Ticket::EventModulePost###9600-TicketDynamicFieldDefault

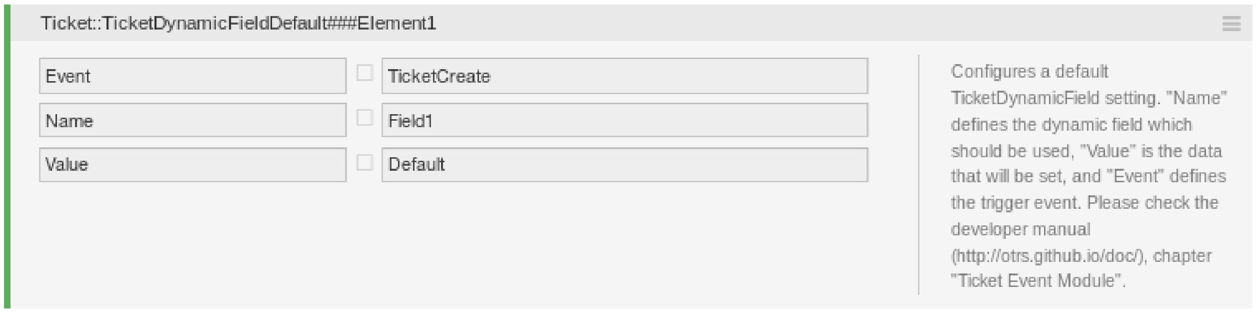
Module	Kernel::System::Ticket::Event::TicketD
Transaction	1

Event module registration. For more performance you can define a trigger event (e. g. Event => TicketCreate). This is only possible if all Ticket dynamic fields need the same event.

Fig. 60: Activate Ticket Event Module

Example: activate *Field1* in `TicketCreate` event:

1. Go to the *System Configuration* screen.
2. Navigate to *Core* → *Ticket* → *DynamicFieldDefault* and search for the setting `Ticket::TicketDynamicFieldDefault###Element1`.
3. Click on the *Edit this setting* button to activate the setting.
4. Click on the tick button on the right to save the setting.
5. Deploy the modified system configuration.


Ticket::TicketDynamicFieldDefault###Element1

Event	TicketCreate
Name	Field1
Value	Default

Configures a default TicketDynamicField setting. "Name" defines the dynamic field which should be used, "Value" is the data that will be set, and "Event" defines the trigger event. Please check the developer manual (http://otrs.github.io/doc/), chapter "Ticket Event Module".

Fig. 61: Activate Dynamic Field in Ticket Create Event

Note: This configuration can be set in any of the 16 `Ticket::TicketDynamicFieldDefault###Element` settings.

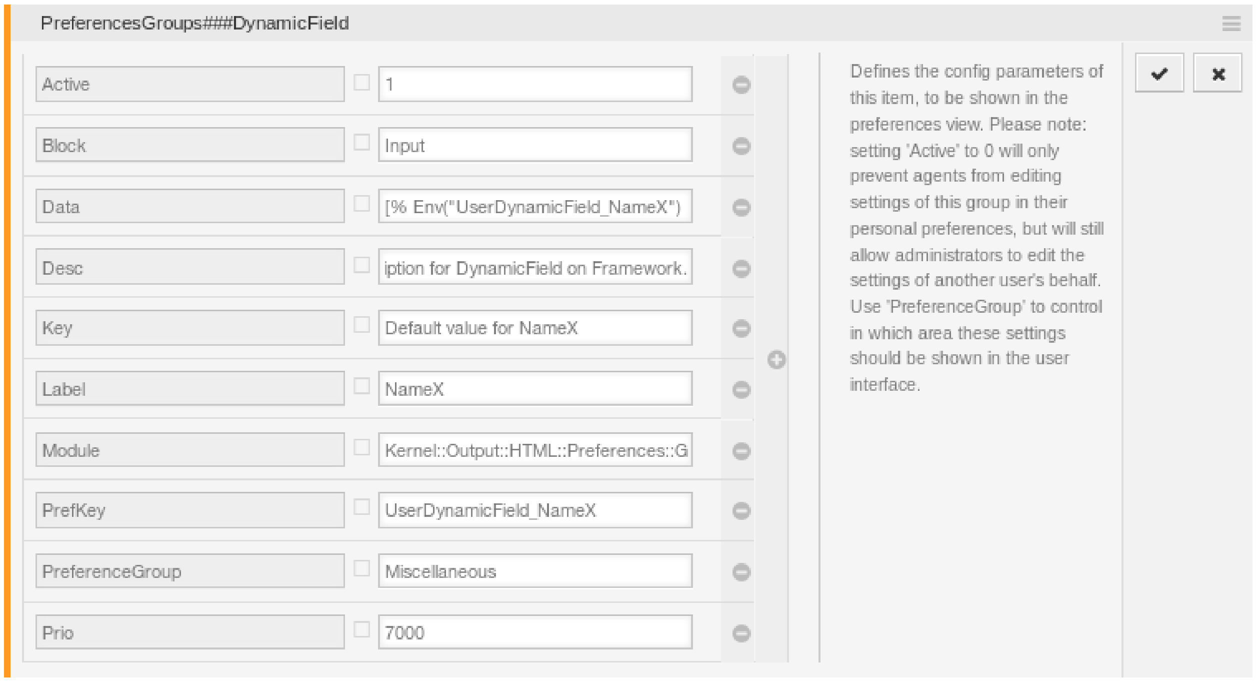
See also:

If more than 16 fields needs to be set up, a custom XML file must be placed in `$OTRS_HOME/Kernel/Config/Files/XML` directory to extend this feature.

5.2.8 Set Default Value via User Preferences

The dynamic field default value can be overwritten with a user defined value stored in the personal preferences.

1. Go to the *System Configuration* screen.
2. Navigate to *Frontend* → *Agent* → *View* → *Preferences* and search for the setting `PreferencesGroups###DynamicField`.
3. Click on the *Edit this setting* button to activate the setting.
4. Click on the tick button on the right to save the setting.
5. Deploy the modified system configuration.



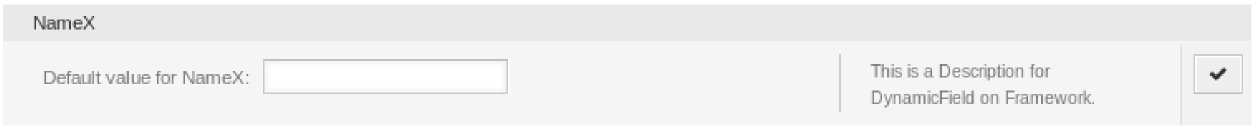
PreferencesGroups###DynamicField

Active	1
Block	Input
Data	[% Env("UserDynamicField_NameX")
Desc	iption for DynamicField on Framework.
Key	Default value for NameX
Label	NameX
Module	Kernel::Output::HTML::Preferences::G
PrefKey	UserDynamicField_NameX
PreferenceGroup	Miscellaneous
Prio	7000

Defines the config parameters of this item, to be shown in the preferences view. Please note: setting 'Active' to 0 will only prevent agents from editing settings of this group in their personal preferences, but will still allow administrators to edit the settings of another user's behalf. Use 'PreferenceGroup' to control in which area these settings should be shown in the user interface.

Fig. 62: Activate Dynamic Field in Personal Preferences

Click on your avatar on the top left corner, and select *Personal Preferences* → *Miscellaneous* to add a default value for the dynamic field.



NameX

Default value for NameX:

This is a Description for DynamicField on Framework.

Fig. 63: Dynamic Field in Personal Preferences

This setting is an example of how to create an entry in the user preferences screen to set an exclusive dynamic field `Name_X` default value for the selected user. The limitation of this setting is that it only permits the use of one dynamic field. If two or more fields will use this feature, it is necessary to create a custom XML configuration file to add more settings similar to this one.

Note: If more settings are added in a new XML each setting name needs to be unique in the system and different than `PreferencesGroups###DynamicField`. For example:

- `PreferencesGroups###101-DynamicField-Field1`
- `PreferencesGroups###102-DynamicField-Field2`
- `PreferencesGroups###My-Field1`
- `PreferencesGroups###My-Field2`

5.3 Generic Agent

Processing tickets require often a workflow. Let's say "if-then" activities.

If specific conditions match like:

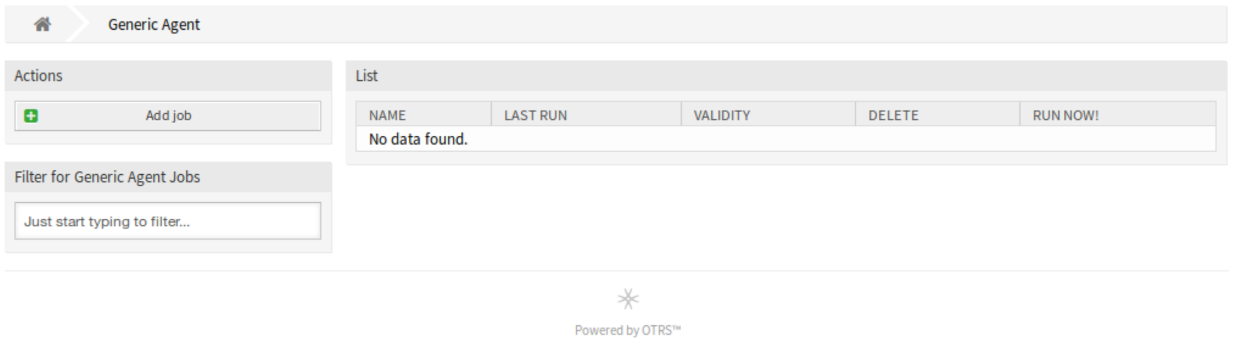
- A ticket is from one particular customer.
- A ticket is assigned to an appropriate queue.
- A ticket has a defined priority.
- A ticket contains defined keywords.

Outlined activities must be performed like changing the ticket priority, moving the ticket to another group, assigning a service to a ticket, and many more.

Also time-based activities can be required like cleaning up the spam-queue once a week.

OTRS supports this with the *Generic Agent*. Here, simple or compound time and event-based tasks are configurable in the OTRS front end without the requirement to learn a scripting language. Depending on search criteria, and time or event criteria, tickets will automatically be acted upon.

Use this screen to manage generic agent jobs in the system. A fresh OTRS installation contains no generic agent jobs by default. The generic agent job management screen is available in the *Generic Agent* module of the *Processes & Automation* group.



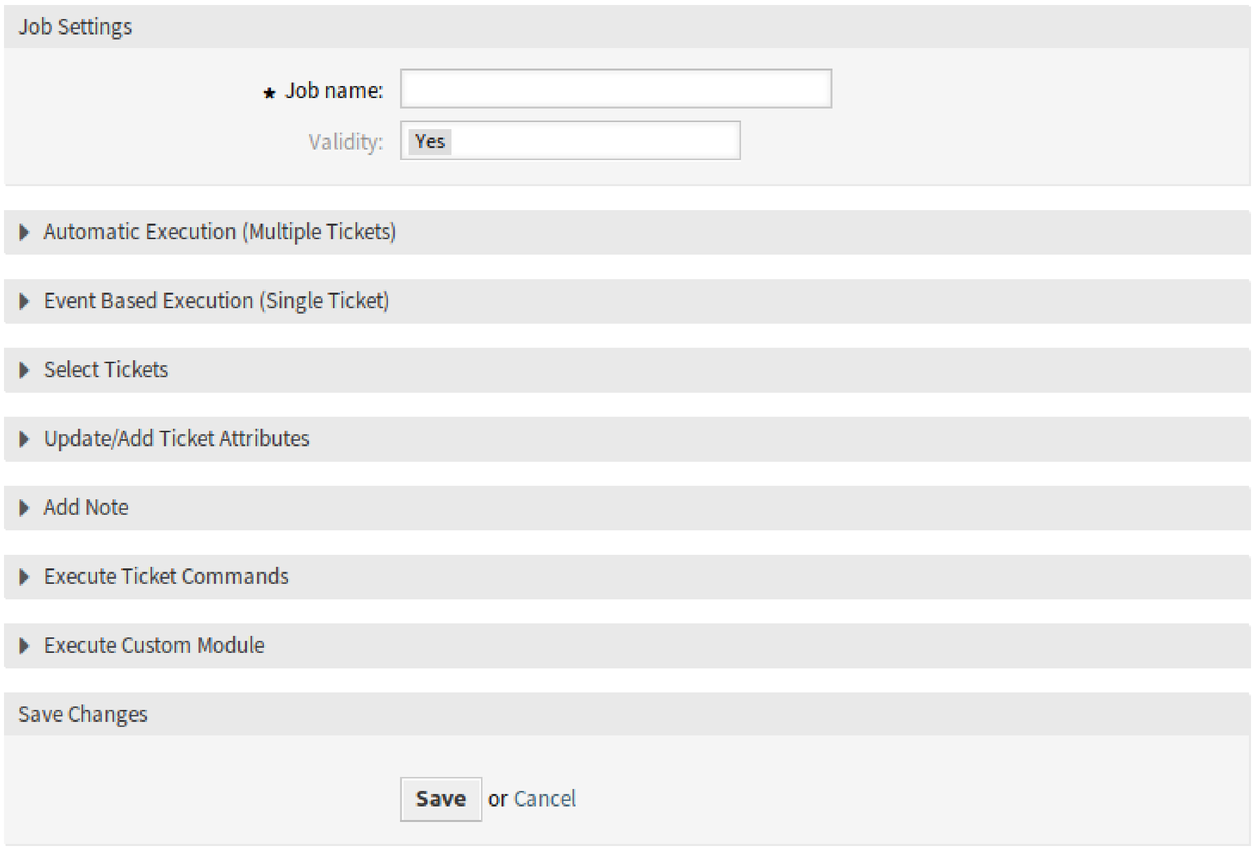
NAME	LAST RUN	VALIDITY	DELETE	RUN NOW!
No data found.				

Fig. 64: Generic Agent Management Screen

5.3.1 Manage Generic Agent Jobs

To create a new generic agent job:

1. Click on the *Add Job* button in the left sidebar.
2. Fill in the required fields.
3. Click on the *Save* button.



Job Settings

★ Job name:

Validity: Yes

▸ Automatic Execution (Multiple Tickets)

▸ Event Based Execution (Single Ticket)

▸ Select Tickets

▸ Update/Add Ticket Attributes

▸ Add Note

▸ Execute Ticket Commands

▸ Execute Custom Module

Save Changes

Fig. 65: Create New Generic Agent Job Screen

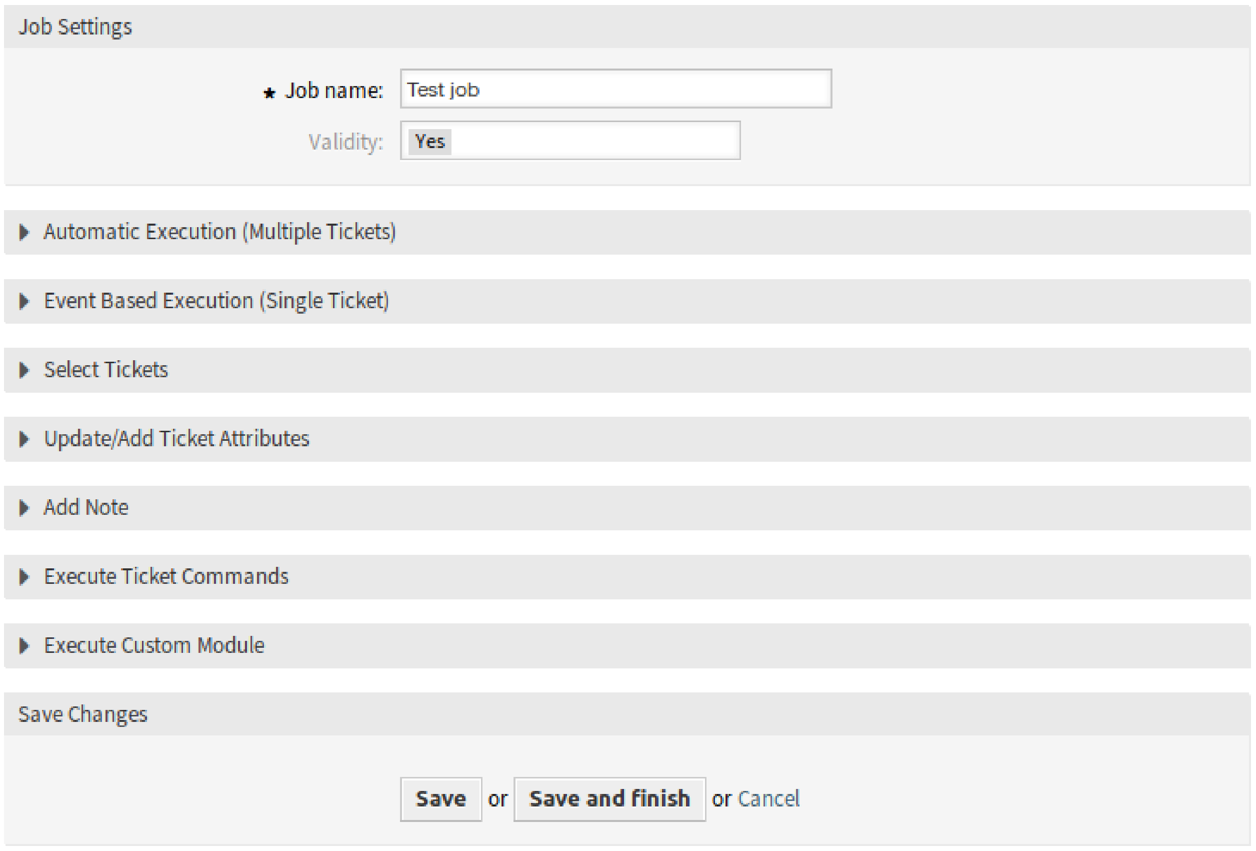
To edit a generic agent job:

1. Click on a generic agent job in the list of generic agent jobs.
2. Modify the fields.
3. Click on the *Save* or *Save and finish* button.

To delete a generic agent job:

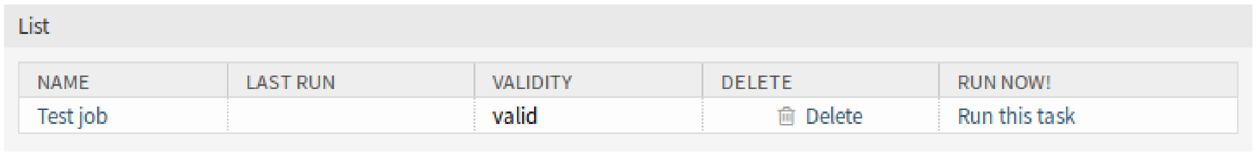
1. Click on the trash icon in the fourth column of the overview table.
2. Click on the *Confirm* button.

Note: If several generic agent jobs are added to the system, use the filter box to find a particular generic agent job by just typing the name to filter.



★ Job name:	Test job
Validity:	Yes

Fig. 66: Edit Generic Agent Job Screen



NAME	LAST RUN	VALIDITY	DELETE	RUN NOW!
Test job		valid	Delete	Run this task

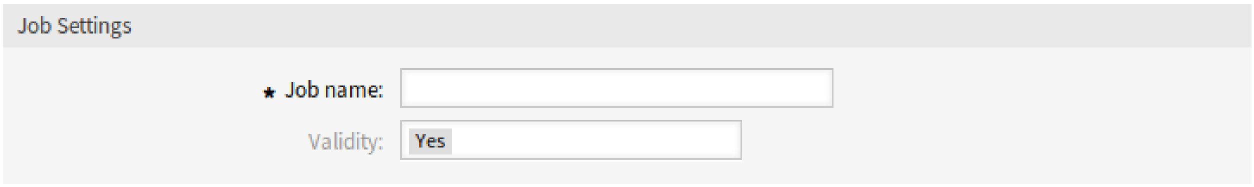
Fig. 67: Delete Generic Agent Job Screen

Warning: The maximum number of 30 *valid* generic agent jobs should not be exceeded. Exceeding this limit may affect the system performance.

5.3.2 Generic Agent Job Settings

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

General Job Settings


Job Settings

★ Job name:	
Validity:	Yes

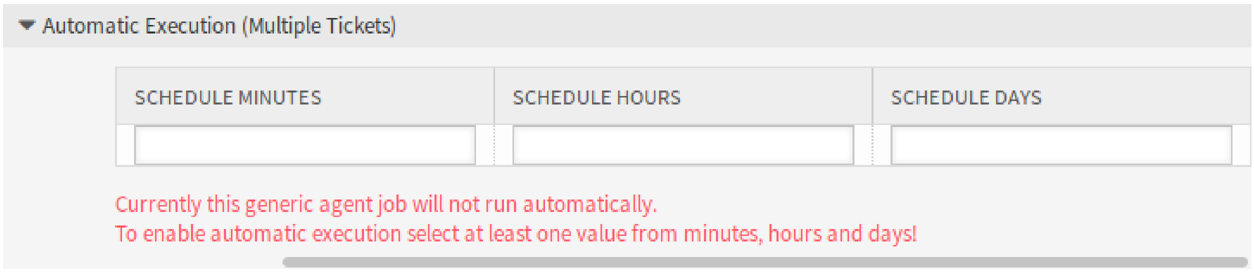
Fig. 68: Job Settings - General

Job Name *
The name of this resource. Any type of characters can be entered to this field including uppercase letters and spaces. The name will be displayed in the overview table.

Validity
Set the validity of this resource. This resource can be used in OTRS only, if this field is set to *Yes*. Setting this field to *No* will disable the use of the resource.

Automatic Execution

Fill in this section to set the times for automatic execution of the job.



SCHEDULE MINUTES	SCHEDULE HOURS	SCHEDULE DAYS

Currently this generic agent job will not run automatically.
To enable automatic execution select at least one value from minutes, hours and days!

Fig. 69: Job Settings - Automatic Execution

Schedule minutes
Select the minutes in which the job has to be executed. For example if *10* is selected, the job will be executed in every hour in 00:10, 01:10, 02:10, etc.

Warning: The automatic execution should not be performed more than one per hour. Exceeding this limit may affect your system performance.

Schedule hours
Select the hours in which the job has to be executed. For example if *10* is selected for minutes and *03* is selected for hours, the job will be executed in every day in 03:10.

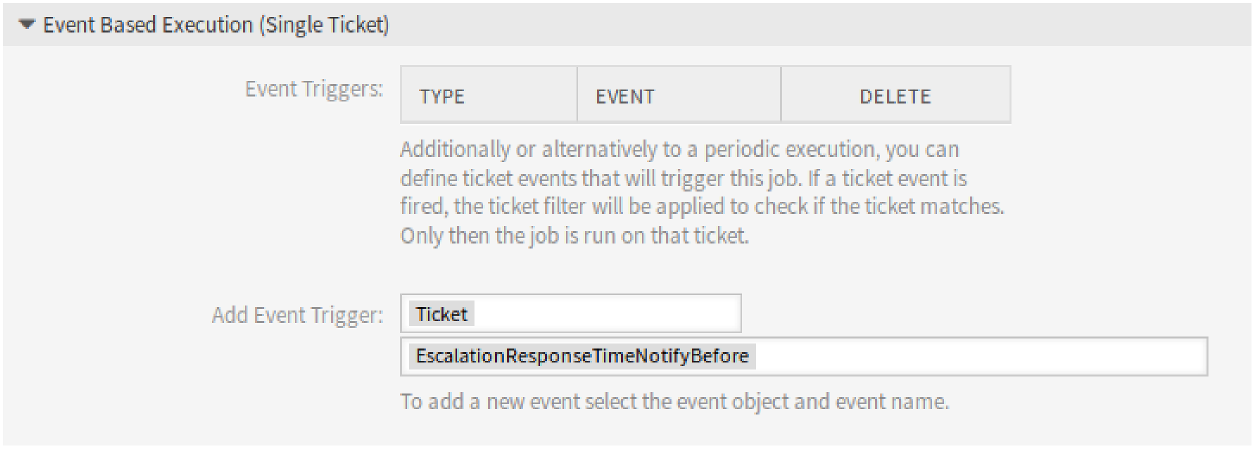
Schedule days
Select the days in which the job has to be executed. For example if *10* is selected for minutes, *03* is selected for hours and *Fri* is selected for days, the job will be executed in every week in Friday 03:10.

Note: Scheduled tasks are using the local system time provided by the operating system OTRS is running on.

Warning: Times scheduled during daylight saving time start and daylight saving time end can have unexpected effects. At daylight saving time start it can be skipped and at daylight saving time end it can be executed twice. It is highly not recommended to schedule times when the daylight saving time starts or ends according to the server timezone.

Event Based Execution

Fill in this section to set events that trigger the execution of the job.



Additionally or alternatively to a periodic execution, you can define ticket events that will trigger this job. If a ticket event is fired, the ticket filter will be applied to check if the ticket matches. Only then the job is run on that ticket.

Ticket

EscalationResponseTimeNotifyBefore

To add a new event select the event object and event name.

Fig. 70: Job Settings - Event Based Execution

Event Triggers
This is a list of already added events. If no events are added yet, the list contains no elements. Elements can be deleted by clicking on the trash icon in the last column.

Add Event Trigger
Select an object and a related event that will trigger the job.

Select Tickets

This section contains all the attributes, that you can use to select the affected tickets.

Dynamic fields of type attachment could be used only in this section. However, it is not updatable by generic agent and this type of dynamic fields are not listed in *Update/Add Ticket Attributes* section.

Fields are self-explained, so there is no additional explanation here.

Update/Add Ticket Attributes

Fill in this section to update and/or add ticket attributes.

Dynamic fields of type date or date and time can be set to absolute, relative or no date. The values can be positive like *+5 days* or negative like *-10 hours* based on the current time.

Dynamic fields of type attachment are not listed in this section.

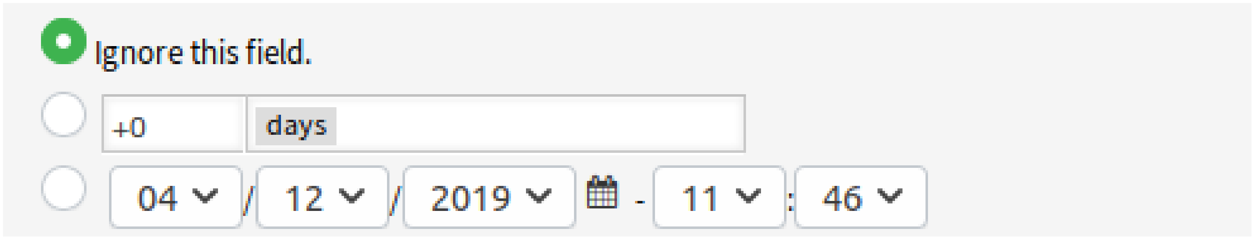


Fig. 71: Date Field in Generic Agent Screen

Fields are self-explained, so there is no additional explanation here.

Add Note

Fill in this section to add a note to tickets that are affected by job execution.

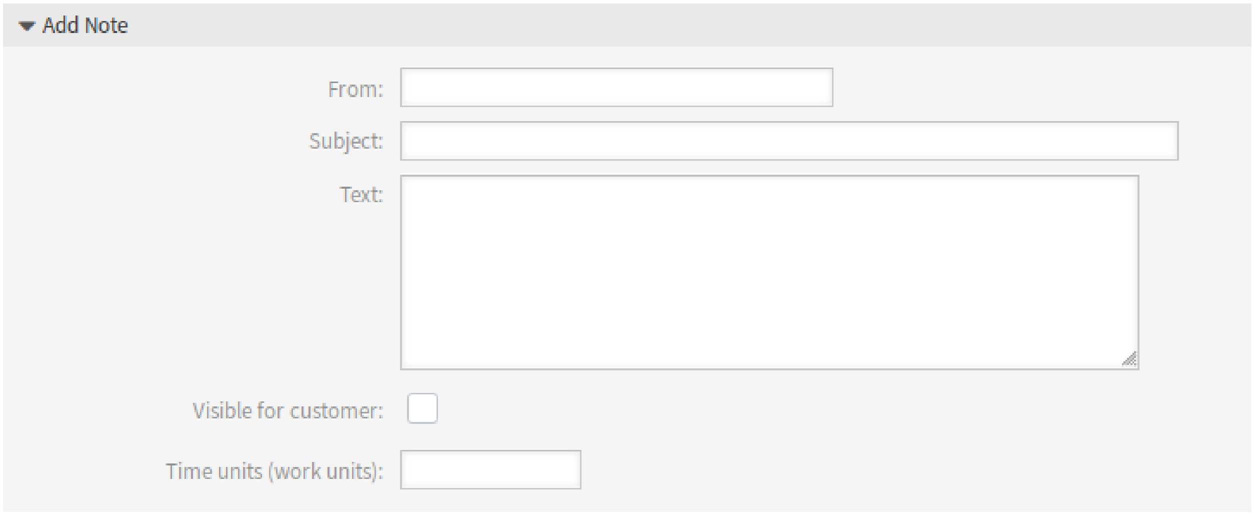
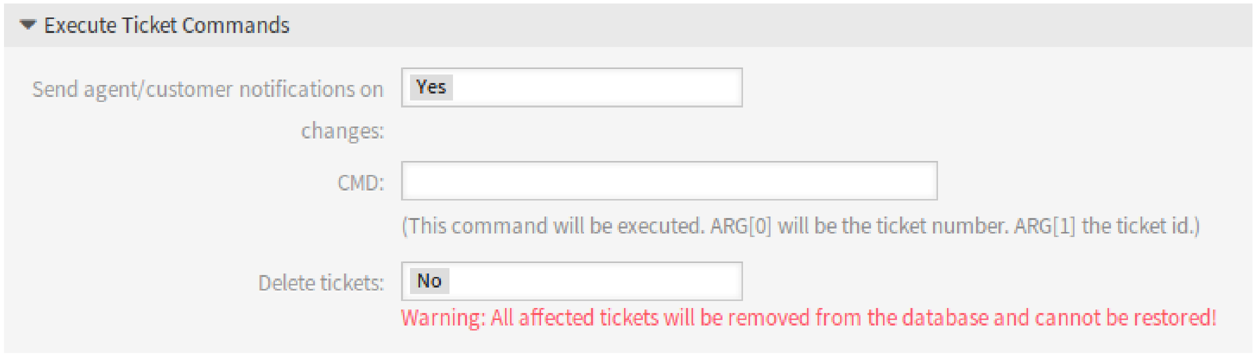


Fig. 72: Job Settings - Add Note

Execute Ticket Commands

Fill in this section, if you would like to make execute a custom command with the job.



Execute Ticket Commands

Send agent/customer notifications on changes:	Yes
CMD:	
	(This command will be executed. ARG[0] will be the ticket number. ARG[1] the ticket id.)
Delete tickets:	No
	Warning: All affected tickets will be removed from the database and cannot be restored!

Fig. 73: Job Settings - Execute Ticket Commands

Send agent/customer notifications on changes
If *Yes* is selected, a notification will be sent to agents and customers about ticket changes.

CMD
Enter a command here, that will be executed. ARG[0] will be the ticket number. ARG[1] the ticket ID. Use `::` as directory separator, if the value contains a path.

See also:

Commands to be run by OTRS are blocked by default due to security reasons. You have to add the command to the allow list as described in *Allow Program Safe to Run* chapter.

Note: This feature is only available to *On-Premise* customers. If you are a *Managed* customer, this feature is taken care of by the *Customer Solutions Team* in **OTRS**. Please contact us via support@otrs.com or in the OTRS Portal.

Delete tickets
If *Yes* is selected, the generic agent job will delete the matched tickets.

Warning: All affected tickets will be removed from the database and cannot be restored!

Execute Custom Module

Note: This feature is only available to *On-Premise* customers. If you are a *Managed* customer, this feature is taken care of by the *Customer Solutions Team* in **OTRS**. Please contact us via support@otrs.com or in the OTRS Portal.

Fill in this section, if you would like to make execute a custom module with the job.

Module
This is the path for the module to be executed.

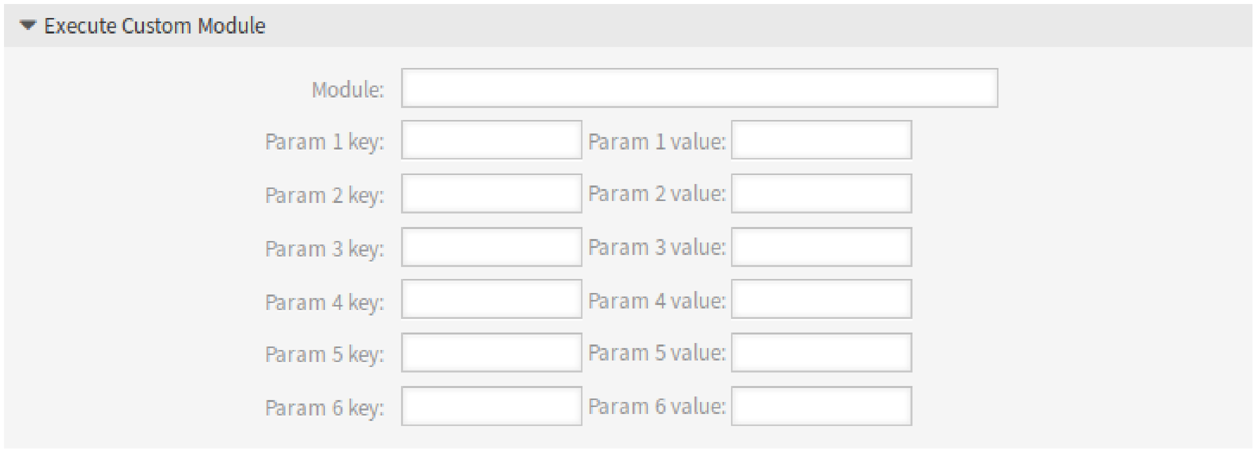


Fig. 74: Job Settings - Execute Custom Module

Note: Use `::` as directory separator, if the value contains a path.

Param key
Enter the key of the parameter, that should be passed to the module.

Param value
Enter the value of the parameter, that should be passed to the module.

5.4 Process Management

Maximizing performance while minimizing human error is a requirement of organizations of all sizes. This need is covered by defined processes and workflows, for recurring tasks. Ensuring all required information is available in the right place, and contacts are informed about their responsibilities like adding information, approving requests, etc.

OTRS supports this requirement by process management. Process tickets help by using the required mandatory and optional fields (see *Dynamic Fields*) that information is not forgotten upon ticket creation or during later steps of the process. Process tickets are simple to handle for customer users and agents, so no intensive training is required.

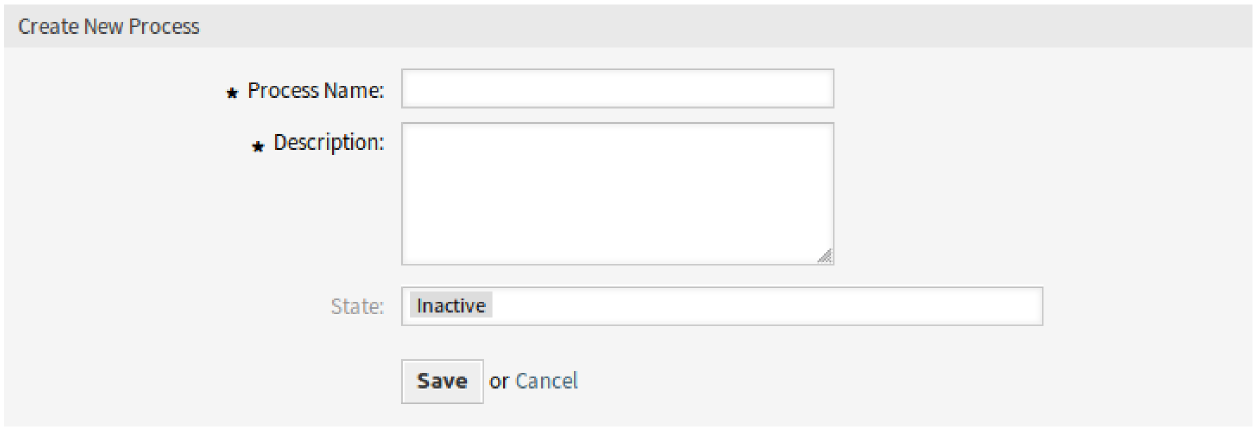
Processes are designed completely and efficiently within the OTRS front end to fit the requirements of your organization.

Use this screen to manage processes in the system. The process management screen is available in the *Process Management* module of the *Processes & Automation* group.

5.4.1 Manage Processes

To create a new process:

1. Click on the *Create New Process* button in the left sidebar.
2. Fill in the required fields.
3. Click on the *Save* button.
4. Add activities, user task activity dialogs, sequence flows and sequence flow actions.
5. Set *State* to *Active*.
6. Deploy all processes.


Create New Process

★ Process Name:	
★ Description:	
State:	Inactive

Fig. 75: Create New Process Screen

To edit a process:

1. Click on a process in the list of processes.
2. Modify the fields and the process path.
3. Click on the *Save* or *Save and finish* button.
4. Deploy all processes.

To copy a process:

1. Click on the copy icon in the fifth column of list of processes.
2. Click on the newly created process to edit it.

To delete a process:

1. Click on a process in the list of processes.
2. Set the *State* option to *Inactive*.
3. Click on the *Save* button. A new *Delete Inactive Process* button will appear in the left sidebar.
4. Click on the *Delete Inactive Process* button.
5. Click on the *Delete* button in the confirmation screen.
6. Deploy all processes.

Warning: Processes are written into file in Perl format. Without deploying, all processes are still in this cache file even if they are deleted or the *State* option is set to *Inactive* or *FadeAway*. Don't forget to deploy all processes after modifications!

Give some time for agents to complete the running process tickets before the process will be deleted. It is possible to mark a process for deletion, i. e. set the process as not to be selected anymore. Process states can be:

Active
Processes can be used in new process tickets.

FadeAway
Agents and customer users cannot select this process for new tickets and neither can they assign this process to an existing ticket, but tickets where this process was assigned to can still use the process. For automation, other processes can still create new tickets with this process or assign this process to existing tickets.

Inactive
Processes are deactivated and cannot be used for new or existing tickets.

To deploy all processes:

1. Click on the *Deploy All Processes* button in the left sidebar.

Note: New or modified processes have to be deployed in order to affect the behavior of the system. Setting the *State* option to *Active* just indicates, which processes should be deployed.

To export a process:

1. Click on the export icon in the fourth column of list of processes.
2. Choose a location in your computer to save the `Export_ProcessEntityID_xxx.yml` file.

Warning: The exported file contains only the process itself, and doesn't contain the *Queues*, *Agents*, *Dynamic Fields*, etc. needed for the process.

To import a process:

1. Click on the *Browse…* button of the *Configuration Import* widget in the left sidebar.
2. Select a previously exported `.yml` file.
3. Click on the *Import process configuration* button.
4. Deploy all processes.

Note: Before import a process, it is still necessary to create all *Queues*, *Agents* and *Dynamic Fields*, as well as to set *System Configuration*, that are needed by each process before the import. If the process requires the use of *Access Control Lists (ACL)* those are also needed to be set manually.

The system includes several examples of predefined processes that can help in some specific cases.

The following Ready2Adopt processes are available from the *Ready2Adopt Processes* widget:

```
Application for leave
Conference Room Reservation
Office Materials Management
Order Request Management
Request for Leave Management
Start RMA
Travel expense
```

To import a Ready2Adopt process:

1. Select a process from the *Ready2Adopt Processes* widget in the left sidebar.
2. Click on the *Import Ready2Adopt process* button.
3. Deploy all processes.

During the import process, the system takes care of creating the needed dynamic fields and/or any needed updates to the system configuration.

Note: If several processes are added to the system, use the filter box to find a particular process by just typing the name to filter.

Warning: The maximum number of 50 *deployed* processes should not be exceeded. Exceeding this limit may affect the system performance.

5.4.2 Process Elements

The element names of the process modeler have been adapted to the *Business Process Model and Notation* (BPMN) ISO naming convention.

Activities

Activities are the basic steps for the process.

Click on the *Activities* item in the *Available Process Elements* widget in the left sidebar. This action will expand the *Activities* options and will collapse all others doing an accordion like effect. Click on the *Create New Activity* button.

The following task activities can be used as basic elements of the process.

Script task activity

Script task activity is executed by the process management module and it can set dynamic field values or manage tickets automatically.

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

Activity name *

The name of this resource. Any type of characters can be entered to this field including uppercase letters and spaces.

Activity type *

The following types of task activities can be used:

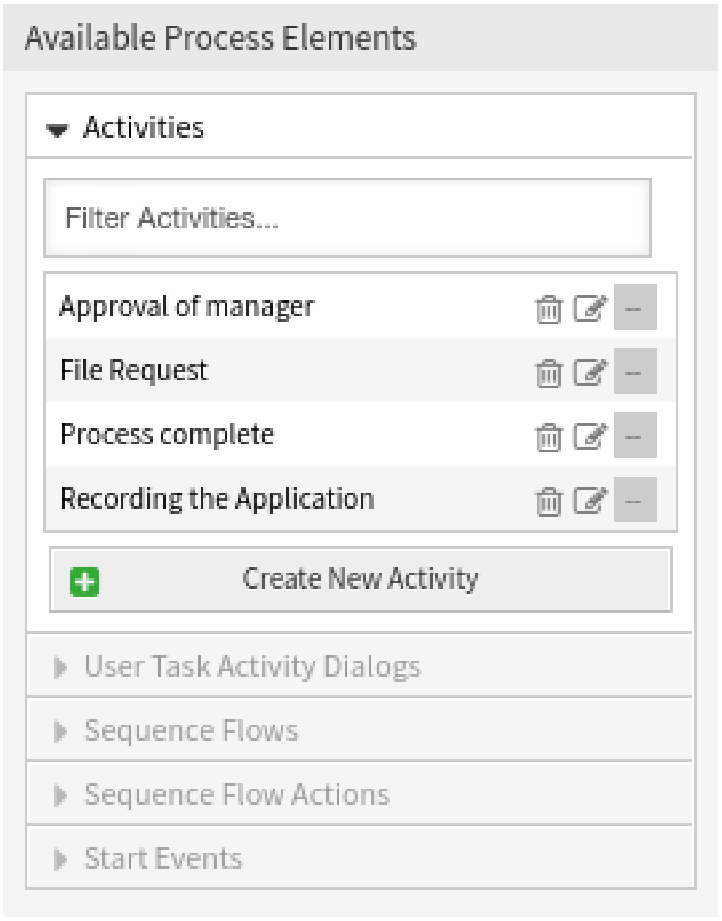
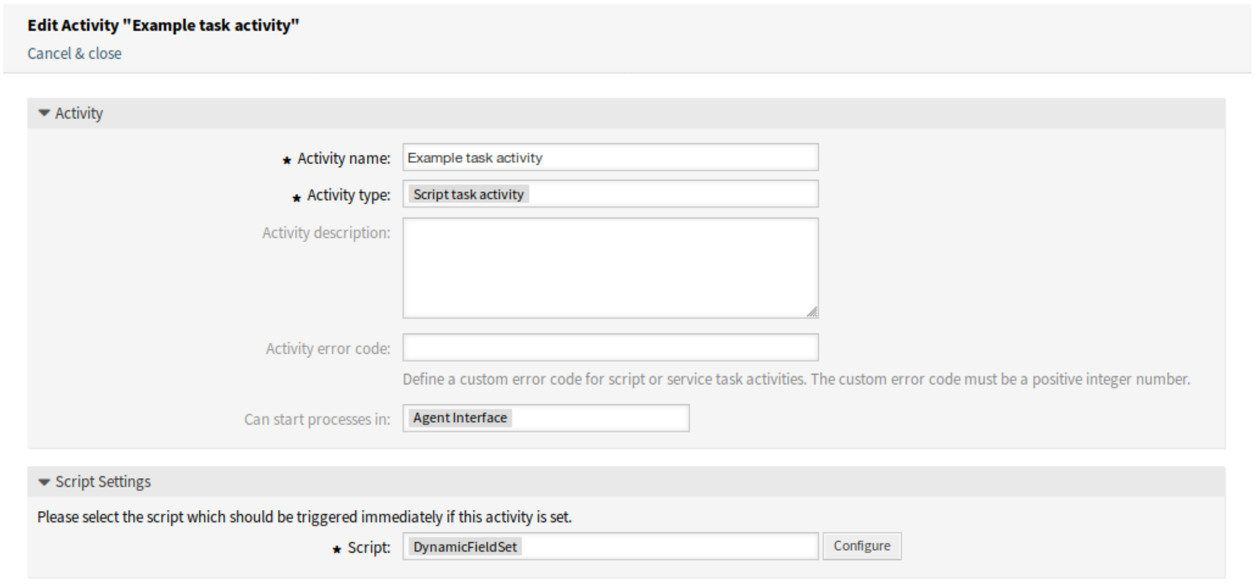


Fig. 76: Activities


Edit Activity "Example task activity"

Cancel & close

Activity

⋆ Activity name:	Example task activity
⋆ Activity type:	Script task activity
Activity description:	
Activity error code:	
	Define a custom error code for script or service task activities. The custom error code must be a positive integer number.
Can start processes in:	Agent Interface

Script Settings

Please select the script which should be triggered immediately if this activity is set.

⋆ Script:	DynamicFieldSet

Fig. 77: Script Task Activity Window

- Script task activity (selected for now)
- Service task activity
- User task activity

Activity description
Add additional information to this resource. It is recommended to always fill this field as a description of the resource with a full sentence for better clarity.

Activity error code
Define a custom error code for script or service task activities. The custom error code must be a positive integer number.

Can start processes in
Defines, where can the processes be started by the agents or customer users. A process can be started in the following interfaces:

- Agent Interface
- Agent and External Interface
- External Interface

Script *
In this drop-down can be selected which script should be triggered immediately if the activity is set. Click on the *Configure* button to add parameters (key-value pairs) for the script.

See also:

Each module has its own and different parameters. Please refer to the *Process Modules* section to learn all required and optional parameters.

Note: Many script task activities need an agent user with the corresponding permissions to complete successfully. In all script task activity configuration dialogs the executing agent user can be replaced with a different one.

This can be done with key `UserID` and the agent user ID with the required permissions as value.

Service task activity
Service task activity uses a web service to complete the task.

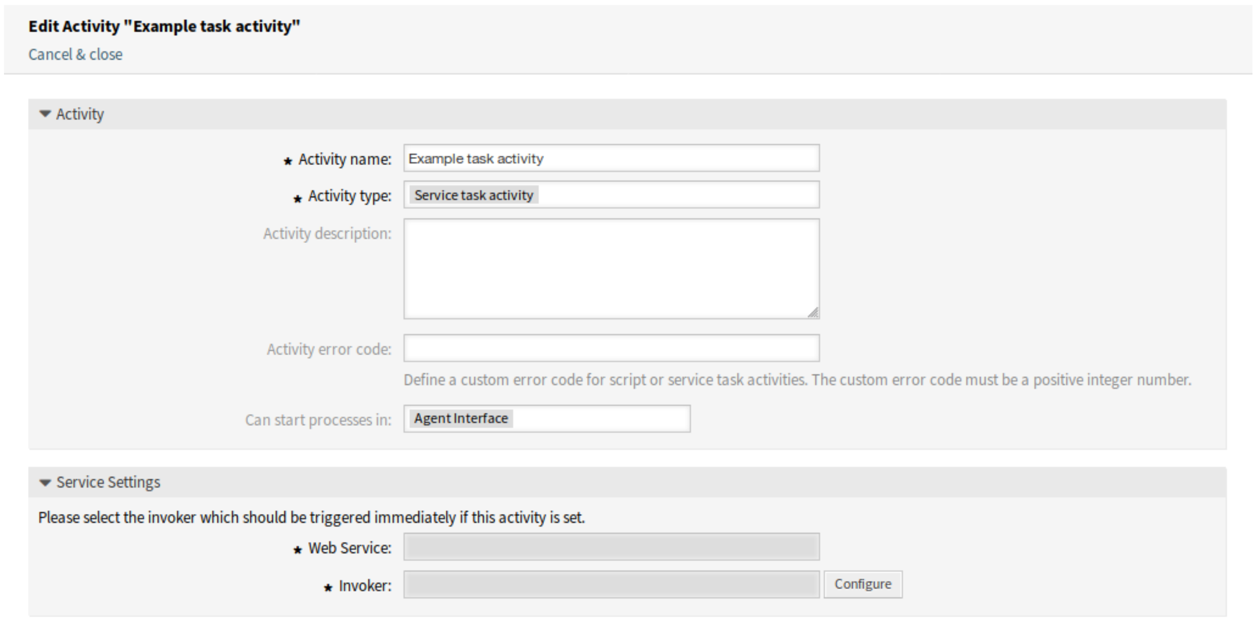
The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

Activity name *
The name of this resource. Any type of characters can be entered to this field including uppercase letters and spaces.

Activity type *
The following types of task activities can be used:

- Script task activity
- Service task activity (selected for now)
- User task activity

Activity description
Add additional information to this resource. It is recommended to always fill this field as a description of the resource with a full sentence for better clarity.


Edit Activity "Example task activity"

Cancel & close

Activity

★ Activity name: Example task activity

★ Activity type: Service task activity

Activity description:

Activity error code:

Define a custom error code for script or service task activities. The custom error code must be a positive integer number.

Can start processes in: Agent Interface

Service Settings

Please select the invoker which should be triggered immediately if this activity is set.

★ Web Service:

★ Invoker:

Fig. 78: Service Task Activity Window

Activity error code
: Define a custom error code for script or service task activities. The custom error code must be a positive integer number.

Can start processes in
: Defines, where can the processes be started by the agents or customer users. A process can be started in the following interfaces:

  - Agent Interface
  - Agent and External Interface
  - External Interface

Web Service *
: Select one of the *Web Services* from the drop-down list.

Invoker *
: Select an invoker for the web service. Click on the *Configure* button to add parameters for the invoker.

User task activity

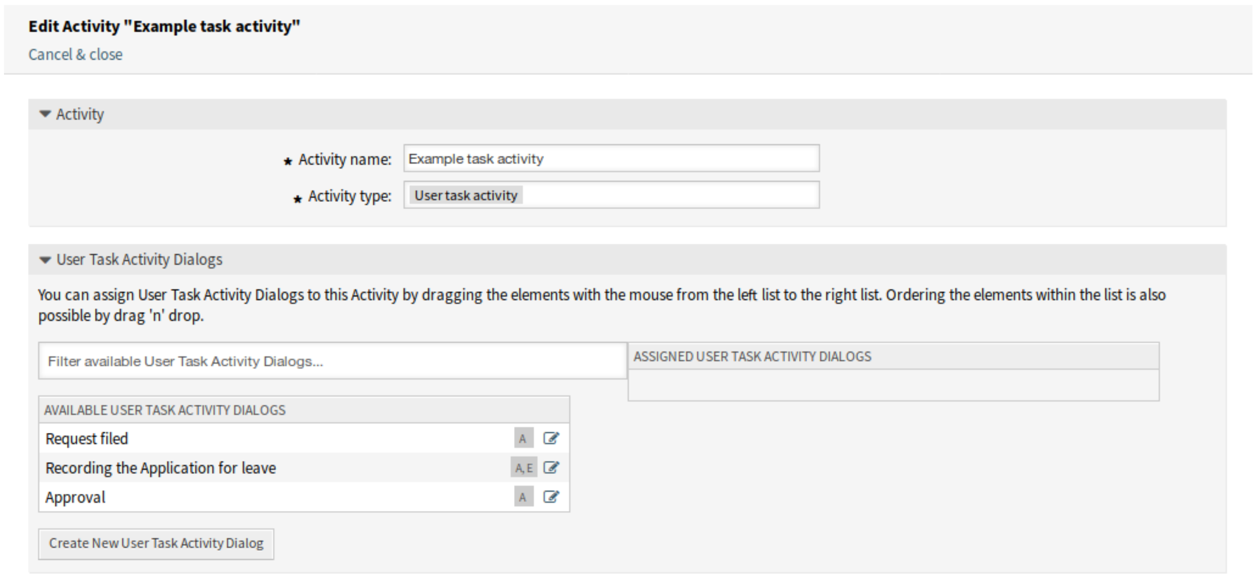
User task activity can be used when the task is being done by an agent or a customer user.

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

Activity name *
: The name of this resource. Any type of characters can be entered to this field including uppercase letters and spaces.

Activity type *
: The following types of task activities can be used:

  - Script task activity
  - Service task activity


Edit Activity "Example task activity"

Cancel & close

Activity

★ Activity name: Example task activity

★ Activity type: User task activity

User Task Activity Dialogs

You can assign User Task Activity Dialogs to this Activity by dragging the elements with the mouse from the left list to the right list. Ordering the elements within the list is also possible by drag 'n' drop.

Filter available User Task Activity Dialogs...

ASSIGNED USER TASK ACTIVITY DIALOGS

AVAILABLE USER TASK ACTIVITY DIALOGS

Request filed	A
Recording the Application for leave	A, E
Approval	A

Fig. 79: User Task Activity Window

- User task activity (selected for now)

User Task Activity Dialogs

You can assign user task activity dialogs to this activity by dragging the elements with the mouse from the left list to the right list. Ordering the elements within the list is also possible by drag and drop.

Click on the *Create New User Task Activity Dialog* button to create new dialogs.

For each configured field in a user task activity dialog, the process modeler is able to choose the following options for the fields:

- Do not show Field
- Show Field
- Show Field As Mandatory

If you use *Show Field* for a select field the empty value needs to be added to the possible values. Otherwise without having an empty value available, this field will be always a mandatory field.

User Task Activity Dialogs

A user task activity dialog contains the actual user interaction of the process and consists of fields that can be displayed to the users in the ticket detail view or can be set automatically by them.

Click on the *User Task Activity Dialogs* item in the *Available Process Elements* widget in the left sidebar. This action will expand the *User Task Activity Dialogs* options and will collapse all others doing an accordion like effect.

Click on the *Create New User Task Activity Dialog* button.

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

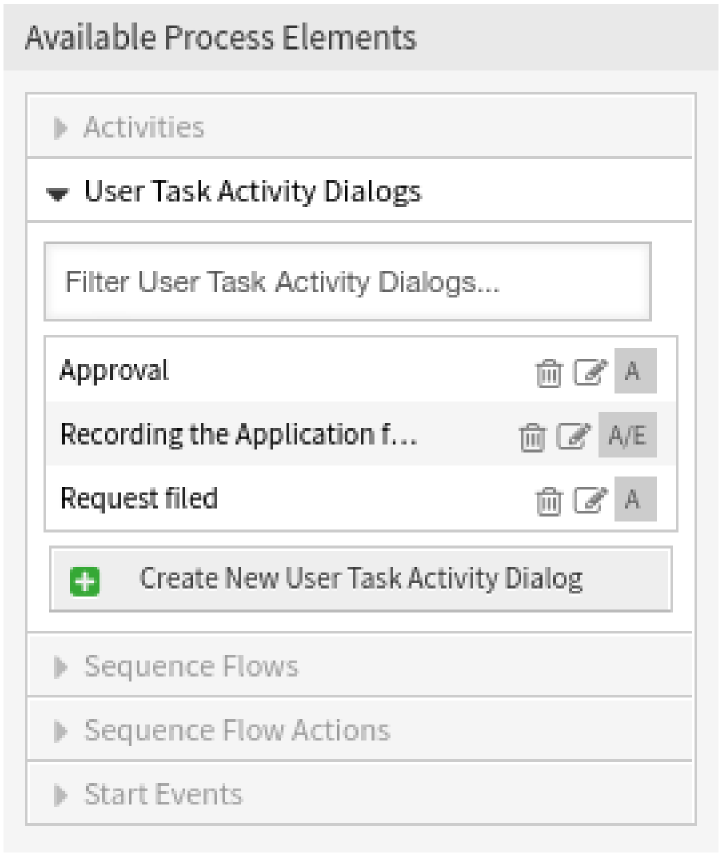
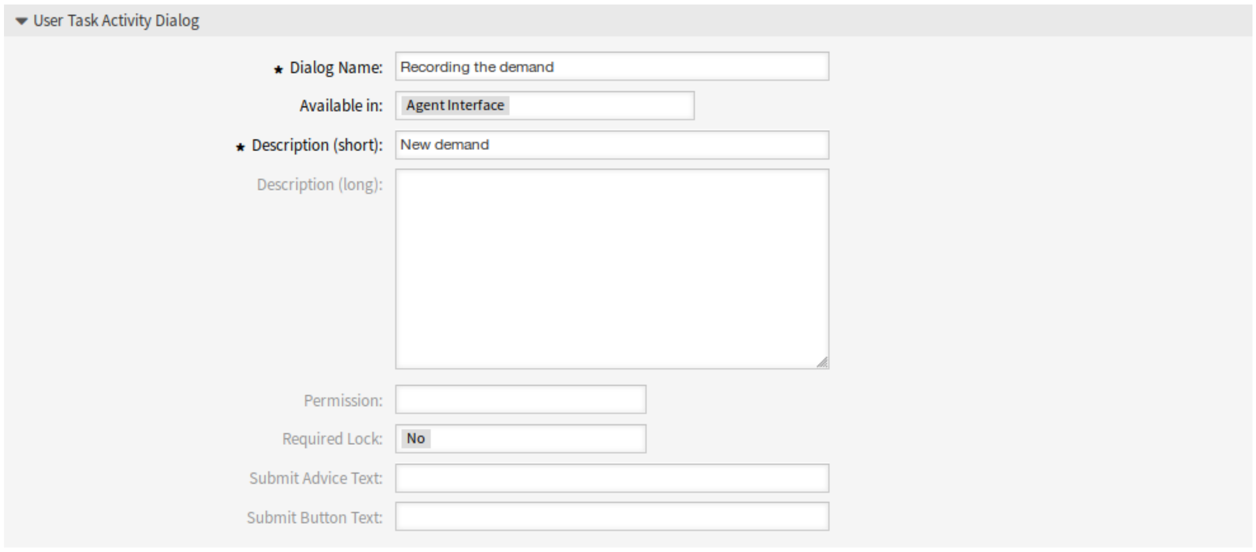

Available Process Elements
Activities
User Task Activity Dialogs
Filter User Task Activity Dialogs...
Approval
A
Recording the Application f...
A/E
Request filed
A
Create New User Task Activity Dialog
Sequence Flows
Sequence Flow Actions
Start Events

Fig. 80: User Task Activity Dialogs


User Task Activity Dialog

* Dialog Name:	Recording the demand
Available in:	Agent Interface
* Description (short):	New demand
Description (long):	
Permission:	
Required Lock:	No
Submit Advice Text:	
Submit Button Text:	

Fig. 81: Add User Task Activity Dialog

Dialog Name *
: The name of this resource. Any type of characters can be entered to this field including uppercase letters and spaces.

Available in
: It determines in which interface the dialog appears.

  Possible values are:

  - Agent Interface
  - External Interface
  - Agent and External Interface

Description (short) *
: The dialog can be explained briefly here. The description is displayed in the user task activity dialog in the ticket detail view.

Description (long)
: The dialog can be explained in more detail here. The description is displayed in the user task activity dialog in the ticket detail view.

Permission
: By means of this setting, the visibility of the process dialogs can be controlled, based on the user's permissions on the queue in which the ticket is currently located. If a user does not have the here configured permissions on the group which is associated to the queue in which the ticket is currently located, the dialog will not be visible in the ticket detail view.

  This permission setting does not affect the visibility of the first user task activity dialog while creating a new process ticket.

Required Lock
: By means of this setting, it is achieved that the ticket is automatically locked to the executing agent or not once an agent opens the user task activity dialog in the ticket detail view.

  Possible values are:

  - *yes*: the ticket is locked to the executing agent while executing
  - *no*: the ticket is not locked to the executing agent while executing

Submit Advice Text
: This text will be shown above the submit button and may contain an advice for the user.

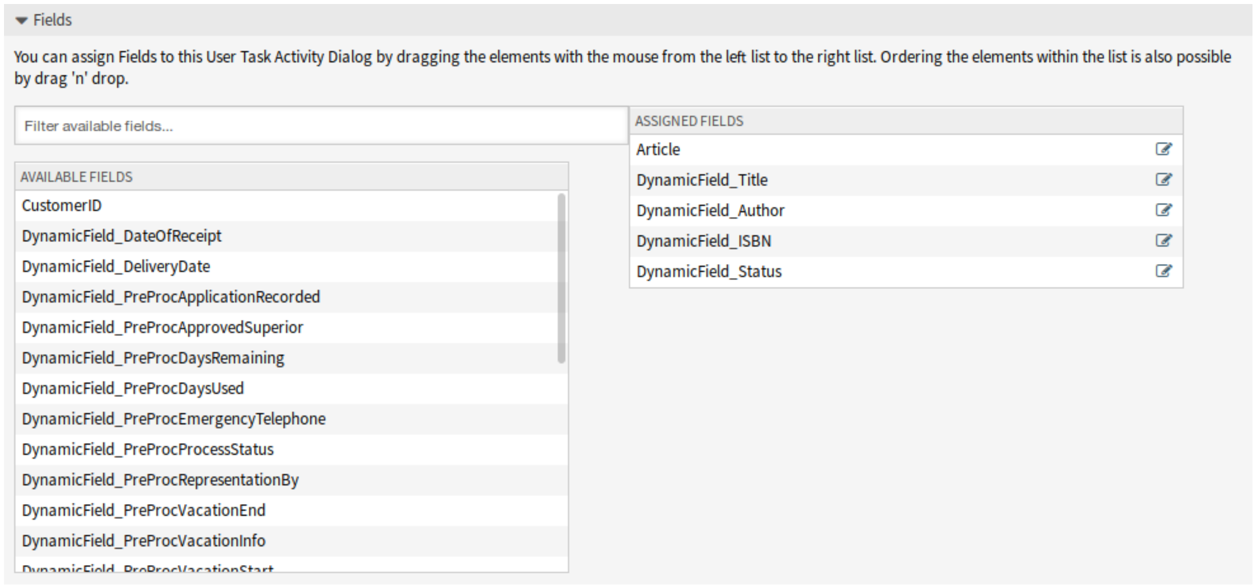
Submit Button Text
: With this text the button label can be changed from *Submit* to the text entered here.

To assign fields to the user task activity dialog simple drag the required field from the *Available Fields* pool and drop into the *Assigned Fields* pool. The order in the *Assigned Fields* pool is the order as the fields will have in the screen. To modify the order simply drag and drop the field within the pool to rearrange it in the correct place.

Available Fields
: All available fields of the system are shown here in an alphabetical order.

Assigned Fields
: The fields assigned to the dialog are displayed here.

As soon as the fields are dropped into the *Assigned Fields* pool another popup screen is shown with some details about the field.



Fields

You can assign Fields to this User Task Activity Dialog by dragging the elements with the mouse from the left list to the right list. Ordering the elements within the list is also possible by drag 'n' drop.

Filter available fields...

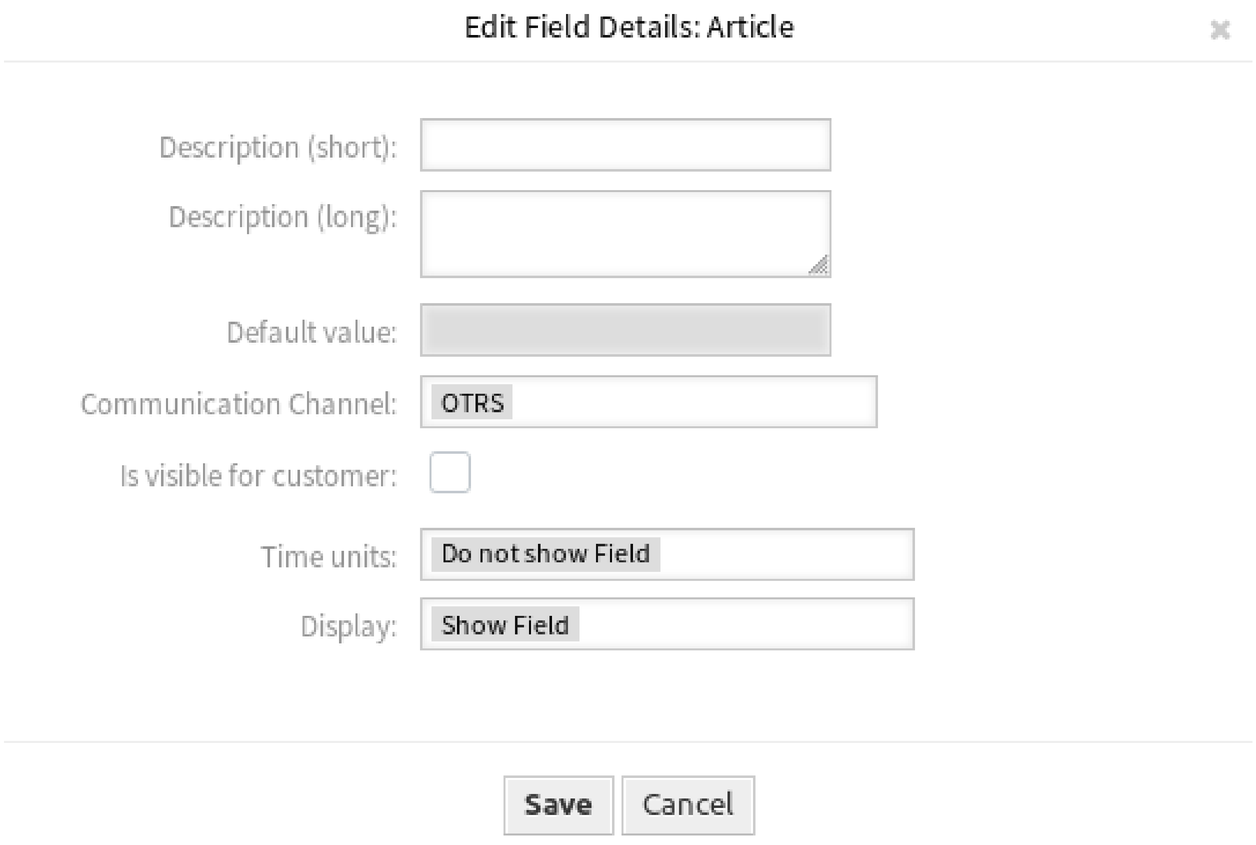
AVAILABLE FIELDS

- CustomerID
- DynamicField_DateOfReceipt
- DynamicField_DeliveryDate
- DynamicField_PreProcApplicationRecorded
- DynamicField_PreProcApprovedSuperior
- DynamicField_PreProcDaysRemaining
- DynamicField_PreProcDaysUsed
- DynamicField_PreProcEmergencyTelephone
- DynamicField_PreProcProcessStatus
- DynamicField_PreProcRepresentationBy
- DynamicField_PreProcVacationEnd
- DynamicField_PreProcVacationInfo
- DynamicField_PreProcVacationStart

ASSIGNED FIELDS

- Article
- DynamicField_Title
- DynamicField_Author
- DynamicField_ISBN
- DynamicField_Status

Fig. 82: Add User Task Activity Dialog Fields



Edit Field Details: Article

Description (short):	
Description (long):	
Default value:	
Communication Channel:	OTRS
Is visible for customer:	☐
Time units:	Do not show Field
Display:	Show Field

Fig. 83: Edit User Task Activity Dialog Fields

Description (short)
A short explanatory text for the field can be specified here. The description will be displayed underneath the field.

Description (long)
A more detailed explanatory text for the field can be specified here. The description will be displayed when hovering over the icon next to the label of the field.

Default value
Defines a default value for that field.

Communication Channel
Defines the communication channel. The process modeler is able to choose the following options for the fields:

- Email
- OTRS
- Phone

Warning: In the same activity dialog, you can use either an article field with email or a regular article field. Using both fields together will cause a process error.

Is visible to customer
If this is checked, the customer user will be able to see the article created by the process.

Time units
For each configured field in a user task activity dialog, the process modeler is able to choose the following options for the fields:

- Do not show Field
- Show Field
- Show Field As Mandatory

Display
For each configured field in a user task activity dialog, the process modeler is able to choose the following options for the fields:

- Do not show Field
- Show Field
- Show Field As Mandatory

The option *Do not show Field* offers the possibility to set a field automatically to a certain value configured as a *Default value*.

If you use *Show Field* for a select field the empty value needs to be added to the possible values. Otherwise without having an empty value available, this field will be always a mandatory field.

Sequence Flows

A sequence flow is used to connect objects of a process and to represent the flow. The sequence flow controls the sequence of activities by checking whether certain defined conditions are met.

Click on the *Sequence Flows* item in the *Available Process Elements* widget in the left sidebar. This action will expand the *Sequence Flows* options and will collapse all others doing an accordion like effect.

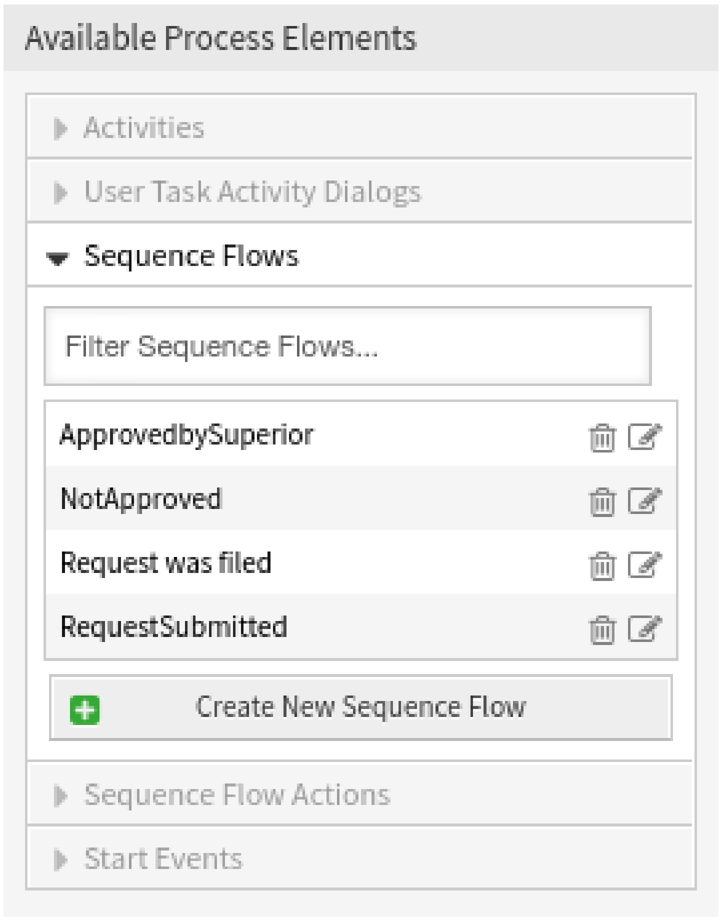


Fig. 84: Sequence Flows

Click on the *Create New Sequence Flow* button.

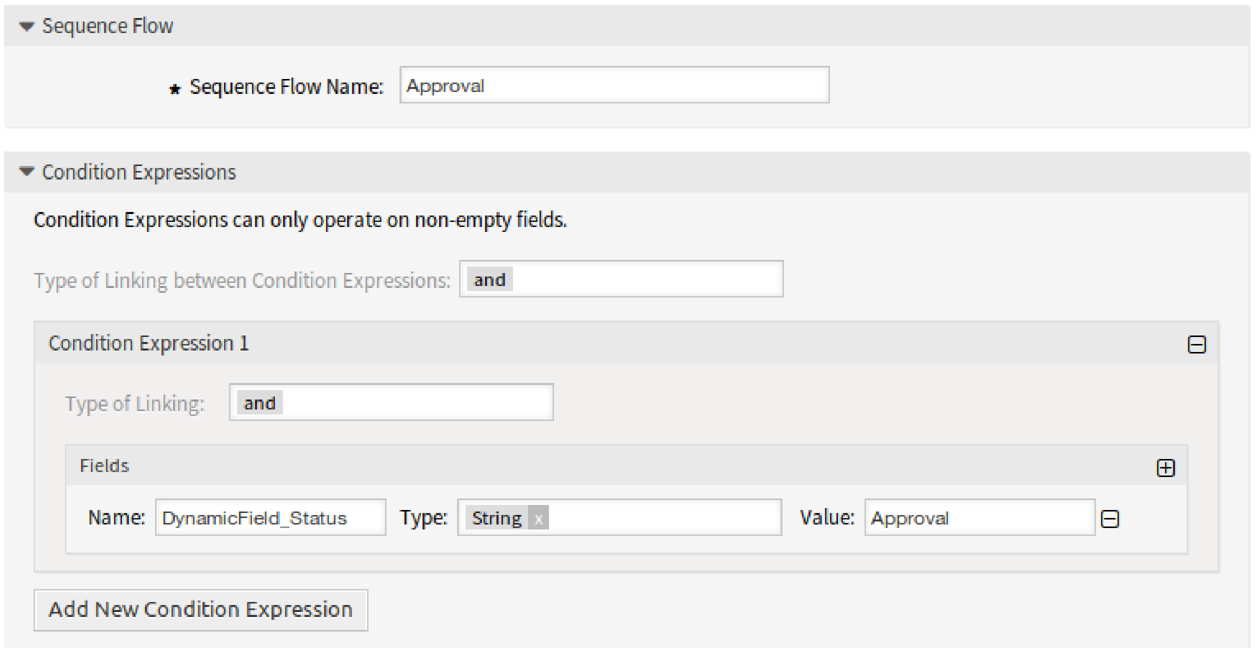
The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

Sequence Flow Name *
: The name of this resource. Any type of characters can be entered to this field including uppercase letters and spaces.

Type of Linking between Condition Expressions
: A sequence flow consists of one or many condition expressions. If the sequence flow has more than one condition expressions you can select the type of linking between the condition expressions.

  Possible logical operators:

  - *and*: the output is true if all inputs are true.
  - *or*: the output is true if one or many of its inputs are true.


Sequence Flow

★ Sequence Flow Name: Approval

Condition Expressions

Condition Expressions can only operate on non-empty fields.

Type of Linking between Condition Expressions: and

Condition Expression 1

Type of Linking: and

Fields

Name: DynamicField_Status Type: String Value: Approval

Add New Condition Expression

Fig. 85: Add Sequence Flow

- *xor*: the output is true only if exactly one of its inputs is true.

A new condition expression can be added by clicking on the *Add New Condition Expression* button.

Type of Linking
A condition expression can have one or many conditions. If your condition expression has more than one condition you can select the type of linking between them.

Possible logical operators:

- *and*: the output is true if all inputs are true.
- *or*: the output is true if one or many of its inputs are true.
- *xor*: the output is true only if exactly one of its inputs is true.

Name
The name field contains the name of the attribute which shall be checked.

In case of dynamic fields the prefix `DynamicField_` has to be used in front of its name. Make sure that the spelling of the attribute name is correct.

Type
Three types of condition checks can be selected here.

- *String*: checks for a fixed value.
- *Regular Expression*: checks for patterns using regular expressions.
- *Validation Module*: checks if a certain logic is met which is contained in the validation module. Validation modules are custom made.

Value
The value field can contain the following values:

- string,
- regular expression,

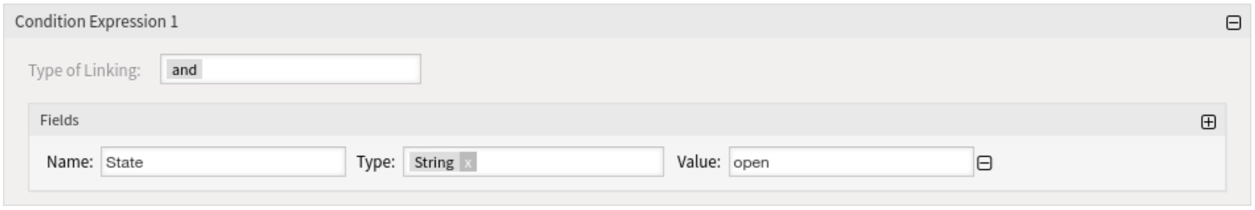
- *OTRS Tags*,
- a combination of the above.

By means of the + symbol on the right side, a new condition consisting of name, type and value can be added.

Examples:

Check if the ticket state is *open*.

- *Name*: `State`
- *Type*: *String*
- *Value*: *open*



Condition Expression 1

Type of Linking:	and

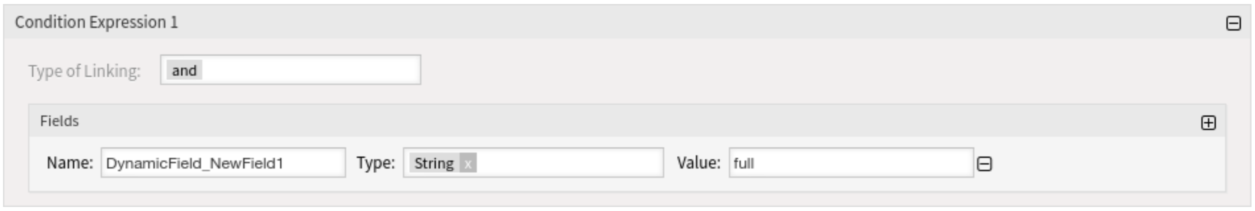
Fields

Name:	State	Type:	String	Value:	open

Fig. 86: Sequence Flow Example

Check if the dynamic field named `NewField1` contains the string *full*.

- *Name*: `DynamicField_NewField1`
- *Type*: *String*
- *Value*: *full*



Condition Expression 1

Type of Linking:	and

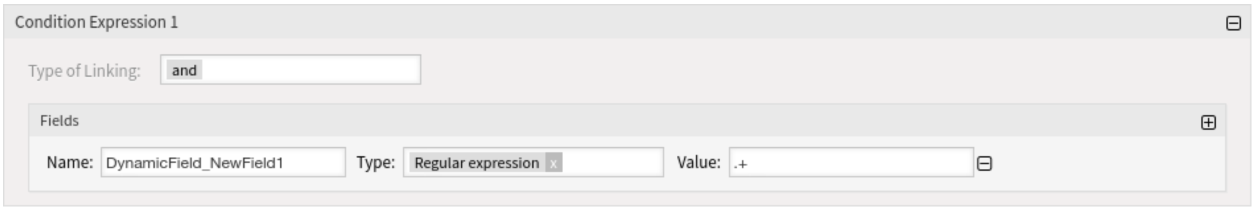
Fields

Name:	DynamicField_NewField1	Type:	String	Value:	full

Fig. 87: Sequence Flow Example

Check if the dynamic field named `NewField1` contains something.

- *Name*: `DynamicField_NewField1`
- *Type*: *Regular expression*
- *Value*: `.+`



Condition Expression 1

Type of Linking:	and

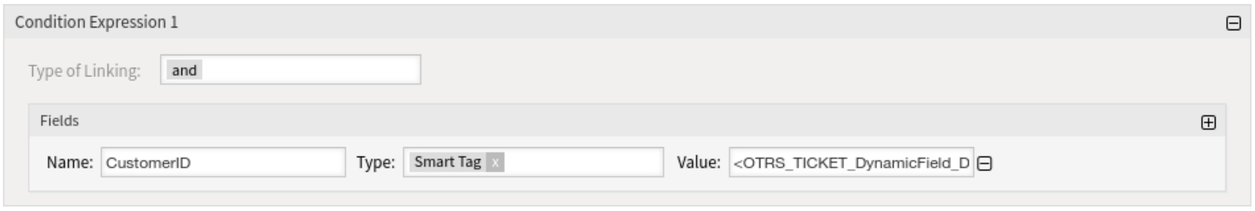
Fields

Name:	DynamicField_NewField1	Type:	Regular expression	Value:	.+

Fig. 88: Sequence Flow Example

Check if the field customer ID has the same content as the dynamic field named `DF1`.

- *Name*: `CustomerID`
- *Type*: *Smart Tag*
- *Value*: `<OTRS_TICKET_DynamicField_DF1>`



Condition Expression 1

Type of Linking: and

Fields

Name: CustomerID Type: Smart Tag Value: <OTRS_TICKET_DynamicField_D

Fig. 89: Sequence Flow Example

Sequence Flow Actions

With sequence flow actions ticket data can be changed or created, mails can be send from a ticket and data can be pushed from one OTRS object to the other.

Sequence flow actions need to be associated to a sequence flow. A sequence flow action is executed if the sequence flow is triggered and it gets executed after the following activity.

Click on the *Sequence Flow Actions* item in the *Available Process Elements* widget in the left sidebar. This action will expand the *Sequence Flow Actions* options and will collapse all others doing an accordion like effect.

Click on the *Create New Sequence Flow Action* button.

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

Sequence Flow Action Name *
The name of this resource. Any type of characters can be entered to this field including uppercase letters and spaces.

Sequence Flow Action Module *
In this drop-down field it can be selected which action module shall be executed.

If both fields are filled in, click on the *Save* button. A new *Configure* button will appear next to the action module field. Click on the *Configure* button and add the needed configuration parameter keys and values.

See also:

Each module has its own and different parameters. Please refer to the *Process Modules* section to learn all required and optional parameters.

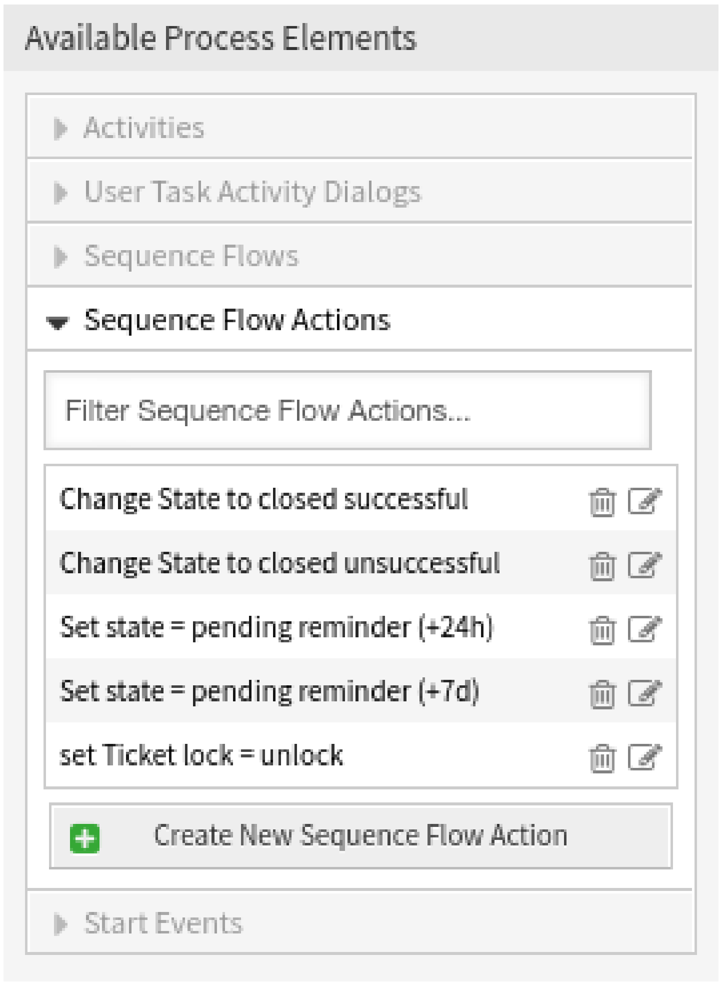
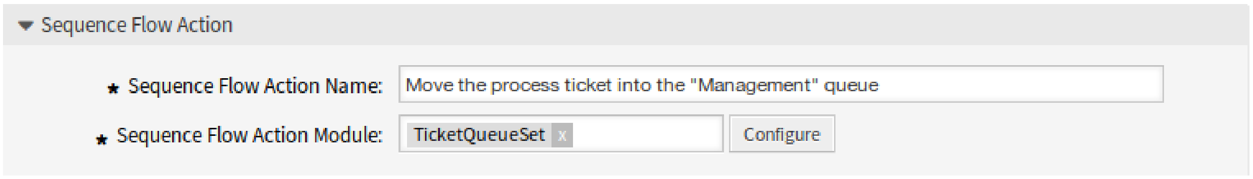
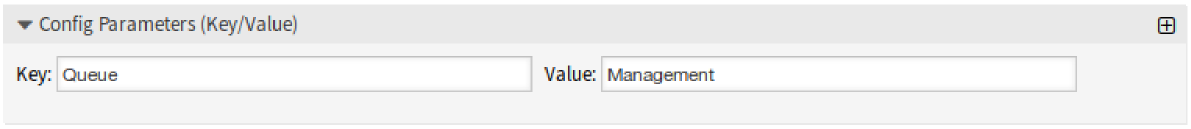

Available Process Elements
Activities
User Task Activity Dialogs
Sequence Flows
Sequence Flow Actions
Filter Sequence Flow Actions...
Change State to closed successful
Change State to closed unsuccessful
Set state = pending reminder (+24h)
Set state = pending reminder (+7d)
set Ticket lock = unlock
Create New Sequence Flow Action
Start Events

Fig. 90: Sequence Flow Actions


Sequence Flow Action

* Sequence Flow Action Name:	Move the process ticket into the "Management" queue
* Sequence Flow Action Module:	TicketQueueSet

Fig. 91: Add Sequence Flow Action


Config Parameters (Key/Value)

Key: Queue Value: Management

Fig. 92: Sequence Flow Action Parameters

Note: Many script task activities need an agent user with the corresponding permissions to complete successfully. In all script task activity configuration dialogs the executing agent user can be replaced with a different one. This can be done with key `UserID` and the agent user ID with the required permissions as value.

Start Events

Processes can contain a normal start point and a scheduled start point. The schedule of the start event can be configured via an interface.

Click on the *Start Events* item in the *Available Process Elements* widget in the left sidebar. This action will expand the *Start Events* options and will collapse all others doing an accordion like effect. Click on the *Create New Start Event* button.

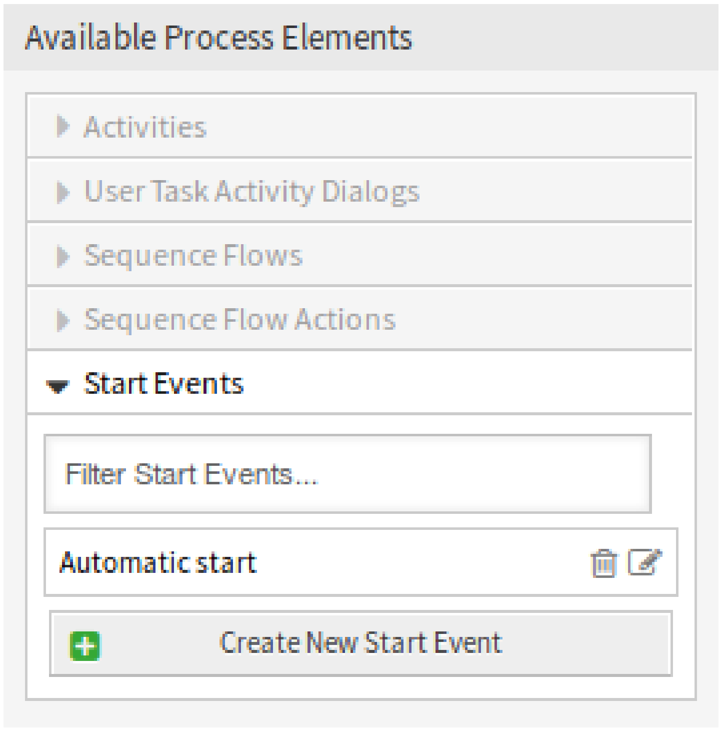
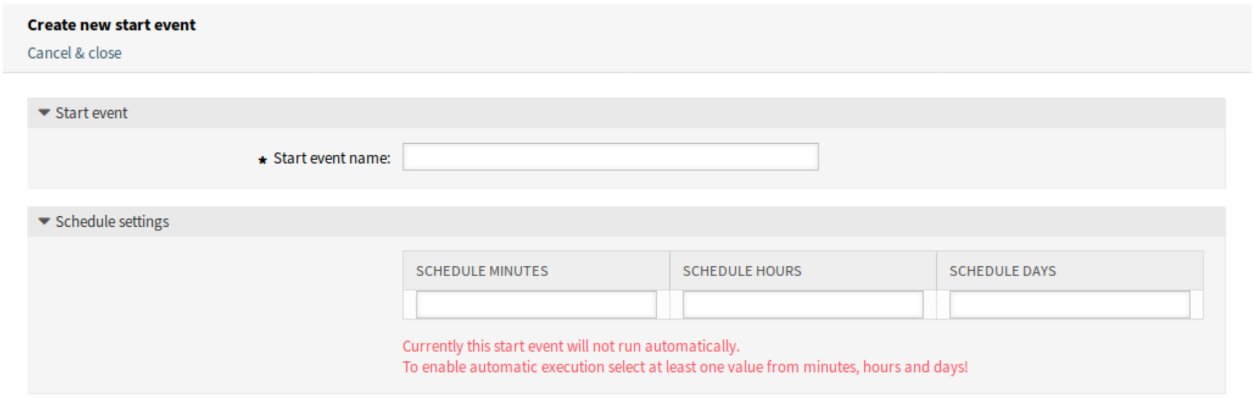


Fig. 93: Start Events

In the opened popup screen fill in the *Start Event Name* as well as set the schedule times.



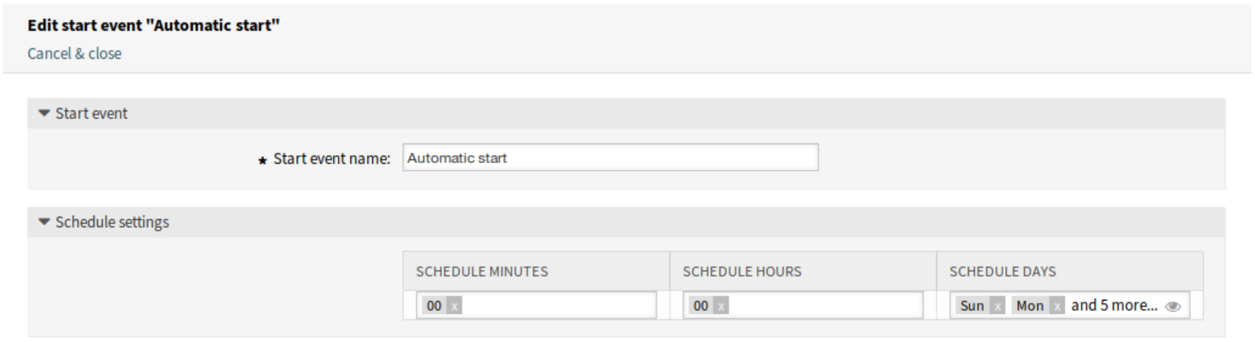
Currently this start event will not run automatically.
To enable automatic execution select at least one value from minutes, hours and days!

Fig. 94: Create New Start Event Screen

Start event name
A name of the scheduled based start point.

Schedule settings
Cron settings of the scheduled start point.

To edit an already defined start event, just click on its name in the *Available Process Elements* widget.


Edit start event "Automatic start"
Cancel & close

Start event

★ Start event name: Automatic start

Schedule settings

SCHEDULE MINUTES	SCHEDULE HOURS	SCHEDULE DAYS
00 x	00 x	Sun x Mon x and 5 more...

Fig. 95: Edit Start Event Screen

Warning: Times scheduled during daylight saving time start and daylight saving time end can have unexpected effects. At daylight saving time start it can be skipped and at daylight saving time end it can be executed twice. It is highly not recommended to schedule times when the daylight saving time starts or ends according to the server timezone.

To add a configured start event to the process canvas screen a free activity (not connected to other activities) is necessary. The scheduled based start event can not be connected directly to the normal start point of the process. In addition, the timer start event is not moveable. It is absolutely required to connect the scheduled based start event to a free activity. This first scheduled based activity can then be connected with a normal activity.

The initial activity of the timer start event can be any type of activity. In case of a user task activity, the process ticket is created and the process stays in this activity.

5.4.3 Process Modules

Process management modules can be used in sequence flow actions and script task activities. The modules are scripts which can interact with other objects like tickets, knowledge base articles, configuration items or appointments. Using these modules makes it possible to change the attributes of the process ticket or other objects.

Some built-in modules are shipped with the framework.

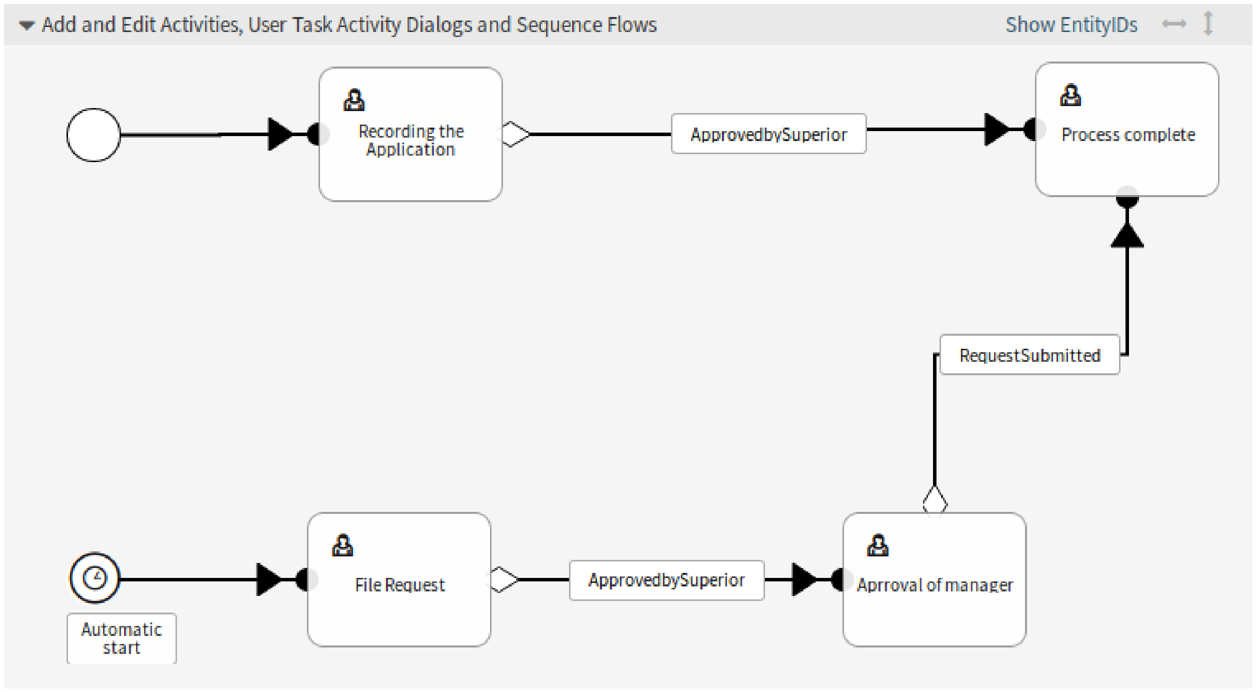
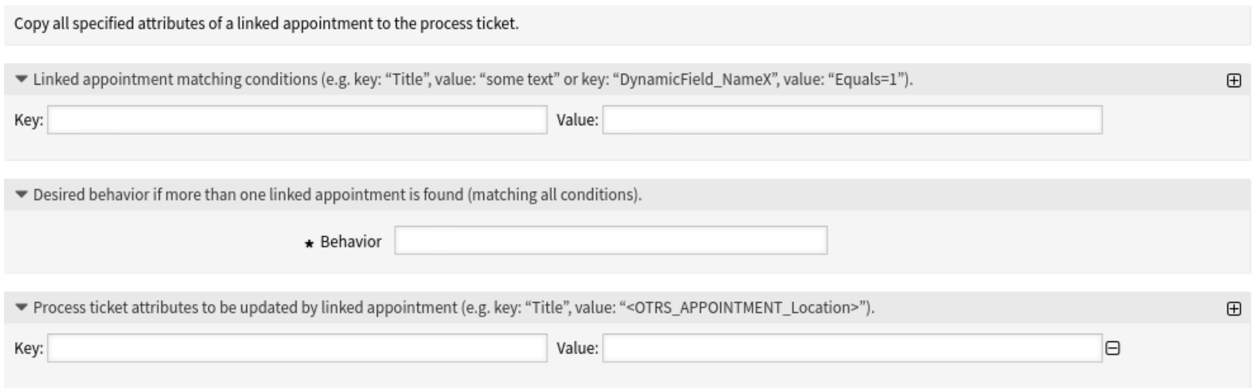


Fig. 96: Start Event on Canvas

AppointmentDataPull

A module to fetch data from a linked appointment.



Copy all specified attributes of a linked appointment to the process ticket.

Linked appointment matching conditions (e.g. key: “Title”, value: “some text” or key: “DynamicField_NameX”, value: “Equals=1”).

Key: Value:

Desired behavior if more than one linked appointment is found (matching all conditions).

* Behavior

Process ticket attributes to be updated by linked appointment (e.g. key: “Title”, value: “<OTRS_APPOINTMENT_Location>”).

Key: Value:

Fig. 97: Process Management Module AppointmentDataPull

Linked appointment condition section

This section is used to search for appointments.

Filters can be added with key-value pairs. There is logical AND relation between the filters if more than one filter is added. Logical OR relation can be added by multiple values separated by , .

Desired behavior section

If more than one appointments are found in the section above, the desired behavior can be defined here.

Possible values:

- Copy attributes from the appointment that was found first
- Copy attributes from the appointment that was found last
- Ignore appointment, do not copy anything

Process ticket attributes section

With this module the process ticket attributes can be updated. The key is the attribute of the process ticket. The value can be a pre-defined text, an attribute from the linked appointment in form of an OTRS tag or a concatenation of both. The `<OTRS_APPOINTMENT_*>` OTRS tag prefix can be used here.

The key `UserID` can override the executing agent user. For this it has to contain the agent user ID with the required permissions for the sequence flow action.

Examples:

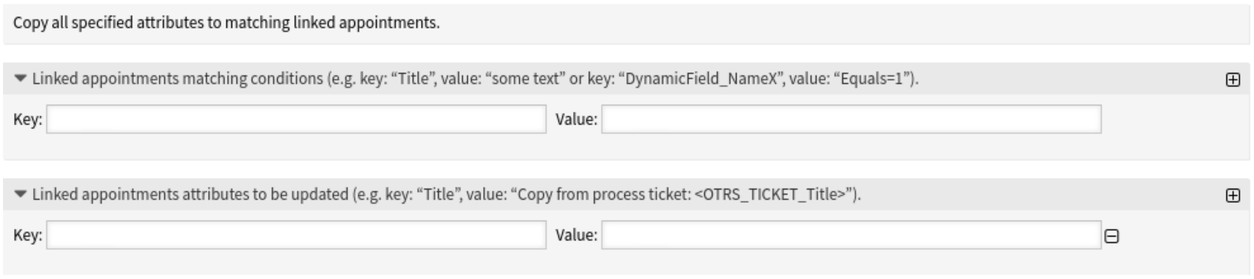
Key	Value
`Priority`	`5 very high`
`DynamicField_Location`	`<OTRS_APPOINTMENT_Location>`
`Title`	`From: <OTRS_APPOINTMENT_Title>`

See also:

See the AppointmentDataPull and the AppointmentSearch() API reference.

`AppointmentDataPush`

A module to insert data to a linked appointment.



Copy all specified attributes to matching linked appointments.

Linked appointments matching conditions (e.g. key: "Title", value: "some text" or key: "DynamicField_NameX", value: "Equals=1").

Key: Value:

Linked appointments attributes to be updated (e.g. key: "Title", value: "Copy from process ticket: <OTRS_TICKET_Title>").

Key: Value:

Fig. 98: Process Management Module AppointmentDataPush

Linked appointment condition section

This section is used to search in the linked appointments.

Filters can be added with key-value pairs. There is logical AND relation between the filters if more than one filter is added. Logical OR relation can be added by multiple values separated by `,`.

Linked appointment attributes section

Here can be set the linked appointment attributes to be updated. The key is the attribute of the linked appointment. The value can be a pre-defined text, an attribute from the process ticket in form of an OTRS tag or a concatenation of both. The `<OTRS_TICKET_*>` OTRS tag prefix can be used here.

The key `UserID` can override the executing agent user. For this it has to contain the agent user ID with the required permissions for the sequence flow action.

Examples:

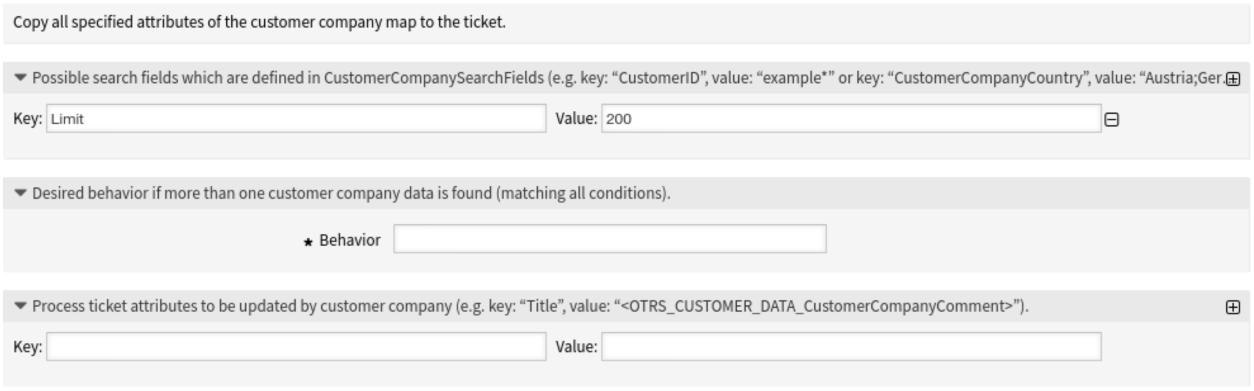
Key	Value
`state`	`closed successful`
`Priority`	`<OTRS_TICKET_Priority>`
`Title`	`Process: <OTRS_TICKET_DynamicField_ProcessManagementProcessID>`

See also:

See the AppointmentDataPush and the AppointmentSearch() API reference.

CustomerCompanyDataPull

A module to pull data from customer company map into current ticket.



Copy all specified attributes of the customer company map to the ticket.

Possible search fields which are defined in CustomerCompanySearchFields (e.g. key: "CustomerID", value: "example*" or key: "CustomerCompanyCountry", value: "Austria;Ger.

Key: Limit Value: 200

Desired behavior if more than one customer company data is found (matching all conditions).

★ Behavior

Process ticket attributes to be updated by customer company (e.g. key: "Title", value: "<OTRS_CUSTOMER_DATA_CustomerCompanyComment>").

Key: Value:

Fig. 99: Process Management Module CustomerCompanyDataPull

Possible search fields section

This section is used to search for customer companies.

Filters can be added with key-value pairs. There is logical AND relation between the filters if more than one filter is added. Logical OR relation can be added by multiple values separated by `;`.

The key `Limit` limits the number of customer companies returned.

The possible search fields are defined in the `CustomerCompanySearchFields` array of `Kernel/Config.pm`. If no customer company search fields are defined here then `Kernel/Config/Defaults.pm` will be used.

Desired behavior section

If more than one customer companies are found in the section above, the desired behavior can be defined here.

Possible values:

- Copy attributes from customer company that was found first
- Copy attributes from customer company that was found last
- Ignore customer company data, do not copy anything

Process ticket attributes section

With this module the process ticket attributes can be updated. The key is the attribute of the process ticket. The value can be a pre-defined text, an attribute of the found customer company in form of an OTRS tag or a concatenation of both. The `<OTRS_CUSTOMERCOMPANY_DATA_*>` OTRS tag prefix can be used here.

The key `UserID` can override the executing agent user. For this it has to contain the agent user ID with the required permissions for the sequence flow action.

Examples:

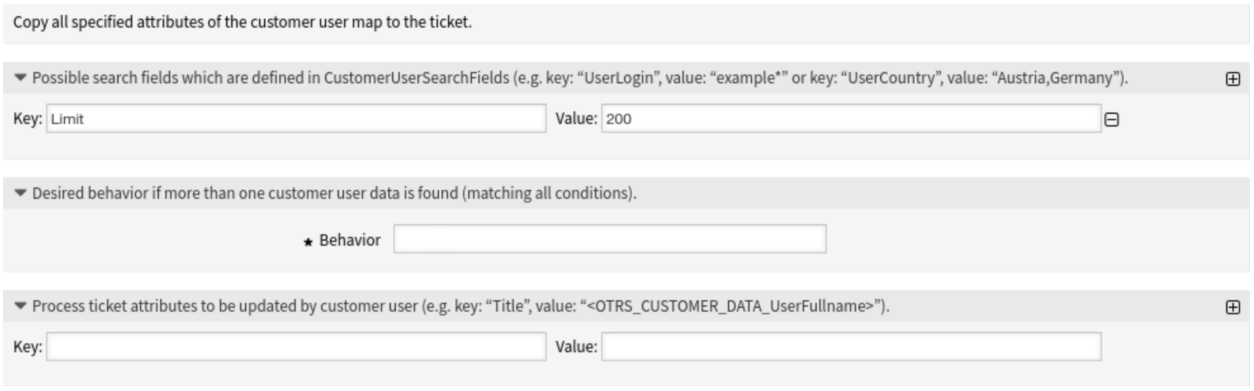
Key	Value
`Priority`	`5 very high`
`DynamicField_Name`	`<OTRS_CUSTOMERCOMPANY_DATA_CustomerCompanyName>`
`DynamicField_Search`	`https://startpage.com/?q=<OTRS_CUSTOMERCOMPANY_DATA_CustomerCompanyNa`

See also:

See the CustomerCompanyDataPull and the CustomerCompanySearchDetail() API reference.

`CustomerUserDataPull`

A module to pull data from customer user map into current ticket.


Copy all specified attributes of the customer user map to the ticket.

Possible search fields which are defined in CustomerUserSearchFields (e.g. key: "UserLogin", value: "example*" or key: "UserCountry", value: "Austria,Germany").

Key: Limit Value: 200

Desired behavior if more than one customer user data is found (matching all conditions).

★ Behavior

Process ticket attributes to be updated by customer user (e.g. key: "Title", value: "<OTRS_CUSTOMER_DATA_UserFullname>").

Key: Value:

Fig. 100: Process Management Module CustomerUserDataPull

Possible search fields section

This section is used to search for customer users.

Filters can be added with key-value pairs. There is logical AND relation between the filters if more than one filter is added. Logical OR relation can be added by multiple values separated by `,`.

The key `Limit` limits the number of customer users returned.

The possible search fields are defined in the `CustomerUserSearchFields` array of `Kernel/Config.pm`. If no customer user search fields are defined here then `Kernel/Config/Defaults.pm` will be used.

Desired behavior section

If more than one customer users are found in the section above, the desired behavior can be defined here.

Possible values:

- Copy attributes from customer user that was found first
- Copy attributes from customer user that was found last

- Ignore customer user data, do not copy anything

Process ticket attributes section

With this module the process ticket attributes can be updated. The key is the attribute of the process ticket. The value can be a pre-defined text, an attribute of the found customer user in form of an OTRS tag or a concatenation of both. The `<OTRS_CUSTOMER_DATA_*>` OTRS tag prefix can be used here.

The key `UserID` can override the executing agent user. For this it has to contain the agent user ID with the required permissions for the sequence flow action.

Examples:

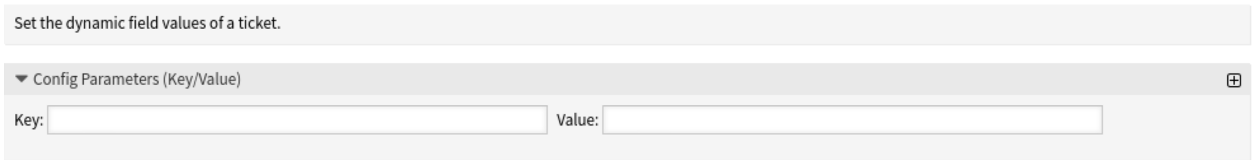
Key	Value
`Priority`	`5 very high`
`DynamicField_Name`	`<OTRS_CUSTOMER_DATA_UserFullname>`
`Dynamic-Field_Search`	`https://startpage.com/?q=<OTRS_CUSTOMER_DATA_UserFullname>`

See also:

See the CustomerUserDataPull and the CustomerSearchDetail() API reference.

`DynamicFieldSet`

A module to set the dynamic field values of a ticket.



Set the dynamic field values of a ticket.

Config Parameters (Key/Value)

Key:

Value:

Fig. 101: Process Management Module DynamicFieldSet

Configuration parameters section

The dynamic field values of the process ticket can be set here. The key is the attribute of the process ticket. The value can be a pre-defined text, an attribute from the process ticket in form of an OTRS tag or a concatenation of both. The `<OTRS_TICKET_*>` OTRS tag prefix can be used here.

Examples:

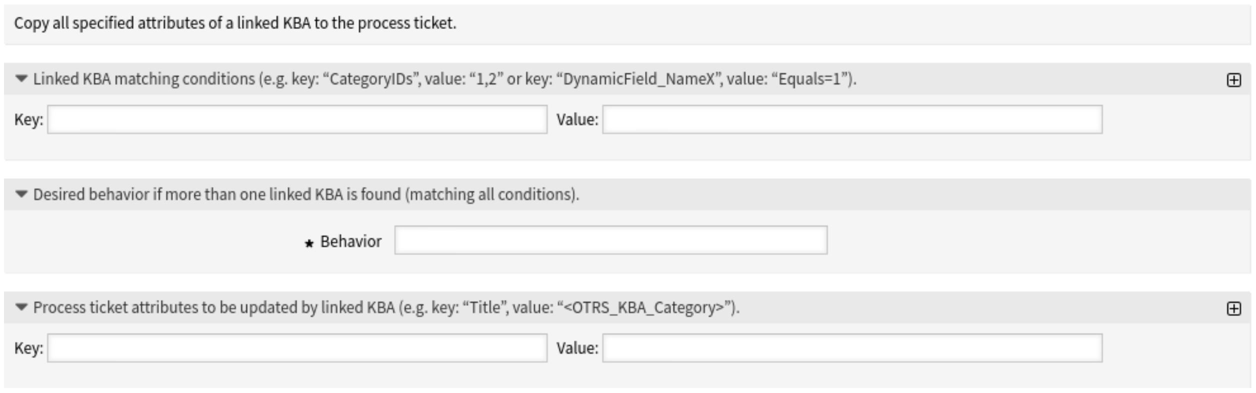
Key	Value
`Approved`	`1`
`User_ID`	`123`

See also:

See the DynamicFieldSet API reference.

`KBADataPull`

A module to fetch data from a linked knowledge base article.



Copy all specified attributes of a linked KBA to the process ticket.

Linked KBA matching conditions (e.g. key: "CategoryIDs", value: "1,2" or key: "DynamicField_NameX", value: "Equals=1").

Key: Value:

Desired behavior if more than one linked KBA is found (matching all conditions).

★ Behavior

Process ticket attributes to be updated by linked KBA (e.g. key: "Title", value: "<OTRS_KBA_Category>").

Key: Value:

Fig. 102: Process Management Module KBADataPull

Linked KBA condition section

This section is used to search for knowledge base articles.

Filters can be added with key-value pairs. There is logical AND relation between the filters if more than one filter is added. Logical OR relation can be added by multiple values separated by `,`.

Desired behavior section

If more than one knowledge base articles are found in the section above, the desired behavior can be defined here.

Possible values:

- Copy attributes from the KBA that was found first
- Copy attributes from the KBA that was found last
- Ignore KBA, do not copy anything

Process ticket attributes section

With this module the process ticket attributes can be updated. The key is the attribute of the process ticket. The value can be a pre-defined text, an attribute from the linked knowledge base article in form of an OTRS tag or a concatenation of both. The `<OTRS_KBA_*>` OTRS tag prefix can be used here.

The key `UserID` can override the executing agent user. For this it has to contain the agent user ID with the required permissions for the sequence flow action.

Examples:

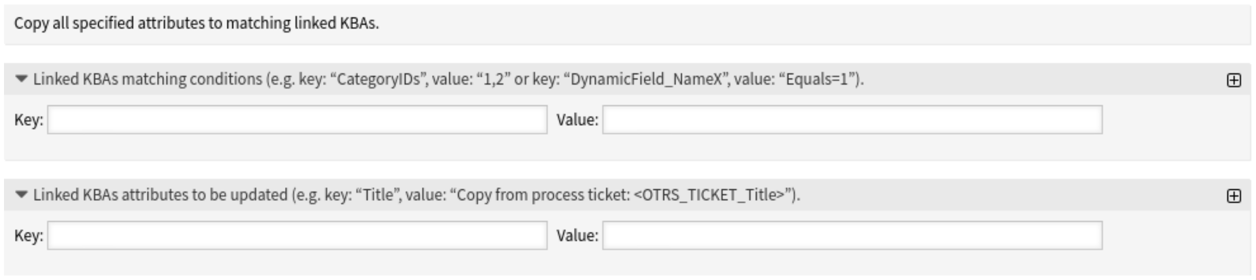
Key	Value
`Priority`	`5 very high`
`DynamicField_Solution`	`<OTRS_KBA_Field3>`
`Title`	`Category: <OTRS_KBA_Field3>`

See also:

See the KBADataPull and the FAQSearch() API reference.

`KBADataPush`

A module to insert data to the linked knowledge base articles.


Copy all specified attributes to matching linked KBAs.

Linked KBAs matching conditions (e.g. key: "CategoryIDs", value: "1,2" or key: "DynamicField_NameX", value: "Equals=1").

Key:		Value:	

Linked KBAs attributes to be updated (e.g. key: "Title", value: "Copy from process ticket: <OTRS_TICKET_Title>").

Key:		Value:	

Fig. 103: Process Management Module KBADataPush

Linked KBA condition section
This section is used to search in the linked knowledge base articles.

Filters can be added with key-value pairs. There is logical AND relation between the filters if more than one filter is added. Logical OR relation can be added by multiple values separated by `,`.

Linked KBA attributes section
Here can be set the linked knowledge base article attributes to be updated. The key is an attribute of the linked knowledge base article. The value can be a pre-defined text, an attribute from the process ticket in form of an OTRS tag or a concatenation of both. The `<OTRS_TICKET_*>` OTRS tag prefix can be used here.

The key `UserID` can override the executing agent user. For this it has to contain the agent user ID with the required permissions for the sequence flow action.

Examples:

Key	Value
`DynamicField_Name`	`<OTRS_TICKET_DynamicField_Name>`
`DynamicField_Text`	`Title: <OTRS_TICKET_Title>`
`UserID`	`17`

See also:

See the KBADataPush and the FAQSearch() API reference.

`LinkWithAppointment`

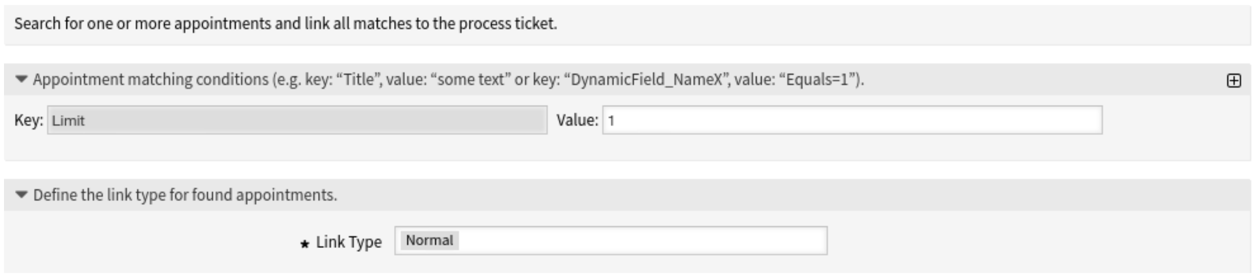
A module to link appointments.

Appointment matching section
This section is used to search for appointments.

Filters can be added with key-value pairs. There is logical AND relation between the filters if more than one filter is added. Logical OR relation can be added by multiple values separated by `,`.

The key `Limit` limits the number of appointments returned. This field is mandatory.

The key `UserID` can override the executing agent user. For this it has to contain the agent user ID with the required permissions for the sequence flow action.


Search for one or more appointments and link all matches to the process ticket.

Appointment matching conditions (e.g. key: “Title”, value: “some text” or key: “DynamicField_NameX”, value: “Equals=1”).

Key: Limit Value: 1

Define the link type for found appointments.

★ Link Type Normal

Fig. 104: Process Management Module LinkWithAppointment

Link type section

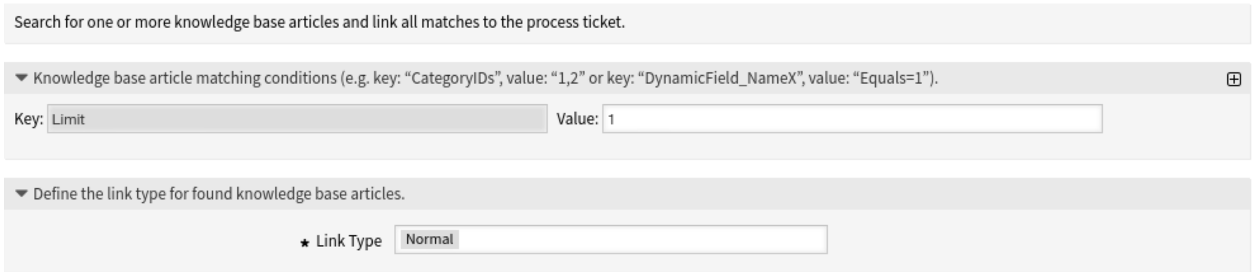
Here can be defined the link type for found appointments. Possible values are *Normal*, *Parent* or *Child*.

See also:

See the LinkWithAppointment and the AppointmentSearch() API reference.

`LinkWithKBA`

A module to link knowledge base articles.


Search for one or more knowledge base articles and link all matches to the process ticket.

Knowledge base article matching conditions (e.g. key: “CategoryIDs”, value: “1,2” or key: “DynamicField_NameX”, value: “Equals=1”).

Key: Limit Value: 1

Define the link type for found knowledge base articles.

★ Link Type Normal

Fig. 105: Process Management Module LinkWithKBA

Knowledge base article matching section

This section is used to search for knowledge base articles.

Filters can be added with key-value pairs. There is logical AND relation between the filters if more than one filter is added. Logical OR relation can be added by multiple values separated by `,`.

The key `Limit` limits the number of knowledge base articles returned. This field is mandatory.

The key `UserID` can override the executing agent user. For this it has to contain the agent user ID with the required permissions for the sequence flow action.

Link type section

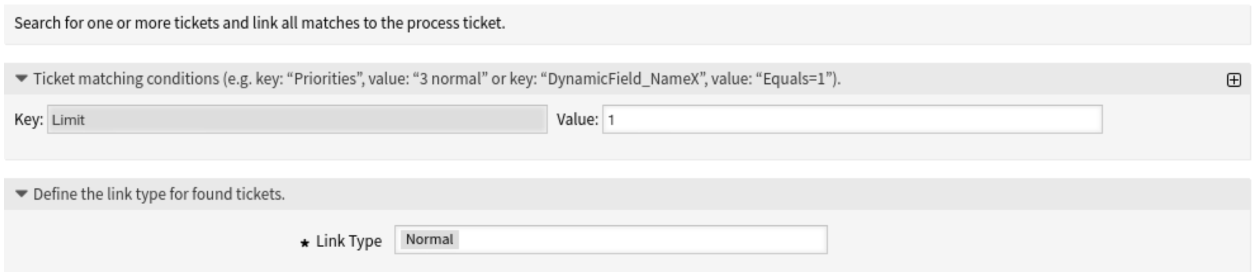
Here can be defined the link type for found knowledge base articles. Possible values are *Normal*, *Parent* or *Child*.

See also:

See the LinkWithKBA and the FAQSearch() API reference.

`LinkWithTicket`

A module to link other tickets.


Search for one or more tickets and link all matches to the process ticket.

Ticket matching conditions (e.g. key: "Priorities", value: "3 normal" or key: "DynamicField_NameX", value: "Equals=1").

Key: Limit Value: 1

Define the link type for found tickets.

★ Link Type: Normal

Fig. 106: Process Management Module LinkWithTicket

Ticket matching section

This section is used to search for tickets.

Filters can be added with key-value pairs. There is logical AND relation between the filters if more than one filter is added. Logical OR relation can be added by multiple values separated by `,`.

The key `Limit` limits the number of tickets returned. This field is mandatory.

The key `UserID` can override the executing agent user. For this it has to contain the agent user ID with the required permissions for the sequence flow action.

Link type section

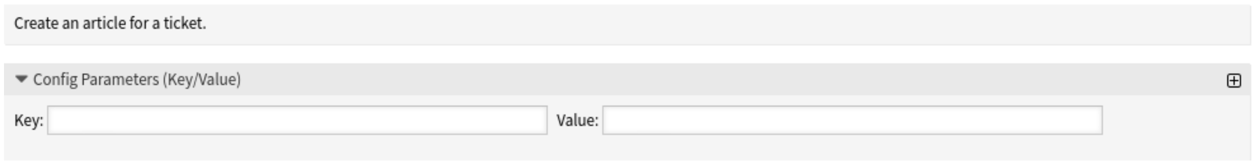
Here can be defined the link type for found tickets. Possible values are *Normal*, *Parent* or *Child*.

See also:

See the LinkWithTicket and the TicketSearch() API reference.

`TicketArticleCreate`

A module to create an article for a ticket.


Create an article for a ticket.

Config Parameters (Key/Value)

Key: Value:

Fig. 107: Process Management Module TicketArticleCreate

Configuration parameters section

Here can be set the article attributes of a ticket. The key is the attribute of the article which will be created. The value can be a pre-defined text, an attribute from the process ticket in form of an OTRS tag or a concatenation of both. The `<OTRS_TICKET_*>` OTRS tag prefix can be used here.

The key `UserID` can override the executing agent user. For this it has to contain the agent user ID with the required permissions for the sequence flow action.

Examples:

Key	Value

Body	`<table cellspacing="2" cellpadding="2" border="1">`
	`  <thead>`
	`    <tr>`
	`      <th>Field</th>`
	`      <th>Content</th>`
	`    </tr>`
	`  </thead>`
	`  <tbody>`
	`    <tr>`
	`      <td>Book Title</td>`
	`      <td><OTRS_TICKET_DynamicField_Title></td>`
	`    </tr>`
	`    <tr>`
	`      <td>Book Author</td>`
	`      <td><OTRS_TICKET_DynamicField_Author></td>`
	`    </tr>`
	`    <tr>`
	`      <td>ISBN</td>`
	`      <td><OTRS_TICKET_DynamicField_ISBN></td>`
	`    </tr>`
	`  </tbody>`
	`</table>`

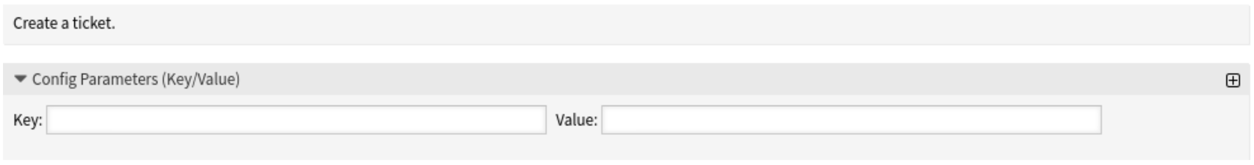
CommunicationChannel	Internal
ContentType	text/html;charset=UTF-8
HistoryComment	Last man standing
HistoryType	AddNote
IsVisibleForCustomer	1
SenderType	agent
Subject	Summarization of Former Process Ticket Content: <OTRS_TICKET_Title>

See also:

See the TicketArticleCreate and the ArticleCreate() API reference.

`TicketCreate`

A module to create a ticket.


Create a ticket.

Config Parameters (Key/Value)

Key:

Value:

Fig. 108: Process Management Module TicketCreate

Configuration parameters section

Here can be set the ticket attributes. The key is the attribute of the ticket which will be created. The value can be a pre-defined text, an attribute from the process ticket in form of an OTRS tag or a concatenation of both. The `<OTRS_TICKET_*>` OTRS tag prefix can be used here.

The key `UserID` can override the executing agent user. For this it has to contain the agent user ID with the required permissions for the sequence flow action.

Examples:

Key	Value
`CustomerID`	`<OTRS_TICKET_CustomerID>`
`CustomerUser`	`<OTRS_TICKET_CustomerUserID>`
`DynamicField_ProcessManagementActivityID`	`Activity-1a1924ad9c1a6b23f70fc58a80961760`
`DynamicField_ProcessManagementProcessID`	`Process-95a06ad414cf371ebc4b82c8c2a3c389`
`LinkAs`	`Child`
`Lock`	`unlock`
`OwnerID`	`1`
`Priority`	`3 normal`
`Queue`	`Postmaster`
`State`	`open`
`Title`	`Subtask of: <OTRS_TICKET_Title>`

See also:

See the TicketCreate and the TicketCreate() API reference.

`TicketCustomerSet`

A module to set the customer of a ticket.

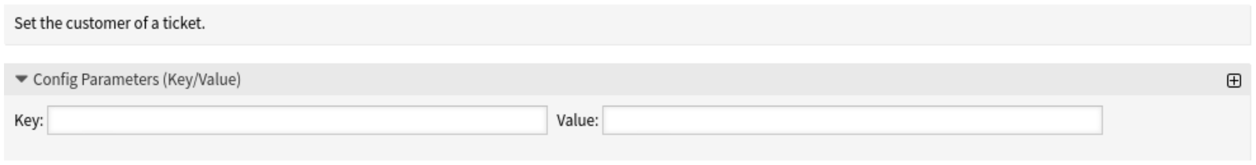

Set the customer of a ticket.

Fig. 109: Process Management Module TicketCustomerSet

Configuration parameters section

Here can be set the customer of the ticket. The key is the attribute of the process ticket. The value can be a pre-defined text, an attribute from the process ticket in form of an OTRS tag or a concatenation of both. The `<OTRS_TICKET_*>` OTRS tag prefix can be used here.

The key `UserID` can override the executing agent user. For this it has to contain the agent user ID with the required permissions for the sequence flow action.

Examples:

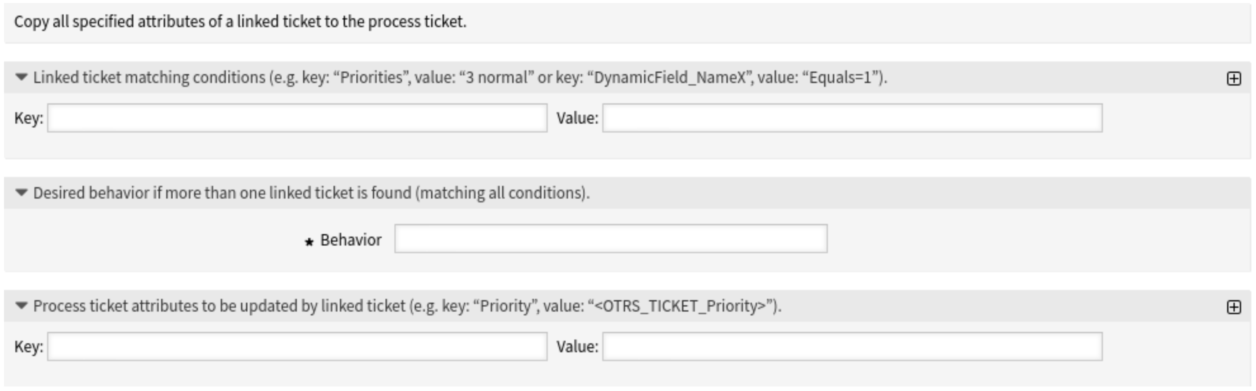
Key	Value
`CustomerID`	`client123`
`CustomerUserID`	`client-user-123`

See also:

See the TicketCustomerSet and the TicketCustomerSet() API reference.

`TicketDataPull`

A module to fetch data from a linked ticket.


Copy all specified attributes of a linked ticket to the process ticket.

Linked ticket matching conditions (e.g. key: “Priorities”, value: “3 normal” or key: “DynamicField_NameX”, value: “Equals=1”).

Key:
Value:

Desired behavior if more than one linked ticket is found (matching all conditions).

★ Behavior

Process ticket attributes to be updated by linked ticket (e.g. key: “Priority”, value: “<OTRS_TICKET_Priority>”).

Key:
Value:

Fig. 110: Process Management Module TicketDataPull

Linked ticket condition section

This section is used to search for tickets.

Filters can be added with key-value pairs. There is logical AND relation between the filters if more than one filter is added. Logical OR relation can be added by multiple values separated by `,`.

Desired behavior section

If more than one tickets are found in the section above, the desired behavior can be defined here.

Possible values:

- Copy attributes from the ticket that was found first
- Copy attributes from the ticket that was found last
- Ignore ticket, do not copy anything

Process ticket attributes section

With this module the process ticket attributes can be updated. The key is the attribute of the process ticket. The value can be a pre-defined text, an attribute from the linked ticket in form of an OTRS tag or a concatenation of both. The `<OTRS_TICKET_*>` OTRS tag prefix can be used here.

The key `UserID` can override the executing agent user. For this it has to contain the agent user ID with the required permissions for the sequence flow action.

Examples:

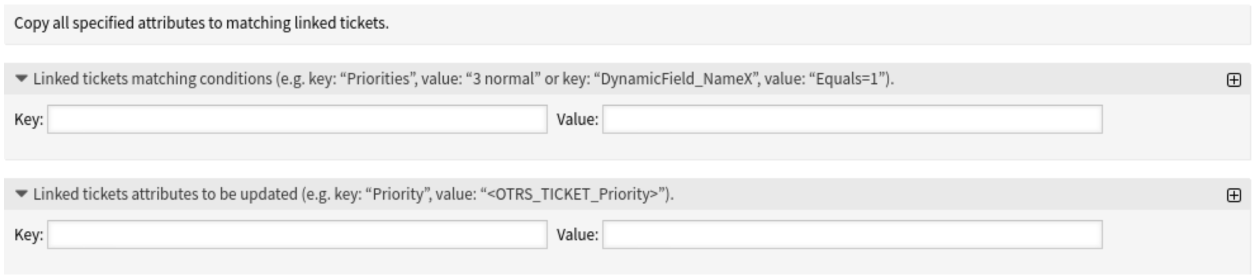
Key	Value
`Priority`	`5 very high`
`DynamicField_Priority`	`<OTRS_TICKET_Prioroty>`
`Title`	`From: <OTRS_TICKET_DynamicField_Title>`

See also:

See the TicketDataPull and the TicketSearch() API reference.

TicketDataPush

A module to insert data to the linked tickets.


Copy all specified attributes to matching linked tickets.

Linked tickets matching conditions (e.g. key: "Priorities", value: "3 normal" or key: "DynamicField_NameX", value: "Equals=1").

Key: Value:

Linked tickets attributes to be updated (e.g. key: "Priority", value: "<OTRS_TICKET_Priority>").

Key: Value:

Fig. 111: Process Management Module TicketDataPush

Linked ticket condition section
This section is used to search in the linked tickets.

Filters can be added with key-value pairs. There is logical AND relation between the filters if more than one filter is added. Logical OR relation can be added by multiple values separated by `,`.

Linked ticket attributes section
Here can be set the linked ticket attributes to be updated. The key is the attribute of the linked ticket. The value can be a pre-defined text, an attribute from the process ticket in form of an OTRS tag or a concatenation of both. The `<OTRS_TICKET_*>` OTRS tag prefix can be used here.

The key `UserID` can override the executing agent user. For this it has to contain the agent user ID with the required permissions for the sequence flow action.

Examples:

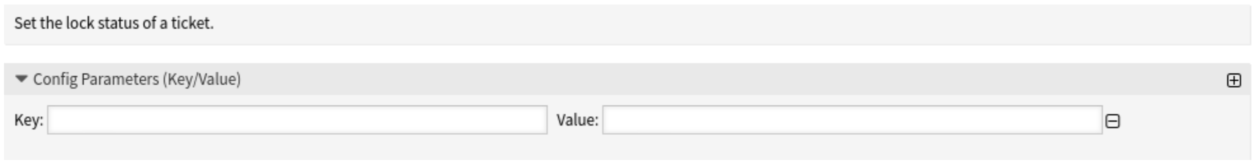
Key	Value
`state`	`closed successful`
`Priority`	`<OTRS_TICKET_Priority>`
`Title`	`Process: <OTRS_TICKET_DynamicField_ProcessManagementProcessID>`

See also:

See the TicketDataPush and the TicketSearch() API reference.

TicketLockSet

A module to set the lock status of a ticket.


Set the lock status of a ticket.

Config Parameters (Key/Value)

Key: Value:

Fig. 112: Process Management Module TicketLockSet

Configuration parameters section
Here can be set the lock status of a ticket.

The key `UserID` can override the executing agent user. For this it has to contain the agent user ID with the required permissions for the sequence flow action.

Examples:

Key	Value
`Lock`	`locked`
`LockID`	`1`

See also:

See the TicketLockSet and the TicketLockSet() API reference.

`TicketOwnerSet`

A module to set the owner of a ticket.

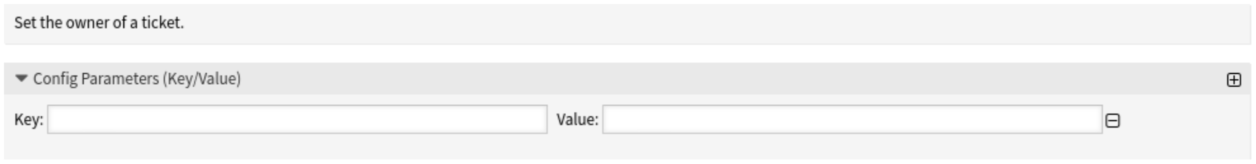


Fig. 113: Process Management Module TicketOwnerSet

Configuration parameters section

Here can be set the owner of a ticket.

The key `UserID` can override the executing agent user. For this it has to contain the agent user ID with the required permissions for the sequence flow action.

Examples:

Key	Value
`Owner`	`root@localhost`
`OwnerID`	`1`

See also:

See the TicketOwnerSet and the TicketOwnerSet() API reference.

`TicketQueueSet`

A module to move a ticket to a new queue.

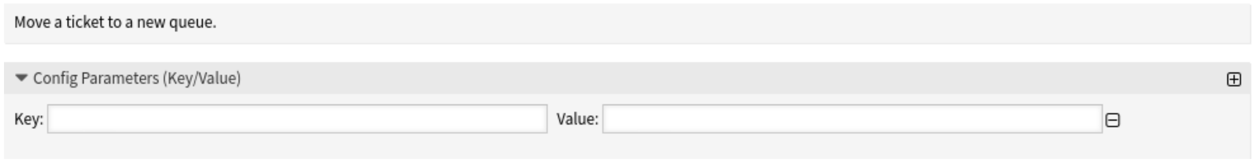


Fig. 114: Process Management Module TicketQueueSet

Configuration parameters section

Here can be set the queue of a ticket.

The key `UserID` can override the executing agent user. For this it has to contain the agent user ID with the required permissions for the sequence flow action.

Examples:

Key	Value
`Queue`	`Misc`
`QueueID`	`1`

See also:

See the TicketQueueSet and the TicketQueueSet() API reference.

`TicketResponsibleSet`

A module to set the responsible agent of a ticket.

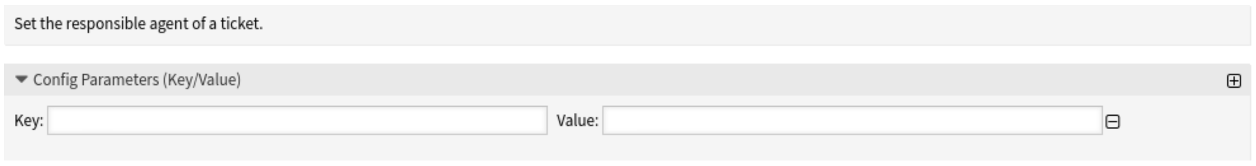

Set the responsible agent of a ticket.

Fig. 115: Process Management Module TicketResponsibleSet

Configuration parameters section

Here can be set the responsible agent of a ticket.

The key `UserID` can override the executing agent user. For this it has to contain the agent user ID with the required permissions for the sequence flow action.

Examples:

Key	Value
`Responsible`	`root@localhost`
`ResponsibleID`	`1`

See also:

See the TicketResponsibleSet and the TicketResponsibleSet() API reference.

`TicketSendEmail`

A module to send an email from a ticket.

Configuration parameters for recipients section

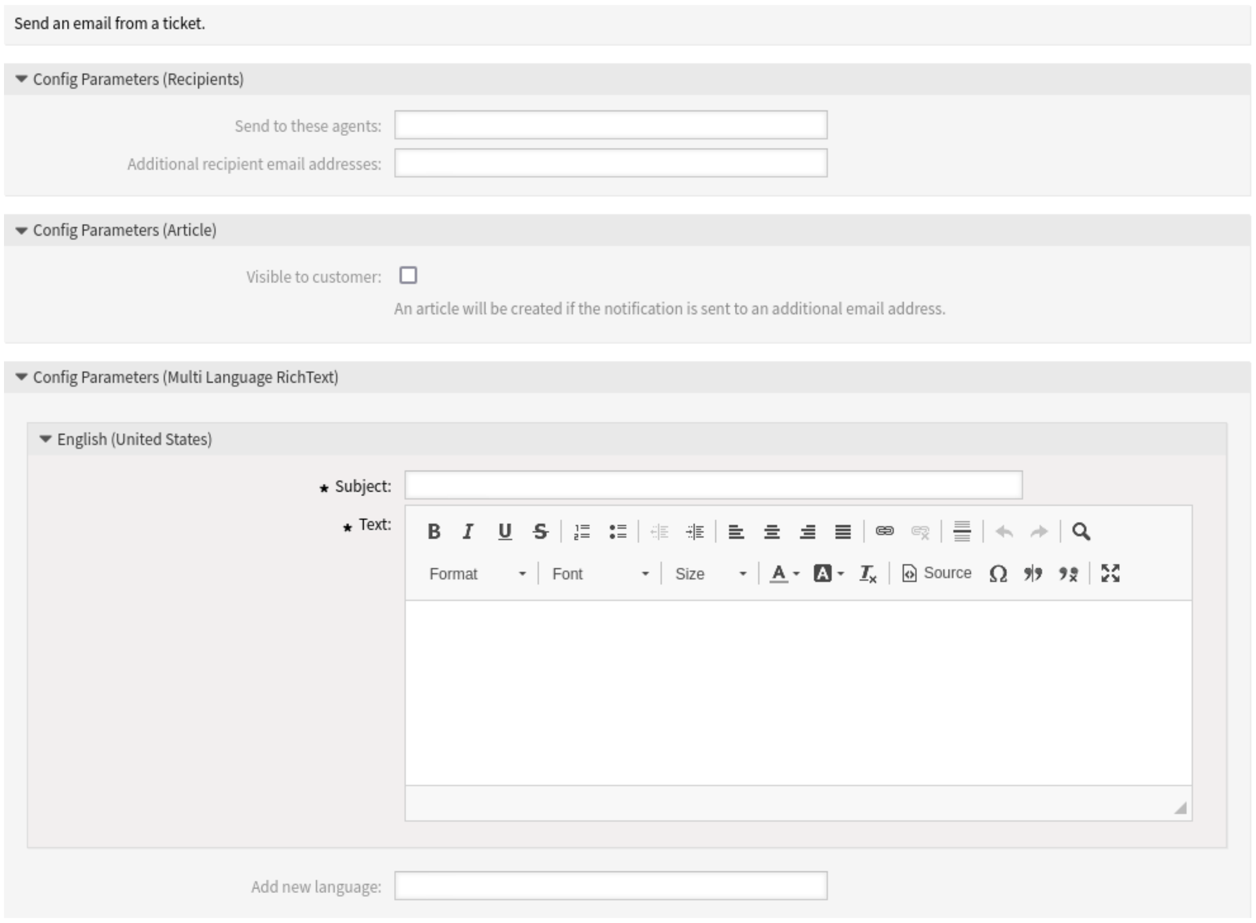
Here can be set the recipients of the email.

Send to these agents

Select the agents who will receive the email.

Additional recipient email addresses

Additional email addresses can be added here.


Send an email from a ticket.

Config Parameters (Recipients)

Send to these agents:	
Additional recipient email addresses:	

Config Parameters (Article)

Visible to customer: ☐

An article will be created if the notification is sent to an additional email address.

Config Parameters (Multi Language RichText)

English (United States)

★ Subject:	
★ Text:	

Add new language:

Fig. 116: Process Management Module TicketSendEmail

Configuration parameters for article section
Here can be set if the article is visible to customer.

Configuration parameters for email section
Here can be set the subject and the body of the email. Multiple languages are supported.

The key `UserID` can override the executing agent user. For this it has to contain the agent user ID with the required permissions for the sequence flow action.

See also:

See the TicketSendEmail API reference.

`TicketServiceSet`

A module to set the service of a ticket.

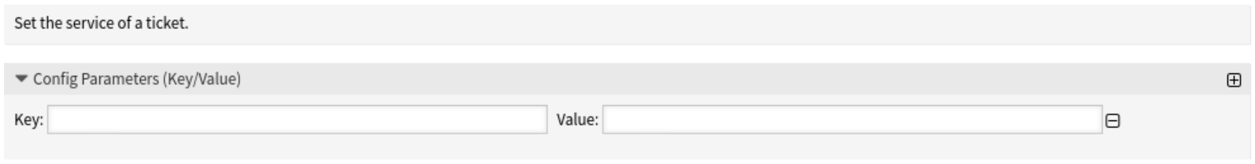


Fig. 117: Process Management Module TicketServiceSet

Configuration parameters section
Here can be set the service of a ticket.

The key `UserID` can override the executing agent user. For this it has to contain the agent user ID with the required permissions for the sequence flow action.

Examples:

Key	Value
`Service`	`MyService::Subservice`
`ServiceID`	`123`

See also:

See the TicketServiceSet and the TicketServiceSet() API reference.

`TicketSLASet`

A module to set the SLA of a ticket.

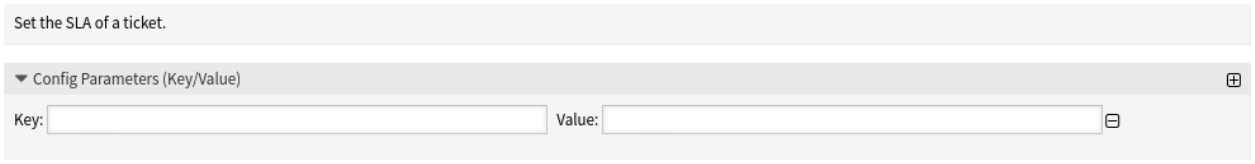


Fig. 118: Process Management Module TicketSLASet

Configuration parameters section

Here can be set the service level agreement of a ticket.

The key `UserID` can override the executing agent user. For this it has to contain the agent user ID with the required permissions for the sequence flow action.

Examples:

Key	Value
`SLA`	`MySLA`
`SLAID`	`123`

See also:

See the TicketSLASet and the TicketSLASet() API reference.

`TicketStateSet`

A module to set the state of a ticket.

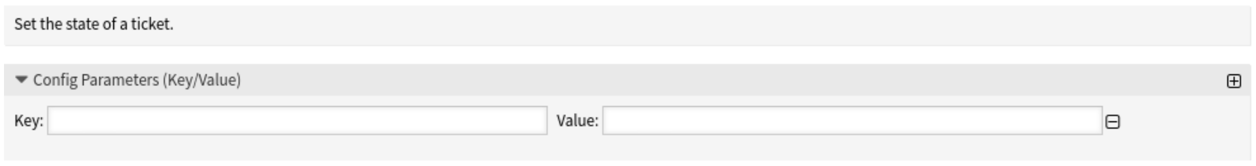

Set the state of a ticket.

Fig. 119: Process Management Module TicketStateSet

Configuration parameters section

Here can be set the state of a ticket.

The key `UserID` can override the executing agent user. For this it has to contain the agent user ID with the required permissions for the sequence flow action.

Examples:

Key	Value
`State`	`open`
`StateID`	`1`

See also:

See the TicketStateSet and the TicketStateSet() API reference.

`TicketTitleSet`

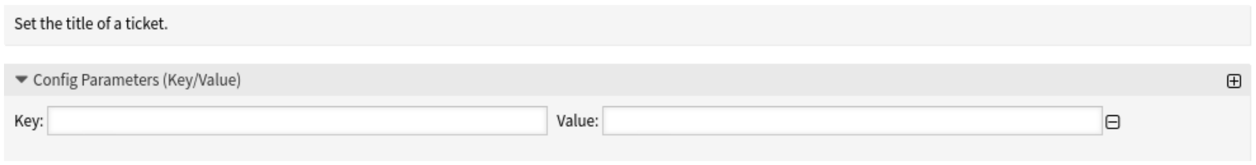
A module to set the title of a ticket.

Configuration parameters section

Here can be set the title of a ticket.

The key `UserID` can override the executing agent user. For this it has to contain the agent user ID with the required permissions for the sequence flow action.

Examples:


Set the title of a ticket.

Config Parameters (Key/Value)

Key: Value:

Fig. 120: Process Management Module TicketTitleSet

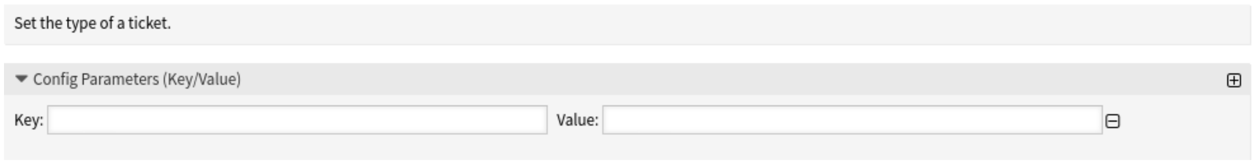
Key	Value
`Title`	`Some ticket title`

See also:

See the TicketTitleSet and the TicketTitleUpdate() API reference.

`TicketTypeSet`

A module to set the type of a ticket.


Set the type of a ticket.

Config Parameters (Key/Value)

Key: Value:

Fig. 121: Process Management Module TicketTypeSet

Configuration parameters section

Here can be set the type of a ticket.

The key `UserID` can override the executing agent user. For this it has to contain the agent user ID with the required permissions for the sequence flow action.

Examples:

Key	Value
`Type`	`Default`
`TypeID`	`1`

See also:

See the TicketTypeSet and the TicketTypeSet() API reference.

5.4.4 Example process

Processes are more complex than other resources in OTRS. To create a process, you need to do several steps. The following chapters shows you, how to define a process from the specification and create the needed resources. Let' s see an example to make it more demonstrative. We will define a book order process.

Process Specification

The book order process has four states.

Recording the demand
Before an order will be placed, the demand for literature by an employee will be recorded. The following book is needed in our example:

```
Title: Prozessmanagement für Dummies
Autor: Thilo Knuppertz
ISBN: 3527703713
```

Approval by manager
The head of the employee' s department needs to decide on the order. In case of a denial, a reason should be recorded by the manager. In case of approval, the order is passed to the purchasing department.

Processing by purchasing department
Purchasing now has the task to find out where the book can be ordered with the best conditions. If it is out of stock, this can be recorded in the order. In case of a successful order purchasing will record the supplier, the price and the delivery date.

Processing by the mail room
The shipment will arrive at the company. The incoming goods department checks the shipment and records the date of receipt. Now the employee will be informed that their order has arrived and is ready to be collected.

Introduce The Process Elements

If we assume that a ticket acts in this workflow like an accompanying document that can receive change notes, we already have a clear picture of process tickets.

From the analysis of the example process we can identify the following necessary items:

- Possibility to record data, let' s call this *user task activity dialog*.
- Check which can react to changed data automatically, let' s call this *sequence flow*.
- Change which can be applied to a process ticket after successful transitions of a process ticket, let' s call this *sequence flow action*.
- A possibility to offer more than just one user task activity dialog to be available. In our example this is needed when the manager must have the choice between *Approve* and *Deny*. Let' s call this *activity*.

Now, with activities, user task activity dialogs, sequence flows and sequence flow actions we have the necessary tools to model the individual steps of our example. What is still missing is an area where for each workflow the order of the steps can be specified. Let' s call this *process*.

Create Necessary Resources

Before the creation of the process and its parts is necessary to prepare the system. We need to define some *Queues*, *Agents* and *Dynamic Fields* as well as set some *System Configuration* settings.

Create the following *Queues*:

- Management
- Employees
- Purchasing
- Post office

Create the following *Agents*:

- Manager
- Employee

Create the following *Dynamic Fields*:

Object	Type	Name	Label	Possible values
Ticket	Text	`Title`	Title	
Ticket	Text	`Author`	Author	
Ticket	Text	`ISBN`	ISBN	
Ticket	Dropdown	`Status`	Status	• Approval • Approval denied • Approved • Order denied • Order placed • Shipment received
Ticket	Text	`Supplier`	Supplier	
Ticket	Text	`Price`	Price	
Ticket	Date	`DeliveryDate`	Delivery date	
Ticket	Date	`DateOfReceipt`	Date of receipt	

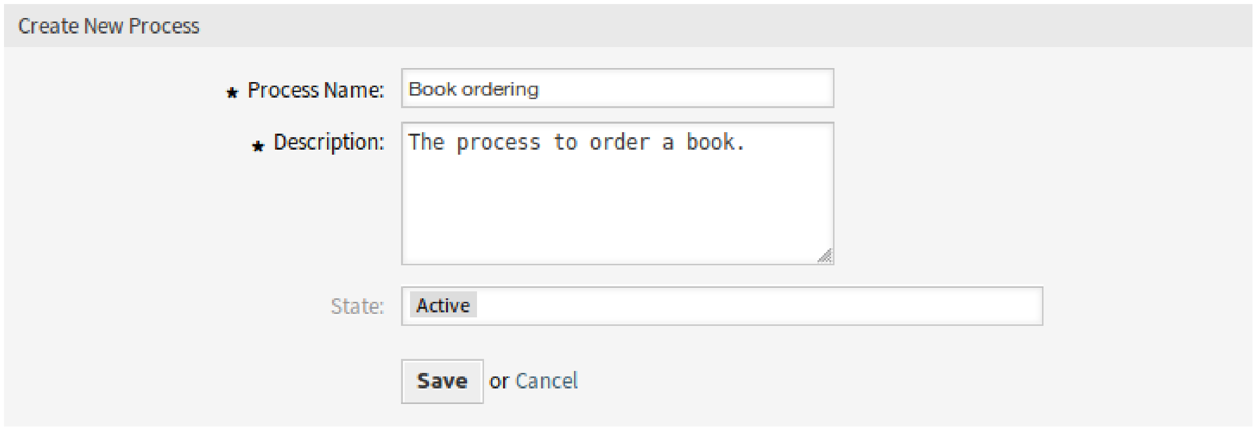
Set the following *System Configuration* settings:

- Ticket::Responsible
  - Enabled
- AgentFrontend::TicketDetailView::Widget::BusinessProcessInformation###DynamicField
  - `Author` → *1 - Enabled*
  - `DateOfReceipt` → *1 - Enabled*
  - `DeliveryDate` → *1 - Enabled*
  - `ISBN` → *1 - Enabled*
  - `Price` → *1 - Enabled*

   - `Status` → *1 - Enabled*
   - `Supplier` → *1 - Enabled*
   - `Title` → *1 - Enabled*
- AgentFrontend::TicketDetailView::Widget::BusinessProcessInformation###DynamicFieldGroups
   - `Book` → `Title,Author,ISBN`
   - `General` → `Status`
   - `Order` → `Price,Supplier,DeliveryDate`
   - `Shipment` → `DateOfReceipt`

Note: Don't forget to deploy the modified system configuration settings.

Now, go back to the *Process Management* screen and click on the *Create New Process*. Fill in the required fields.


Create New Process

Process Name: Book ordering

Description: The process to order a book.

State: Active

Fig. 122: Book Ordering - Create New Process

The new process is created. You can add some process element now.

Create User Task Activity Dialogs

Click on the *User Task Activity Dialogs* item in the *Available Process Elements* widget in the left sidebar. This action will expand the *User Task Activity Dialogs* options and will collapse all others doing an accordion like effect. Click on the *Create New User Task Activity Dialog* button.

In the opened popup screen fill in the *Dialog Name* as well as the *Description (short)* fields. For this example we will leave all other fields as the default.

To assign fields to the user task activity dialog simple drag the required field from the *Available Fields* pool and drop into the *Assigned Fields* pool. The order in the *Assigned Fields* pool is the order as the fields will have in the screen. To modify the order simply drag and drop the field within the pool to rearrange it in the correct place.

In this example we will use:

- `Article` field for comments.

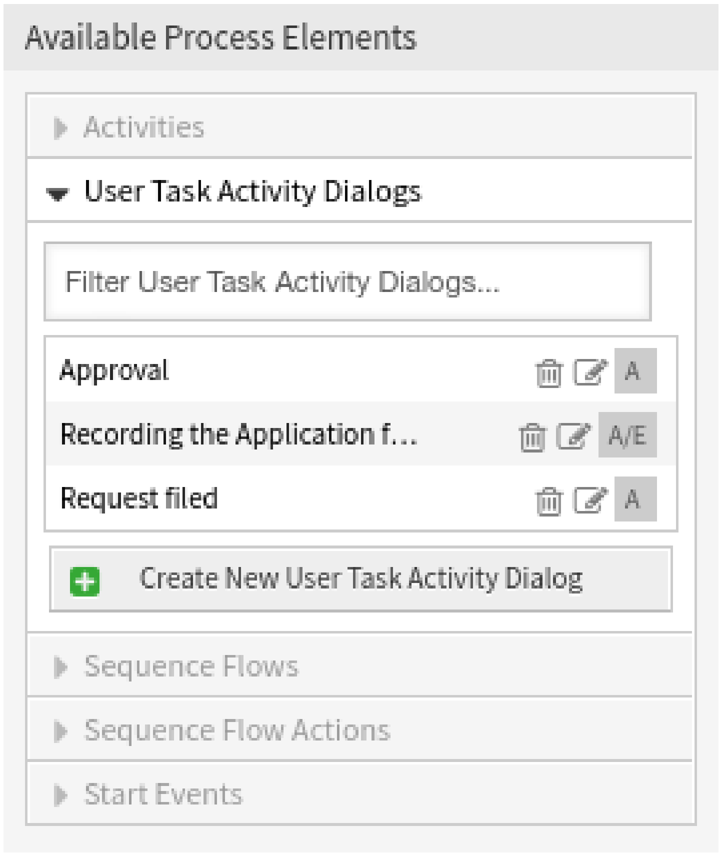
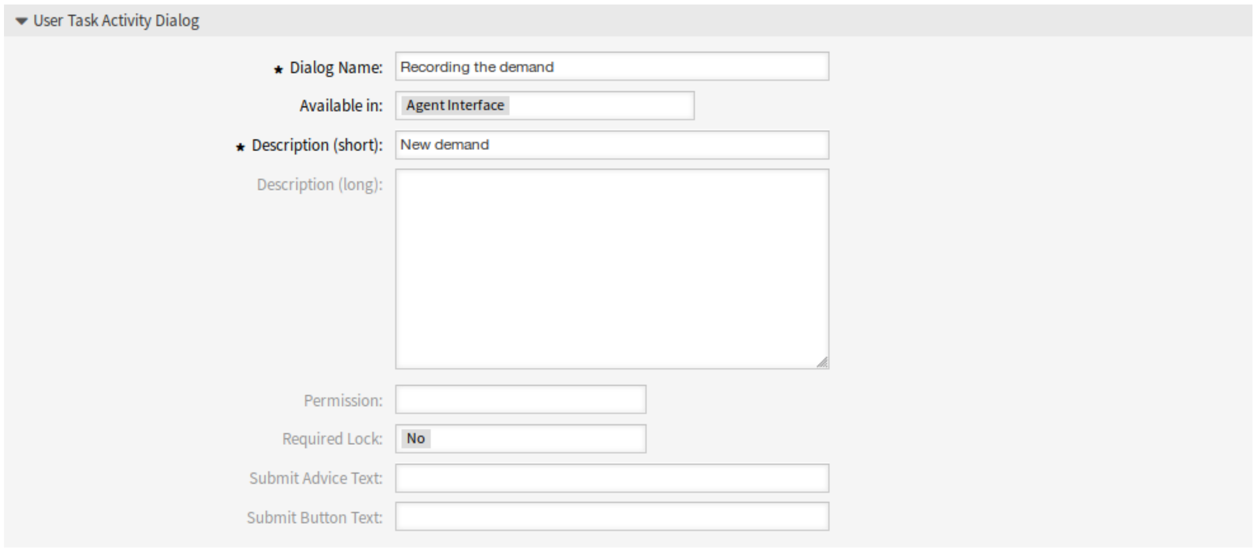

Available Process Elements
Activities
User Task Activity Dialogs
Filter User Task Activity Dialogs...
Approval
A
Recording the Application f...
A/E
Request filed
A
Create New User Task Activity Dialog
Sequence Flows
Sequence Flow Actions
Start Events

Fig. 123: Book Ordering - User Task Activity Dialogs



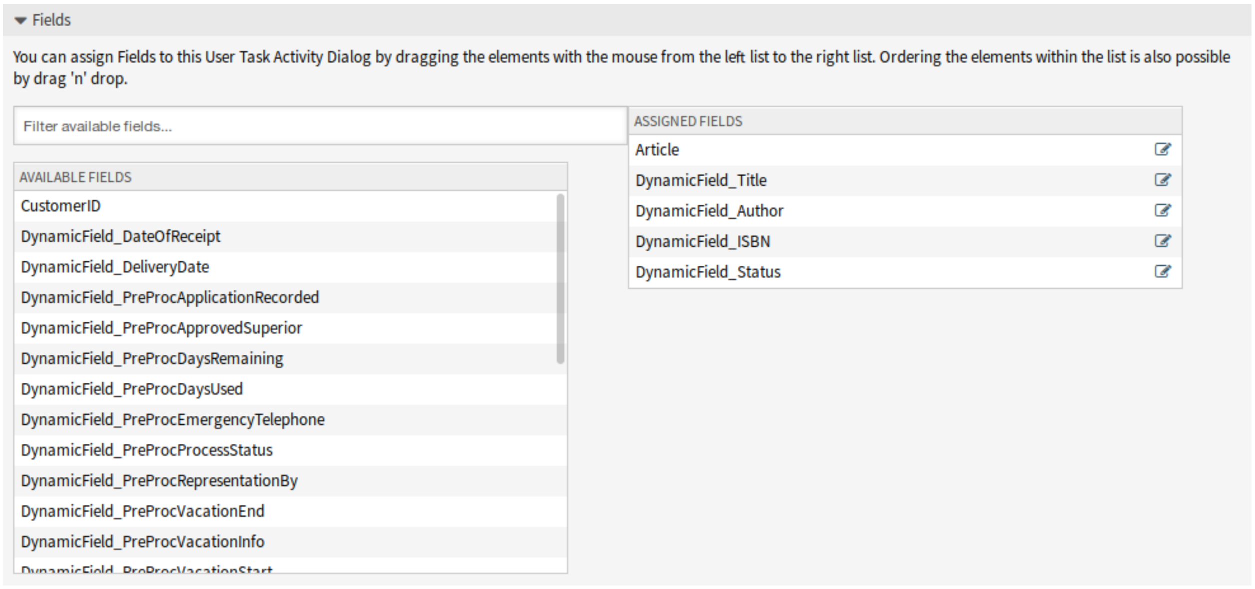
★ Dialog Name:	Recording the demand
Available in:	Agent Interface
★ Description (short):	New demand
Description (long):	
Permission:	
Required Lock:	No
Submit Advice Text:	
Submit Button Text:	

Fig. 124: Book Ordering - Add User Task Activity Dialog

- `DynamicField_Title`, `DynamicField_Author`, `DynamicField_ISBN` fields for the data to be collected for the order.
- `DynamicField_Status` with the possibility to choose *Approval*.

Drag these fields from the *Available Fields* pool and drop into the *Assigned Fields* pool.

Note: In this screen all dynamic fields has the prefix `DynamicField_` as in `DynamicField_Title`. Do not confuse with the field `Title` that is the ticket title.



You can assign Fields to this User Task Activity Dialog by dragging the elements with the mouse from the left list to the right list. Ordering the elements within the list is also possible by drag 'n' drop.

Available fields:
- CustomerID
- DynamicField_DateOfReceipt
- DynamicField_DeliveryDate
- DynamicField_PreProcApplicationRecorded
- DynamicField_PreProcApprovedSuperior
- DynamicField_PreProcDaysRemaining
- DynamicField_PreProcDaysUsed
- DynamicField_PreProcEmergencyTelephone
- DynamicField_PreProcProcessStatus
- DynamicField_PreProcRepresentationBy
- DynamicField_PreProcVacationEnd
- DynamicField_PreProcVacationInfo
- DynamicField_PreProcVacationStart

Assigned fields:
- Article
- DynamicField_Title
- DynamicField_Author
- DynamicField_ISBN
- DynamicField_Status

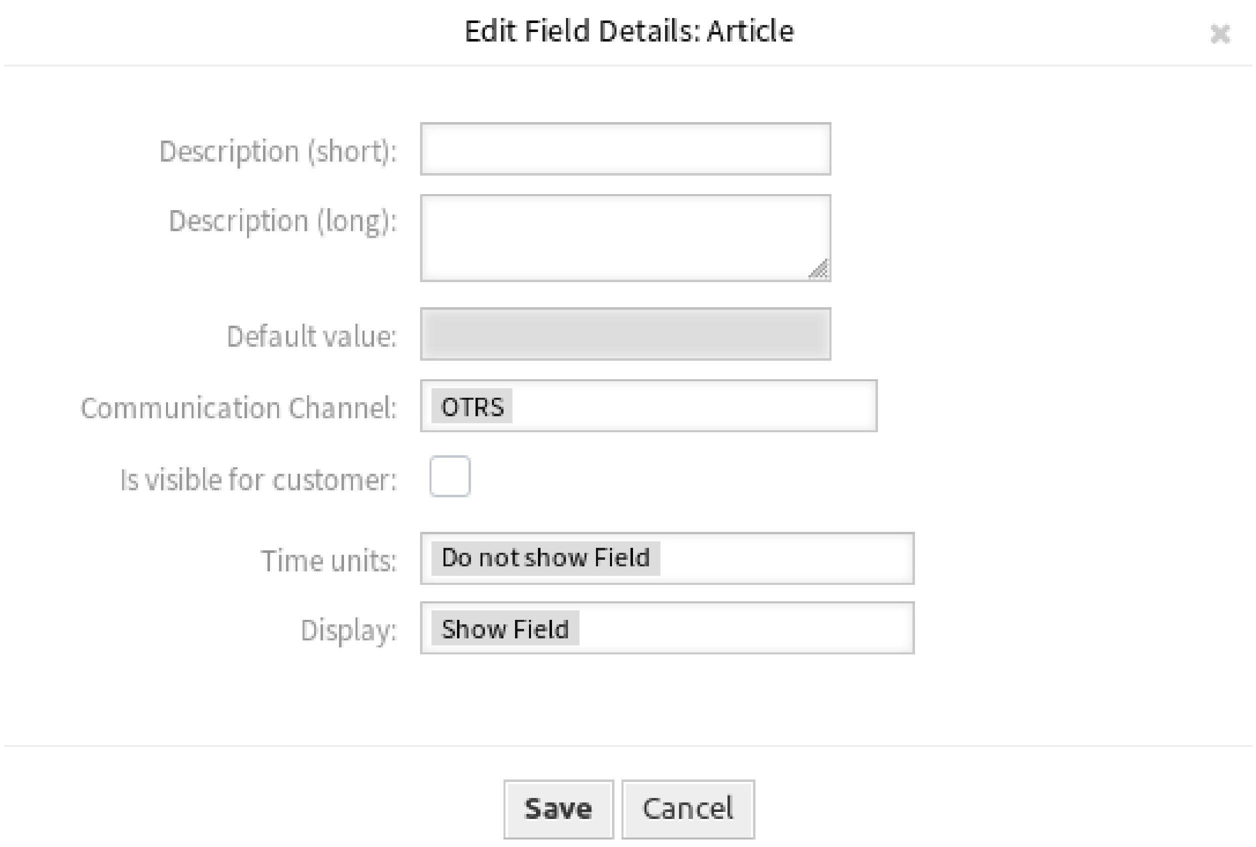
Fig. 125: Book Ordering - Add User Task Activity Dialog Fields

As soon as the fields are dropped into the *Assigned Fields* pool another popup screen is shown with some details about the field. We will leave the default options and only for `Article` fields we should make sure that the *Communication Channel* field is set to *OTRS* and that the *Is visible for customer* is not checked.

After all fields are filled in, click on the *Save and finish* button to save the changes and go back to the project management screen.

Create the following user task activity dialogs with fields:

- *Recording the demand* (already created before)
  - `Article` field for comments.
  - `DynamicField_Title`, `DynamicField_Author`, `DynamicField_ISBN` fields for the data to be collected for the order.
  - `DynamicField_Status` with the possibility to choose *Approval*.
- *Approval denied*
  - `Article` field for comments.
  - `DynamicField_Status` with the possibility to choose *Approval denied*.
- *Approved*



Description (short):	
Description (long):	
Default value:	
Communication Channel:	OTRS
Is visible for customer:	☐
Time units:	Do not show Field
Display:	Show Field

Fig. 126: Book Ordering - Edit User Task Activity Dialog Fields

   - `DynamicField_Status` with the possibility to choose *Approved*.

- *Order denied*
   - `Article` field for comments.
   - `DynamicField_Status` with the possibility to choose *Order denied*.
- *Order placed*
   - `DynamicField_Supplier`, `DynamicField_Price`, `DynamicField_DeliveryDate` fields for purchasing.
   - `DynamicField_Status` with the possibility to choose *Order placed*.
- *Shipment received*
   - `DynamicField_DateOfReceipt` for the mail room.
   - `DynamicField_Status` with the possibility to choose *Shipment received*.

Create Sequence Flows

Click on the *Sequence Flows* item in the *Available Process Elements* widget in the left sidebar. This action will expand the *Sequence Flows* options and will collapse all others doing an accordion like effect. Click on the *Create New Sequence Flow* button.

In the opened popup screen fill in the *Sequence Flow Name*. For this example in the *Condition Expressions* we will use just one condition expression and just one field. For both we can leave the *Type of Linking* as *and* and we will use the filed match type value as *String*.

After all fields are filled in, click on the *Save and finish* button to save the changes and go back to the project management screen.

Create the following sequence flows:

- *Approval* (already created before)

   Check if the `DynamicField_Status` is set to *Approval*.
- *Approval denied*

   Check if the `DynamicField_Status` field is set to *Approval denied*.
- *Approved*

   Check if the `DynamicField_Status` field is set to *Approved*.
- *Order denied*

   Check if the `DynamicField_Status` field is set to *Order denied*.
- *Order placed*

   Check if the `DynamicField_Status` field is set to *Order placed*.
- *Shipment received*

   Check if the `DynamicField_Status` field is set to *Shipment received*.

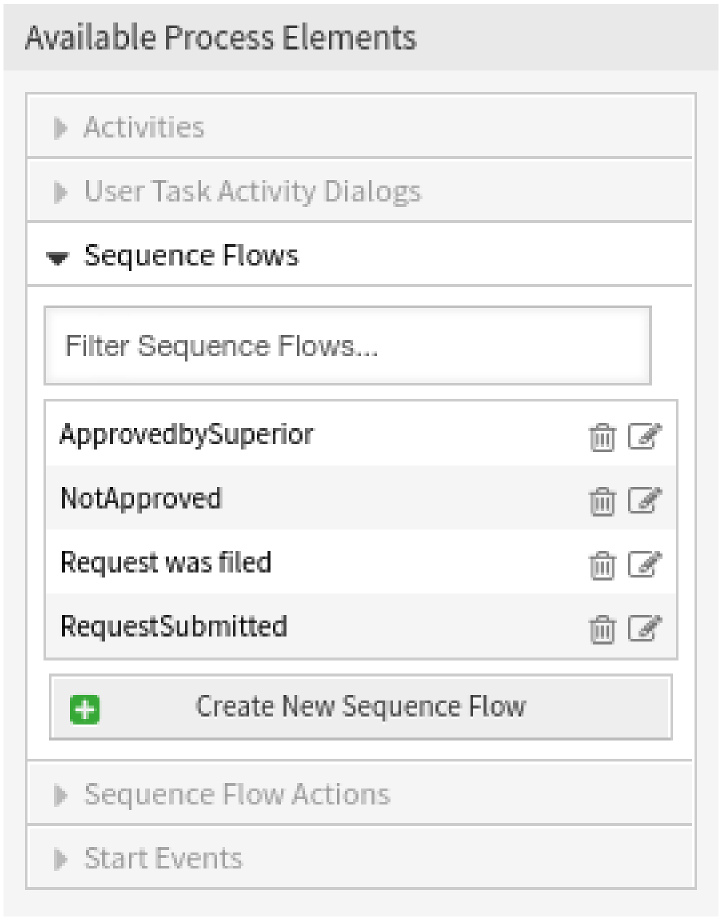
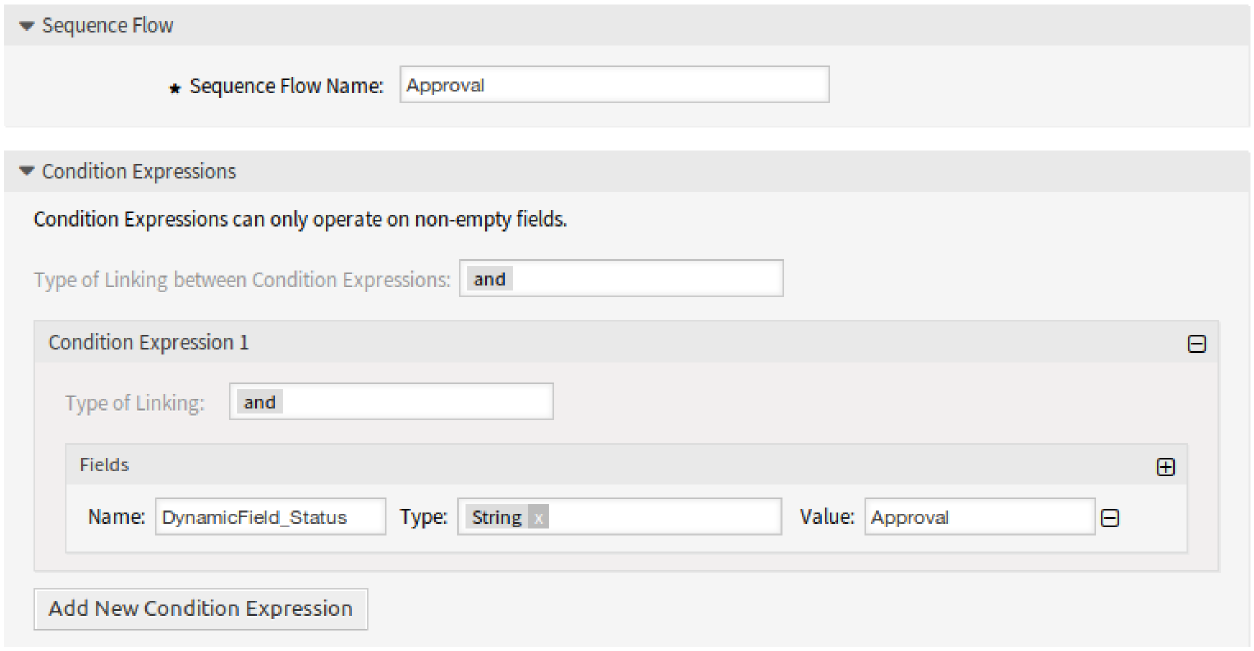


Fig. 127: Book Ordering - Sequence Flows



Sequence Flow Name: Approval

Condition Expressions can only operate on non-empty fields.

Type of Linking between Condition Expressions: and

Type of Linking: and

Name: DynamicField_Status Type: String Value: Approval

Fig. 128: Book Ordering - Add Sequence Flow

Create Sequence Flow Actions

Click on the *Sequence Flow Actions* item in the *Available Process Elements* widget in the left sidebar. This action will expand the *Sequence Flow Actions* options and will collapse all others doing an accordion like effect. Click on the *Create New Sequence Flow Action* button.

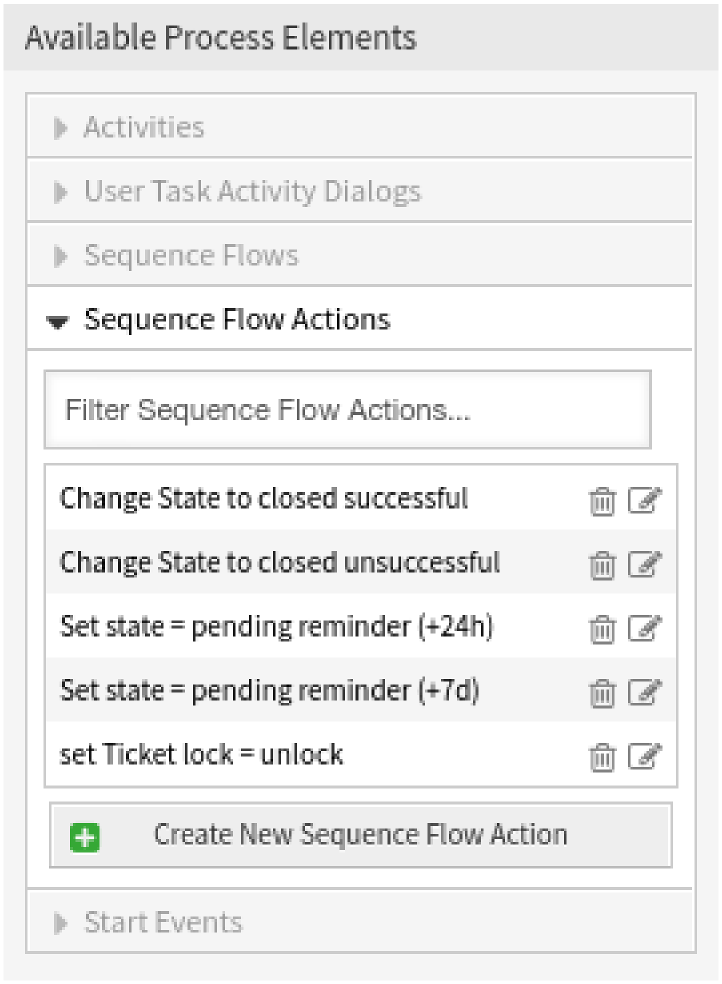
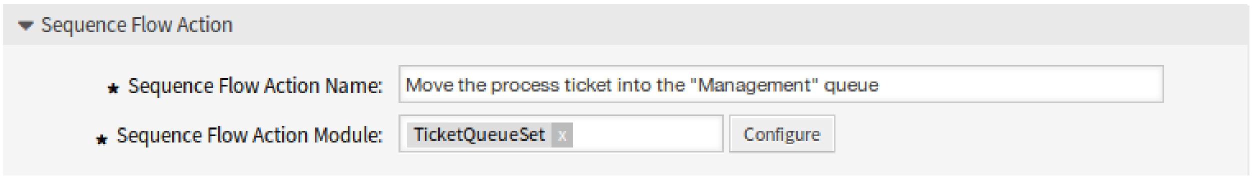

Available Process Elements
Activities
User Task Activity Dialogs
Sequence Flows
Sequence Flow Actions
Filter Sequence Flow Actions...
Change State to closed successful
Change State to closed unsuccessful
Set state = pending reminder (+24h)
Set state = pending reminder (+7d)
set Ticket lock = unlock
Create New Sequence Flow Action
Start Events

Fig. 129: Book Ordering - Sequence Flow Actions

In the opened popup screen fill in the *Sequence Flow Action Name* and the *Sequence Flow Action module* then click on the *Save* button. A new *Configure* button will appear next to the module field.



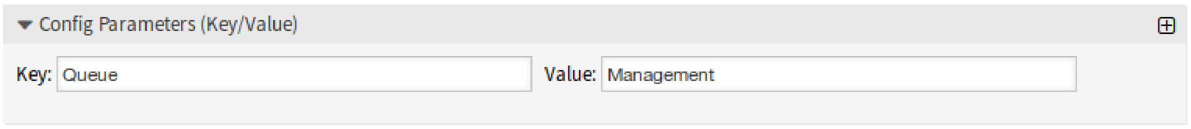
★ Sequence Flow Action Name:	Move the process ticket into the "Management" queue
★ Sequence Flow Action Module:	TicketQueueSet

Fig. 130: Book Ordering - Add Sequence Flow Action

Click on the *Configure* button and add the needed configuration parameter keys and values.

See also:

Each module has its own and different parameters. Please refer to the *Process Modules* section to learn all required and optional parameters.



Key: Queue Value: Management

Fig. 131: Book Ordering - Sequence Flow Action Parameters

After all fields are filled in, click on the *Save and finish* button to save the changes and go back to the project management screen.

Create the following sequence flow actions:

- *Move the process ticket into the "Management" queue* (already created before)

  To be executed when the sequence flow *Approval* applied.

- *Change ticket responsible to "Manager"*

  To be executed when the sequence flow *Approval* applied.

- *Move process ticket into the "Employees" queue*

  To be executed when:

  - The sequence flow *Approval denied* applied.
  - The sequence flow *Order denied* applied.
  - The sequence flow *Shipment received* applied.

- *Change ticket responsible to "Employee"*

  To be executed when:

  - The sequence flow *Approval denied* applied.
  - The sequence flow *Order denied* applied.
  - The sequence flow *Shipment received* applied.

- *Move process ticket into the "Purchasing" queue*

  To be executed when the sequence flow *Approved* applied.

- *Move process ticket into the "Post office" queue*

  To be executed when the sequence flow *Order placed* applied.

- *Close ticket successfully*

  To be executed when the sequence flow *Shipment received* applied.

- *Close ticket unsuccessfully*

  To be executed when:

  - The sequence flow *Approval denied* applied.
  - The sequence flow *Order denied* applied.

There are places where the same sequence flow actions should be executed. Therefore it is reasonable to make it possible to link sequence flow actions freely with sequence flows to be able to reuse them.

Create Activities

Click on the *Activities* item in the *Available Process Elements* widget in the left sidebar. This action will expand the *Activities* options and will collapse all others doing an accordion like effect. Click on the *Create New Activity* button.

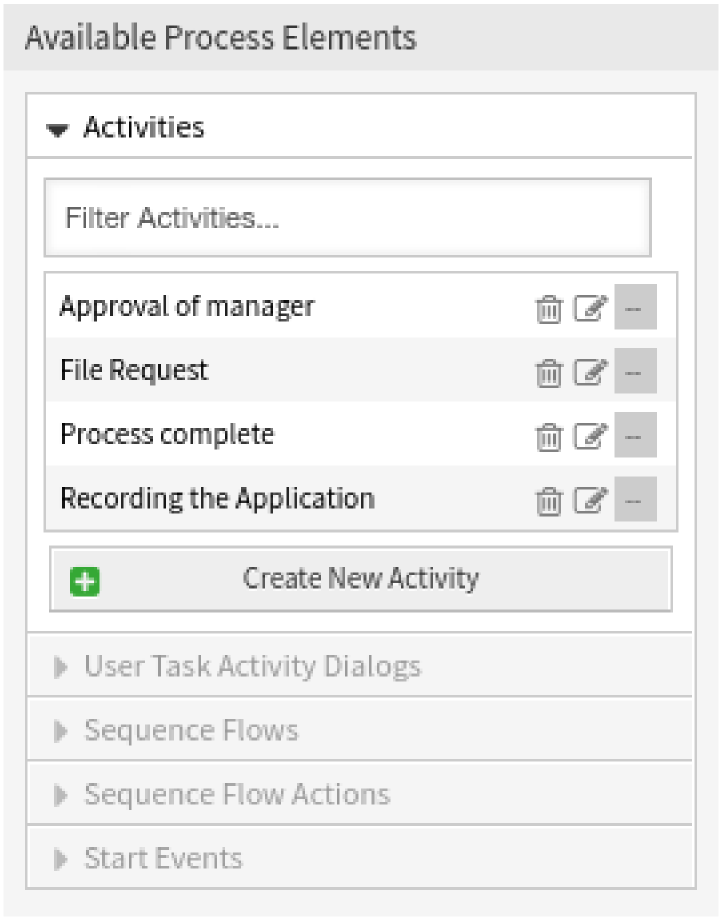
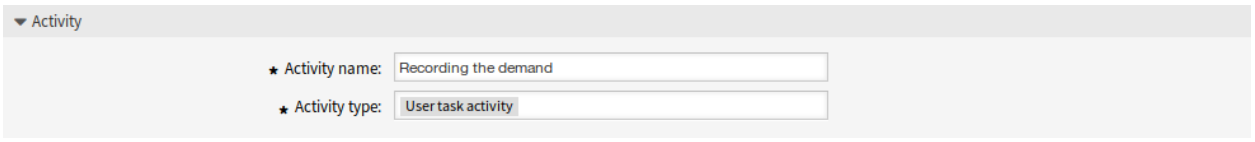


Fig. 132: Book Ordering - Activities

In the opened popup screen fill in the *Activity name* field and select *User task activity* from the *Activity type* drop-down.



Activity name: Recording the demand
Activity type: User task activity

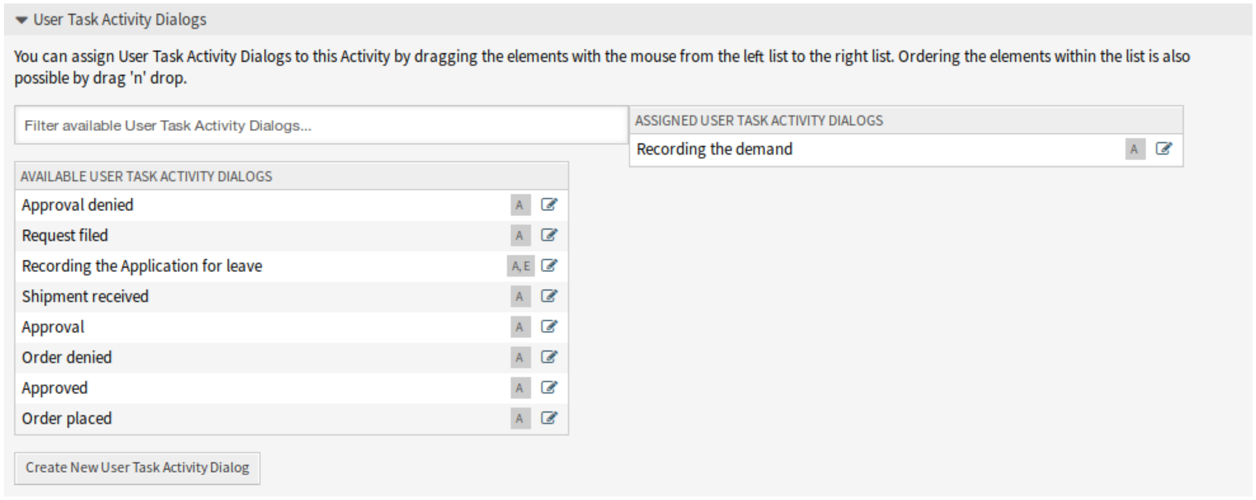
Fig. 133: Book Ordering - Add Activity

To assign dialogs to the activity simple drag the required dialogs from the *Available User Task Activity Dialogs* pool and drop into the *Assigned User Task Activity Dialogs* pool. The order in the *Assigned User Task Activity*

Dialogs pool is the order as the dialogs will be presented in the ticket detail view. To modify the order simply drag and drop the dialog within the pool to rearrange it in the correct place.

Note: This order is specially important in the first activity, since the first user task activity dialog for this activity is the only one that is presented when the process starts.

In this example we need to assign only the *Recording the demand* user task activity dialog. Drag this dialog from the *Available User Task Activity Dialogs* pool and drop into the *Assigned User Task Activity Dialogs* pool.


User Task Activity Dialogs

You can assign User Task Activity Dialogs to this Activity by dragging the elements with the mouse from the left list to the right list. Ordering the elements within the list is also possible by drag 'n' drop.

Filter available User Task Activity Dialogs...

AVAILABLE USER TASK ACTIVITY DIALOGS	
Approval denied	A
Request filed	A
Recording the Application for leave	A, E
Shipment received	A
Approval	A
Order denied	A
Approved	A
Order placed	A

ASSIGNED USER TASK ACTIVITY DIALOGS	
Recording the demand	A

Fig. 134: Book Ordering - Assign User Task Activity Dialog

After all fields are filled in, click on the *Save and finish* button to save the changes and go back to the project management screen.

Create the following activities:

- *Recording the demand* (already created before)

  Assign the user task activity dialog *Recording the demand*.

- *Approval*

  Assign the user task activity dialogs *Approval denied* and *Approved*.

- *Order*

  Assign the user task activity dialogs *Order denied* and *Order placed*.

- *Incoming*

  Assign the user task activity dialog *Shipment received*.

- *Process complete*

  This is an activity without possible user task activity dialogs. It will be set after *Approval denied*, *Order denied* or *Shipment received* and represents the end of the process.

Now we can clearly see that activities are precisely defined states of a process ticket. After a successful sequence flow a process ticket moves from one activity to another.

Create Process Path

Let us conclude our example with the last missing piece in the puzzle, the process as the a flow describer. In our case this is the whole ordering workflow. Other processes could be office supply ordering or completely different processes.

The process has a starting point which consists of the start activity and the start user task activity dialog. For any new book order, the first user task activity dialog of the first activity is the first screen that is displayed. If this is completed and saved, the process ticket will be created and can follow the configured workflow.

The process also contains the directions for how the process ticket can move through the process. Let's call this *process path*. It consists of the start activity, one or more sequence flows (possibly with sequence flow actions) and other activities.

Assuming that the activities has already assigned their user task activity dialogs, drag an activity from the accordion in the *Available Process Elements* widget in the left sidebar and drop it into the canvas area below the process information. Notice that an arrow from the process start (white circle) to the activity is placed automatically. This is the first activity and its first user task activity dialog is the first screen that will be shown when the process starts.

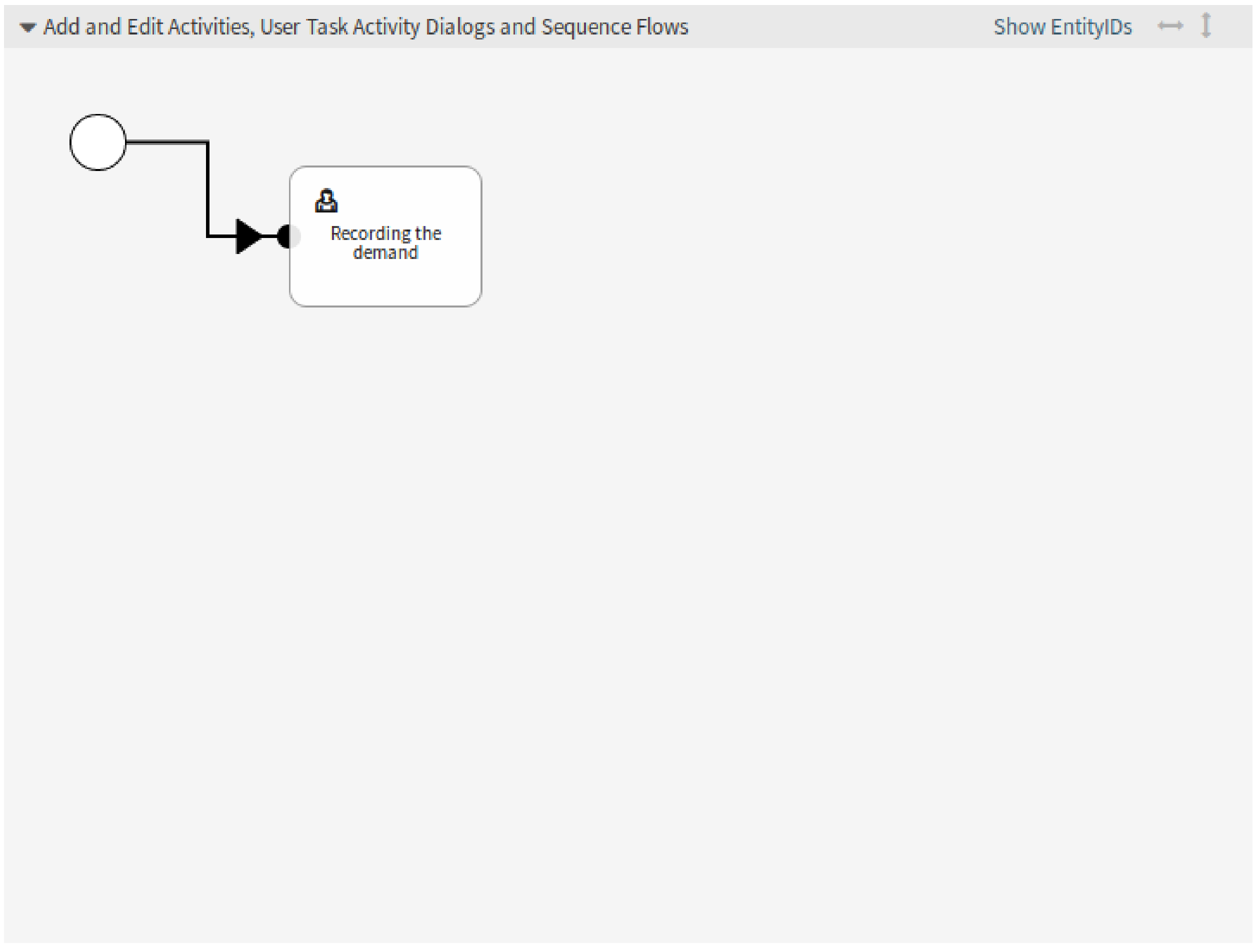

Add and Edit Activities, User Task Activity Dialogs and Sequence Flows
Show EntityIDs
Recording the demand

Fig. 135: Book Ordering - First Activity On Canvas

Next, drag another activity into the canvas too. Now we will have two activities in the canvas. The first one is connected to the start point and the second has no connections. You can hover the mouse over each activity to reveal their own activity dialogs.

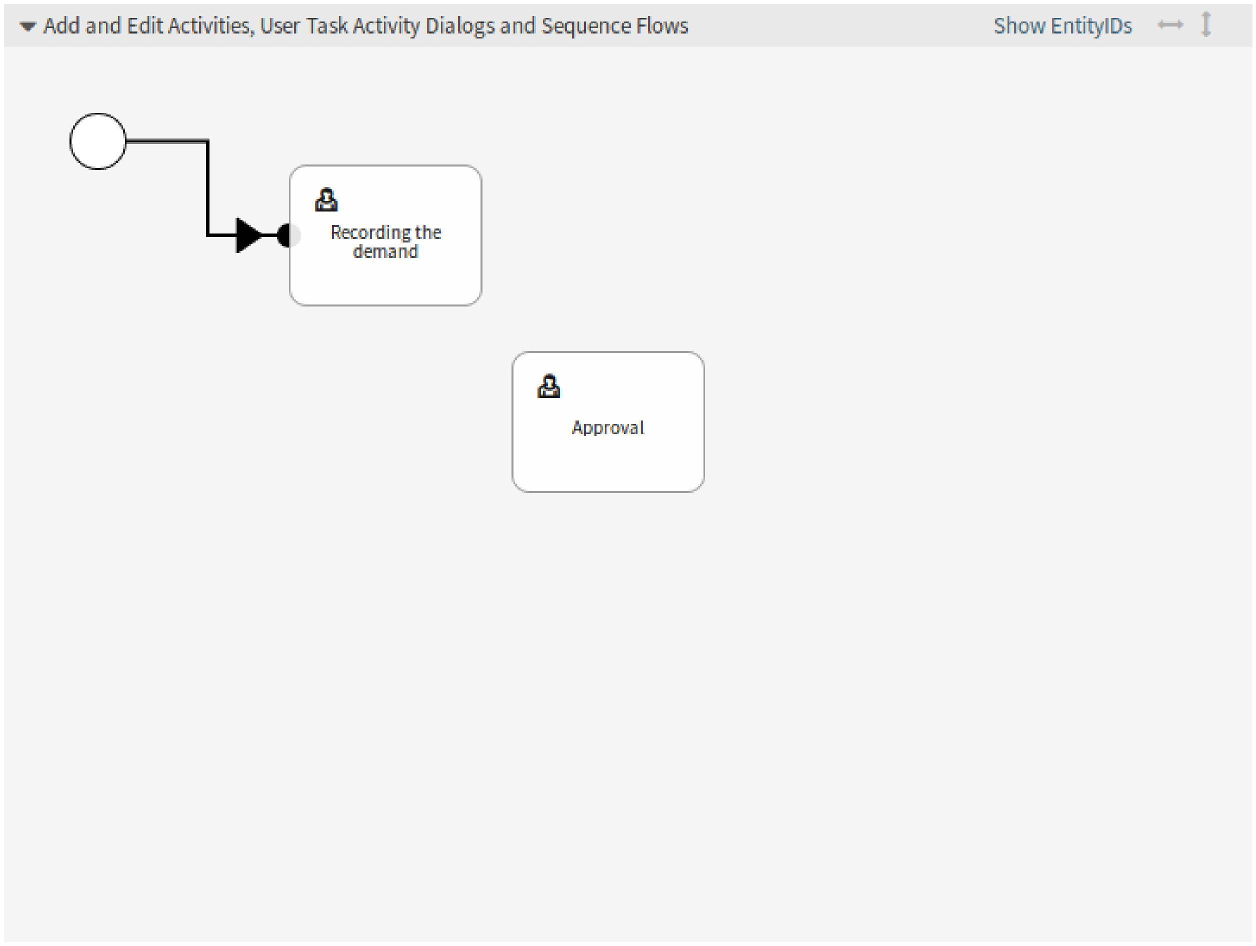

Add and Edit Activities, User Task Activity Dialogs and Sequence Flows
Show EntityIDs
Recording the demand
Approval

Fig. 136: Book Ordering - Second Activity On Canvas

Then let's create the process path (connection) between this two activities. For this we will use the sequence flows. Click on sequence flow in the accordion, drag a sequence flow and drop it inside the first activity. As soon as the sequence flow is dropped the end point of the sequence flow arrow will be placed next to the process start point. Drag the sequence flow arrow end point and drop it inside the other activity to create the connection between the activities.

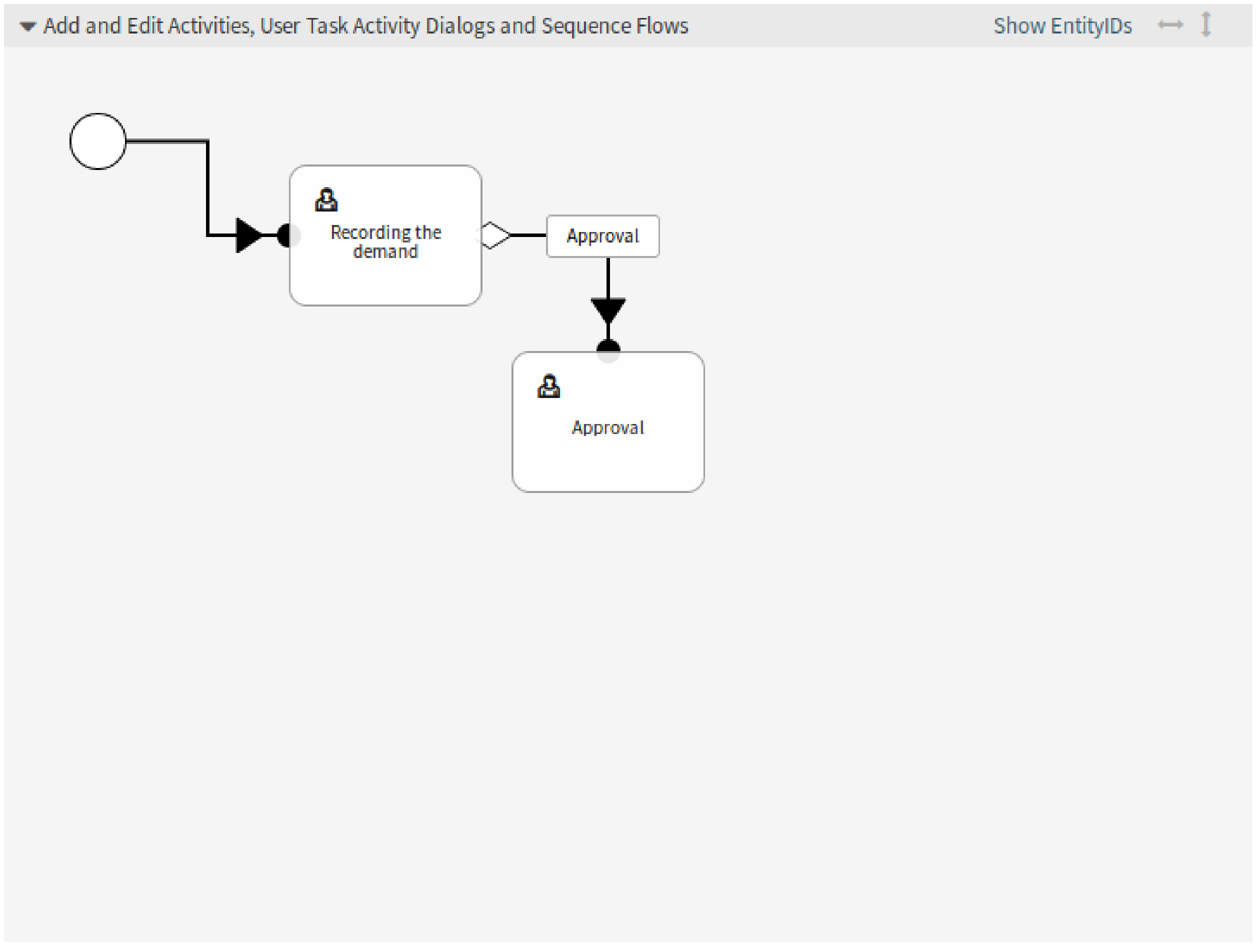

Add and Edit Activities, User Task Activity Dialogs and Sequence Flows
Show EntityIDs
Recording the demand
Approval
Approval

Fig. 137: Book Ordering - First Sequence Flow On Canvas

Now that the process path between the actions is defined, then we need to assign the sequence flow actions to the sequence flow. Double click the sequence flow label in the canvas to open a new popup window.

After the sequence flow actions are assigned, click on the *Save* button to go back to the main process edit screen. Click on *Save* button below the canvas to save all other changes.

Complete the process path by adding the following activities, sequence flows and sequence flow actions:

- *Recording the demand* (already created before)

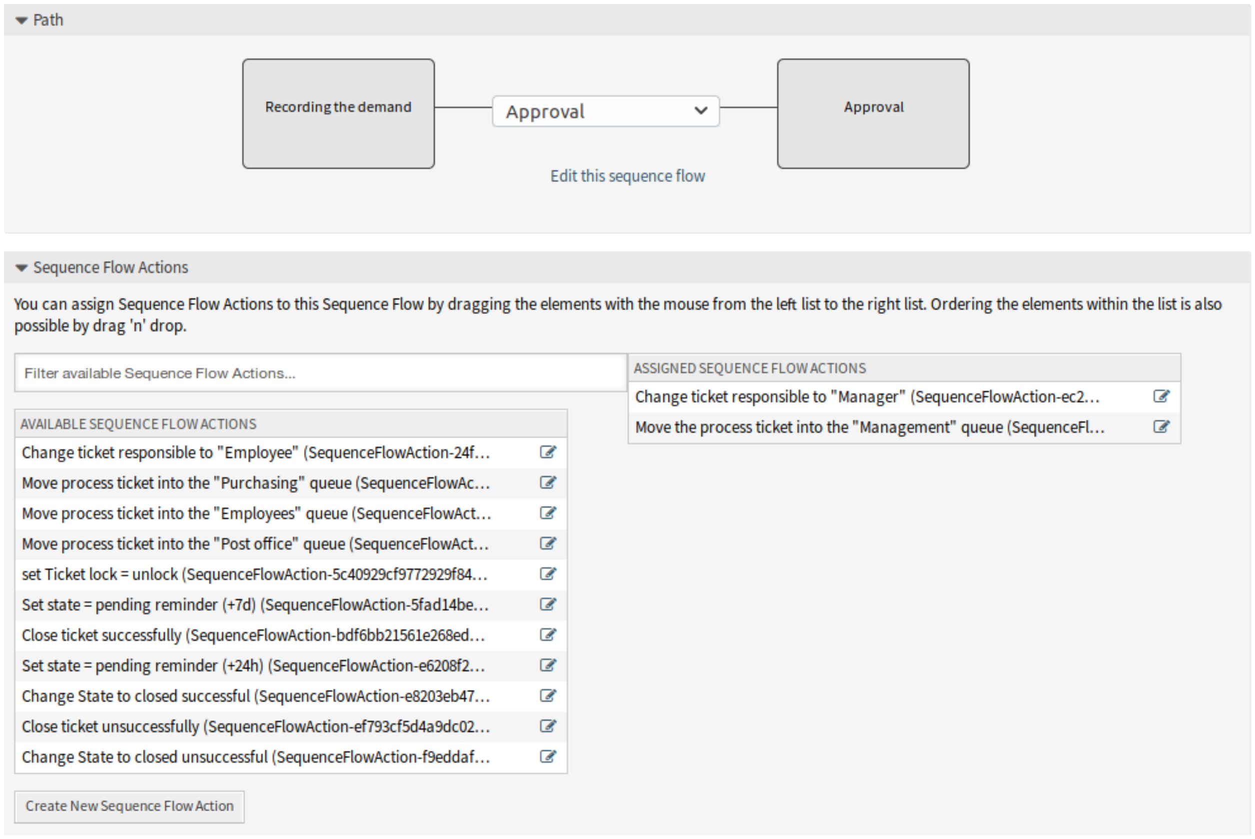
  Possible sequence flow: *Approval*

  Starting activity: *Recording the demand*

  Next activity: *Approval*

  If the condition of this activity is fulfilled, the ticket will move to activity *Approval*.

  Additionally, the following sequence flow actions are executed:



You can assign Sequence Flow Actions to this Sequence Flow by dragging the elements with the mouse from the left list to the right list. Ordering the elements within the list is also possible by drag 'n' drop.

AVAILABLE SEQUENCE FLOW ACTIONS
Change ticket responsible to "Employee" (SequenceFlowAction-24f…
Move process ticket into the "Purchasing" queue (SequenceFlowAc…
Move process ticket into the "Employees" queue (SequenceFlowAct…
Move process ticket into the "Post office" queue (SequenceFlowAct…
set Ticket lock = unlock (SequenceFlowAction-5c40929cf9772929f84…
Set state = pending reminder (+7d) (SequenceFlowAction-5fad14be…
Close ticket successfully (SequenceFlowAction-bdf6bb21561e268ed…
Set state = pending reminder (+24h) (SequenceFlowAction-e6208f2…
Change State to closed successful (SequenceFlowAction-e8203eb47…
Close ticket unsuccessfully (SequenceFlowAction-ef793cf5d4a9dc02…
Change State to closed unsuccessful (SequenceFlowAction-f9eddaf…

ASSIGNED SEQUENCE FLOW ACTIONS
Change ticket responsible to "Manager" (SequenceFlowAction-ec2…
Move the process ticket into the "Management" queue (SequenceFl…

Fig. 138: Book Ordering - Assign First Sequence Flow Action

- *Move the process ticket into the "Management" queue*
- *Change ticket responsible to "Manager"*

The activity *Recording the demand* is a defined step of the process ticket, where there is the possibility for the sequence flow *Approval*. If this applies, the ticket will move to the next activity *Approval*, and the sequence flow actions *Move the process ticket into the "Management" queue* and *Change ticket responsible to "Manager"* are executed. In the activity *Approval*, the user task activity dialogs *Approval denied* and *Approved* are available.

- *Approval*

  Possible sequence flow: *Approval denied*

  Starting activity: *Approval*

  Next activity: *Process complete*

  If this matches, the process ticket will move to activity *Process complete*.

  Additionally, the following sequence flow actions are executed:

  - *Move process ticket into the "Employees" queue*
  - *Change ticket responsible to "Employee"*
  - *Close ticket unsuccessfully*

  Possible sequence flow: *Approved*

  Starting activity: *Approval*

  Next activity: *Order*

  If this matches, the process ticket will move to activity *Order*.

  Additionally, the following sequence flow actions are executed:

  - *Move process ticket into the "Purchasing" queue*

  We can see that from the current activity, which defines a step of the process ticket, there are one or more possibilities for sequence flow which have exactly one target activity (and possibly one or more sequence flow actions).

- *Order*

  Possible sequence flow: *Order denied*

  Starting activity: *Order*

  Next activity: *Process complete*

  If this matches, the process ticket will move to activity *Process complete*.

  Additionally, the following sequence flow actions are executed:

  - *Move process ticket into the "Employees" queue*
  - *Change ticket responsible to "Employee"*
  - *Close ticket unsuccessfully*

  Possible sequence flow: *Order placed*

  Starting activity: *Order*

  Next activity: *Incoming*

  If this matches, the process ticket will move to activity *Incoming*.

Additionally, the following sequence flow actions are executed:

- *Move process ticket into the "Post office" queue*

- *Incoming*

  Possible sequence flow: *Shipment received*

  Starting activity: *Incoming*

  Next activity: *Process complete*

  If this matches, the process ticket will move to activity *Process complete*.

  Additionally, the following sequence flow actions are executed:

  - *Move process ticket into the "Employees" queue*
  - *Change ticket responsible to "Employee"*
  - *Close ticket successfully*

The complete process path for the book ordering process will then look like this:

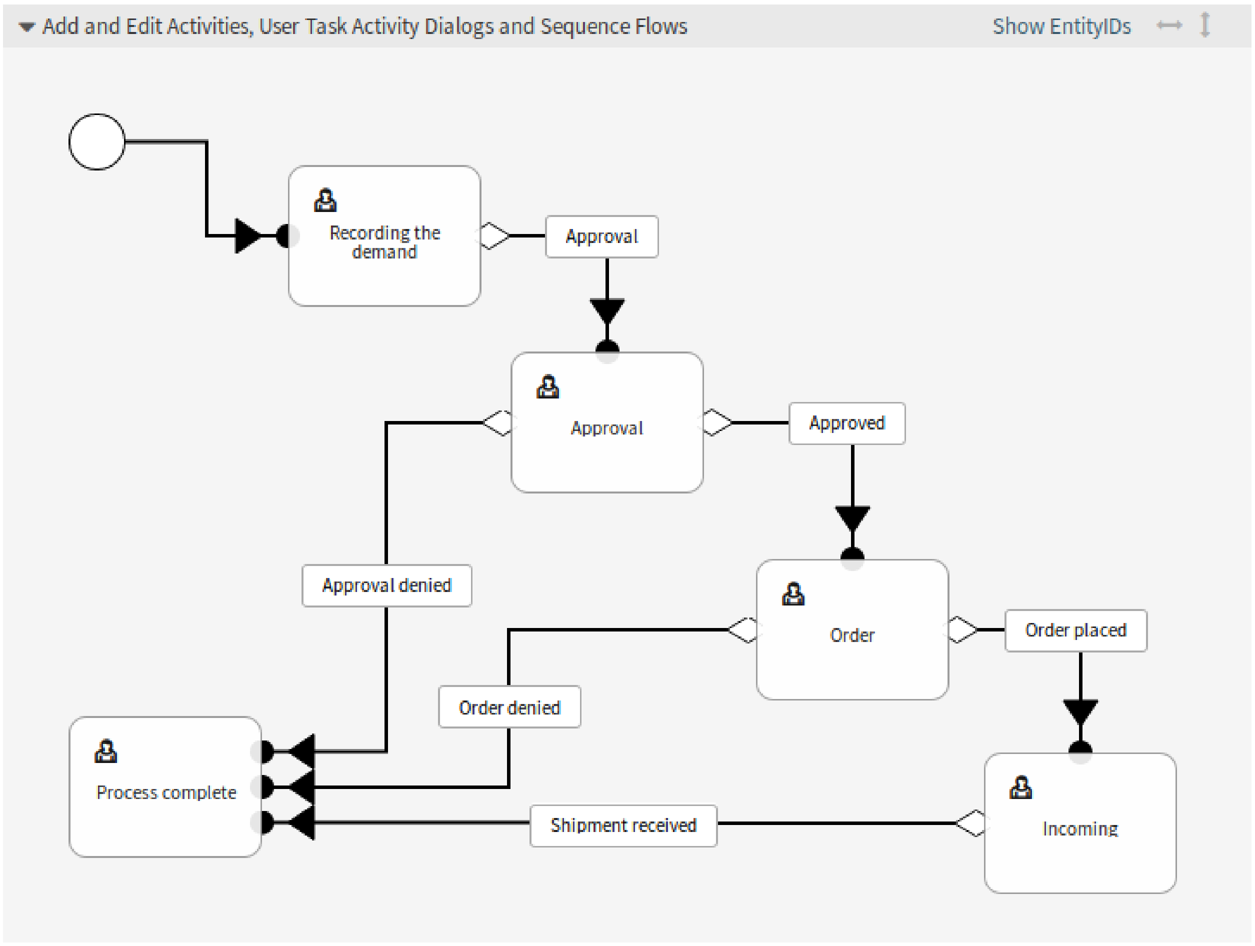


Fig. 139: Book Ordering - Process Complete

After you finish the process path, click on *Save and finish* button below the canvas to go back to the process management screen.

Click on the *Deploy All Processes* button in the left sidebar. This will gather all processes information form the database and create a cache file (in Perl language). This cache file is actually the processes configuration that the system will use to create or use process tickets.

Note: Any change that is made on the process will require to re-deploy the process in order to get the change reflected in the system.

Create Access Control Lists

With the help of *Access Control Lists (ACL)*, the selectable values in process tickets can be limited. Some ACLs have to be defined for the book ordering process to operate correctly.

In this section, all necessary ACLs are defined. Each ACL is added here in YAML format, so you can copy them, save them as separate `.yml` files and import them in the ACL management screen.

Warning: The exported ACLs contain the activity dialog IDs from the system, where they were exported from. Do not forget to change the IDs based on your process. Otherwise the ACLs will not work.

See also:

Use the *Show EntityIDs* link in the header of the process canvas to see the entity IDs. For the activity dialogs, hover the mouse over the name in the list of activity dialogs in the left sidebar to see the ID.

`001-ACL-BookOrderingStatus`

This ACL enables only the *Approval* value for the `Status` dynamic field in the *Recording the demand* activity dialog.

```
---
- ChangeBy: root@localhost
  ChangeTime: 2020-04-18 15:46:16
  Comment: Approval
  ConfigChange:
    Possible:
      Ticket:
        DynamicField_Status:
        - Approval
  ConfigMatch:
    Properties:
      Process:
        ActivityDialogEntityID:
        - ActivityDialog-bfa31751ee47f8d8ec3a15e4cf1de732
  CreateBy: root@localhost
  CreateTime: 2020-04-18 15:42:06
  Description: ''
  ID: 1
  Name: 001-ACL-BookOrderingStatus
  StopAfterMatch: 0
  ValidID: 1
```

`002-ACL-BookOrderingStatus`

This ACL enables only the *Approval denied* value for the `Status` dynamic field in the *Approval denied* activity dialog.

```
---
- ChangeBy: root@localhost
  ChangeTime: 2020-04-18 15:46:08
  Comment: Approval denied
  ConfigChange:
    Possible:
      Ticket:
        DynamicField_Status:
        - Approval denied
  ConfigMatch:
    Properties:
      Process:
        ActivityDialogEntityID:
        - ActivityDialog-1ce810fd3668ce799f25cf968b139427
  CreateBy: root@localhost
  CreateTime: 2020-04-18 15:44:21
  Description: ''
  ID: 2
  Name: 002-ACL-BookOrderingStatus
  StopAfterMatch: 0
  ValidID: 1
```

003-ACL-BookOrderingStatus
This ACL enables only the *Approved* value for the `Status` dynamic field in the *Approved* activity dialog.

```
---
- ChangeBy: root@localhost
  ChangeTime: 2020-04-18 15:47:04
  Comment: Approved
  ConfigChange:
    Possible:
      Ticket:
        DynamicField_Status:
        - Approved
  ConfigMatch:
    Properties:
      Process:
        ActivityDialogEntityID:
        - ActivityDialog-96b8e0d7f8a0e69e170f7871cbb83e15
  CreateBy: root@localhost
  CreateTime: 2020-04-18 15:46:22
  Description: ''
  ID: 3
  Name: 003-ACL-BookOrderingStatus
  StopAfterMatch: 0
  ValidID: 1
```

004-ACL-BookOrderingStatus
This ACL enables only the *Order denied* value for the `Status` dynamic field in the *Order denied* activity dialog.

```
---
- ChangeBy: root@localhost
  ChangeTime: 2020-04-18 15:48:07
  Comment: Order denied
  ConfigChange:
    Possible:
```

(continues on next page)

(continued from previous page)

```
      Ticket:
        DynamicField_Status:
        - Order denied
  ConfigMatch:
    Properties:
      Process:
        ActivityDialogEntityID:
        - ActivityDialog-5b60db9960a9cd488f448e3308cc8b4f
  CreateBy: root@localhost
  CreateTime: 2020-04-18 15:47:07
  Description: ''
  ID: 4
  Name: 004-ACL-BookOrderingStatus
  StopAfterMatch: 0
  ValidID: 1
```

005-ACL-BookOrderingStatus

This ACL enables only the *Order placed* value for the `Status` dynamic field in the *Order placed* activity dialog.

```
---
- ChangeBy: root@localhost
  ChangeTime: 2020-04-18 15:48:51
  Comment: Order placed
  ConfigChange:
    Possible:
      Ticket:
        DynamicField_Status:
        - Order placed
  ConfigMatch:
    Properties:
      Process:
        ActivityDialogEntityID:
        - ActivityDialog-a756ccae6ae83f356faa8333549a87f0
  CreateBy: root@localhost
  CreateTime: 2020-04-18 15:48:13
  Description: ''
  ID: 5
  Name: 005-ACL-BookOrderingStatus
  StopAfterMatch: 0
  ValidID: 1
```

006-ACL-BookOrderingStatus

This ACL enables only the *Shipment received* value for the `Status` dynamic field in the *Shipment received* activity dialog.

```
---
- ChangeBy: root@localhost
  ChangeTime: 2020-04-18 15:49:41
  Comment: Shipment received
  ConfigChange:
    Possible:
      Ticket:
        DynamicField_Status:
        - Shipment received
  ConfigMatch:
```

(continues on next page)

(continued from previous page)

```
    Properties:
      Process:
        ActivityDialogEntityID:
        - ActivityDialog-885f547d9a0e07aa6e2703af59ec08ae
  CreateBy: root@localhost
  CreateTime: 2020-04-18 15:48:57
  Description: ''
  ID: 6
  Name: 006-ACL-BookOrderingStatus
  StopAfterMatch: 0
  ValidID: 1
```

Note: Don't forget to deploy the imported ACLs.

Create Process Ticket

The book ordering process is ready to use. Go to the *New Process Ticket* screen in the agent interface, and find the book ordering process.

5.5 Web Services

In a connected world, a ticket system needs to be able to react to requests from other systems and also to send requests or information to other systems:

- CRM systems
- Project management systems
- Documentation management systems
- and many more

The ticket system must be reachable by other services without manual intervention by an agent.

OTRS supports this requirement by the *Generic Interface*. It empowers the administrator to create a web service for a specific task without scripting language knowledge. OTRS reacts on incoming REST or SOAP requests and create objects or provides object data to other systems transparently.

A web service is a communication method between two systems, in our case OTRS and a remote system. In its configuration, the *operation* or *invoker* determine the direction of communication, and the *mapping* and *transport* take care of how the data is received and interpreted.

In its configuration it can be defined what actions the web service can perform internally (operation), what actions the OTRS request can perform remote system (invokers), how data is converted from one system to the other (mapping), and over which protocol the communication will take place (transport).

The generic interface is the framework that makes it possible to create web services for OTRS in a predefined way, using already made building blocks that are independent from each other and interchangeable.

5.5.1 Web Service Documentation

5.5.2 Generic Interface

The generic interface consists of a multiple layer framework that lets OTRS communicate with other systems via a web service. This communication could be bi-directional:

- OTRS as provider: OTRS acts as a server listening to requests from the external system, processing the information, performing the requested action, and answering the request.
- OTRS as requester: OTRS acts as a client collecting information, sending the request to the remote system, and waiting for the response.

Generic Interface Layers

The generic interface is build based on a layer model, to be flexible and easy to customize.

A layer is a set of files, which control how the generic interface performs different parts of a web service. Using the right configuration, one can build different web services for different external systems without creating new modules.

Note: If the remote system does not support the current bundled modules of the generic interface, special modules need to be developed for that specific web service.

Network Transport

This layer is responsible for the correct communication with the remote system. It receives requests and generates responses when acting as provider, and generates requests and receives responses when acting as requester.

Requester communication could be initiated during an event triggered by a generic interface module or any other OTRS module. This event is caught by the event handler and depending on the configuration the event will be processed directly by the requester object or delegated to the scheduler (a separated daemon designed to process tasks asynchronously).

Data Mapping

This layer is responsible for translating data structures between OTRS and the remote system (data internal and data external layers). Usually remote systems have different data structures than OTRS (including different values and names for those values), and here resides the importance of the layer to change the received information into something that OTRS can understand and on the opposite way send the information to each remote system using their data dictionaries.

Example: *Priority* (OTRS) might be called *Prio* in a remote system and it could be that value *1 very low* (OTRS) should be mapped to *Information* in the remote system.

Controller

Controllers are collections of similar operations or invokers. For example, a ticket controller might contain several standard ticket operations. Custom controllers can be implemented, for example a `TicketExternalCompany` controller which may contain similar functions as the standard ticket controller, but with a different data interface, or function names (to adapt to the remote system function names) or complete different code.

One application for generic interface could be to synchronize information with one remote system that only can talk with another remote system of the same kind. In this case new controllers needs to be developed and the operations and invokers has to emulate the remote system behavior in such way that the interface that OTRS exposes is similar to the interface of the remote system.

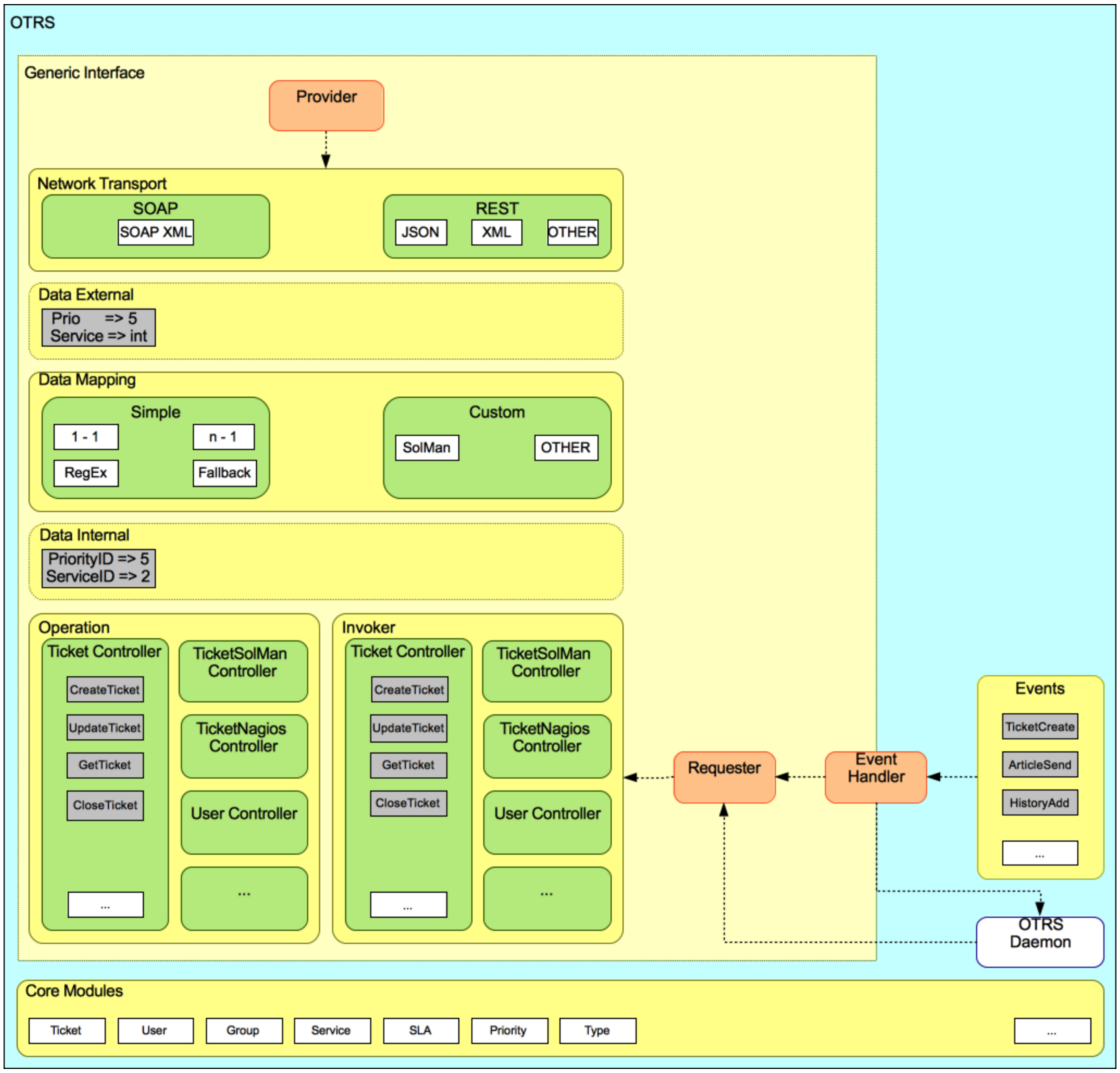


Fig. 140: Generic Interface Layers

Operation (OTRS as a provider)
An operation is a single action that can be performed within OTRS. All operations have the same programming interface, they receive the data into one specific parameter, and return a data structure with a success status, potential error message and returning data.

Normally operations uses the already mapped data (internal) to call core modules and perform actions in OTRS like: create a ticket, update a user, invalidate a queue, send a notification, etc. An operation has full access to the OTRS API to perform the action.

Invoker (OTRS as a requester)
An invoker is an action that OTRS performs against a remote system. Invokers use the OTRS core modules to process and collect the needed information to create the request. When the information is ready it has to be mapped to the remote system format in order to be sent to the remote system, that will process the information, execute the action and send the response back, to either process the success or handle errors.

Generic Interface Communication Flow

The generic interface has a defined flow to perform actions as a provider and as a requester. These flows are described below:

OTRS as Provider

Remote Request:

1. HTTP request
   - OTRS receives HTTP request and passes it through the layers.
   - The provider module is in charge to execute and control these actions.
2. Network transport
   - The network transport module decodes the data payload and separates the operation name from the rest of the data.
   - The operation name and the operation data are returned to the provider.
3. Data external
   - Data as sent from the remote system (this is not a module based layer).
4. Mapping
   - The data is transformed from the external system format to the OTRS internal format as specified in the mapping configuration for this operation (mapping for incoming request data).
   - The already transformed data is returned to the provider.
5. Data internal
   - Data as transformed and prepared to be passed to the operation (This is not a module based layer).
6. Operation
   - Receives and validates data.
   - Performs user access control.
   - Executes the action.

OTRS Response:

1. Operation
   - Returns result data to the provider.
2. Data internal
   - Data as returned from operation.
3. Mapping
   - The data is transformed back to the remote system format as specified in the mapping configuration (mapping for outgoing response data).
   - The already transformed data is returned to the provider.
4. Data external
   - Data as transformed and prepared to be passed to network transport as response.
5. Network transport
   - Receives the data already in the remote system format.
   - Constructs a valid response for this network transport type.
6. HTTP response
   - The response is sent back to the web service client.
   - In the case of an error, an error response is sent to the remote system (e.g. SOAP fault, HTTP error, etc).

OTRS as Requester

OTRS Request:

1. Event trigger handler
   - Based on the web service configuration determines if the request will be synchronous or asynchronous.
     - Synchronous
       - A direct call to the requester is made in order to create a new request and to pass it through the layers.
     - Asynchronous
       - Create a new generic interface (requester) task for the OTRS daemon (by delegating the request execution to the scheduler daemon, the user experience could be highly improved, otherwise all the time needed to prepare the request and the remote execution will be added to the OTRS events that trigger those requests).
       - In its next cycle the OTRS daemon process reads the new task and creates a call to the requester that will create a new request and then passes it through the layers.
2. Invoker
   - Receives data from the event.
   - Validates received data (if needed).
   - Call core modules to complement the data (if needed).

   - Return the request data structure or send a stop communication signal to the requester, to gracefully cancel the request.

3. Data internal
   - Data as passed from the invoker (this is not a module based layer).
4. Mapping
   - The data is transformed to the remote system format as specified in the mapping configuration (mapping for outgoing response data).
   - The already transformed data is returned to the requester.
5. Data external
   - Data as transformed and prepared for sending to the remote system.
6. Network transport
   - Receives the remote operation name and the data already transformed to the remote system format from the requester.
   - Constructs a valid request for the network transport.
   - Sends the request to the remote system and waits for the response.

Remote Response:

1. Network transport
   - Receives the response and decodes the data payload.
   - Returns the data to the requester.
2. Data external
   - Data as received from the remote system.
3. Mapping
   - The data is transformed from the external system format to the OTRS internal format as specified in the mapping configuration for this operation (mapping for incoming response data).
   - The already transformed data is returned to the requester.
4. Data internal
   - Data as transformed and ready to be passed back to the requester.
5. Invoker
   - Receives return data.
   - Handles the data as needed specifically by each invoker (included error handling if any).
   - Return the Invoker result and data to the Requester.
6. Event handler or OTRS daemon
   - Receives the data from the requester. In the case of the OTRS daemon this data might contain information to create a task in the future.

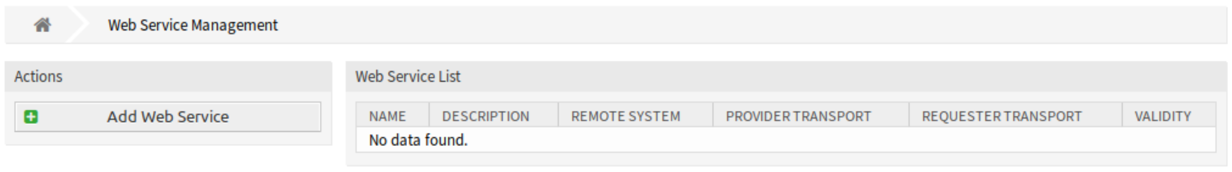
5.5.3 Manage Web Services

A web service is a communication method between two systems, in our case OTRS and a remote system.

The heart of the web service is its configuration, where it is defined what actions the web service can perform internally (operation), what actions the OTRS request can perform remote system (invokers), how data is converted from one system to the other (mapping), and over which protocol the communication will take place (transport).

The generic interface is the framework that makes it possible to create web services for OTRS in a predefined way, using already made building blocks that are independent from each other and interchangeable.

Use this screen to manage web services in the system. A fresh OTRS installation contains no web service by default. The web service management screen is available in the *Web Services* module of the *Processes & Automation* group.

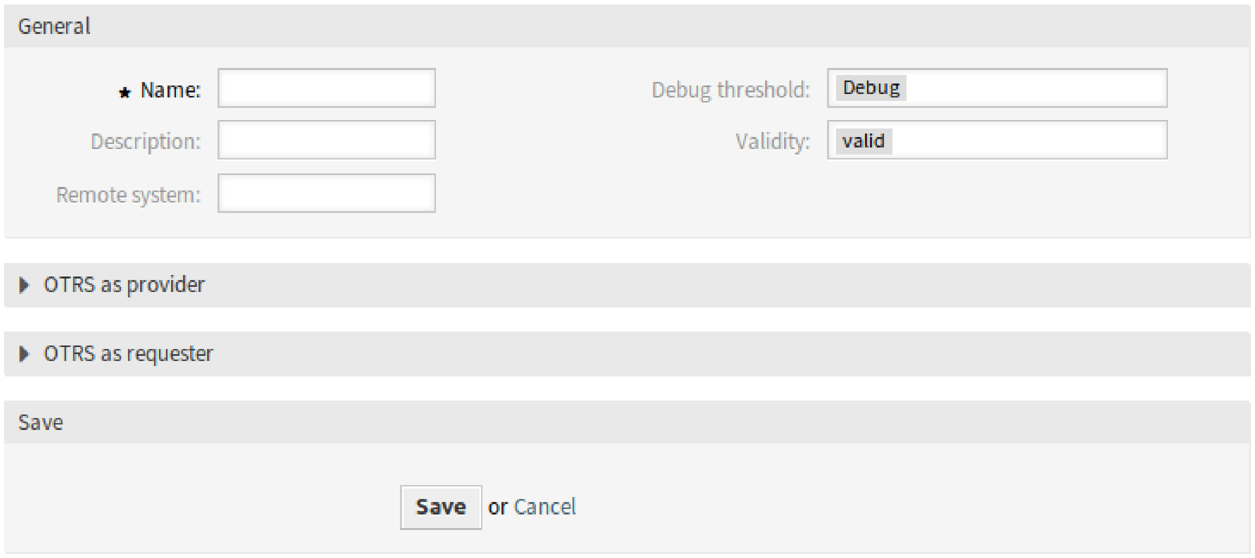


NAME	DESCRIPTION	REMOTE SYSTEM	PROVIDER TRANSPORT	REQUESTER TRANSPORT	VALIDITY
No data found.					

Fig. 141: Web Service Management Screen

To create a web service:

1. Click on the *Add Web Service* button in the left sidebar.
2. Fill in the required fields.
3. Click on the *Save* button.



Debug threshold: Debug

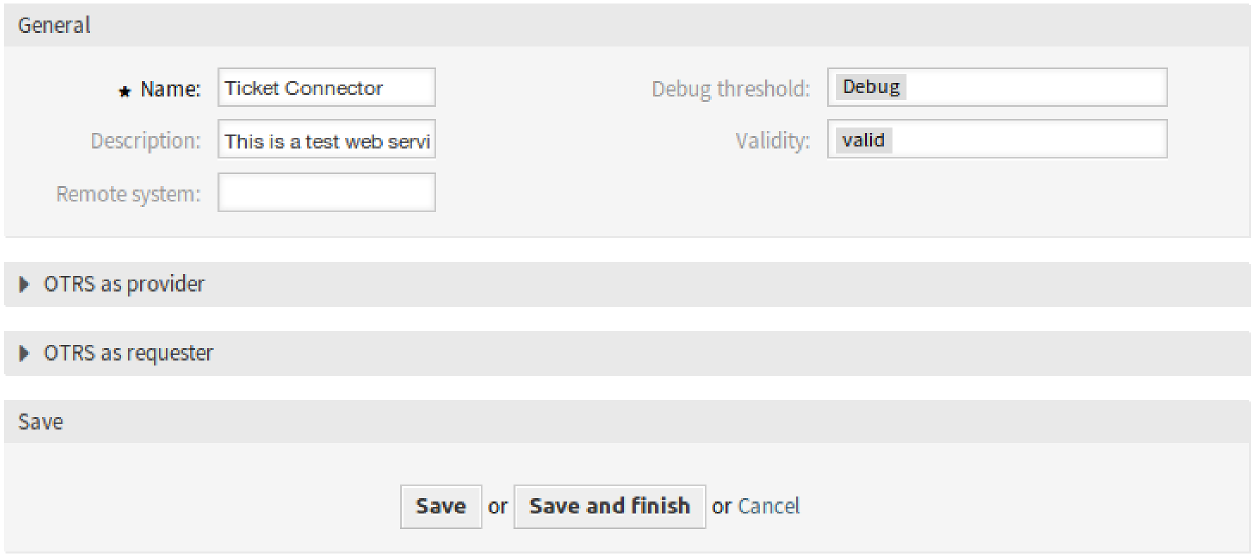
Validity: valid

Fig. 142: Create New Web Service Screen

To edit a web service:

1. Click on a web service in the list of web services.

2. Modify the fields.
3. Click on the *Save* or *Save and finish* button.

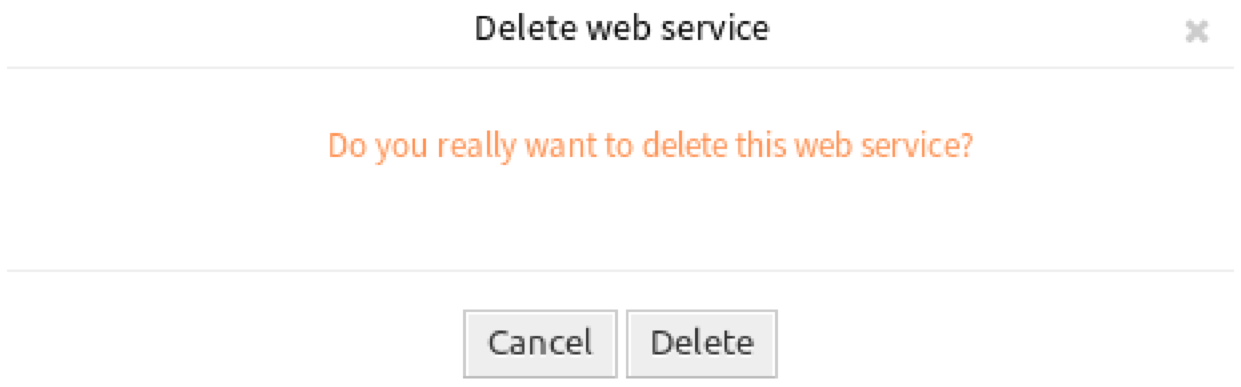


★ Name:	Ticket Connector	Debug threshold:	Debug
Description:	This is a test web servi	Validity:	valid
Remote system:			

Fig. 143: Edit Web Service Screen

To delete a web service:

1. Click on a web service in the list of web services.
2. Click on the *Delete Web Service* button in the left sidebar.
3. Click on the *Delete* button in the confirmation dialog.



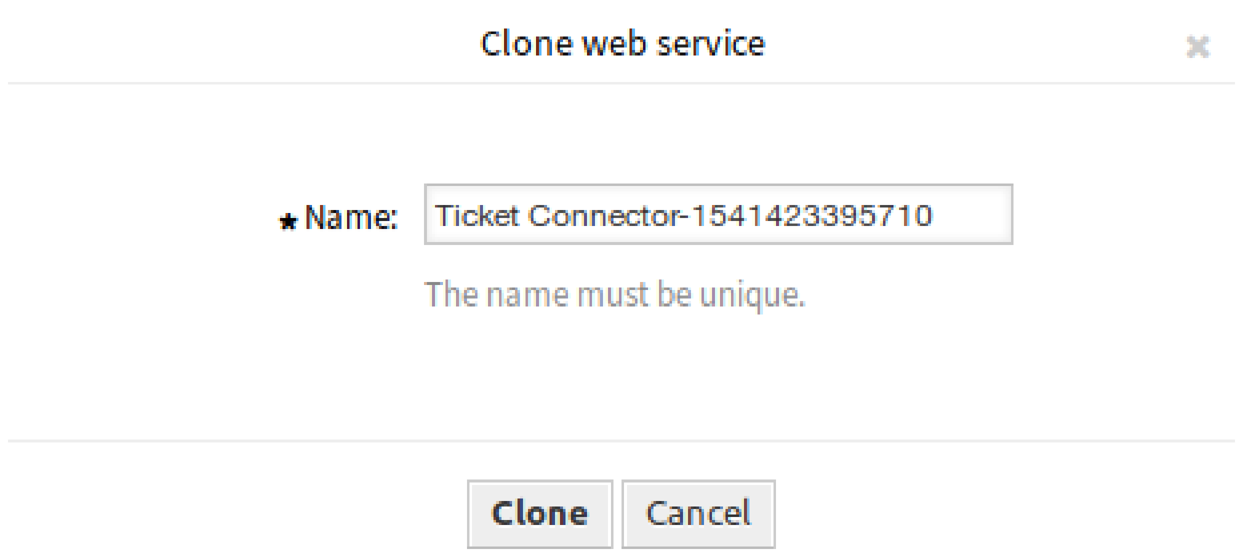
Do you really want to delete this web service?

Fig. 144: Delete Web Service Screen

To clone a web service:

1. Click on a web service in the list of web services.
2. Click on the *Clone Web Service* button in the left sidebar.
3. Enter a new name for the web service.

To export a web service:



Ticket Connector-1541423395710

The name must be unique.

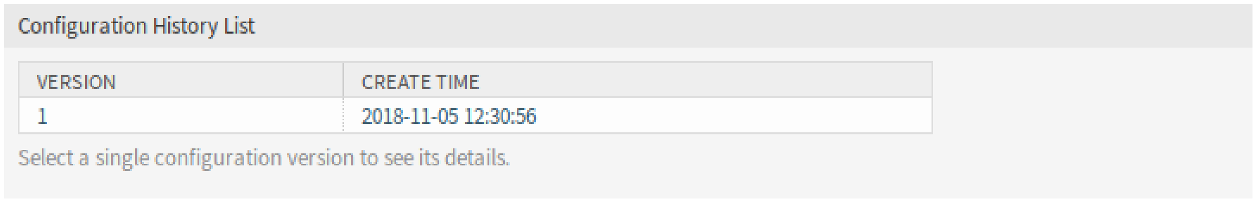
Fig. 145: Clone Web Service Screen

1. Click on a web service in the list of web services.
2. Click on the *Export Web Service* button in the left sidebar.
3. Choose a location in your computer to save the `Export_ACL.yml` file.

Warning: All stored passwords in the web service configuration will be exported in plain text format.

To see the configuration history of a web service:

1. Click on a web service in the list of web services.
2. Click on the *Configuration History* button in the left sidebar.



VERSION	CREATE TIME
1	2018-11-05 12:30:56

Select a single configuration version to see its details.

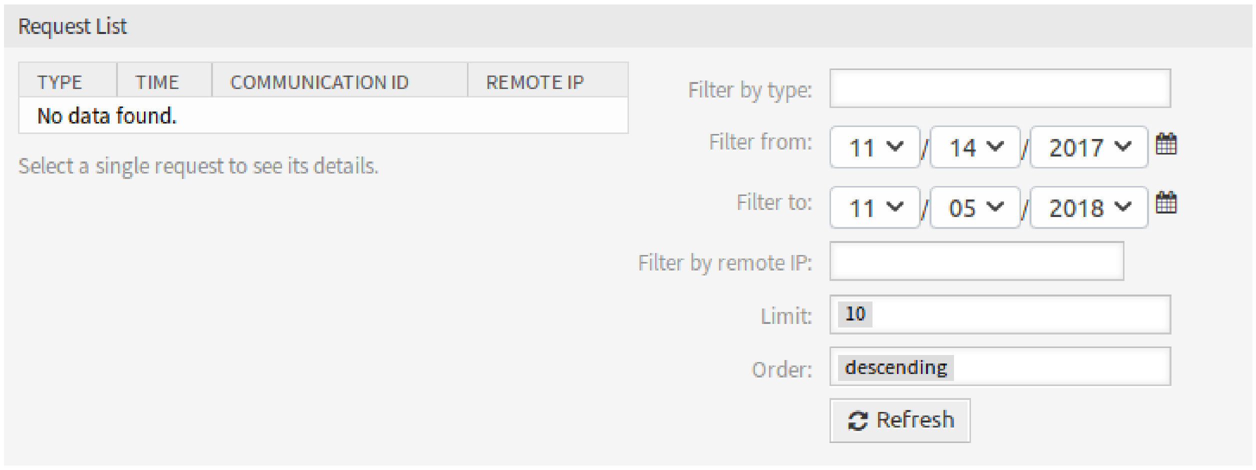
Fig. 146: Web Service Configuration History Screen

To use the debugger for a web service:

1. Click on a web service in the list of web services.
2. Click on the *Debugger* button in the left sidebar.

To import a web service:

1. Click on the *Add Web Service* button in the left sidebar.
2. Click on the *Import Web Service* button in the left sidebar.
3. Click on the *Browse*… button in the dialog.


Request List

TYPE	TIME	COMMUNICATION ID	REMOTE IP
No data found.			

Select a single request to see its details.

Filter by type:	
Filter from:	11 / 14 / 2017
Filter to:	11 / 05 / 2018
Filter by remote IP:	
Limit:	10
Order:	descending

Fig. 147: Web Service Debugger Screen

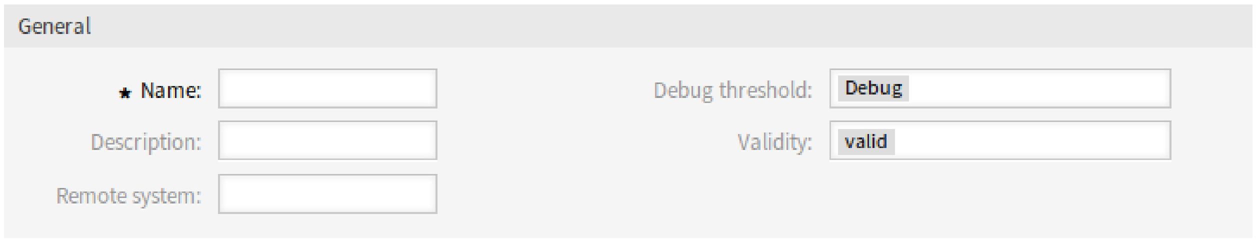
4. Select a previously exported `.yml` file.
5. Add a name for the imported web service (optional). Otherwise the name will be taken from the configuration file name.
6. Click on the *Import* button.

5.5.4 Web Service Settings

The web service configuration needs to be saved on each level. This means that if a setting is changed, links to other, deeper parts of the configuration will be disabled forcing you to save the current configuration level. After saving the disabled links will be re-enabled again allowing you to continue with the configuration.

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

General Web Service Settings


General

⭑ Name:		Debug threshold:	Debug
Description:		Validity:	valid
Remote system:			

Fig. 148: Web Service Settings - General

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

Name *
The name of this resource. Any type of characters can be entered to this field including uppercase letters and spaces. The name will be displayed in the overview table.

Description
Like comment, but longer text can be added here.

Remote system
This field can be used to add a description for the remote system.

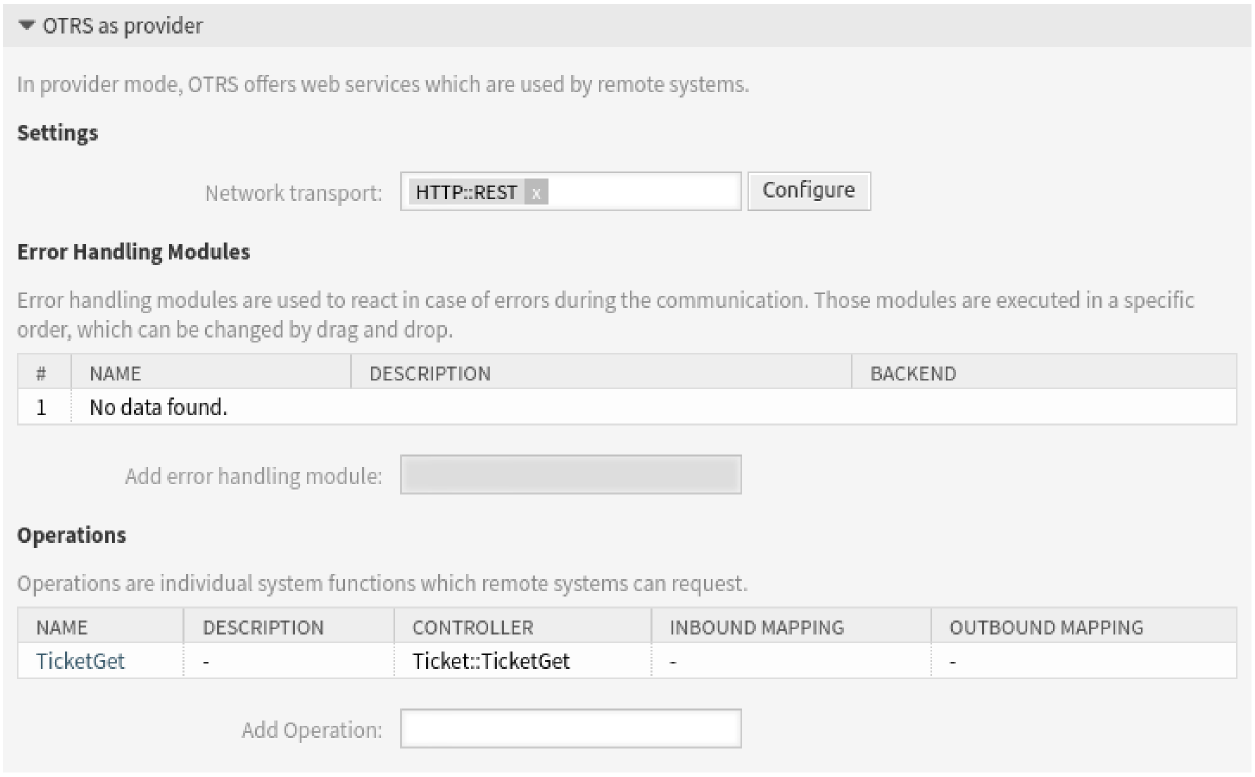
Debug threshold
The default value is *Debug*. When configured in this manner all communication logs are registered in the database. Each subsequent debug threshold value is more restrictive and discards communication logs of lower order than the one set in the system.

Debug threshold levels (from lower to upper):

- Debug
- Info
- Notice
- Error

Validity
Set the validity of this resource. Each resource can be used in OTRS only, if this field is set to *valid*. Setting this field to *invalid* or *invalid-temporarily* will disable the use of the resource.

Provider Web Service Settings



In provider mode, OTRS offers web services which are used by remote systems.

Settings

Network transport: HTTP::REST

Error Handling Modules

Error handling modules are used to react in case of errors during the communication. Those modules are executed in a specific order, which can be changed by drag and drop.

#	NAME	DESCRIPTION	BACKEND
1	No data found.		

Add error handling module:

Operations

Operations are individual system functions which remote systems can request.

NAME	DESCRIPTION	CONTROLLER	INBOUND MAPPING	OUTBOUND MAPPING
TicketGet	-	Ticket::TicketGet	-	-

Add Operation:

Fig. 149: Web Service Settings - OTRS as Provider

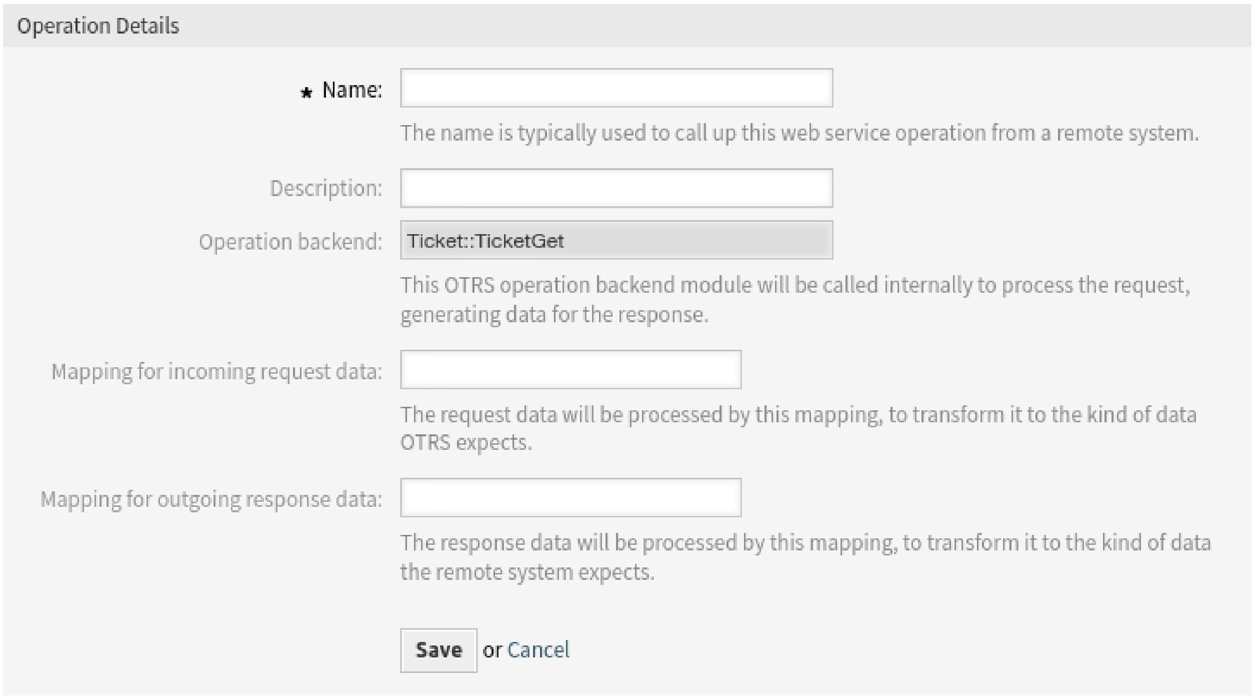
The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

Network transport
Select which network transport would you like to use with the web service. Possible values are *HTTP::REST* and *HTTP::SOAP*.

Note: After selecting the transport method, you have to save the configuration with clicking on the *Save* button. A *Configuration* button will be displayed next to this field.

Configuration
The *Configure* button is visible only, after a network transport was selected and saved. See the configuration for *OTRS as Provider - HTTP::REST* and *OTRS as Provider - HTTP::SOAP* below.

Add Operation
This option is visible only, after a network transport was selected and saved. Selecting an operation will open a new screen for its configuration.


Operation Details

⁎ Name:	
	The name is typically used to call up this web service operation from a remote system.
Description:	
Operation backend:	Ticket::TicketGet
	This OTRS operation backend module will be called internally to process the request, generating data for the response.
Mapping for incoming request data:	
	The request data will be processed by this mapping, to transform it to the kind of data OTRS expects.
Mapping for outgoing response data:	
	The response data will be processed by this mapping, to transform it to the kind of data the remote system expects.

Fig. 150: Web Service Settings - OTRS as Provider - Operation

Each operation requires a valid agent user login name and a password or a session ID. This session ID can be obtained using the `SessionCreate` operation from session connector that is available by default in OTRS.

`Link::LinkAdd`
This operation is used to create a link between two objects.

`Link::LinkDelete`
This operation is used to remove a link between two objects.

`Link::LinkDeleteAll`
This operation is used to remove all link of an object.

`Link::LinkList`
This operation shows all links of an object, optionally restricted by another object, link type and link direction.

Link::PossibleLinkList
This operation shows all possible link types between objects that are registered in the OTRS system.

Link::PossibleObjectsList
This operation shows all objects that can be used for linking.

Link::PossibleTypesList
This operation shows all possible link types between two given objects.

There are more detailed explanation and usage examples in the *Link Object Connector* section of the *Tutorials* chapter.

Due to the nature of the generic interface and the operations included in OTRS an external software is needed to send the requests to the OTRS system.

For testing we recommend the use of:

- OTRS Perl SOAP Requester script: some of the variables in this script such as `URL`, `NameSpace` and `Operation` will need to be changed to match the current web service, the operation to be executed and the data to be sent.
- SoapUI by SMARTBEAR: this is an open source software designed to test web services using SOAP messages.

OTRS as Provider - HTTP::REST

The configuration might be a bit more complicated, as it grows dynamically for each configured operation by adding route mapping for each operation and valid request methods for each operation to the default transport settings.

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

Route mapping for Operation '<OperationName>' *
Define the route that should get mapped to this operation. Variables marked by a `:` will get mapped to the entered name and passed along with the others to the mapping (e.g. `/Ticket/:TicketID`).

In this setting a resource path is set. This path must be defined according to the needs of the web service considering that the path in conjunction with the HTTP request method determines the generic interface operation to be executed.

Path can contain variables in the form of `:<VariableName>`. Each path string that fits on the position of the variable name will be added to the request payload using the variable name defined in this setting. Examples:

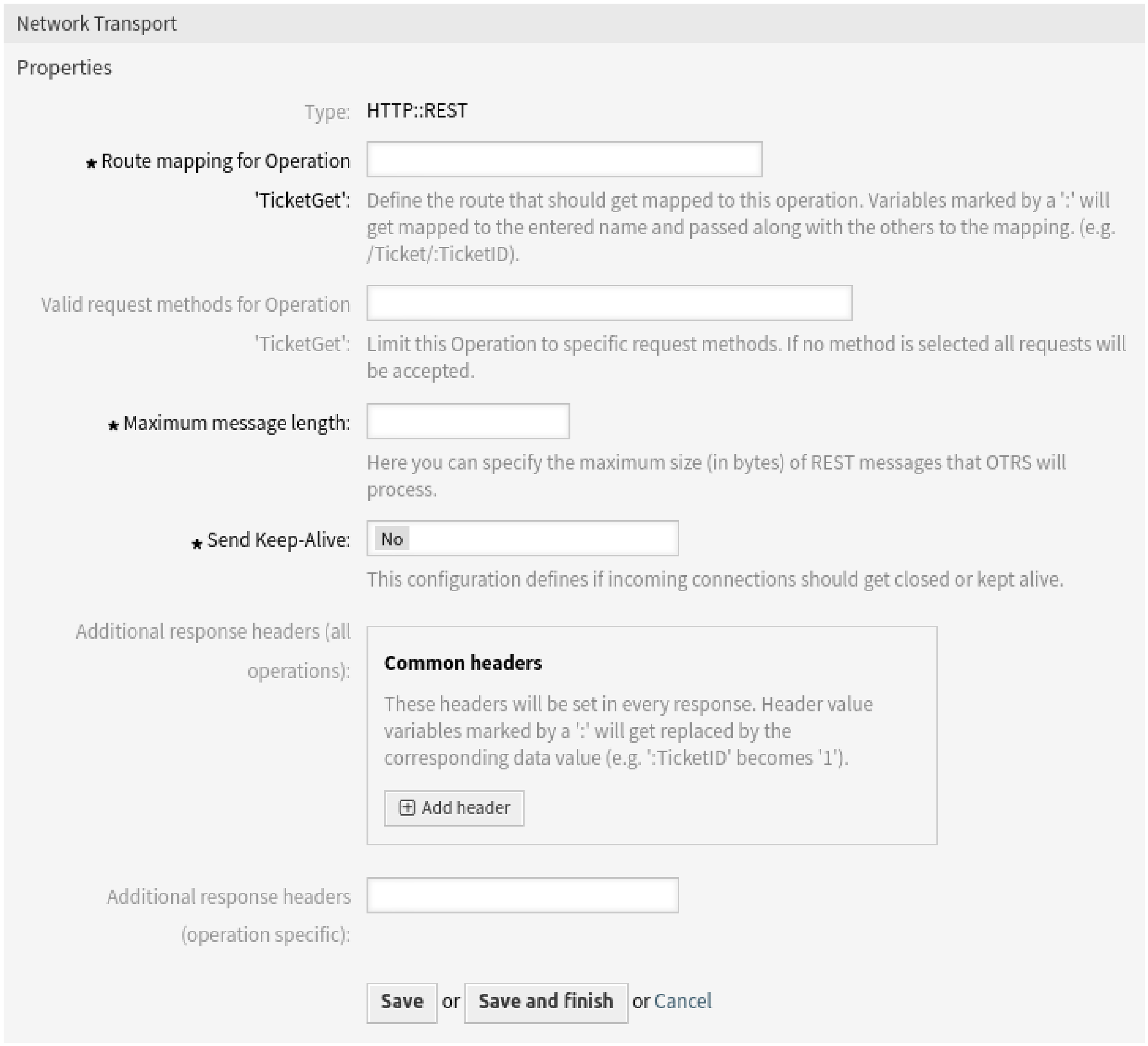
Valid requests for `/Resource` route mapping:

```
https://localhost/otrs/nph-genericinterface.pl/Webservice/Test/Resource
https://localhost/otrs/nph-genericinterface.pl/Webservice/Test/Resource?Param1=One
```

Invalid requests for `/Resource` route mapping:

```
https://localhost/otrs/nph-genericinterface.pl/Webservice/Test/Resource/
https://localhost/otrs/nph-genericinterface.pl/Webservice/Test/Resource/
↪OtherResource
https://localhost/otrs/nph-genericinterface.pl/Webservice/Test/Resource/
↪OtherResource?Param1=One
```

Valid requests for `/Resource/:ID` route mapping:


Network Transport

Properties

Type: HTTP::REST

★ Route mapping for Operation 'TicketGet': Define the route that should get mapped to this operation. Variables marked by a ':' will get mapped to the entered name and passed along with the others to the mapping. (e.g. /Ticket/:TicketID).

Valid request methods for Operation 'TicketGet': Limit this Operation to specific request methods. If no method is selected all requests will be accepted.

★ Maximum message length: Here you can specify the maximum size (in bytes) of REST messages that OTRS will process.

★ Send Keep-Alive: No

This configuration defines if incoming connections should get closed or kept alive.

Additional response headers (all operations):

Common headers

These headers will be set in every response. Header value variables marked by a ':' will get replaced by the corresponding data value (e.g. ':TicketID' becomes '1').

Additional response headers (operation specific):

Fig. 151: Web Service Settings - OTRS as Provider - HTTP::REST

```
https://localhost/otrs/nph-genericinterface.pl/Webservice/Test/Resource/1
https://localhost/otrs/nph-genericinterface.pl/Webservice/Test/Resource/1?
↪Param1=One
```

In both cases `ID` = *1* will be sent to the operation as part of the payload. In the second case also `Param1` = *One* will be added, depending on the HTTP request method other parameters will be added if they come as a JSON string in the request header.

Invalid requests for `/Resource/:ID` route mapping:

```
https://localhost/otrs/nph-genericinterface.pl/Webservice/Test/Resource
https://localhost/otrs/nph-genericinterface.pl/Webservice/Test/Resource?Param1=One
```

Valid requests for `/Resource/OtherResource/:ID/:Color` route mapping:

```
https://localhost/otrs/nph-genericinterface.pl/Webservice/Test/Resource/
↪OtherResource/1/Red
https://localhost/otrs/nph-genericinterface.pl/Webservice/Test/Resource/
↪OtherReosurce/123/Blue?Param1=One
```

In the first example `ID` = *1* and `Color` = *Red*, while in the second `ID` = *123* and `Color` = *Blue*.

Invalid requests for `/Resource/OtherResource/:ID/:Color` route mapping:

```
https://localhost/otrs/nph-genericinterface.pl/Webservice/Test/Resource/1
https://localhost/otrs/nph-genericinterface.pl/Webservice/Test/Resource/
↪OtherResource/1
https://localhost/otrs/nph-genericinterface.pl/Webservice/Test/Resource/
↪OtherResource/1?Param1=One
```

In the first example the part of the path `/OtherResource` is missing as well as the `:Color` variable. In the second example just the `:Color` variable is missing.

Valid request methods for Operation '<OperationName>'
Limit this operation to specific request methods. If no method is selected all requests will be accepted.

The HTTP request methods to determine the operation to use together with the route mapping, possible options: CONNECT, DELETE, GET, HEAD, OPTIONS, PATCH, POST, PUT and TRACE.

Totally different operations can share exactly the same mapping path, but the request method must be unique for each operation, in order to determine correctly the operation to use on each request.

Maximum message length *
Specifies the maximum size (in bytes) for REST messages that OTRS will process.

Send Keep-Alive *
This configuration defines if incoming connections should get closed or kept alive.

Additional response headers (all operations)
Optionally, you may want to define additional response headers for all operations. These may be used to add static header values to every response. Just click on the *Add header* button and fill both header and value fields. There is no limit in number of additional header lines.

Header value variables marked by a `:` will get replaced by the corresponding data value (e.g. `:TicketID` becomes `1`).

Additional response headers (operation specific)
These headers will be set in responses for the selected operation. The purpose of this setting is the same as above.

Header value variables marked by a `:` will get replaced by the corresponding data value (e.g. `:TicketID` becomes `1`).

OTRS as Provider - HTTP::SOAP

It is quite simple to configure the HTTP::SOAP protocol as provider.

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

Check SOAPAction *
Set to *Yes* in order to check the received `SOAPAction` header (if not empty). Set to *No* in order to ignore the received `SOAPAction` header.

SOAPAction scheme *
Select how `SOAPAction` should be constructed. Some web services send a specific construction.

SOAPAction separator *
Character to use as separator between name space and SOAP operation. Usually .Net web services use `/` as separator.

Namespace *
URI to give SOAP methods a context, reducing ambiguities.

Request name scheme *
Select how SOAP request function wrapper should be constructed. `FunctionName` is used as example for actual invoker or operation name. `FreeText` is used as example for actual configured value.

Response name scheme *
Select how SOAP response function wrapper should be constructed. `FunctionName` is used as example for actual invoker or operation name. `FreeText` is used as example for actual configured value.

Maximum message length *
Specifies the maximum size (in bytes) for SOAP messages that OTRS will process.

Additional response headers (all operations)
Optionally, you may want to define additional response headers for all operations. These may be used to add static header values to every response. Just click on the *Add header* button and fill both header and value fields. There is no limit in number of additional header lines.

Header value variables marked by a `:` will get replaced by the corresponding data value (e.g. `:TicketID` becomes `1`).

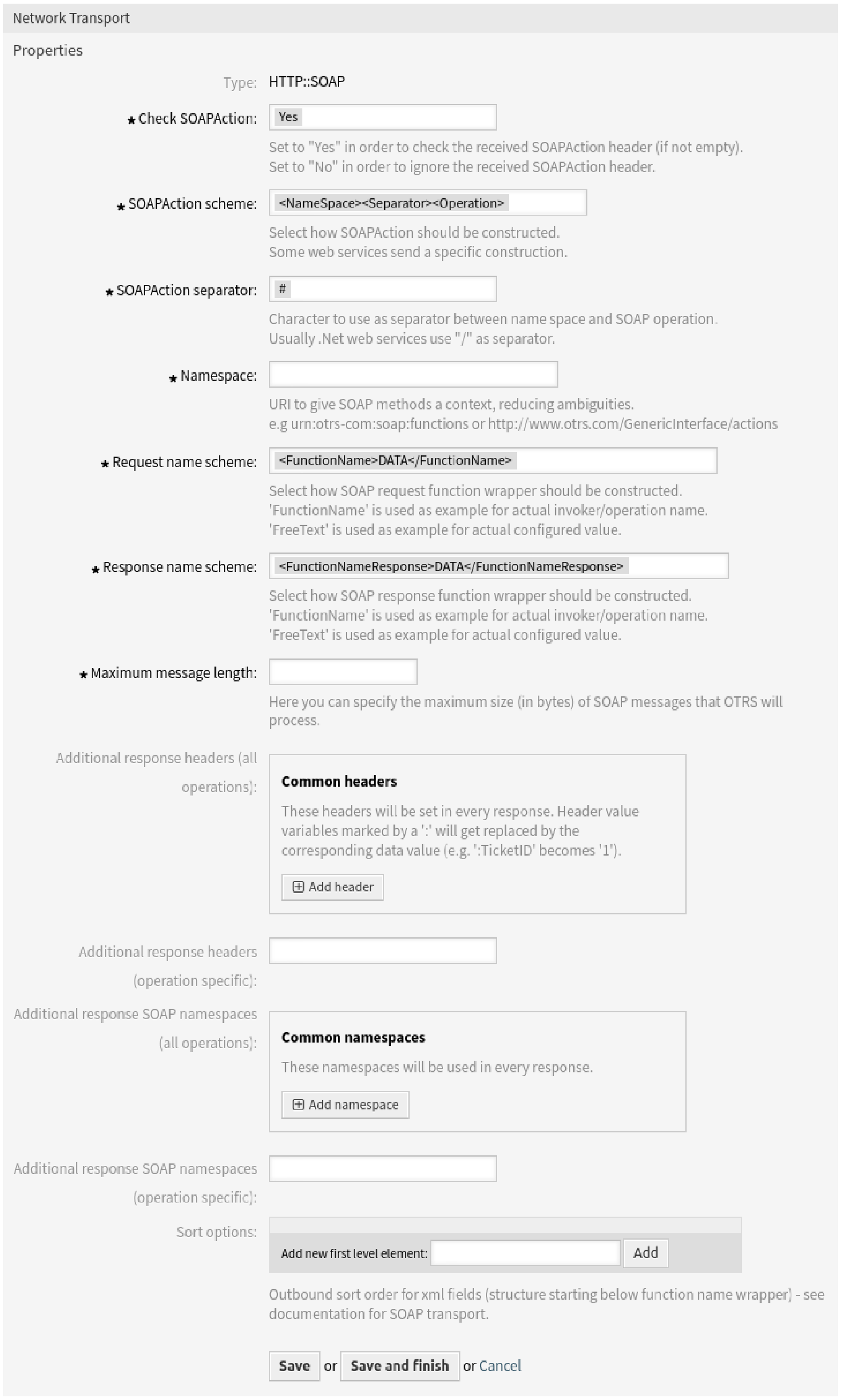
Additional response headers (operation specific)
These headers will be set in responses for the selected operation. The purpose of this setting is the same as above.

Header value variables marked by a `:` will get replaced by the corresponding data value (e.g. `:TicketID` becomes `1`).

Additional response SOAP namespaces (all operations)
These namespaces will be used in every response.

Additional response SOAP namespaces (operation specific)
These namespaces will be used in responses for this specific operation.

Note: Some headers are blocked for safety purposes. If needed, the list of blocked headers can be changed in the following system configuration using the settings:



Network Transport

Properties

Type:	HTTP::SOAP
⋆ Check SOAPAction:	Yes Set to "Yes" in order to check the received SOAPAction header (if not empty). Set to "No" in order to ignore the received SOAPAction header.
⋆ SOAPAction scheme:	<NameSpace><Separator><Operation> Select how SOAPAction should be constructed. Some web services send a specific construction.
⋆ SOAPAction separator:	# Character to use as separator between name space and SOAP operation. Usually .Net web services use "/" as separator.
⋆ Namespace:	URI to give SOAP methods a context, reducing ambiguities. e.g urn:otrs-com:soap:functions or http://www.otrs.com/GenericInterface/actions
⋆ Request name scheme:	<FunctionName>DATA</FunctionName> Select how SOAP request function wrapper should be constructed. 'FunctionName' is used as example for actual invoker/operation name. 'FreeText' is used as example for actual configured value.
⋆ Response name scheme:	<FunctionNameResponse>DATA</FunctionNameResponse> Select how SOAP response function wrapper should be constructed. 'FunctionName' is used as example for actual invoker/operation name. 'FreeText' is used as example for actual configured value.
⋆ Maximum message length:	Here you can specify the maximum size (in bytes) of SOAP messages that OTRS will process.
Additional response headers (all operations):	**Common headers** These headers will be set in every response. Header value variables marked by a ':' will get replaced by the corresponding data value (e.g. ':TicketID' becomes '1'). Add header
Additional response headers (operation specific):	
Additional response SOAP namespaces (all operations):	**Common namespaces** These namespaces will be used in every response. Add namespace
Additional response SOAP namespaces (operation specific):	
Sort options:	Add new first level element: Add Outbound sort order for xml fields (structure starting below function name wrapper) - see documentation for SOAP transport.

Save or Save and finish or Cancel

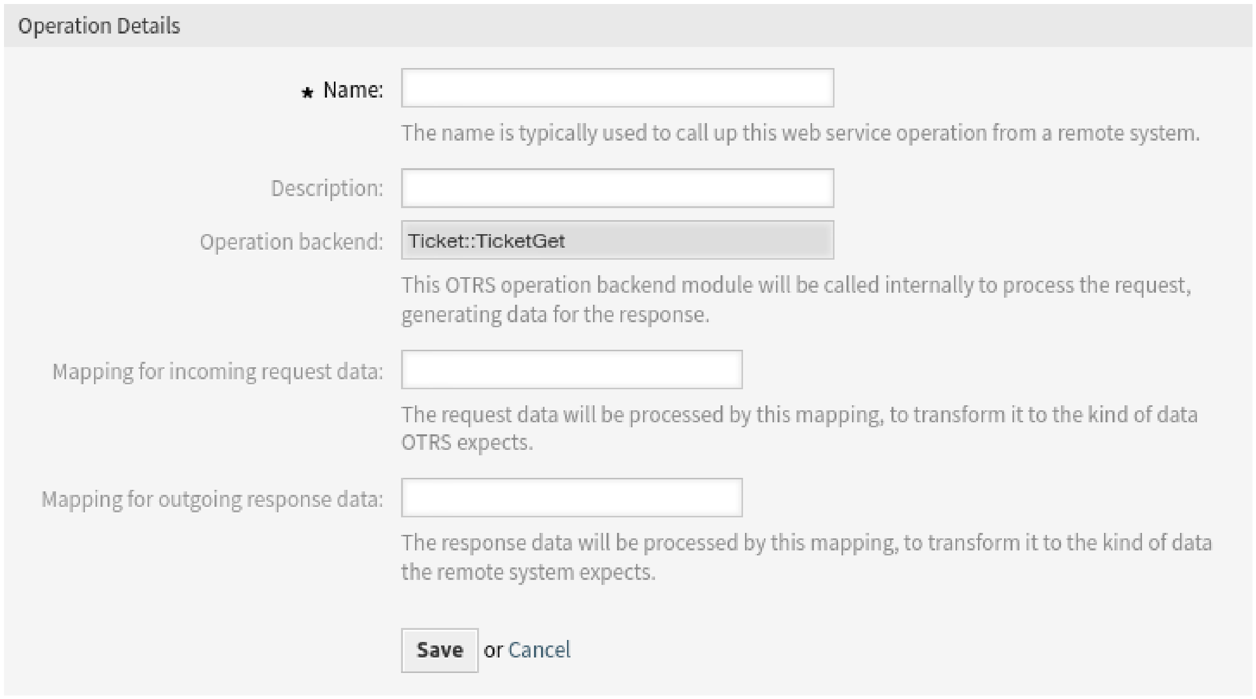
Fig. 152: Web Service Settings - OTRS as Provider - HTTP::SOAP

- `GenericInterface::Invoker::OutboundHeaderBlacklist`
- `GenericInterface::Operation::OutboundHeaderBlacklist`

Sort options
Outbound sort order for XML fields (structure starting below function name wrapper) - see documentation for SOAP transport.

Web Service Operation

The actions that can be performed when you are using OTRS as a provider are called *operations*. Each operation belongs to a controller. Controllers are collections of operations or invokers, normally operations from the same controller need similar settings and share the same configuration dialog. But each operation can have independent configuration dialogs if needed.


Operation Details

⋆ Name:
The name is typically used to call up this web service operation from a remote system.

Description:

Operation backend: Ticket::TicketGet
This OTRS operation backend module will be called internally to process the request, generating data for the response.

Mapping for incoming request data:
The request data will be processed by this mapping, to transform it to the kind of data OTRS expects.

Mapping for outgoing response data:
The response data will be processed by this mapping, to transform it to the kind of data the remote system expects.

Fig. 153: Add Web Service Operation Screen

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

Name *
The name of this resource. Any type of characters can be entered to this field including uppercase letters and spaces. The name will be displayed in the overview table.

Description
Add additional information to this resource. It is recommended to always fill this field as a description of the resource with a full sentence for better clarity, because the description will be also displayed in the overview table.

Operation backend
This OTRS operation back end module will be called internally to process the request, generating data for the response.

The operation back end is pre-populated and cannot be edited. You will see this parameter when you choose the operation on the web service edit screen. The field is only informative.

Mapping for incoming request data
The request data will be processed by this mapping, to transform it to the kind of data OTRS expects.

Mapping for outgoing response data
The response data will be processed by this mapping, to transform it to the kind of data the remote system expects.

Include Ticket Data
Whether to include ticket data in response or not.

Mappings are fields that normally appear on every operation, but other special fields can appear in non default configuration dialogs to fulfill specific needs of the operation.

Normally there are two mapping configuration sections for each operation, one for the incoming data and another one for the outgoing data. You can choose different mapping types (back ends) for each mapping direction, since their configuration is independent from each other and also independent from the operation back end. The normal and most common practice is that the operation uses the same mapping type in both cases (with inverted configuration). The complete mapping configuration is done in a separate screen which depends on the mapping type.

In the left part of the screen on the action column you have the options to go back to web service (discarding all changes since the last save) and delete. If you click on the last one, a dialog will open and ask you if you like to remove the operation. Click on the *Delete* button to confirm the removal of the operation and its configuration or click on the *Cancel* button to close the delete dialog.

Requester Web Service Settings

The network transport configuration for the requester is similar to the configuration for the provider.

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

Network transport
Select which network transport would you like to use with the web service. Possible values are *HTTP::REST* and *HTTP::SOAP*.

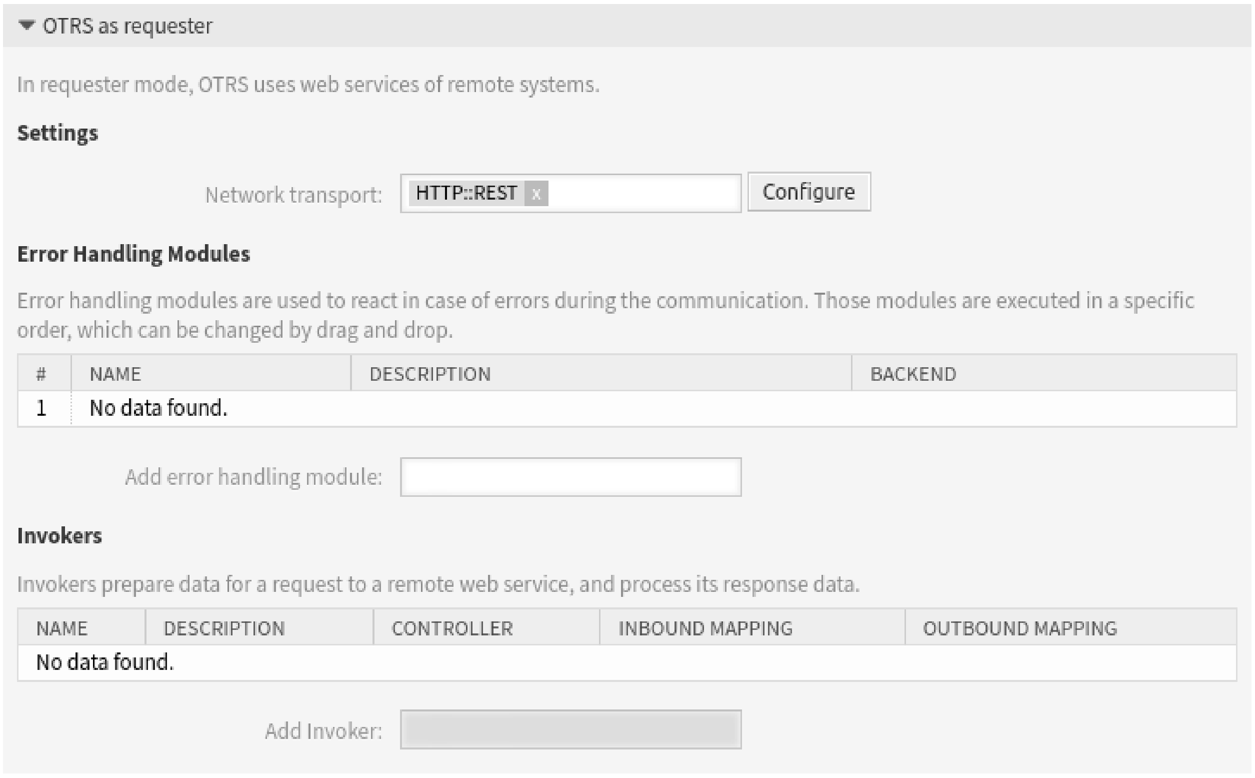
Note: After selecting the transport method, you have to save the configuration with clicking on the *Save* button. A *Configuration* button will be displayed next to this field.

Configuration
The *Configure* button is visible only, after a network transport was selected and saved. See the configuration for *OTRS as Requester - HTTP::REST* and *OTRS as Requester - HTTP::SOAP* below.

It is possible to use both object and array format as a JSON response of the remote system. However, in the case it is an array, system stores it as an object internally, where `ArrayData` is used as a key and a value is an array. Because of that, responded JSON array can be mapped efficiently, but has to be considered as an object described above (key is `ArrayData`, but * can also be used as wildcard).

Add error handling module
This option is visible only, after a network transport was selected and saved. Selecting an error handling module will open a new screen for its configuration.

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.


▾ OTRS as requester

In requester mode, OTRS uses web services of remote systems.

Settings

Network transport: HTTP::REST x | Configure

Error Handling Modules

Error handling modules are used to react in case of errors during the communication. Those modules are executed in a specific order, which can be changed by drag and drop.

#	NAME	DESCRIPTION	BACKEND
1	No data found.		

Add error handling module:

Invokers

Invokers prepare data for a request to a remote web service, and process its response data.

NAME	DESCRIPTION	CONTROLLER	INBOUND MAPPING	OUTBOUND MAPPING
No data found.				

Add Invoker:

Fig. 154: Web Service Settings - OTRS as Requester

Name *
The name of this resource. Any type of characters can be entered to this field including uppercase letters and spaces. The name will be displayed in the overview table.

Description
Add additional information to this resource. It is recommended to always fill this field as a description of the resource with a full sentence for better clarity, because the description will be also displayed in the overview table.

Invoker filter
Only execute error handling module for selected invokers.

Error message content filter
Enter a regular expression to restrict which error messages should cause error handling module execution. Error message subject and data (as seen in the debugger error entry) will be considered for a match.

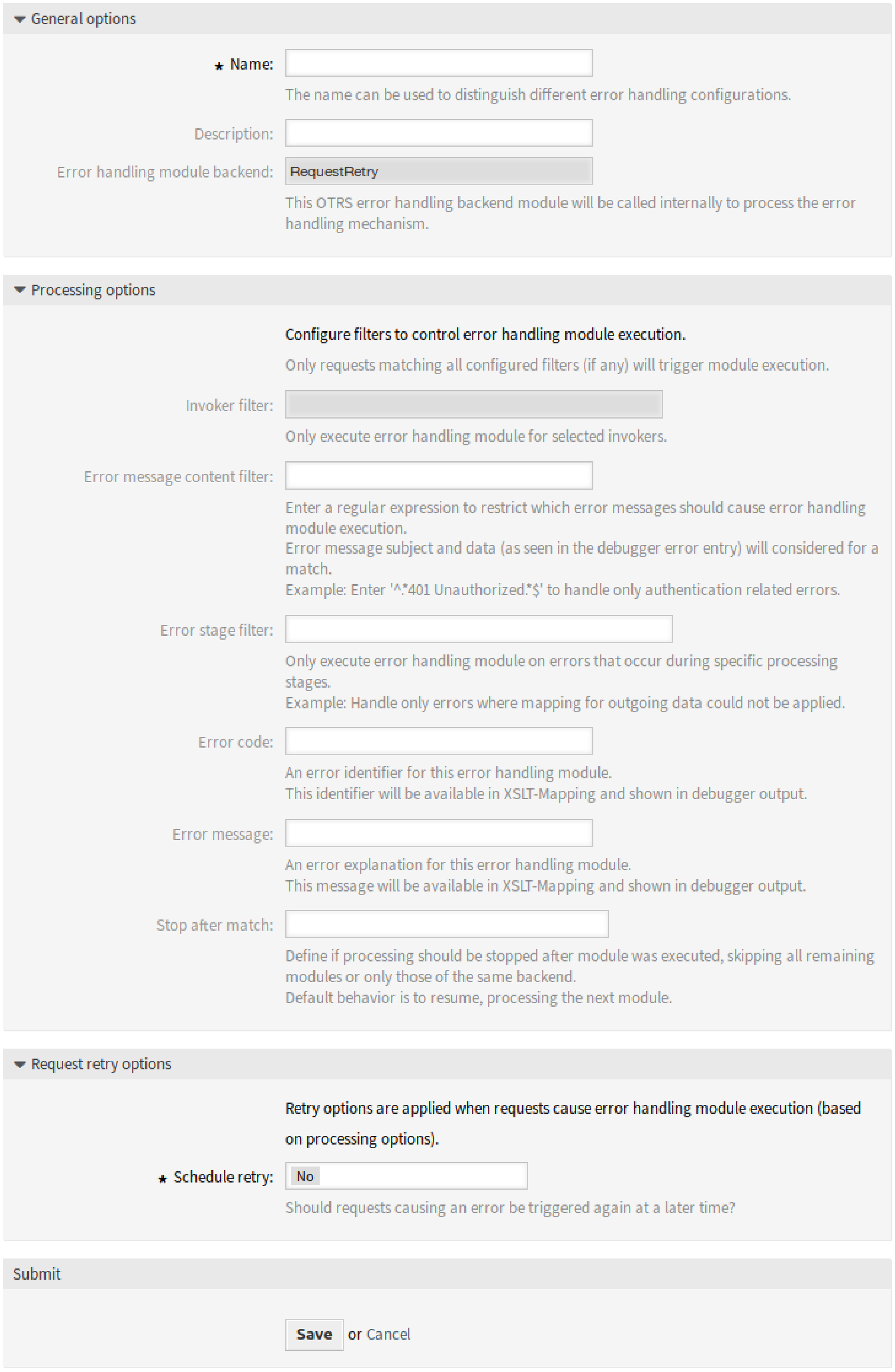
Example: Enter `^.*401 Unauthorized.*$` to handle only authentication related errors.

Error stage filter
Only execute error handling module on errors that occur during specific processing stages.

Example: Handle only errors where mapping for outgoing data could not be applied.

Error code
An error identifier for this error handling module. This identifier will be available in XSLT mapping and shown in debugger output.

Error message
An error explanation for this error handling module. This message will be available in XSLT mapping



General options

★ Name:

The name can be used to distinguish different error handling configurations.

Description:

Error handling module backend: RequestRetry

This OTRS error handling backend module will be called internally to process the error handling mechanism.

Processing options

Configure filters to control error handling module execution.

Only requests matching all configured filters (if any) will trigger module execution.

Invoker filter:

Only execute error handling module for selected invokers.

Error message content filter:

Enter a regular expression to restrict which error messages should cause error handling module execution.
Error message subject and data (as seen in the debugger error entry) will considered for a match.
Example: Enter '^.*401 Unauthorized.*$' to handle only authentication related errors.

Error stage filter:

Only execute error handling module on errors that occur during specific processing stages.
Example: Handle only errors where mapping for outgoing data could not be applied.

Error code:

An error identifier for this error handling module.
This identifier will be available in XSLT-Mapping and shown in debugger output.

Error message:

An error explanation for this error handling module.
This message will be available in XSLT-Mapping and shown in debugger output.

Stop after match:

Define if processing should be stopped after module was executed, skipping all remaining modules or only those of the same backend.
Default behavior is to resume, processing the next module.

Request retry options

Retry options are applied when requests cause error handling module execution (based on processing options).

★ Schedule retry: No

Should requests causing an error be triggered again at a later time?

Submit

Fig. 155: Web Service Settings - OTRS as Provider - Error Handling Module

and shown in debugger output.

Stop after match
Defines if processing should be stopped after module was executed, skipping all remaining modules or only those of the same back end. Default behavior is to resume, processing the next module.

OTRS as Requester - HTTP::REST

In the case of HTTP::REST, this configuration also grows dynamically depending on the configured invokers. Authentication and SSL options are similar to the ones in HTTP::SOAP.

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

Endpoint *
URI of the remote system to indicate specific location for accessing a web service.

Timeout *
Timeout value for requests.

Authentication
An optional authentication mechanism to access the remote system. Select an authentication mechanism from the list and additional fields will appear.

Credential *
Select a credential that has been added in the *Credentials* screen. Click on the *Add credential* button to open the credential management screen.

Certification Authority (CA) Certificate
The full path and name of the certification authority certificate file that validates SSL certificate.

Certification Authority (CA) Directory
The full path of the certification authority directory where the CA certificates are stored in the file system.

Use Proxy Options *
Show or hide proxy options to connect to the remote system.

Controller mapping for Invoker '<InvokerName>' *
The controller that the invoker should send requests to.

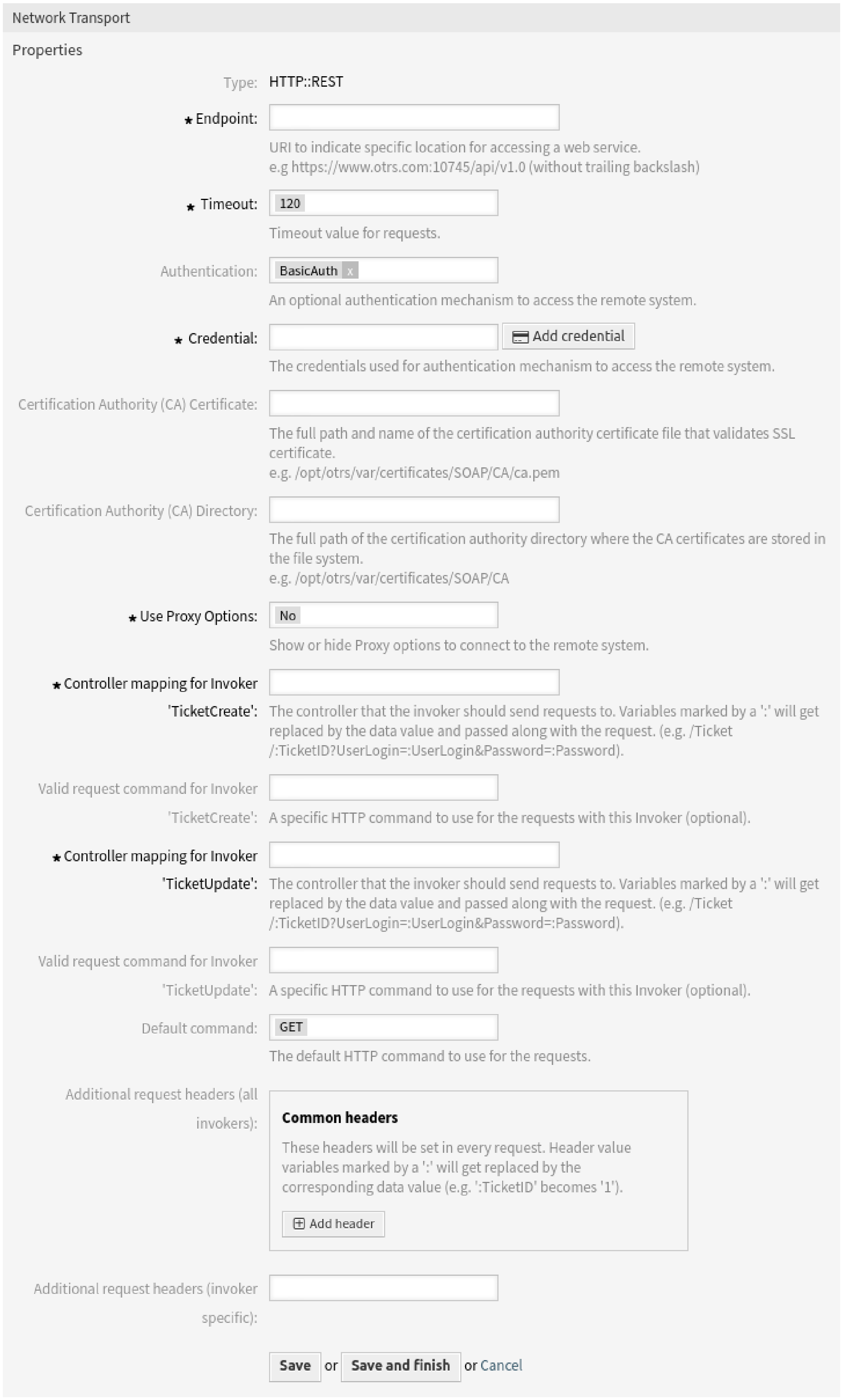
Variables marked by a `:` will get replaced by the data value and passed along with the request (e.g. `/Ticket/:TicketID?UserLogin=:UserLogin&Password=:Password`).

Valid request command for Invoker '<InvokerName>'
A specific HTTP command to use for the requests with this invoker (optional).

Default command
The default HTTP command to use for the requests. Possible options: CONNECT, DELETE, GET, HEAD, OPTIONS, PATCH, POST, PUT and TRACE. If no command is selected, *Default command* is used.

Additional request headers (all invokers)
Optionally, you may want to define additional request headers for all invokers. These may be used to add static header values to every request. Just click on the *Add header* button and fill both header and value fields. There is no limit in number of additional header lines.

Header value variables marked by a `:` will get replaced by the corresponding data value (e.g. `:TicketID` becomes `1`).



Network Transport

Properties

Type:	HTTP::REST
★ Endpoint:	URI to indicate specific location for accessing a web service. e.g https://www.otrs.com:10745/api/v1.0 (without trailing backslash)
★ Timeout:	120 Timeout value for requests.
Authentication:	BasicAuth x An optional authentication mechanism to access the remote system.
★ Credential:	Add credential The credentials used for authentication mechanism to access the remote system.
Certification Authority (CA) Certificate:	The full path and name of the certification authority certificate file that validates SSL certificate. e.g. /opt/otrs/var/certificates/SOAP/CA/ca.pem
Certification Authority (CA) Directory:	The full path of the certification authority directory where the CA certificates are stored in the file system. e.g. /opt/otrs/var/certificates/SOAP/CA
★ Use Proxy Options:	No Show or hide Proxy options to connect to the remote system.
★ Controller mapping for Invoker 'TicketCreate':	The controller that the invoker should send requests to. Variables marked by a ':' will get replaced by the data value and passed along with the request. (e.g. /Ticket /:TicketID?UserLogin=:UserLogin&Password=:Password).
Valid request command for Invoker 'TicketCreate':	A specific HTTP command to use for the requests with this Invoker (optional).
★ Controller mapping for Invoker 'TicketUpdate':	The controller that the invoker should send requests to. Variables marked by a ':' will get replaced by the data value and passed along with the request. (e.g. /Ticket /:TicketID?UserLogin=:UserLogin&Password=:Password).
Valid request command for Invoker 'TicketUpdate':	A specific HTTP command to use for the requests with this Invoker (optional).
Default command:	GET The default HTTP command to use for the requests.
Additional request headers (all invokers):	**Common headers** These headers will be set in every request. Header value variables marked by a ':' will get replaced by the corresponding data value (e.g. ':TicketID' becomes '1'). Add header
Additional request headers (invoker specific):	

Save or Save and finish or Cancel

Fig. 156: Web Service Settings - OTRS as Requester - HTTP::REST

Additional request headers (invoker specific)
These headers will be set in requests for the selected invoker. The purpose of this setting is the same as above.

Header value variables marked by a `:` will get replaced by the corresponding data value (e.g. `:TicketID` becomes `1`).

Note: Some headers are blocked for safety purposes. If needed, the list of blocked headers can be changed in the following system configuration using the settings:

- `GenericInterface::Invoker::OutboundHeaderBlacklist`
- `GenericInterface::Operation::OutboundHeaderBlacklist`

OTRS as Requester - HTTP::SOAP

For the requester HTTP::SOAP network transport there are more fields to be set.

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

Endpoint *
URI of the remote system to indicate specific location for accessing a web service.

Timeout *
Timeout value for requests.

Set SOAPAction *
Set to *Yes* in order to send a filled `SOAPAction` header. Set to *No* in order to send an empty `SOAPAction` header.

SOAPAction scheme *
Select how `SOAPAction` should be constructed. Some web services require a specific construction.

SOAPAction separator *
Character to use as separator between name space and SOAP operation. Usually .Net web services use `/` as separator.

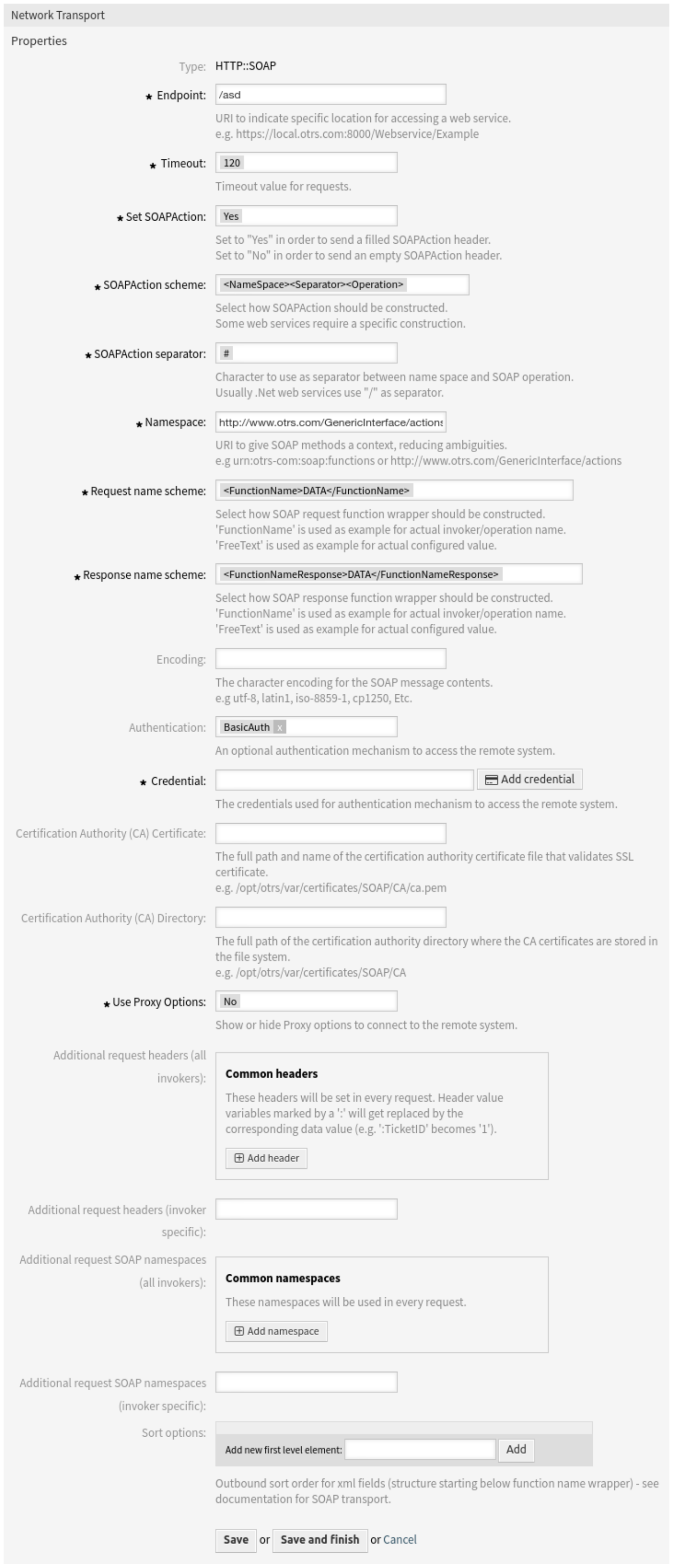
Namespace *
URI to give SOAP methods a context, reducing ambiguities.

Request name scheme *
Select how SOAP request function wrapper should be constructed. `FunctionName` is used as example for actual invoker or operation name. `FreeText` is used as example for actual configured value.

Response name scheme *
Select how SOAP response function wrapper should be constructed. `FunctionName` is used as example for actual invoker or operation name. `FreeText` is used as example for actual configured value.

Encoding
The character encoding for the SOAP message contents.

Authentication
An optional authentication mechanism to access the remote system. Select an authentication mechanism from the list and additional fields will appear.



Network Transport

Properties

Type:	HTTP::SOAP
★ Endpoint:	/asd URI to indicate specific location for accessing a web service. e.g. https://local.otrs.com:8000/Webservice/Example
★ Timeout:	120 Timeout value for requests.
★ Set SOAPAction:	Yes Set to "Yes" in order to send a filled SOAPAction header. Set to "No" in order to send an empty SOAPAction header.
★ SOAPAction scheme:	<NameSpace><Separator><Operation> Select how SOAPAction should be constructed. Some web services require a specific construction.
★ SOAPAction separator:	# Character to use as separator between name space and SOAP operation. Usually .Net web services use "/" as separator.
★ Namespace:	http://www.otrs.com/GenericInterface/actions URI to give SOAP methods a context, reducing ambiguities. e.g urn:otrs-com:soap:functions or http://www.otrs.com/GenericInterface/actions
★ Request name scheme:	<FunctionName>DATA</FunctionName> Select how SOAP request function wrapper should be constructed. 'FunctionName' is used as example for actual invoker/operation name. 'FreeText' is used as example for actual configured value.
★ Response name scheme:	<FunctionNameResponse>DATA</FunctionNameResponse> Select how SOAP response function wrapper should be constructed. 'FunctionName' is used as example for actual invoker/operation name. 'FreeText' is used as example for actual configured value.
Encoding:	The character encoding for the SOAP message contents. e.g utf-8, latin1, iso-8859-1, cp1250, Etc.
Authentication:	BasicAuth An optional authentication mechanism to access the remote system.
★ Credential:	Add credential The credentials used for authentication mechanism to access the remote system.
Certification Authority (CA) Certificate:	The full path and name of the certification authority certificate file that validates SSL certificate. e.g. /opt/otrs/var/certificates/SOAP/CA/ca.pem
Certification Authority (CA) Directory:	The full path of the certification authority directory where the CA certificates are stored in the file system. e.g. /opt/otrs/var/certificates/SOAP/CA
★ Use Proxy Options:	No Show or hide Proxy options to connect to the remote system.
Additional request headers (all invokers):	**Common headers** These headers will be set in every request. Header value variables marked by a ':' will get replaced by the corresponding data value (e.g. ':TicketID' becomes '1'). Add header
Additional request headers (invoker specific):	
Additional request SOAP namespaces (all invokers):	**Common namespaces** These namespaces will be used in every request. Add namespace
Additional request SOAP namespaces (invoker specific):	
Sort options:	Add new first level element: Add Outbound sort order for xml fields (structure starting below function name wrapper) - see documentation for SOAP transport.

Fig. 157: Web Service Settings - OTRS as Requester - HTTP::SOAP

Credential *
Select a credential that has been added in the *Credentials* screen. Click on the *Add credential* button to open the credential management screen.

Certification Authority (CA) Certificate
The full path and name of the certification authority certificate file that validates SSL certificate.

Certification Authority (CA) Directory
The full path of the certification authority directory where the CA certificates are stored in the file system.

Use Proxy Options *
Show or hide proxy options to connect to the remote system.

Additional request headers (all invokers)
Optionally, you may want to define additional request headers for all invokers. These headers will be set in every request.

Header value variables marked by a `:` will get replaced by the corresponding data value (e.g. `:TicketID` becomes `1`).

Additional request headers (invoker specific)
These headers will be set in requests for the selected invoker. The purpose of this setting is the same as above.

Header value variables marked by a `:` will get replaced by the corresponding data value (e.g. `:TicketID` becomes `1`).

Additional request SOAP namespaces (all invokers)
These namespaces will be used in every request.

Additional request SOAP namespaces (invoker specific)
These namespaces will be used in requests for this specific invoker.

Note: Some headers are blocked for safety purposes. If needed, the list of blocked headers can be changed in the following system configuration using the settings:

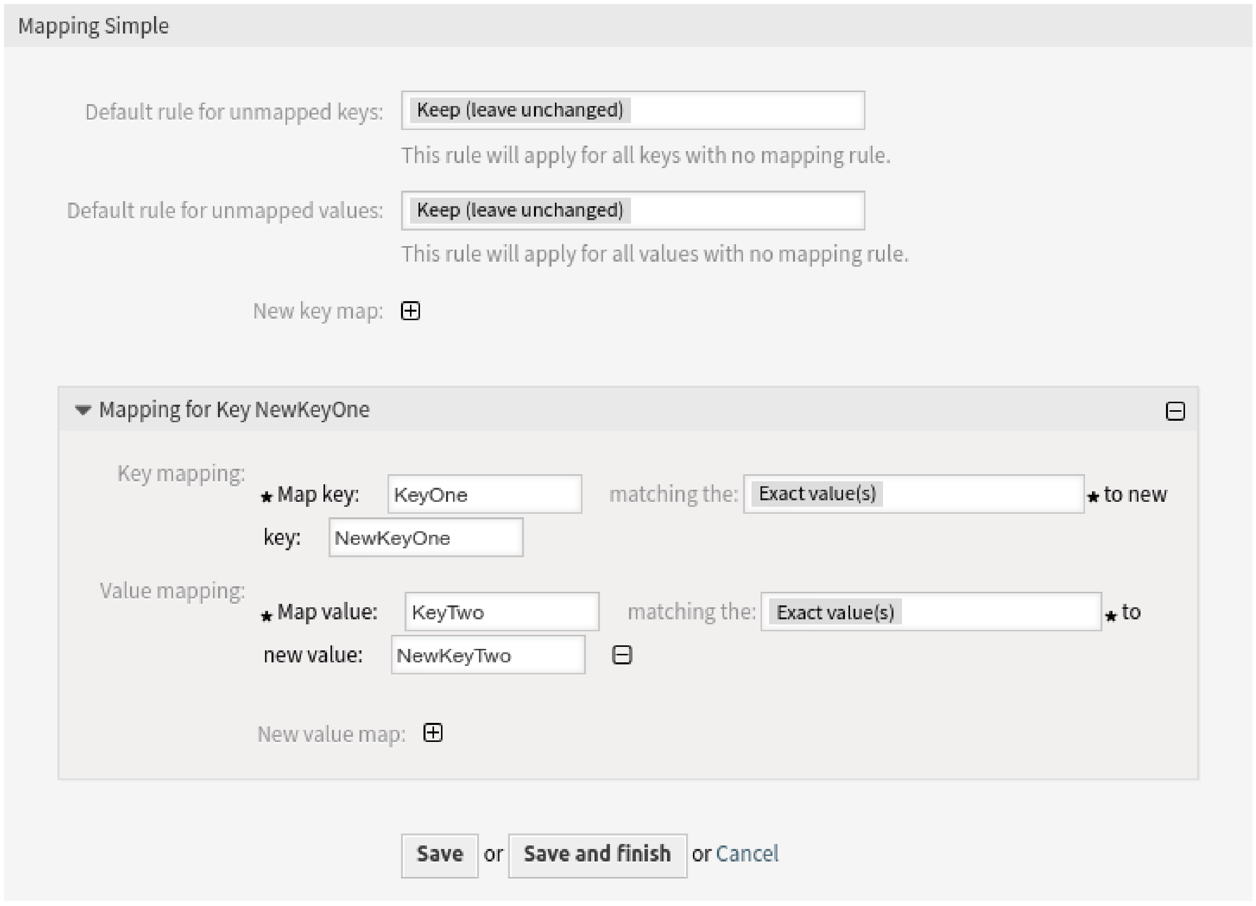
- `GenericInterface::Invoker::OutboundHeaderBlacklist`
- `GenericInterface::Operation::OutboundHeaderBlacklist`

Sort options
Outbound sort order for XML fields (structure starting below function name wrapper) - see documentation for SOAP transport.

Web Service Mapping

There are cases where you need to transform the data from one format to another (map or change data structure), because normally a web service is used to interact with a remote system, that is highly probable that is not another OTRS system and/or could not understand the OTRS data structures and values. In these cases some or all values have to be changed, and sometimes even the names of the values (keys) or even the complete structure, in order to match with the expected data on the other end. To accomplish this task the generic interface mapping layer exists.

Each remote system has it own data structures and it is possible to create new mapping modules for each case (e.g. there is a customized mapping module for *SAP Solution Manager* available as feature), but it is not always necessary. The module `Mapping::Simple` should cover most of the mapping needs.



Mapping Simple

Default rule for unmapped keys: Keep (leave unchanged)

This rule will apply for all keys with no mapping rule.

Default rule for unmapped values: Keep (leave unchanged)

This rule will apply for all values with no mapping rule.

New key map:

Mapping for Key NewKeyOne

Key mapping: * Map key: KeyOne matching the: Exact value(s) * to new key: NewKeyOne

Value mapping: * Map value: KeyTwo matching the: Exact value(s) * to new value: NewKeyTwo

New value map:

Fig. 158: Simple Web Service Mapping

Note: When the `Mapping::Simple` does not cover all mapping needs for a web service, a new mapping module should be created.

This module gives you the opportunity to set default values to map for each key or value for the whole communication data.

At the beginning of the screen you will see a general section where you can set the default rules that will apply for all the unmapped keys and values. There are three options available, these options are listed below:

Keep (leave unchanged)
It does not touch the keys or values in any way.

Ignore (drop key/value pair)
When this is applied to the key it deletes the key and value, because when a key is deleted then in consequence its associated value is deleted too. When this is applied to the value, only the value is deleted, keeping the key, that now will be associated to an empty value.

Map to (use provided value as default)
All keys and/or values without a defined map rule will use this as default. When you select this option a new text field will appear to set this default.

Clicking on the plus button for new key map will display a new box for a single mapping configuration. You can add as many key mappings as needed. Just click on the plus button again and a new mapping box will appear below the existing one. From this mapping boxes you can define a map for a single key, with the next options:

Exact value(s)
The old key string will be changed to a new one if the old key matches exactly.

Regular expression
The key string will be replaced following a regular expression rule.

Pressing the new value map plus button will display a new row for a value map. Here it is also possible to define rules for each value to be mapped with the same options as for the key map (exact value and regular expression). You can add as many values to map as needed, and if you want to delete one of them, just click on the minus button for each mapping value row.

Deleting the complete key mapping section (box) is possible, just push on the minus button located on the up right corner of each box that you want to delete.

If you need to delete a complete mapping configuration, go back to the corresponding operation or invoker screen, look for the mapping direction that you select before and set its value to -, and save the configuration to apply the changes.

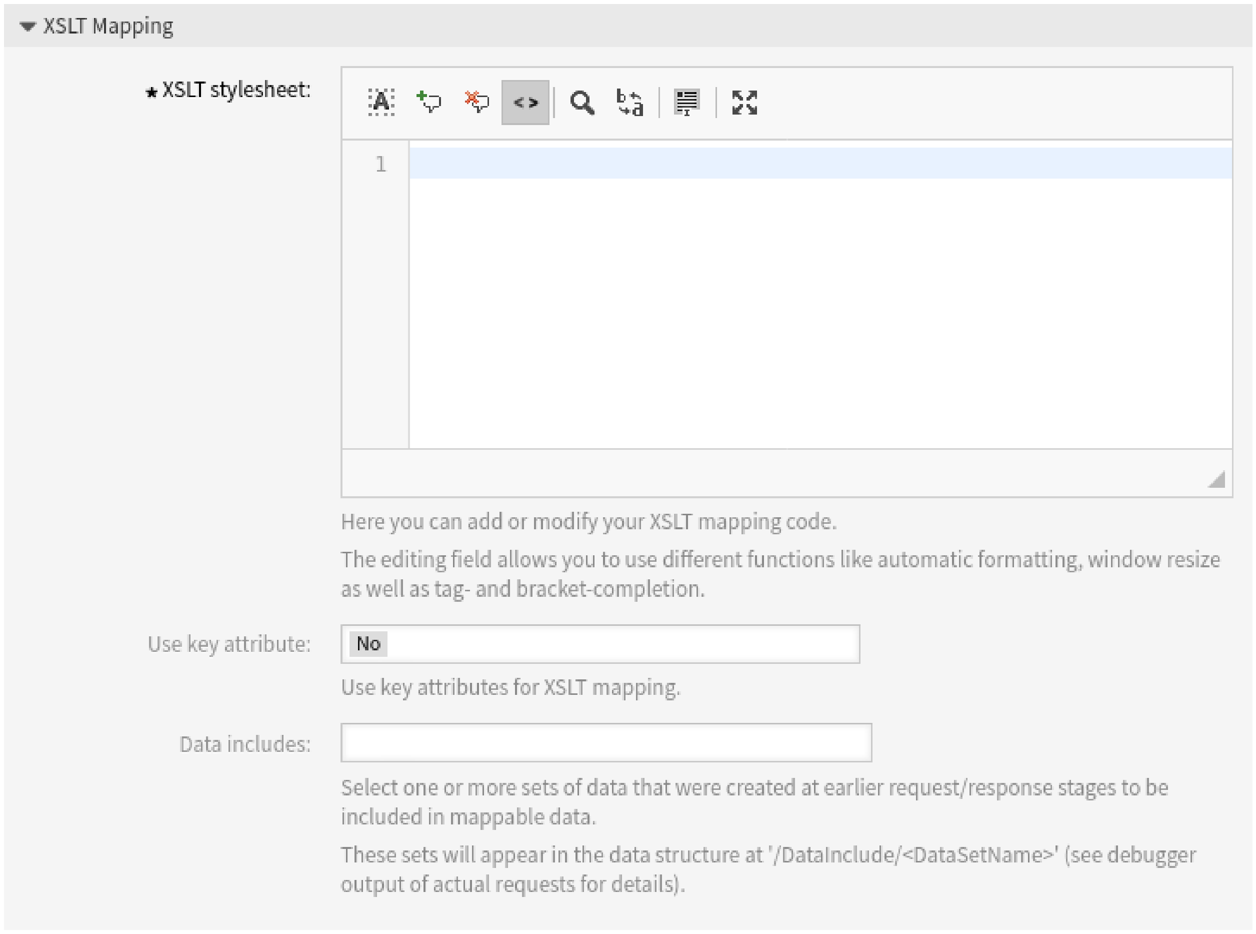
It is possible to define XSLT templates for mapping.

XSLT Mapping

XSLT stylesheet *
Here you can add or modify your XSLT mapping code.

The editing field allows you to use different functions like automatic formatting, window resize as well as tag- and bracket-completion.

Use key attribute
For incoming data this option defines if XML key attributes are converted into a Perl data structure or if they are ignored.



★ XSLT stylesheet:

Here you can add or modify your XSLT mapping code.

The editing field allows you to use different functions like automatic formatting, window resize as well as tag- and bracket-completion.

Use key attribute: No

Use key attributes for XSLT mapping.

Data includes:

Select one or more sets of data that were created at earlier request/response stages to be included in mappable data.

These sets will appear in the data structure at '/DataInclude/<DataSetName>' (see debugger output of actual requests for details).

Fig. 159: XSLT Web Service Incoming Mapping

Example: Incoming XML data

```
<Article>
    <Subject>some subject</Subject>
    <Body>some body</Body>
    <ContentType>text/plain; charset=utf8</ContentType>
    <TimeUnit>1</TimeUnit>
</Article>
<Attachment>
    <Content>someTestData</Content>
    <ContentType>text/plain</ContentType>
    <Filename>test1.txt</Filename>
</Attachment>
<Attachment Content="someTestData" ContentType="text/plain" Filename="test2.txt" /
↪>
```

Resulting Perl data with *Use key attribute* disabled:

```
$VAR1 = {
    Article => {
        Body => 'some body',
        ContentType => 'text/plain; charset=utf8',
        Subject => 'some subject',
        TimeUnit => '1',
    },
    Attachment => [
        {
            Content => 'someTestData',
            ContentType => 'text/plain',
            Filename => 'test1.txt',
        },
        {},
    ],
};
```

Resulting Perl data with *Use key attribute* enabled:

```
$VAR1 = {
    Article => {
        Body => 'some body',
        ContentType => 'text/plain; charset=utf8',
        Subject => 'some subject',
        TimeUnit => '1',
    },
    Attachment => [
        {
            Content => 'someTestData',
            ContentType => 'text/plain',
            Filename => 'test1.txt',
        },
        {
            Content => 'someTestData',
            ContentType => 'text/plain',
            Filename => 'test2.txt',
        },
    ],
};
```

Attribute options

This option must be used in order to use key attributes for outgoing elements. First level options define the elements which should receive key attributes. Second level options define which sub elements should be converted into attributes and attached to the surrounding element. Only two levels of options are considered for key attributes. These will be used for any level of elements in the XML structure (not only the first level).

Please note that sorting of elements in the attribute options is possible but will not affect how key attributes are treated.

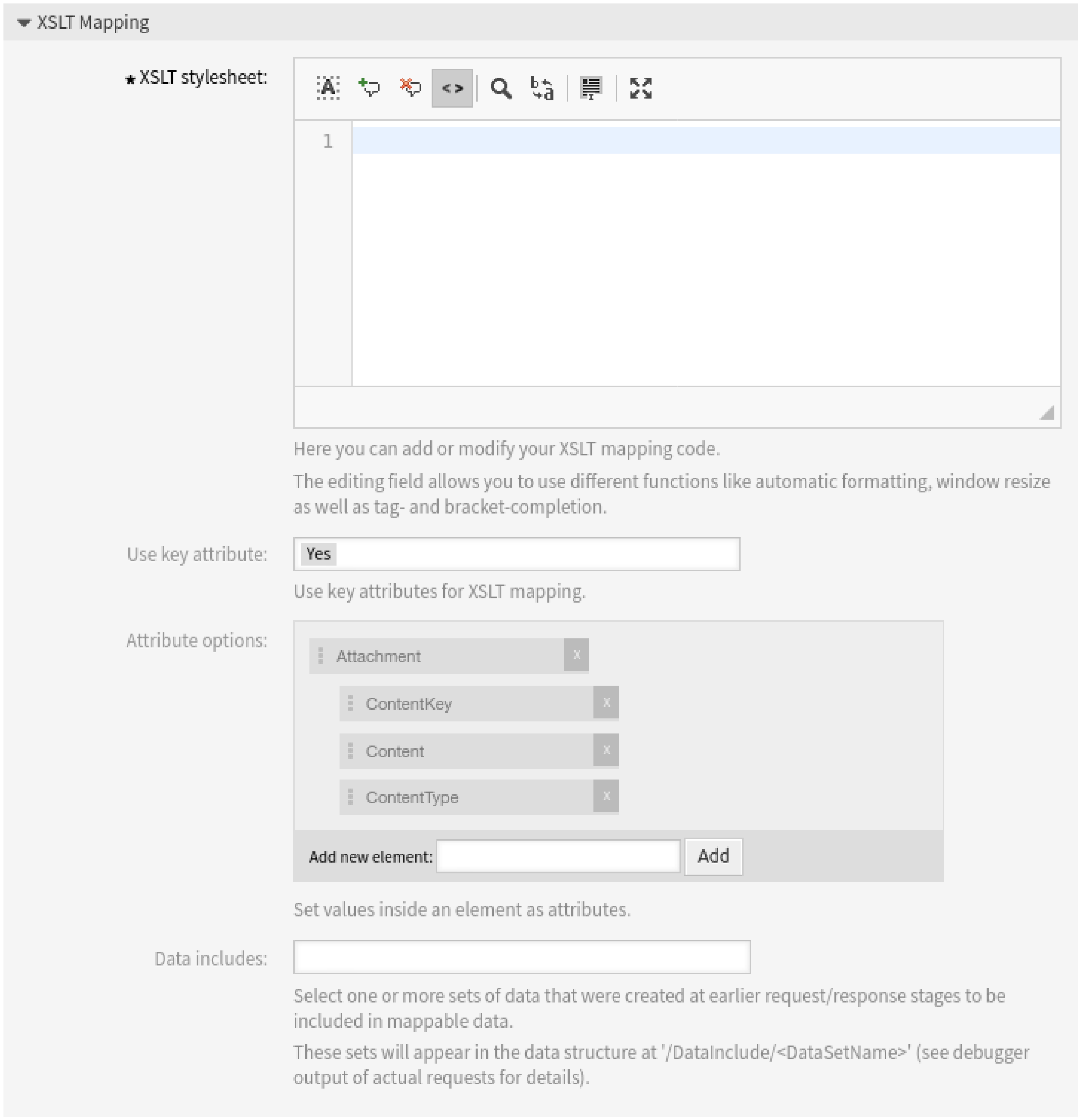
If every sub element of an element is converted into attributes and the element contains a specific `ContentKey` sub element, the content of this sub element will be used as value of the surrounding element. Please see the following example as illustration for these options.

Example: XSLT mapping

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:date="http:/
↪/exslt.org/dates-and-times" version="1.0" extension-element-prefixes="date">
        <xsl:template match="RootElement">
        <xsl:copy>
            <header>
                <messageID>someMessageID</messageID>
                <Attachment>
                    <ContentKey>text1.txt</ContentKey>
                    <Content>someValue</Content>
                    <ContentType>text/plain</ContentType>
                </Attachment>
                <Attachment>
                    <Filename>text2.txt</Filename>
                    <Content>someValue</Content>
                    <ContentType>text/plain</ContentType>
                </Attachment>
                <Attachment>
                    <ContentKey>text3.txt</ContentKey>
                    <Content>someValue</Content>
                    <ContentDisposition>inline</ContentDisposition>
                    <ContentType>text/plain</ContentType>
                </Attachment>
            </header>
            <ticketID>someTicketID</ticketID>
            <returnCode>0</returnCode>
        </xsl:copy>
    </xsl:template>
</xsl:transform>
```

Data includes
Select one or more sets of data that were created at earlier request/response stages to be included in mappable data.

These sets will appear in the data structure at `/DataInclude/<DataSetName>` (see debugger output of actual requests for details).



Here you can add or modify your XSLT mapping code.

The editing field allows you to use different functions like automatic formatting, window resize as well as tag- and bracket-completion.

Use key attribute: Yes

Use key attributes for XSLT mapping.

Attribute options:

- Attachment
  - ContentKey
  - Content
  - ContentType

Set values inside an element as attributes.

Data includes:

Select one or more sets of data that were created at earlier request/response stages to be included in mappable data.

These sets will appear in the data structure at '/DataInclude/<DataSetName>' (see debugger output of actual requests for details).

Fig. 160: XSLT Web Service Outgoing Mapping

Web Service for Dynamic Field

To create a new *dynamic field of type web service* it is necessary to have an already working web service. It requires to have at least one invoker of the type `Generic::PassThrough`. This invoker will be called to fetch the data from the remote server. The original data that it is sent in a request is similar to the following example.

```
{
    DynamicFieldID    => 123,
    DynamicFieldLabel => 'NameX',
    DynamicFieldName  => 'NameX',
    DynamicFieldValue => 'Value',
    Form => {
        # Form fields
        # ...
    },
    Ticket => {
        # Ticket attributes
        # ...
    },
    DynamicField => {
        NameX => 'Value'
        NameY => [ 'Value' ],
    },
    UserID => 123,
},
```

Form
This section contains the fields in the current form in the web browser. This information changes as the screen is filled in.

Ticket
This section (or another source object, e. g. `CustomerUser`) contains the attributes of the object where the dynamic field belongs.

For example in the *New Phone Ticket* screen the section is empty as the ticket is not created yet, but in the *Change Free Fields* screen it contains the information of the current ticket.

DynamicField
This section contains all non empty values of all configured dynamic fields for the current object.

In most cases the data that the remote server requires will be very different from the data provided, so it is highly recommended to use a mapping module for the outgoing data to format it specifically for the remote server call.

The following outgoing mapping example shows an XSLT mapping that discards any data and sets a fixed `UserLogin`, `Password` and `TicketID` (as needed for a `TicketGet` operation).

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:transform
    xmlns:xsl="https://www.w3.org/1999/XSL/Transform"
    xmlns:date="https://exsalt.org/dates-and-times"
    version="1.0"
    extension-element-prefixes="date">

    <xsl:output method="xml" encoding="utf-8" indent="yes" />

    <!-- Don't return unmached tags -->
```

(continues on next page)

(continued from previous page)

```
    <xsl:template match="text()" />

    <!-- Remove empty elements -->
    <xsl:template match="*[not(node())]" />

    <!-- Root template -->
    <xsl:template match="/">
        <RootElement>
            <UserLogin>someuser</UserLogin>
            <Password>somepassword</Password>
            <TicketID>1</TicketID>
        </RootElement>
    </xsl:template>

</xsl:transform>
```

The response from the server can also be very different, so in this case is also very recommended to use a mapping module for the incoming data in order to be able to process the information. The response must be a list of key and value elements.

The following incoming mapping example shows an XSLT mapping that converts the results from a `TicketGet` operation response form the remote server, extracting and formatting the state and queue as needed to be used as options for the dynamic field.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:transform
    xmlns:xsl="https://www.w3.org/1999/XSL/Transform"
    xmlns:date="https://exsalt.org/dates-and-times"
    version="1.0"
    extension-element-prefixes="date">

    <xsl:output method="xml" encoding="utf-8" indent="yes" />

    <!-- Don't return unmached tags -->
    <xsl:template match="text()" />

    <!-- Remove empty elements -->
    <xsl:template match="*[not(node())]" />

    <!-- Root template -->
    <xsl:template match="/">
        <RootElement>
            <xsl:apply-templates />
        </RootElement>
    </xsl:template>

    <xsl:template match="/*/Ticket">
        <PossibleValue>
            <Key>State</Key>
            <Value>
                <xsl:value-of select="/*/Ticket/State" />
            </Value>
        </PossibleValue>
        <PossibleValue>
            <Key>Queue</Key>
            <Value>
```

(continues on next page)

(continued from previous page)

```
                <xsl:value-of select="/*/Ticket/Queue" />
            </Value>
        </PossibleValue>
    </xsl:template>

</xsl:transform>
```

The following web service definition (importable YAML file) can be used for testing the field, but the endpoint must be adapted to match current system. This web service acts as requester and provider and it always returns the state and queue from `TicketID` 1, as possible values to the field.

Note: This example should not be used in conjunction with the development web server.

```
---
Debugger:
  DebugThreshold: debug
  TestMode: '0'
Description: Dynamic Field Web Service Test
FrameworkVersion: 7.0.x git
Provider:
  ErrorHandling: {}
  ErrorHandlingPriority: []
  Operation:
    TicketGet:
      Description: ''
      IncludeTicketData: ''
      MappingInbound: {}
      MappingOutbound: {}
      Type: Ticket::TicketGet
  Transport:
    Config:
      AdditionalHeaders: ~
      MaxLength: '100000000'
      NameSpace: https://www.otrs.org/TicketConnector/
      RequestNameFreeText: ''
      RequestNameScheme: Plain
      ResponseNameFreeText: ''
      ResponseNameScheme: Response
    Type: HTTP::SOAP
RemoteSystem: ''
Requester:
  ErrorHandling: {}
  ErrorHandlingPriority: []
  Invoker:
    TicketGet:
      Description: Get possible values from the other side.
      Events: []
      MappingInbound:
        Config:
          Template: |-
              <?xml version="1.0" encoding="UTF-8"?>
              <!--
              Copyright (C) 2001-2023 OTRS AG, https://otrs.com/
              This software comes with ABSOLUTELY NO WARRANTY. For details, see
```

(continues on next page)

(continued from previous page)

```
the enclosed file COPYING for license information (GPL). If you
did not receive this file, see https://www.gnu.org/licenses/gpl.txt.
-->

<!-- DOCUMENTATION

* Example XML Input *
<RootElement>
    ...
</RootElement>


* Example XML Output *
<RootElement>
    <PossibleValues>
        <Key>???</Key>
        <Value>???</Value>
    </PossibleValues>
    <PossibleValues>
        <Key>???</Key>
        <Value>???</Value>
    </PossibleValues>
    ...
</RootElement>

-->


<xsl:transform
    xmlns:xsl="https://www.w3.org/1999/XSL/Transform"
    xmlns:date="https://exslt.org/dates-and-times"
    version="1.0"
    extension-element-prefixes="date">

    <xsl:output method="xml" encoding="utf-8" indent="yes" />

    <!-- Don't return unmatched tags -->
    <xsl:template match="text()" />

    <!-- Remove empty elements -->
    <xsl:template match="*[not(node())]" />

    <!-- Root template -->
    <xsl:template match="/">
        <RootElement>
            <xsl:apply-templates />
        </RootElement>
    </xsl:template>

    <xsl:template match="/*/Ticket">
        <PossibleValue>
            <Key>State</Key>
            <Value><xsl:value-of select="/*/Ticket/State" /></Value>
        </PossibleValue>
        <PossibleValue>
            <Key>Queue</Key>
```

(continues on next page)

(continued from previous page)

```
                          <Value><xsl:value-of select="/*/Ticket/Queue" /></Value>
                      </PossibleValue>
                  </xsl:template>

              </xsl:transform>
        Type: XSLT
      MappingOutbound:
        Config:
          Template: |-
              <?xml version="1.0" encoding="UTF-8"?>
              <!--
              Copyright (C) 2001-2023 OTRS AG, https://otrs.com/

              This software comes with ABSOLUTELY NO WARRANTY. For details, see
              the enclosed file COPYING for license information (GPL). If you
              did not receive this file, see https://www.gnu.org/licenses/gpl.txt.
              -->

              <!-- DOCUMENTATION

              * Example XML Input *
              <RootElement>
                 ...
              </RootElement>


              * Example XML Output *
              <RootElement>
                  <PossibleValues>
                      <Key>???</Key>
                      <Value>???</Value>
                  </PossibleValues>
                  <PossibleValues>
                      <Key>???</Key>
                      <Value>???</Value>
                  </PossibleValues>
                  ...
              </RootElement>

              -->

              <xsl:transform
                  xmlns:xsl="https://www.w3.org/1999/XSL/Transform"
                  xmlns:date="https://exslt.org/dates-and-times"
                  version="1.0"
                  extension-element-prefixes="date">
                  <xsl:output method="xml" encoding="utf-8" indent="yes" />

                  <!-- Don't return unmatched tags -->
                  <xsl:template match="text()" />

                  <!-- Remove empty elements -->
                  <xsl:template match="*[not(node())]" />

                  <!-- Root template -->
                  <xsl:template match="/">
```

(continues on next page)

(continued from previous page)

```
                      <RootElement>
                          <UserLogin>someuser</UserLogin>
                          <Password>somepassword</Password>
                          <TicketID>1</TicketID>
                      </RootElement>
                  </xsl:template>

              </xsl:transform>
        Type: XSLT
      Type: Generic::PassThrough
  Transport:
    Config:
      Encoding: ''
      Endpoint: https://localhost/otrs/nph-genericinterface.pl/Webservice/
↪GenericConfigItemConnectorSOAP
      NameSpace: https://www.otrs.org/TicketConnector/
      RequestNameFreeText: ''
      RequestNameScheme: Plain
      ResponseNameFreeText: ''
      ResponseNameScheme: Response
      SOAPAction: Yes
      SOAPActionSeparator: '#'
      SSL:
        SSLProxy: ''
        SSLProxyPassword: ''
        SSLProxyUser: ''
    Type: HTTP::SOAP
  UseMappedData: '1'
```

CHAPTER

SIX

AGENT INTERFACE

Completely new user interface has been developed for agents. This setting makes it possible to edit the agent styles that are selectable for all agents.

6.1 Styles

Use this screen to manage styles and choose style variants for use in agent interface. A fresh OTRS installation already contains some variants by default. The style management screen is available in the *Styles* module of the *Agent Interface* group.

The management screen consists of two widget. In the *Edit Styles* widget can be selected the styles for editing. The *Defaults* widget defines, which variant will be used as default for the agent interface. The agents can override this setting and they can select a different variant in they personal preferences, but they can not edit the styles.

6.1.1 Manage Styles

Styles are grouped into four categories: *Bright*, *High Contrast Bright*, *Dark* and *High Contrast Dark*.

To edit a style, select a category first. The edit screen will be opened for the selected category.

The *Edit Layout* section is the same for all categories.

The following settings are available when adding or editing this resource.

Enable
  Select whether the style is available for agents.

Header Logo
  The logo is a small image that is displayed in the header of all pages.

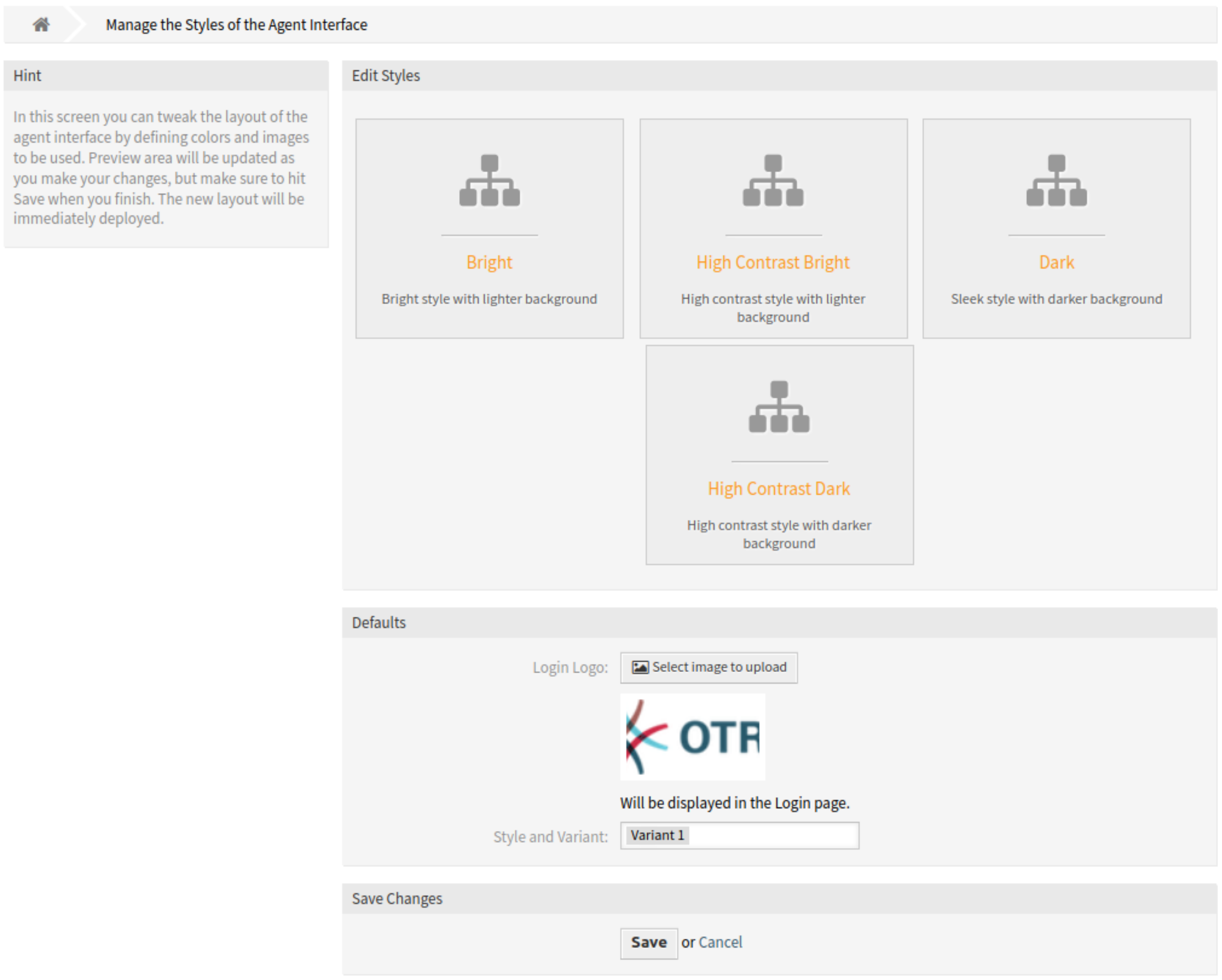
  To change the logo, click on the *Select image to upload* button, and select a new logo image. Recommended file format is PNG.

Favicon
  The favorite icon is an icon that is displayed in the URL bar of the web browser.

  To change the favorite icon, click on the *Select image to upload* button, and select a new icon. This is usually a 16×16 pixel image in PNG or ICO format.

Some categories contain different variants. Each variant can be customize in the same manner.

The following settings are available when adding or editing this resource.


Manage the Styles of the Agent Interface

Hint

In this screen you can tweak the layout of the agent interface by defining colors and images to be used. Preview area will be updated as you make your changes, but make sure to hit Save when you finish. The new layout will be immediately deployed.

Edit Styles

Bright

Bright style with lighter background

High Contrast Bright

High contrast style with lighter background

Dark

Sleek style with darker background

High Contrast Dark

High contrast style with darker background

Defaults

Login Logo: Select image to upload

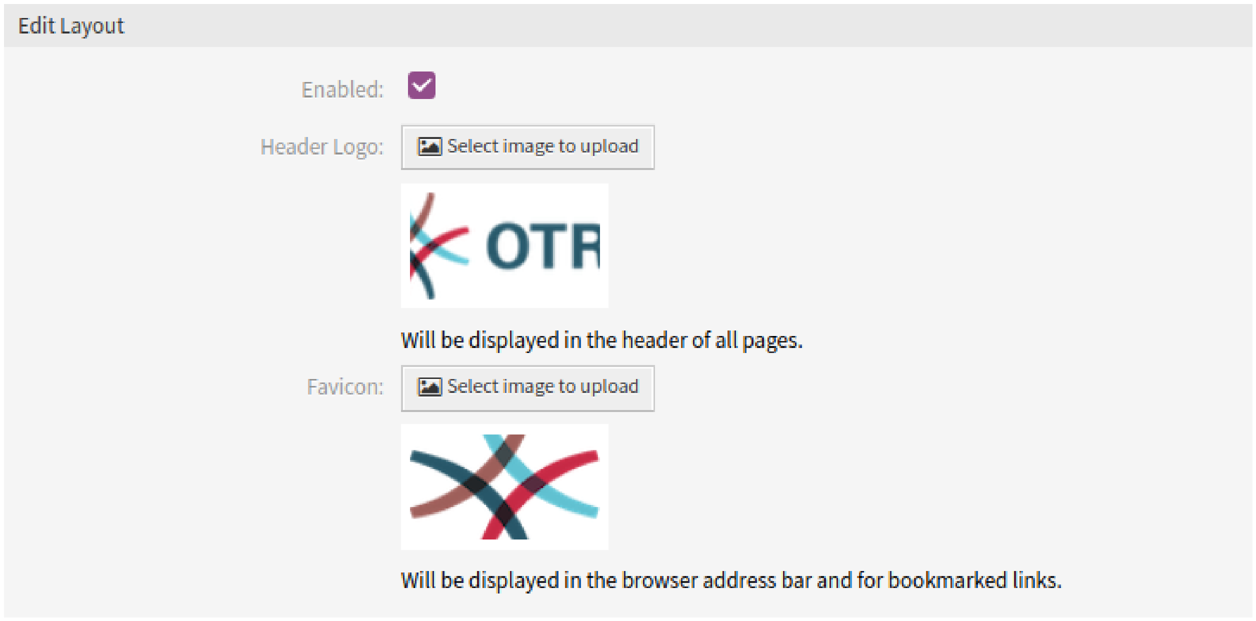
Will be displayed in the Login page.

Style and Variant: Variant 1

Save Changes

Save or Cancel

Fig. 1: Agent Style Management Screen



Will be displayed in the header of all pages.

Will be displayed in the browser address bar and for bookmarked links.

Fig. 2: Agent Style Layout Edit Screen

Enable
Select whether the variant is available for agents.

Name
The name of this resource. Any type of characters can be entered to this field including uppercase letters and spaces.

Primary Color
This color will be used as the background color of primary buttons and the color of textual links and hovered icons.

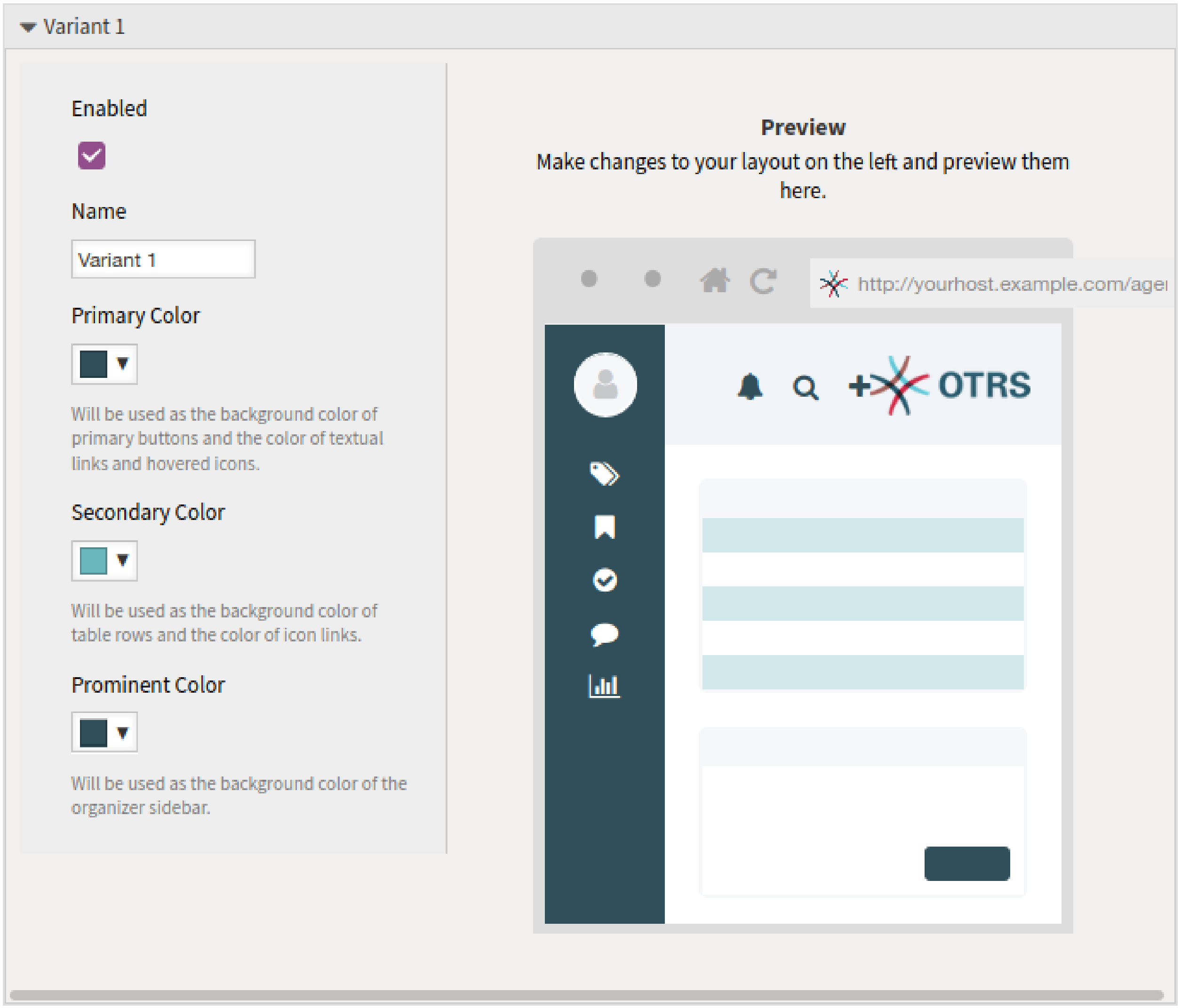
To change the primary color, just select a new color from the color palette. You can chose from the pre-selected colors or define other colors by choosing it from the color selector or typing the hexadecimal value. The new color will be displayed in the preview widget immediately.

Secondary Color
This color will be used as the background color of table rows and the color of icon links.

To change the secondary color, just select a new color from the color palette. You can chose from the pre-selected colors or define other colors by choosing it from the color selector or typing the hexadecimal value. The new color will be displayed in the preview widget immediately.

Prominent Color
This color will be used as the background color of the organizer sidebar.

To change the prominent color, just select a new color from the color palette. You can chose from the pre-selected colors or define other colors by choosing it from the color selector or typing the hexadecimal value. The new color will be displayed in the preview widget immediately.



Variant 1

Enabled ✓

Name: Variant 1

Primary Color

Will be used as the background color of primary buttons and the color of textual links and hovered icons.

Secondary Color

Will be used as the background color of table rows and the color of icon links.

Prominent Color

Will be used as the background color of the organizer sidebar.

Preview

Make changes to your layout on the left and preview them here.

Fig. 3: Agent Style Variant

CHAPTER

SEVEN

EXTERNAL INTERFACE

The need to transport valuable information about service to users is a major one for a service desk. Not always is it possible or plausible to do this via a corporate internet. The customers of a service desk need a one-stop solution for all their service needs.

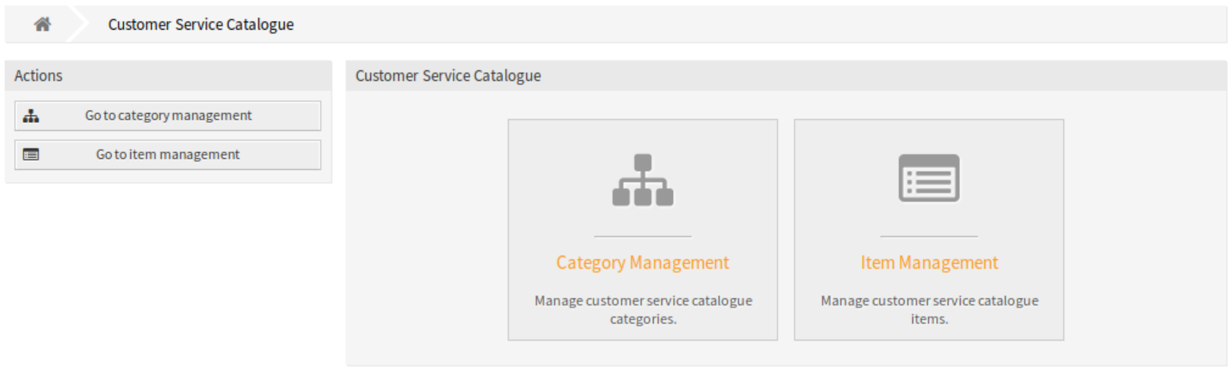
OTRS provides access to create and manage demands, as well as receive valuable information such as the service catalog and knowledge base.

The following chapter describes the administration tools needed to implement corporate identity and manage language-based content.

7.1 Customer Service Catalogue

Use this screen to add categories and items for use in external interface. A fresh OTRS installation doesn't contain any categories or items by default. The catalogue management screen is available in the *Customer Service Catalogue* module of the *External Interface* group.

This module consists of two management screens: a category management screen and an item management screen.


Customer Service Catalogue
Actions
Go to category management
Go to item management
Customer Service Catalogue
Category Management
Item Management

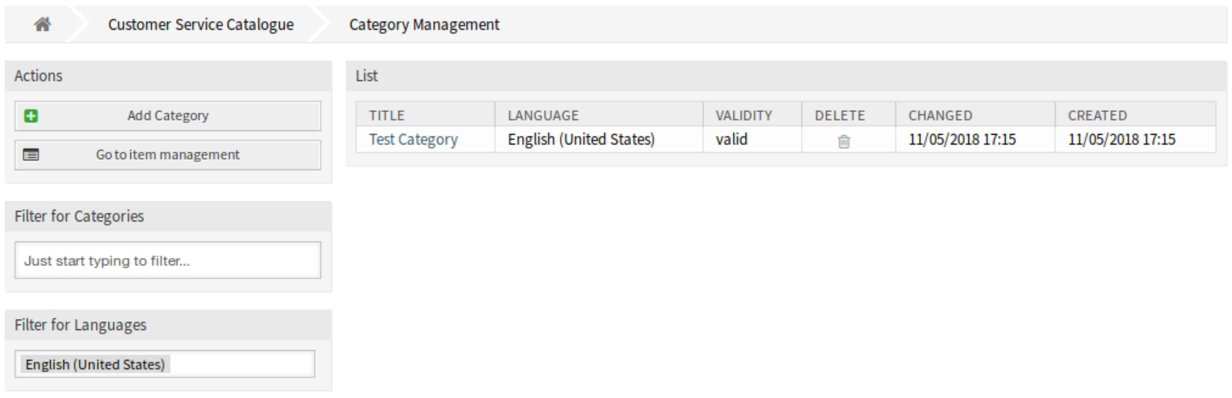
Manage customer service catalogue categories.

Manage customer service catalogue items.

Fig. 1: Customer Service Catalogue Management Screen

7.1.1 Manage Categories

Use this screen to add categories to collect the same items into groups. The *Category Management* screen in available via the *Go to category management* button or via the *Category Management* module.

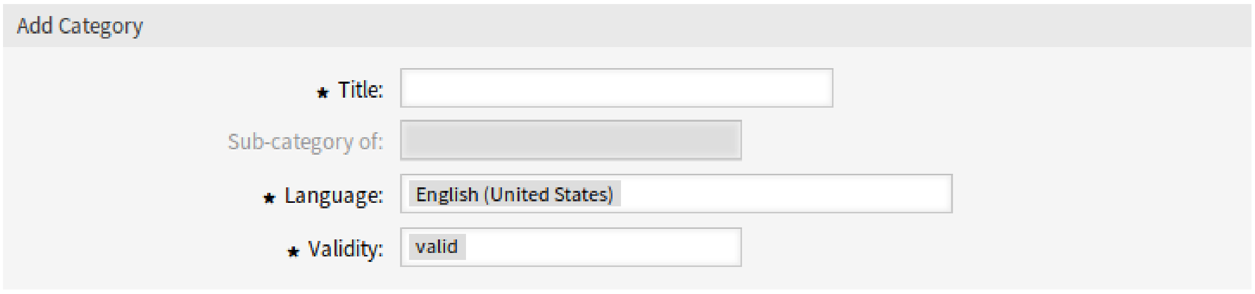


TITLE	LANGUAGE	VALIDITY	DELETE	CHANGED	CREATED
Test Category	English (United States)	valid	🗑	11/05/2018 17:15	11/05/2018 17:15

Fig. 2: Category Management Screen

To add a category:

1. Click on the *Add Category* button in the left sidebar.
2. Fill in the required fields.
3. Click on the *Save* button.



★ Title:	
Sub-category of:	
★ Language:	English (United States)
★ Validity:	valid

Fig. 3: Add Category Screen

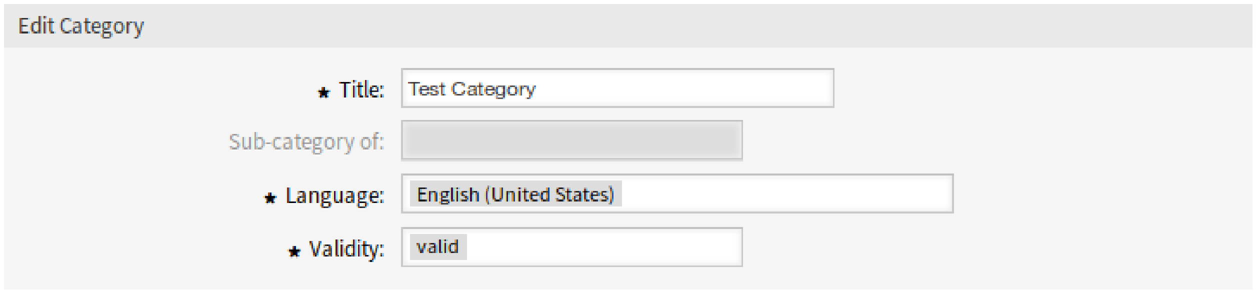
To edit a category:

1. Click on a category in the list of categories.
2. Modify the fields.
3. Click on the *Save* or *Save and finish* button.

To delete a category:

1. Click on the trash icon in the *Delete* column of the overview table.
2. Click on the *Confirm* button.

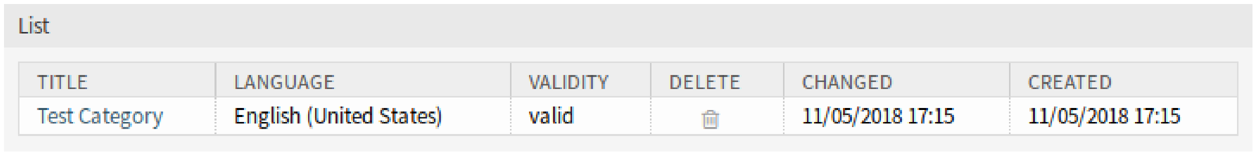
Note: If several categories are added to the system, use the filter box to find a particular category by just typing the name to filter.



Edit Category

⋆ Title:	Test Category
Sub-category of:	
⋆ Language:	English (United States)
⋆ Validity:	valid

Fig. 4: Edit Category Screen



List

TITLE	LANGUAGE	VALIDITY	DELETE	CHANGED	CREATED
Test Category	English (United States)	valid	🗑	11/05/2018 17:15	11/05/2018 17:15

Fig. 5: Delete Category Screen

Category Settings

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

Title *
: The name of this resource. Any type of characters can be entered to this field including uppercase letters and spaces. The name will be displayed in the overview table.

Sub-category of
: It is possible to add the new category under an existing one as sub-category. This will be displayed as *Parent Category::Child Category*.

Language *
: Select a language from the available languages of the system.

Validity *
: Set the validity of this resource. Each resource can be used in OTRS only, if this field is set to *valid*. Setting this field to *invalid* or *invalid-temporarily* will disable the use of the resource.

7.1.2 Manage Items

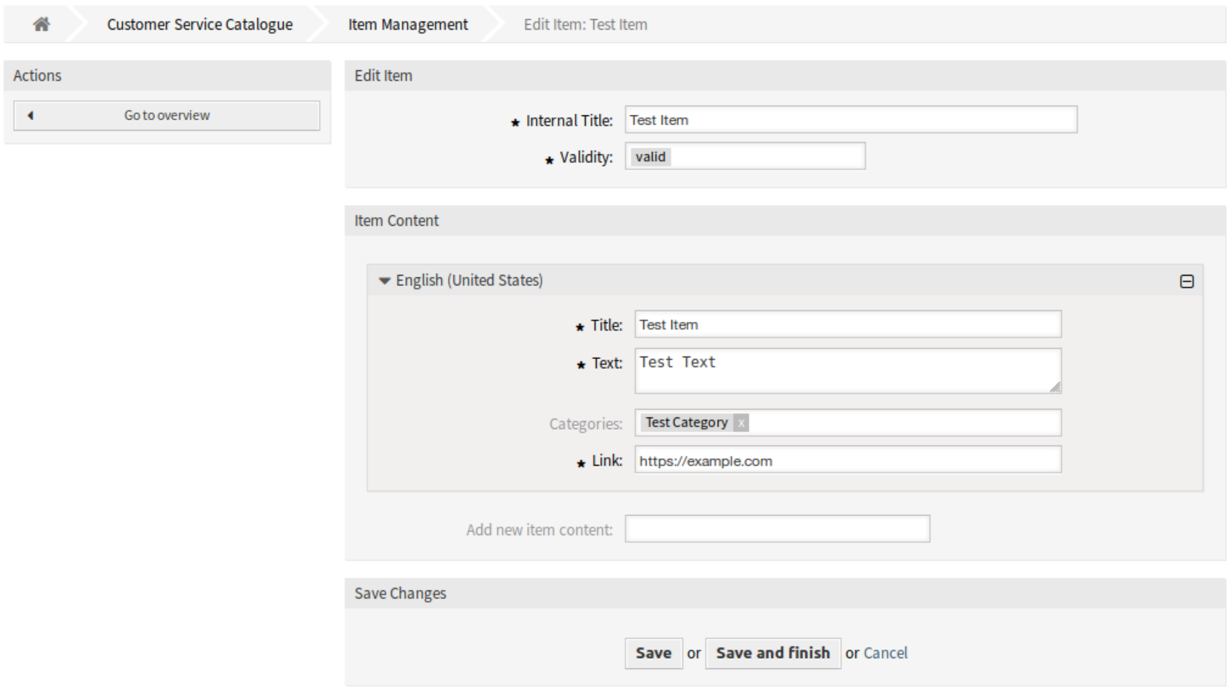
Use this screen to add items to the catalogue. Items can be collected into categories. The *Item Management* screen in available via the *Go to item management* button or via the *Item Management* module.

To add an item:

1. Click on the *Add Item* button in the left sidebar.
2. Fill in the required fields.
3. Click on the *Save* button.

To edit an item:

1. Click on an item in the list of items.
2. Modify the fields.



Customer Service Catalogue | Item Management | Edit Item: Test Item

Actions

Go to overview

Edit Item

Internal Title: Test Item

Validity: valid

Item Content

English (United States)

Title: Test Item

Text: Test Text

Categories: Test Category

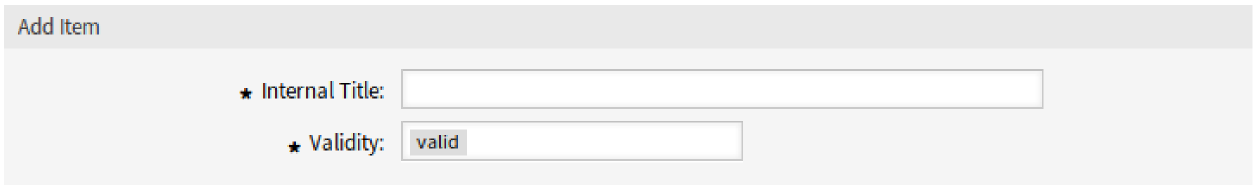
Link: https://example.com

Add new item content:

Save Changes

Save or Save and finish or Cancel

Fig. 6: Item Management Screen



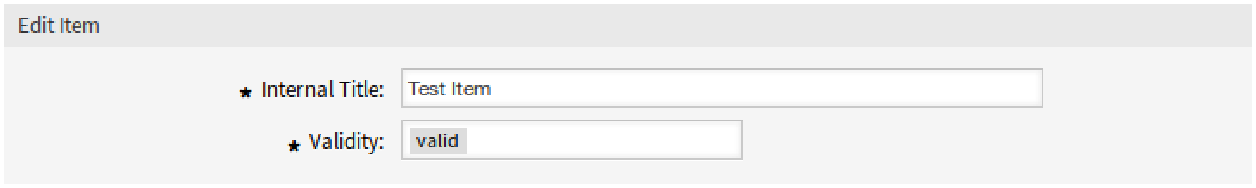
Add Item

Internal Title:

Validity: valid

Fig. 7: Add Item Screen

3. Click on the *Save* or *Save and finish* button.

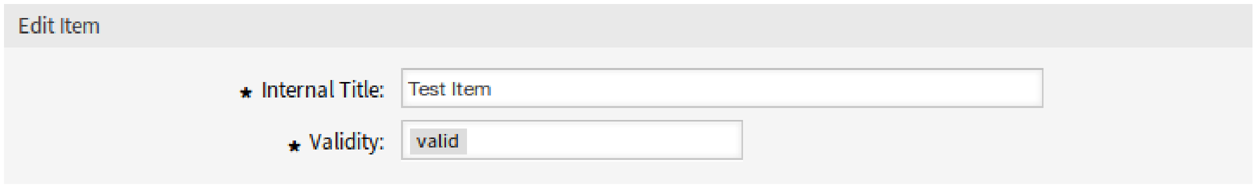

Edit Item

★ Internal Title:	Test Item
★ Validity:	valid

Fig. 8: Edit Item Screen

To delete an item:

1. Click on the trash icon in the *Delete* column of the overview table.
2. Click on the *Confirm* button.


Edit Item

★ Internal Title:	Test Item
★ Validity:	valid

Fig. 9: Delete Item Screen

Note: If several items are added to the system, use the filter box to find a particular item by just typing the name to filter.

Item Settings

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

Internal Title *
: The name of this resource, that is only displayed in the administrator interface. Any type of characters can be entered to this field including uppercase letters and spaces. The name will be displayed in the overview table.

Validity *
: Set the validity of this resource. Each resource can be used in OTRS only, if this field is set to *valid*. Setting this field to *invalid* or *invalid-temporarily* will disable the use of the resource.

Item Content
: In this widget can be added some localized content for the item.

  Title *
  : The name of this resource in the given language. Any type of characters can be entered to this field including uppercase letters and spaces.

  Text *
  : The text for this item in the given language.

  Categories
  : One ore more categories can be selected for the item in which the item should be visible.

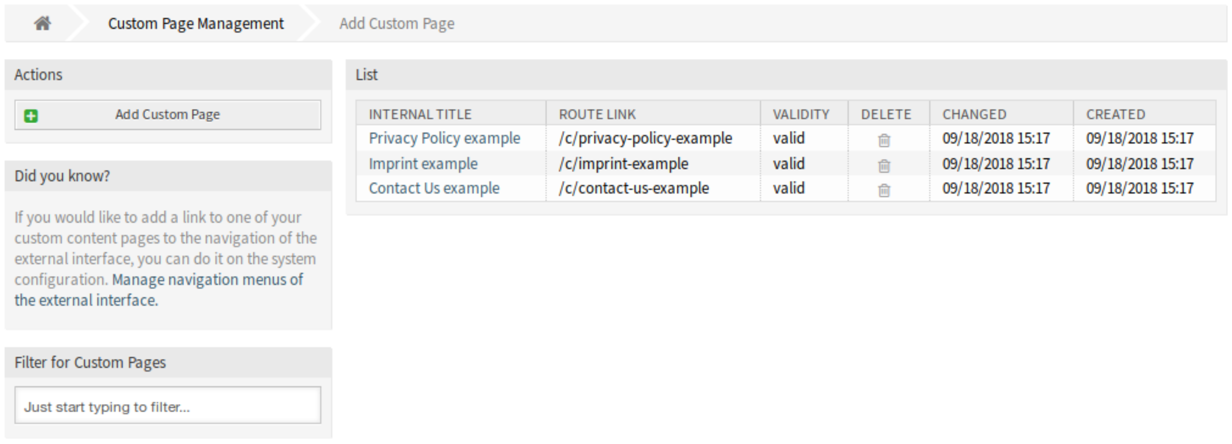
Note: Only those categories can be selected, that have the same language as the selected language for this widget.

Link *
A link to an internal or an external URL.

Add new item content
Select which languages should be added to create localized item content. All added languages can hold its own localized content, that are explained above.

7.2 Custom Pages

Use this screen to add custom pages for use in external interface. A fresh OTRS installation already contains some custom pages by default. The custom page management screen is available in the *Custom Pages* module of the *External Interface* group.



If you would like to add a link to one of your custom content pages to the navigation of the external interface, you can do it on the system configuration. Manage navigation menus of the external interface.

INTERNAL TITLE	ROUTE LINK	VALIDITY	DELETE	CHANGED	CREATED
Privacy Policy example	/c/privacy-policy-example	valid	🗑	09/18/2018 15:17	09/18/2018 15:17
Imprint example	/c/imprint-example	valid	🗑	09/18/2018 15:17	09/18/2018 15:17
Contact Us example	/c/contact-us-example	valid	🗑	09/18/2018 15:17	09/18/2018 15:17

Fig. 10: Custom Page Management Screen

7.2.1 Manage Custom Pages

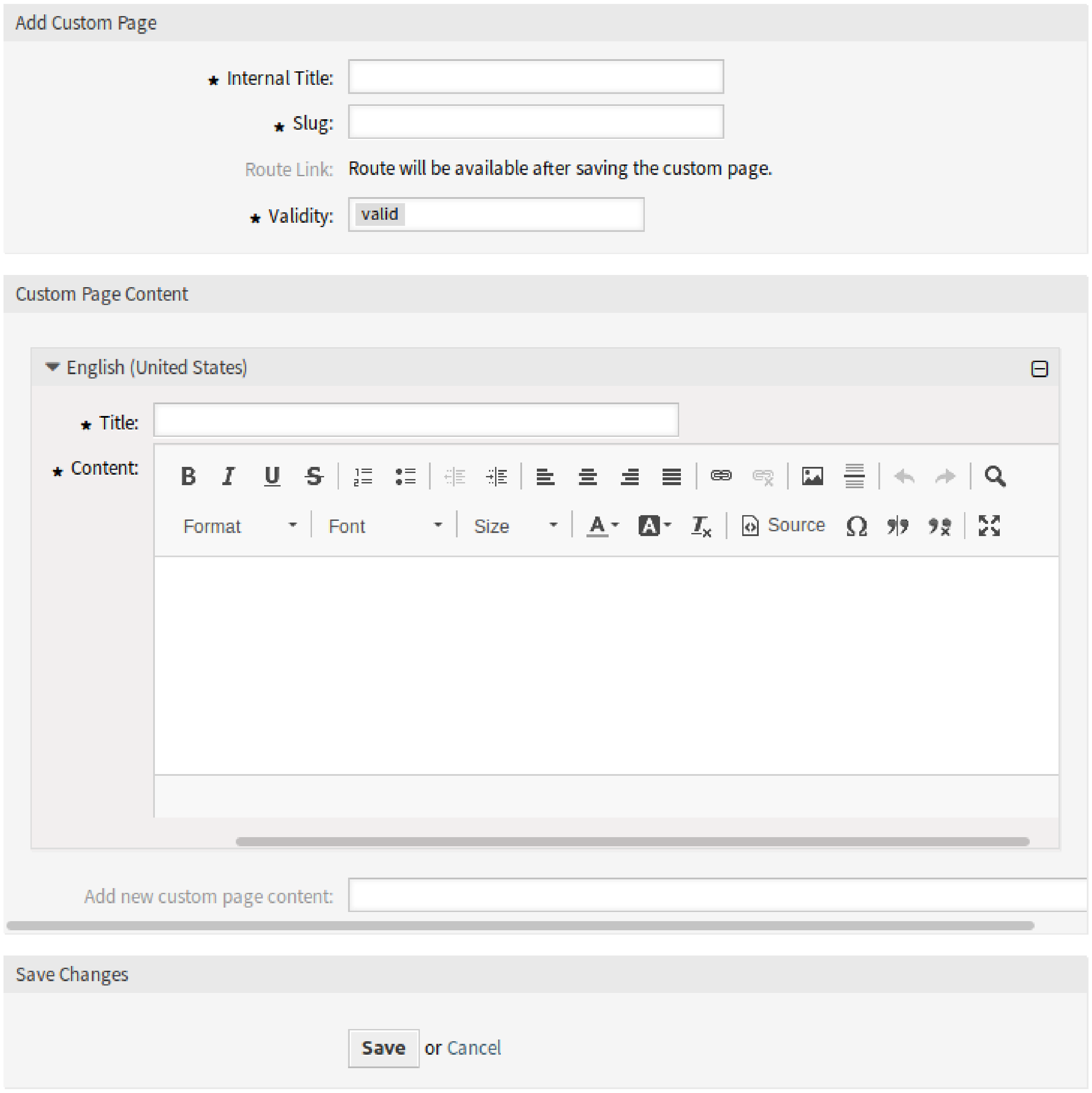
Warning: Make sure to save your changes when you finish. The new configuration will be immediately deployed.

To add a custom page:

1. Click on the *Add Custom Page* button in the left sidebar.
2. Fill in the required fields.
3. Click on the *Save* button.

To edit a custom page:

1. Click on a custom page in the list of custom pages.


Add Custom Page

★ Internal Title:	
★ Slug:	
Route Link:	Route will be available after saving the custom page.
★ Validity:	valid

Custom Page Content

English (United States)

★ Title:	
★ Content:	

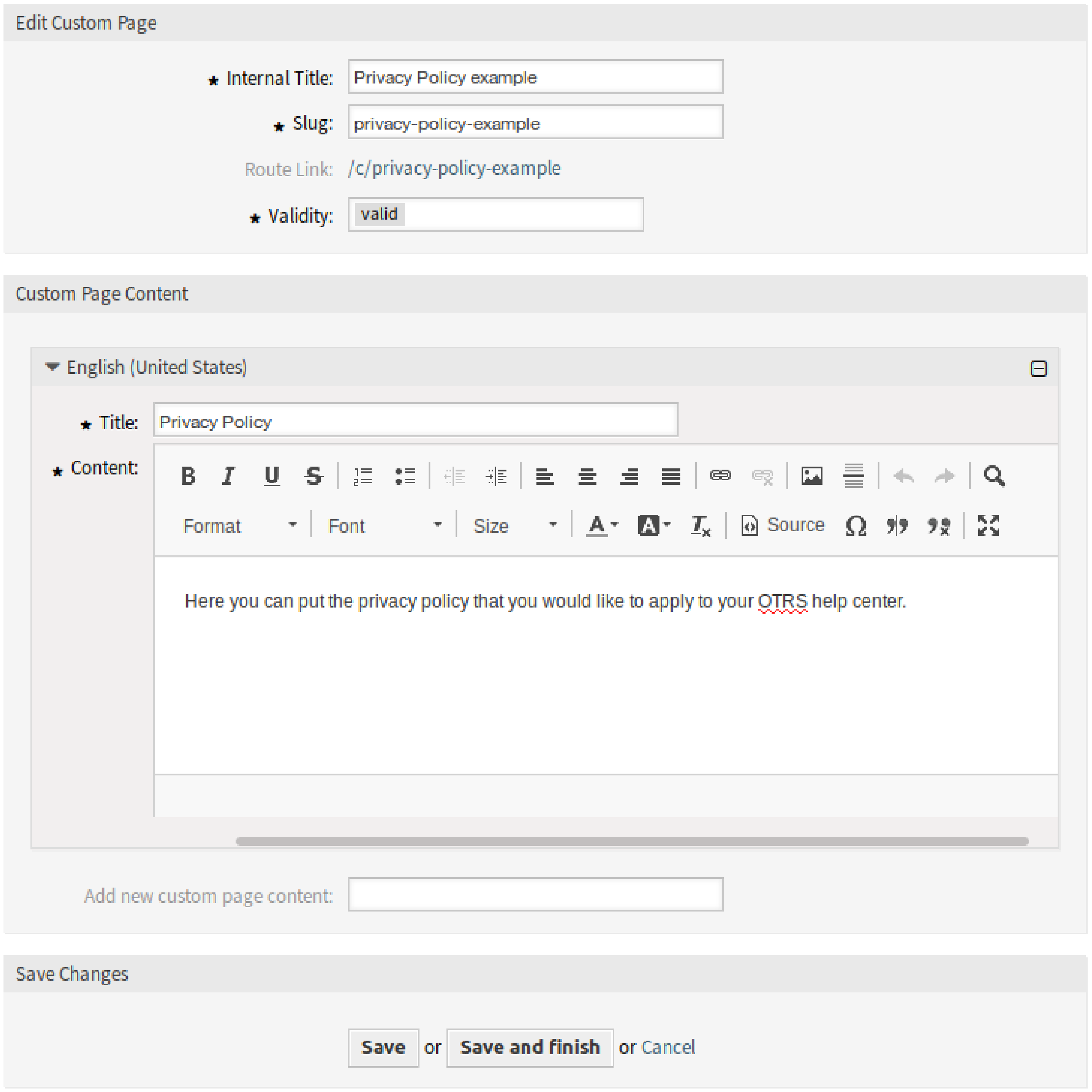
Add new custom page content:

Save Changes

Save or Cancel

Fig. 11: Add Custom Page Screen

2. Modify the fields.
3. Click on the *Save* or *Save and finish* button.



Edit Custom Page

⋆ Internal Title:	Privacy Policy example
⋆ Slug:	privacy-policy-example
Route Link:	/c/privacy-policy-example
⋆ Validity:	valid

Custom Page Content

English (United States)

⋆ Title:	Privacy Policy
⋆ Content:	Here you can put the privacy policy that you would like to apply to your OTRS help center.

Add new custom page content:

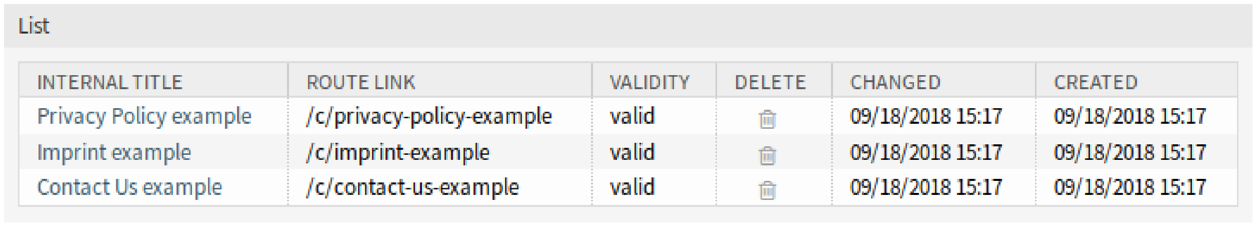
Save Changes

Fig. 12: Edit Custom Page Screen

To delete a custom page:

1. Click on the trash icon in the fourth column of the overview table.
2. Click on the *Confirm* button.

Note: If several custom pages are added to the system, use the filter box to find a particular custom page



List

INTERNAL TITLE	ROUTE LINK	VALIDITY	DELETE	CHANGED	CREATED
Privacy Policy example	/c/privacy-policy-example	valid	🗑	09/18/2018 15:17	09/18/2018 15:17
Imprint example	/c/imprint-example	valid	🗑	09/18/2018 15:17	09/18/2018 15:17
Contact Us example	/c/contact-us-example	valid	🗑	09/18/2018 15:17	09/18/2018 15:17

Fig. 13: Delete Custom Page Screen

by just typing the name to filter.

7.2.2 Custom Page Settings

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

Internal Title *
The name of this resource, that is only displayed in the administrator interface. Any type of characters can be entered to this field including uppercase letters and spaces. The name will be displayed in the overview table.

Slug *
This will be the URL of the custom page. Recommended characters are lowercase letters, numbers and minus sign.

Validity *
Set the validity of this resource. Each resource can be used in OTRS only, if this field is set to *valid*. Setting this field to *invalid* or *invalid-temporarily* will disable the use of the resource.

Custom Page Content
In this widget can be added some localized content for the item.

Title *
The name of this resource in the given language. Any type of characters can be entered to this field including uppercase letters and spaces.

Content *
The text for this item in the given language.

Add new custom page content
Select which languages should be added to create localized item content. All added languages can hold its own localized content, that are explained above.

7.3 Home Page

Use this screen to define home page configuration for different user languages, that are displayed in the external interface. The home page management screen is available in the *Home Page* module of the *External Interface* group.

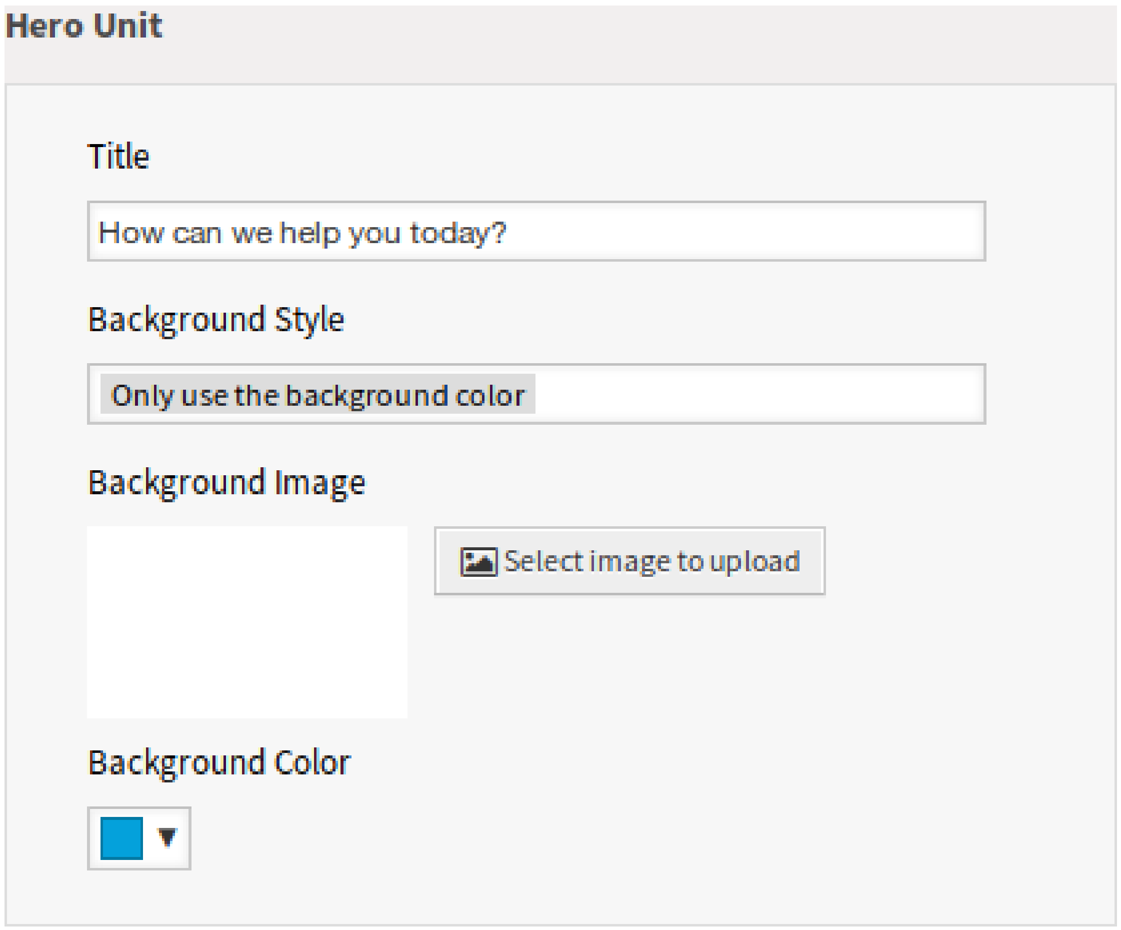
This screen contains several widget for each languages, where localized content can be added.

Warning: Make sure to save your changes when you finish. The new configuration will be immediately deployed.

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

7.3.1 Hero Unit

This is the main widget of the external interface.



Hero Unit

Title

How can we help you today?

Background Style

Only use the background color

Background Image

Background Color

Fig. 14: Hero Unit Widget

Title
This is the most important sentence or motto of the home page.

Background Style
Define the behavior of background image and background color. The following options are available:

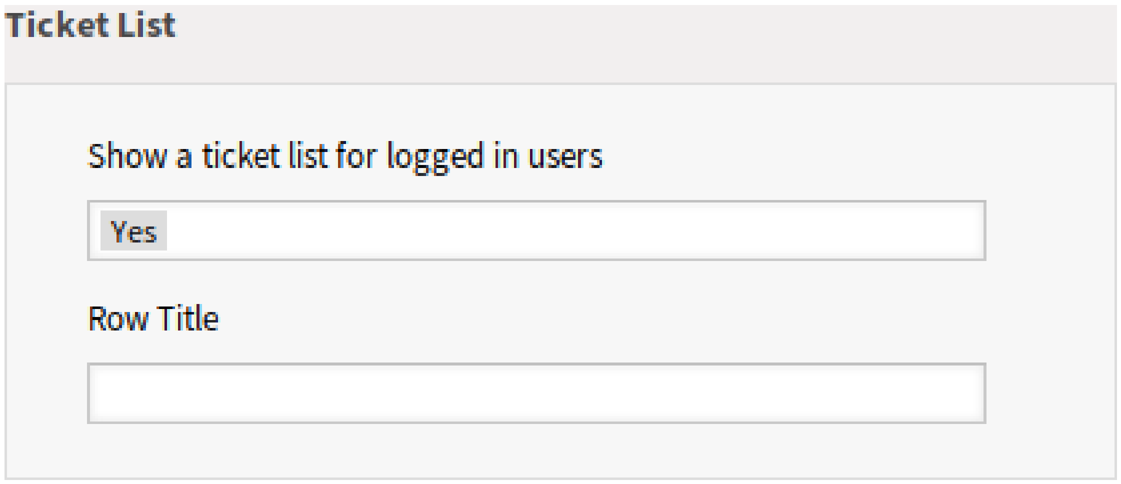
- Only use the background image
- Only use the background color
- Use the background image and overlay it with the selected color

Background Image
Select an image used as background image for the hero unit. To select an image, click on the *Select image to upload* button and chose an image from your file system.

Background Color
To change the background color, just select a new color from the color palette. You can chose from the pre-selected colors or define other colors by choosing it from the color selector or typing the hexadecimal value.

7.3.2 Ticket List

This widget is available only to logged in customer users. If the ticket list is enabled, the tickets of the customer users are displayed under the *Hero Unit* (search field). If the ticket list is disabled, the customer users can see the tickets only in the *My Tickets* screen.


Ticket List

Show a ticket list for logged in users

Yes

Row Title

Fig. 15: Ticket List Widget

Show a ticket list for logged in users
This setting defines if the ticket list should be visible for logged in users or not.

Row Title
This is an optional title for the row, that contains this widget in the external interface.

7.3.3 Image Teasers

The image teasers are displayed in a separate row in the external interface.

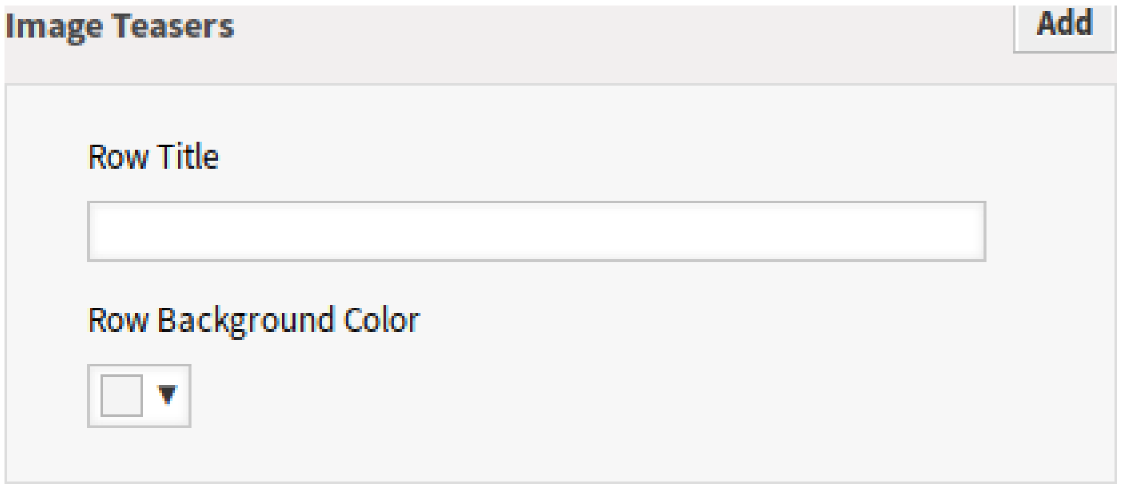


Fig. 16: Image Teasers Widget

Row Title
This is an optional title for the row, that contains this widget in the external interface.

Row Background Color
To change the background color, just select a new color from the color palette. You can chose from the pre-selected colors or define other colors by choosing it from the color selector or typing the hexadecimal value.

You can add up to three image teasers by clicking on the *Add* button in the top right corner of the widget.

Image
Select an image to display it in the image teaser widget. To select an image, click on the *Select image to upload* button and chose an image from your file system.

Title *
The heading text that are displayed in this item.

Text *
The text for this item.

Link Target *
An URL that will be opened after clicking on this item in the external interface.

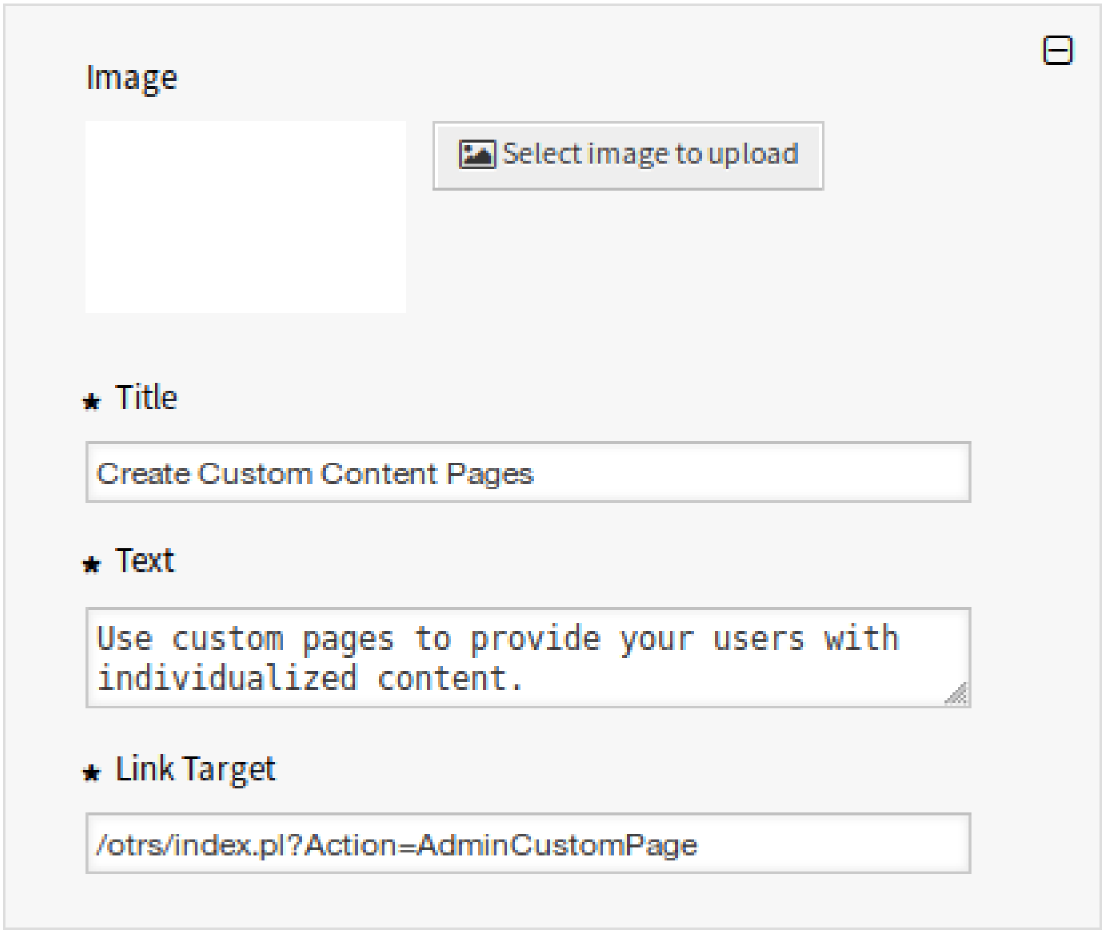
7.3.4 Link Lists

The link lists are displayed in a separate row in the external interface.

Row Title
This is an optional title for the row, that contains this widget in the external interface.

You can add up to four link lists by clicking on the *Add* button in the top right corner of the widget.

Title *
The heading text that are displayed in this item.



★ Title

Create Custom Content Pages

★ Text

Use custom pages to provide your users with individualized content.

★ Link Target

/otrs/index.pl?Action=AdminCustomPage

Fig. 17: Image Teaser

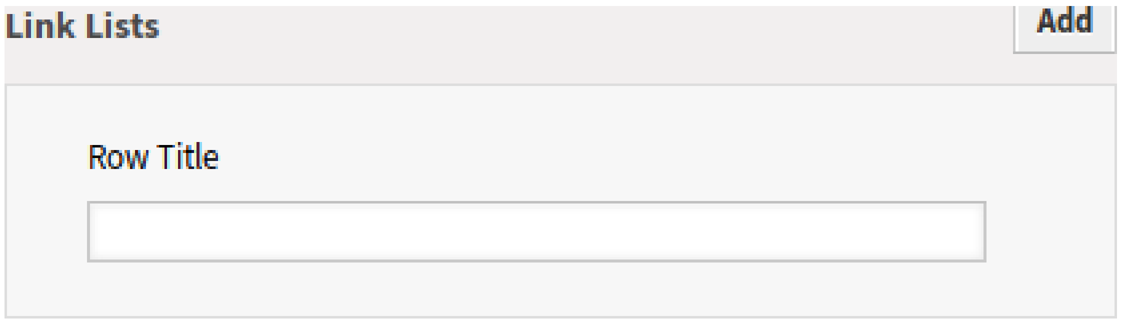


Fig. 18: Link Lists Widget

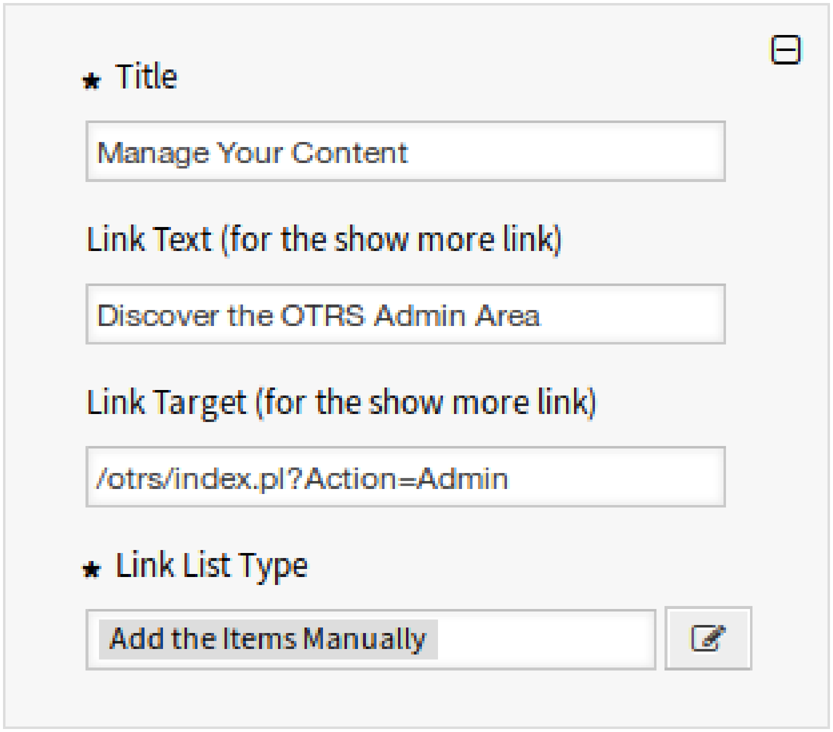

★ Title
Manage Your Content
Link Text (for the show more link)
Discover the OTRS Admin Area
Link Target (for the show more link)
/otrs/index.pl?Action=Admin
★ Link List Type
Add the Items Manually

Fig. 19: Link List

Link Text
Add the text for the *show more* link.

Link Target
Add the URL for the *show more* link.

Link List Type *
Specifies the functionality how the item will be added.

Add the Items Manually
Clicking on the edit button a new widget will be visible to add the items.

Show the Results of a Search
Clicking on the edit button a new widget will be visible to add the items.

This configuration calls the document search. Therefore some tickets, knowledge base articles, service catalog contents are necessary for this function. If the system has public or external contents, this card can show the elements.

7.3.5 Content Cards

The link lists are displayed in a separate row in the external interface.

Row Title
This is an optional title for the row, that contains this widget in the external interface.

You can add up to three content cards by clicking on the *Add* button in the top right corner of the widget.

Title *
The heading text that are displayed in this item.

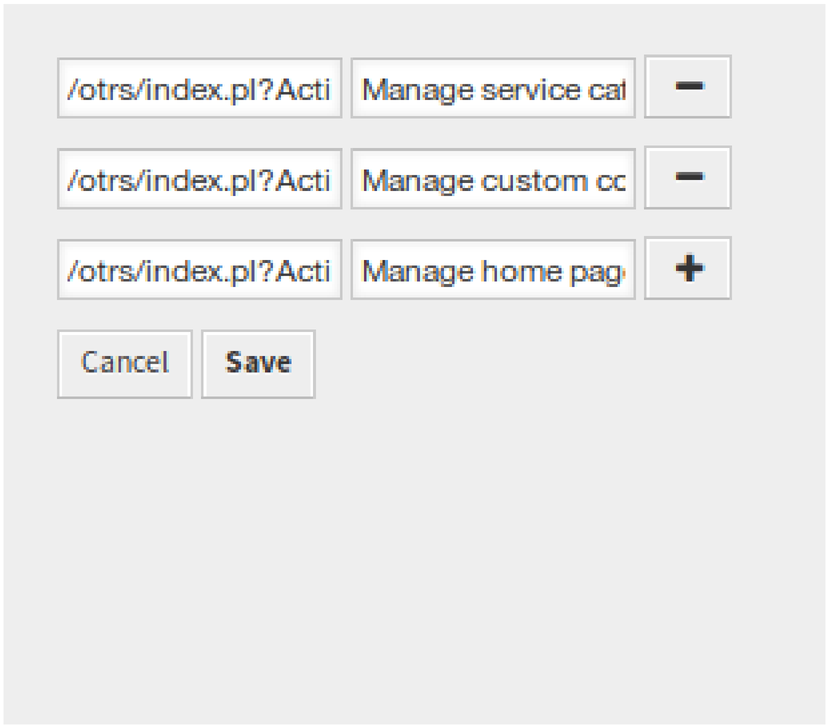


Fig. 20: Add the Items Manually

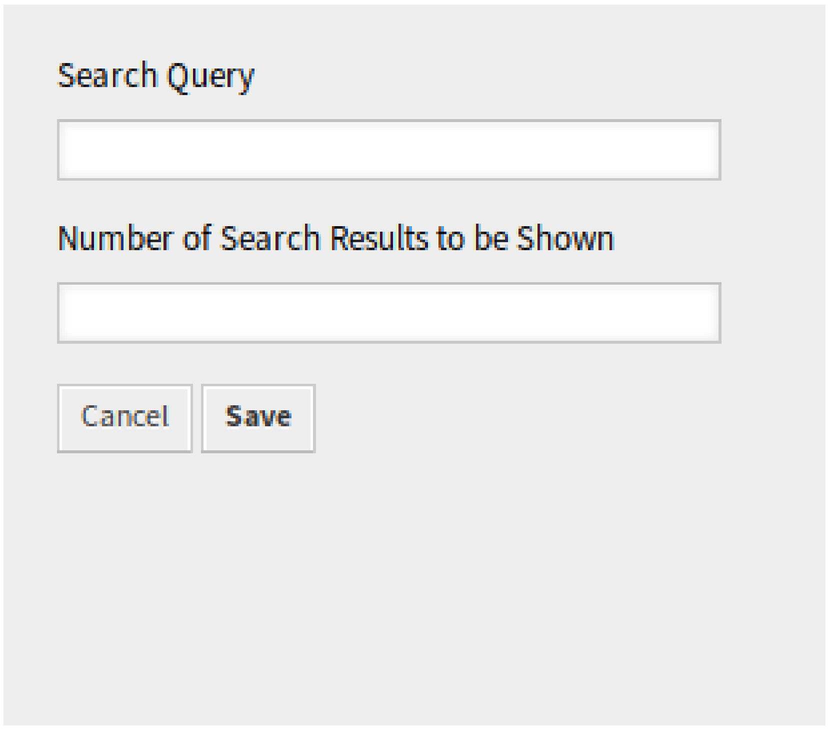
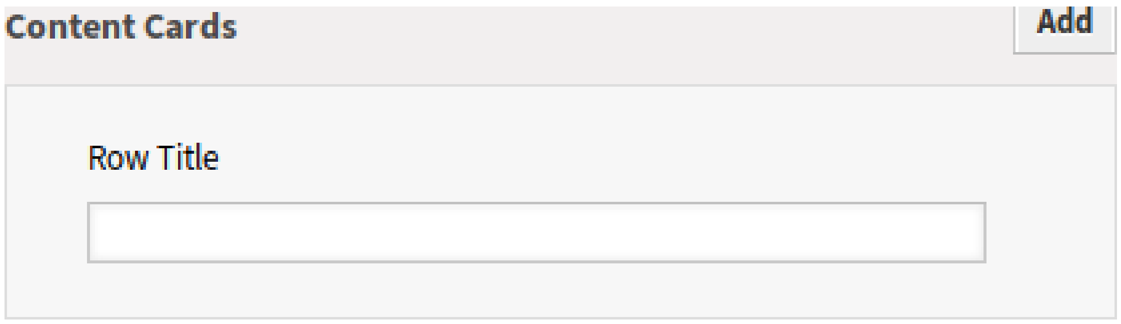


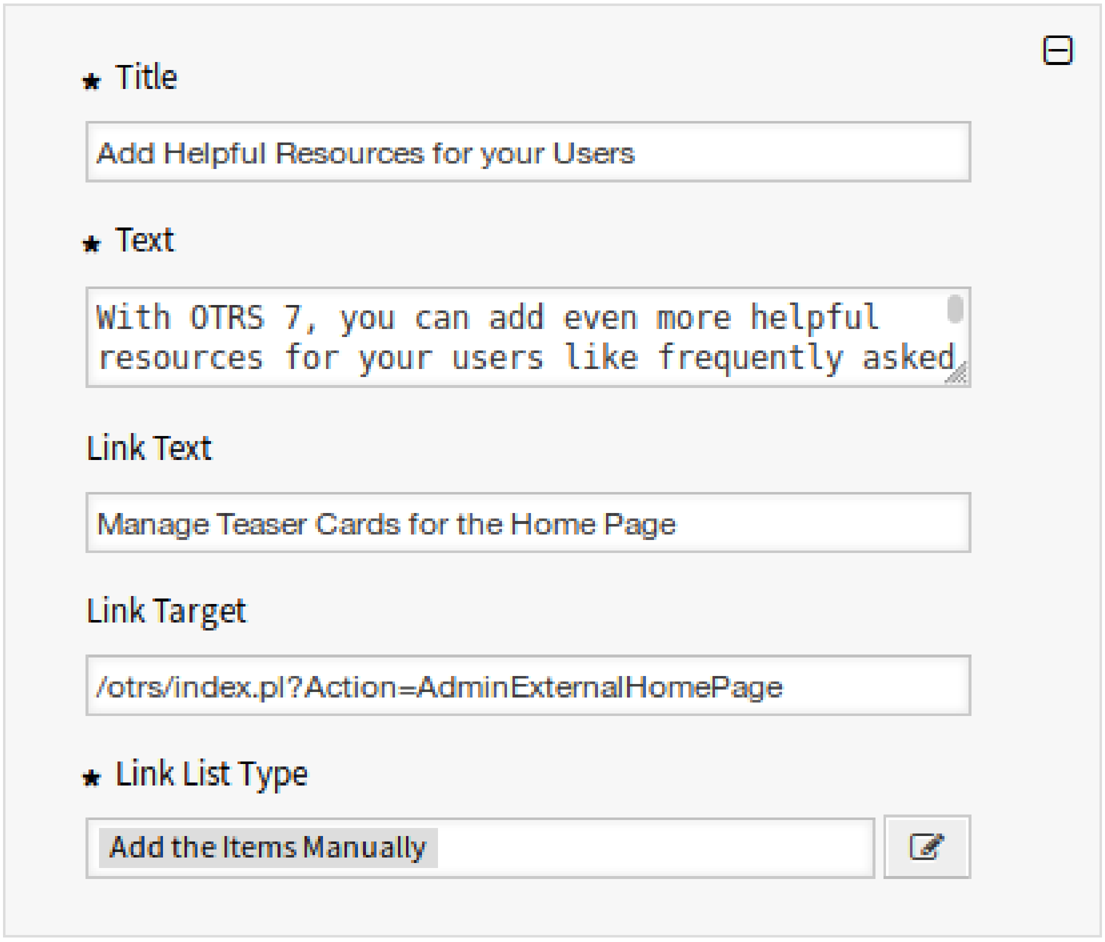
Fig. 21: Show the Results of a Search


Content Cards

Add

Row Title

Fig. 22: Content Cards Widget


★ Title

Add Helpful Resources for your Users

★ Text

With OTRS 7, you can add even more helpful resources for your users like frequently asked

Link Text

Manage Teaser Cards for the Home Page

Link Target

/otrs/index.pl?Action=AdminExternalHomePage

★ Link List Type

Add the Items Manually

Fig. 23: Content Card

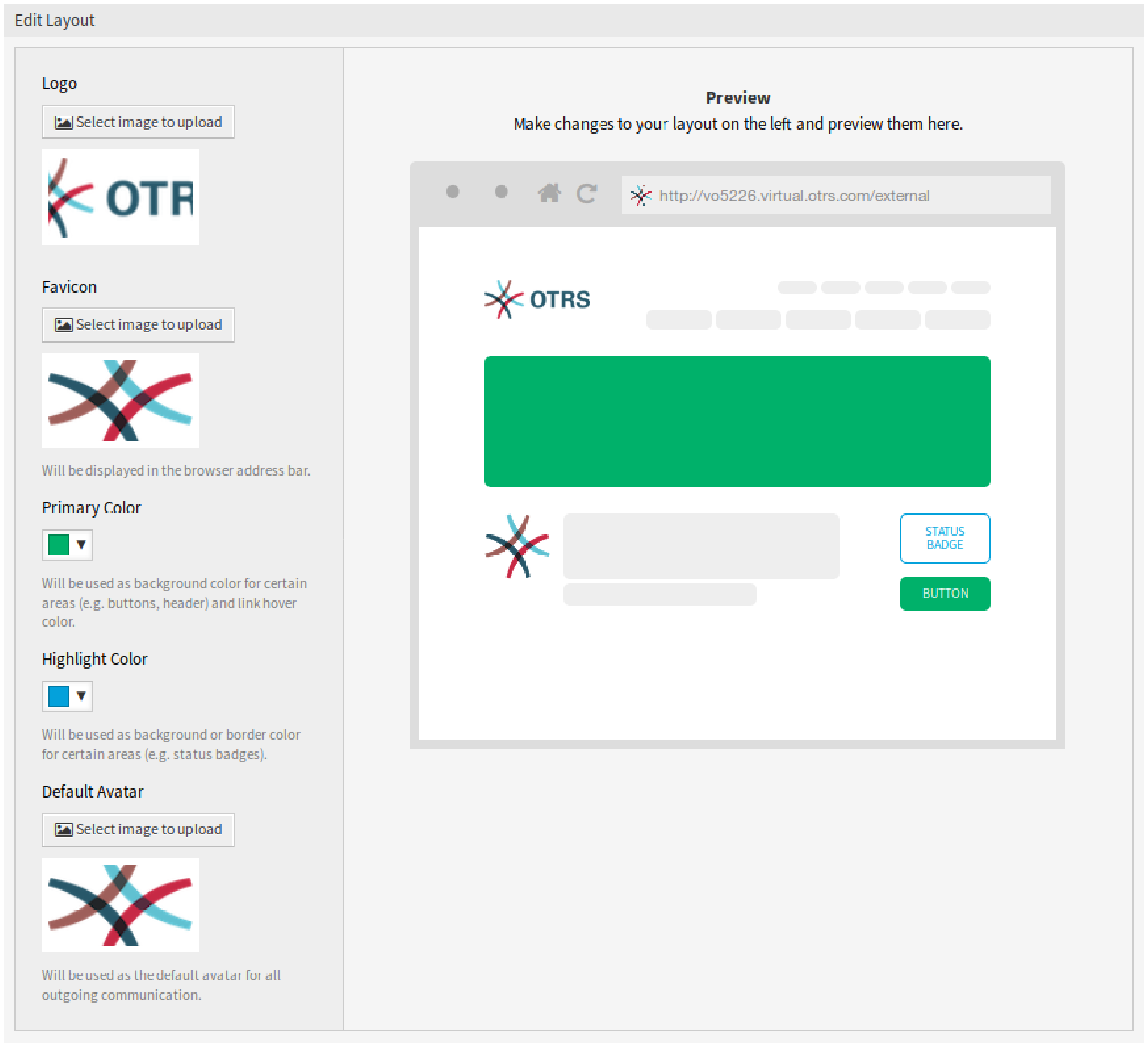
Text *
The text for this item.

Link Text
Add the text for the *show more* link.

Link Target
An URL that will be opened after clicking on this item in the external interface.

7.4 Layout

Use this screen to edit the layout displayed in external interface. The layout management screen is available in the *Layout* module of the *External Interface* group.



Edit Layout

Logo

Select image to upload

Favicon

Select image to upload

Will be displayed in the browser address bar.

Primary Color

Will be used as background color for certain areas (e.g. buttons, header) and link hover color.

Highlight Color

Will be used as background or border color for certain areas (e.g. status badges).

Default Avatar

Select image to upload

Will be used as the default avatar for all outgoing communication.

Preview

Make changes to your layout on the left and preview them here.

Fig. 24: Edit Layout Screen

A fresh OTRS installation already contains a default layout. In this screen all parameters can be changed.

Warning: Make sure to save your changes when you finish. The new configuration will be immediately deployed.

Logo
The logo is a small image that is displayed in the top left corner of the external pages.

To change the logo, click on the *Select image to upload* button, and select a new logo image. Recommended file format is PNG.

Favicon
The favorite icon is an icon that is displayed in the URL bar of the web browser.

To change the favorite icon, click on the *Select image to upload* button, and select a new icon. This is usually a 16×16 pixel image in PNG or ICO format.

Primary Color
Primary color is the most important color of the external pages (see the preview screen).

To change the primary color, just select a new color from the color palette. You can chose from the preselected colors or define other colors by choosing it from the color selector or typing the hexadecimal value. The new color will be displayed in the preview widget immediately.

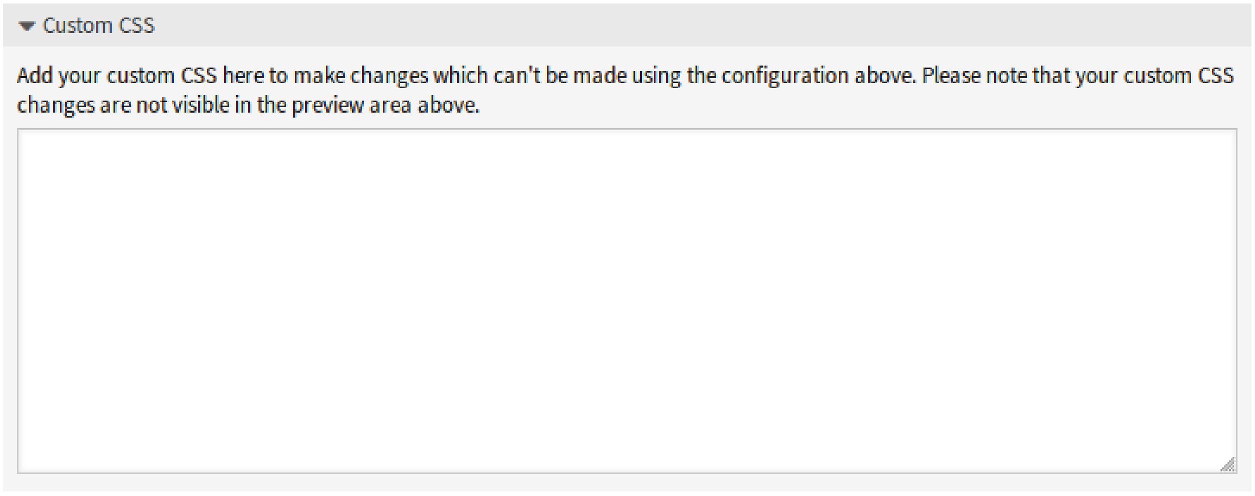
Highlight Color
Highlight color is the second color of the external pages using for status badges, links, etc.

To change the highlight color, just select a new color from the color palette. You can chose from the preselected colors or define other colors by choosing it from the color selector or typing the hexadecimal value. The new color will be displayed in the preview widget immediately.

Default Avatar
Will be used as the default avatar for all outgoing communication.

To change the avatar, click on the *Select image to upload* button, and select a new avatar image.

Custom CSS
Use this text area to add custom CSS to be applied in the external interface.



Add your custom CSS here to make changes which can't be made using the configuration above. Please note that your custom CSS changes are not visible in the preview area above.

Fig. 25: Custom CSS Widget

CHAPTER

EIGHT

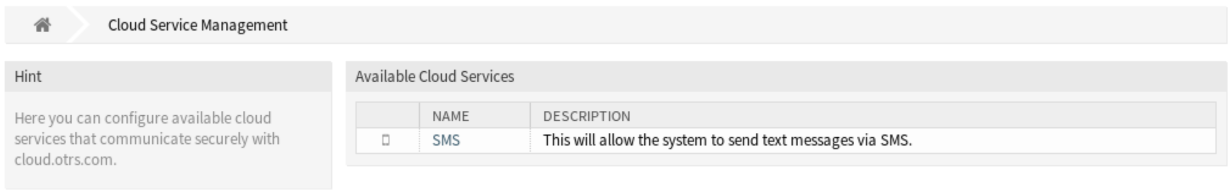
OTRS GROUP SERVICES

A strong partner is good to have when dealing with mission-critical systems. Whether it be maintenance or for consuming cloud services, your partner should be tightly integrated.

The following chapter describes the tools you have available to integrate your system with the powerful cloud service offered by the *OTRS Group*.

8.1 Cloud Services

Use this screen to add cloud services to the system. A fresh OTRS installation doesn' t contain any configured cloud services by default. The cloud service management screen is available in the *Cloud Services* module of the *OTRS Group Services* group.



Cloud Service Management

Hint

Here you can configure available cloud services that communicate securely with cloud.otrs.com.

Available Cloud Services

	NAME	DESCRIPTION
▯	SMS	This will allow the system to send text messages via SMS.

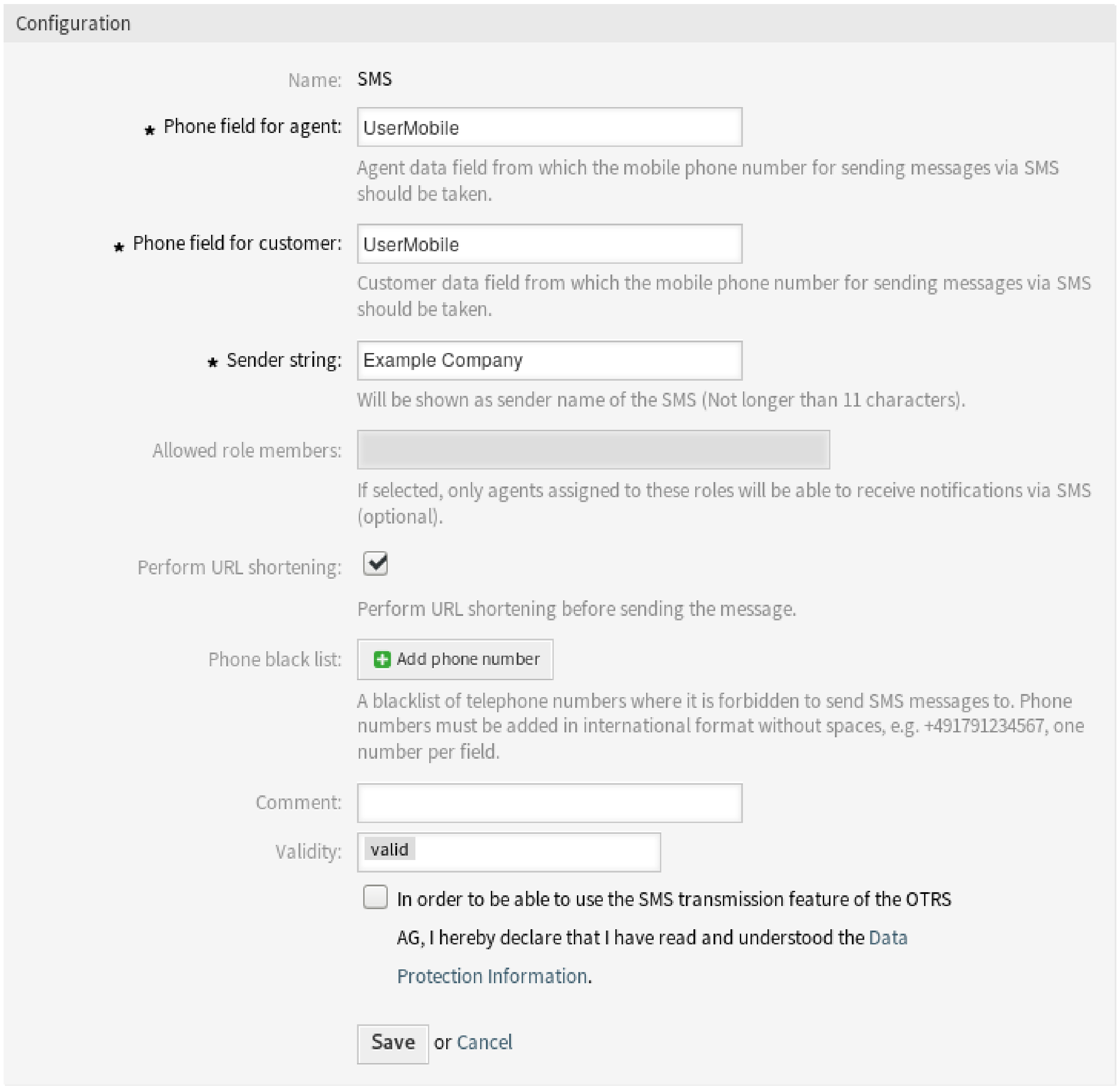
Fig. 1: Cloud Service Management Screen

8.1.1 Manage Cloud Services

Activate SMS Cloud Service

To be able to use SMS cloud service in OTRS, you have to activate it first. To activate the SMS cloud service:

1. Click on the *Activate SMS Cloud Service* button in the left sidebar.
2. Fill in the required fields.
3. Click on the *Save* button.


Configuration

Name:	SMS
★ Phone field for agent:	UserMobile Agent data field from which the mobile phone number for sending messages via SMS should be taken.
★ Phone field for customer:	UserMobile Customer data field from which the mobile phone number for sending messages via SMS should be taken.
★ Sender string:	Example Company Will be shown as sender name of the SMS (Not longer than 11 characters).
Allowed role members:	If selected, only agents assigned to these roles will be able to receive notifications via SMS (optional).
Perform URL shortening:	✓ Perform URL shortening before sending the message.
Phone black list:	Add phone number A blacklist of telephone numbers where it is forbidden to send SMS messages to. Phone numbers must be added in international format without spaces, e.g. +491791234567, one number per field.
Comment:	
Validity:	valid

☐ In order to be able to use the SMS transmission feature of the OTRS AG, I hereby declare that I have read and understood the Data Protection Information.

Fig. 2: Add Cloud Service Screen

Configuration

The following settings are available when adding an SMS cloud service. The fields marked with an asterisk are mandatory.

Phone field for agent *
Agent data field from which the mobile phone number for sending messages via SMS should be taken.

Phone field for customer *
Customer data field from which the mobile phone number for sending messages via SMS should be taken.

Sender string *
Will be shown as sender name of the SMS (Not longer than 11 characters).

Allowed role members
If selected, only agents assigned to these roles will be able to receive notifications via SMS.

Perform URL shortening
Perform URL shortening before sending the message.

Phone black list
A blacklist of telephone numbers where it is forbidden to send SMS messages to. Phone numbers must be added in international format without spaces, e.g. +491791234567, one number per field.

Comment
Add additional information to this resource. It is recommended to always fill this field as a description of the resource with a full sentence for better clarity, because the comment will be also displayed in the overview table.

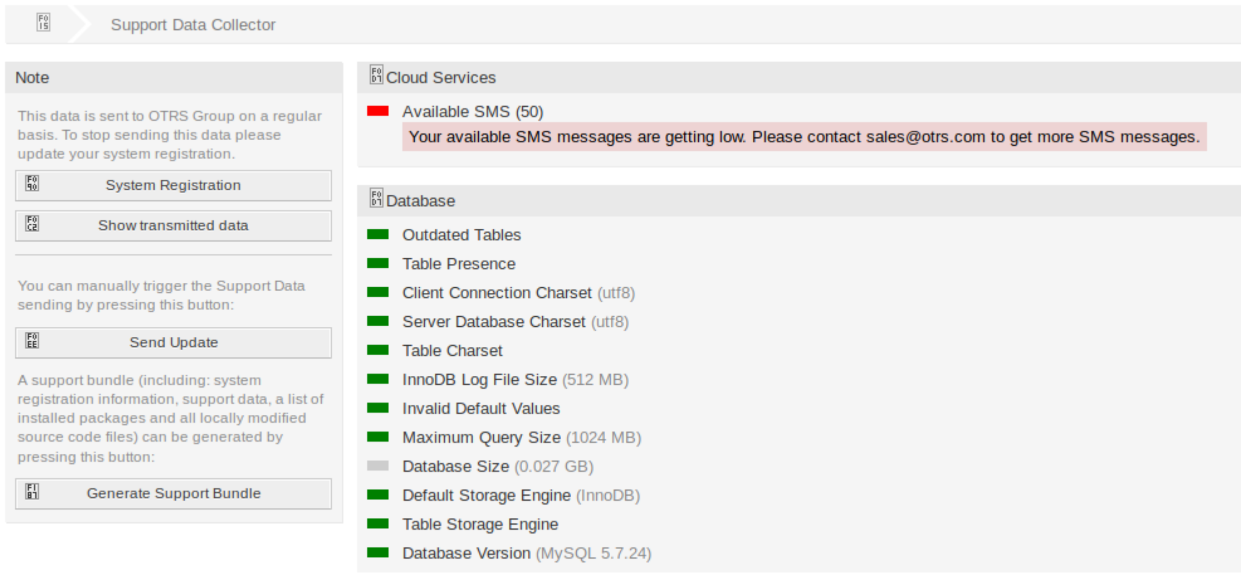
Validity
Set the validity of this resource. Each resource can be used in OTRS only, if this field is set to *valid*. Setting this field to *invalid* or *invalid-temporarily* will disable the use of the resource.

Data Protection Information
In order to be able to use the SMS transmission feature of the OTRS AG, the Data Protection Information needs to be read and understood.

8.2 Support Data Collector

Support data collector is used to collect some data and sent to OTRS Group on a regular basis, if the system is registered.

Use this screen to review the data to be sent to OTRS Group. The support data collector screen is available in the *Support Data Collector* module of the *OTRS Group Services* group.


Support Data Collector

Note

This data is sent to OTRS Group on a regular basis. To stop sending this data please update your system registration.

System Registration

Show transmitted data

You can manually trigger the Support Data sending by pressing this button:

Send Update

A support bundle (including: system registration information, support data, a list of installed packages and all locally modified source code files) can be generated by pressing this button:

Generate Support Bundle

Cloud Services

🟥 Available SMS (50)
Your available SMS messages are getting low. Please contact sales@otrs.com to get more SMS messages.

Database

🟩 Outdated Tables
🟩 Table Presence
🟩 Client Connection Charset (utf8)
🟩 Server Database Charset (utf8)
🟩 Table Charset
🟩 InnoDB Log File Size (512 MB)
🟩 Invalid Default Values
🟩 Maximum Query Size (1024 MB)
⬜ Database Size (0.027 GB)
🟩 Default Storage Engine (InnoDB)
🟩 Table Storage Engine
🟩 Database Version (MySQL 5.7.24)

Fig. 3: Download Support Bundle Screen

8.2.1 Manage Support Data Collector

Support data collector is used to collect some data and sent to OTRS Group on a regular basis, if the system is registered. To register your system:

1. Click on the *System Registration* button in the left sidebar.
2. Follow the registration instructions.

To show what kind of data will be sent:

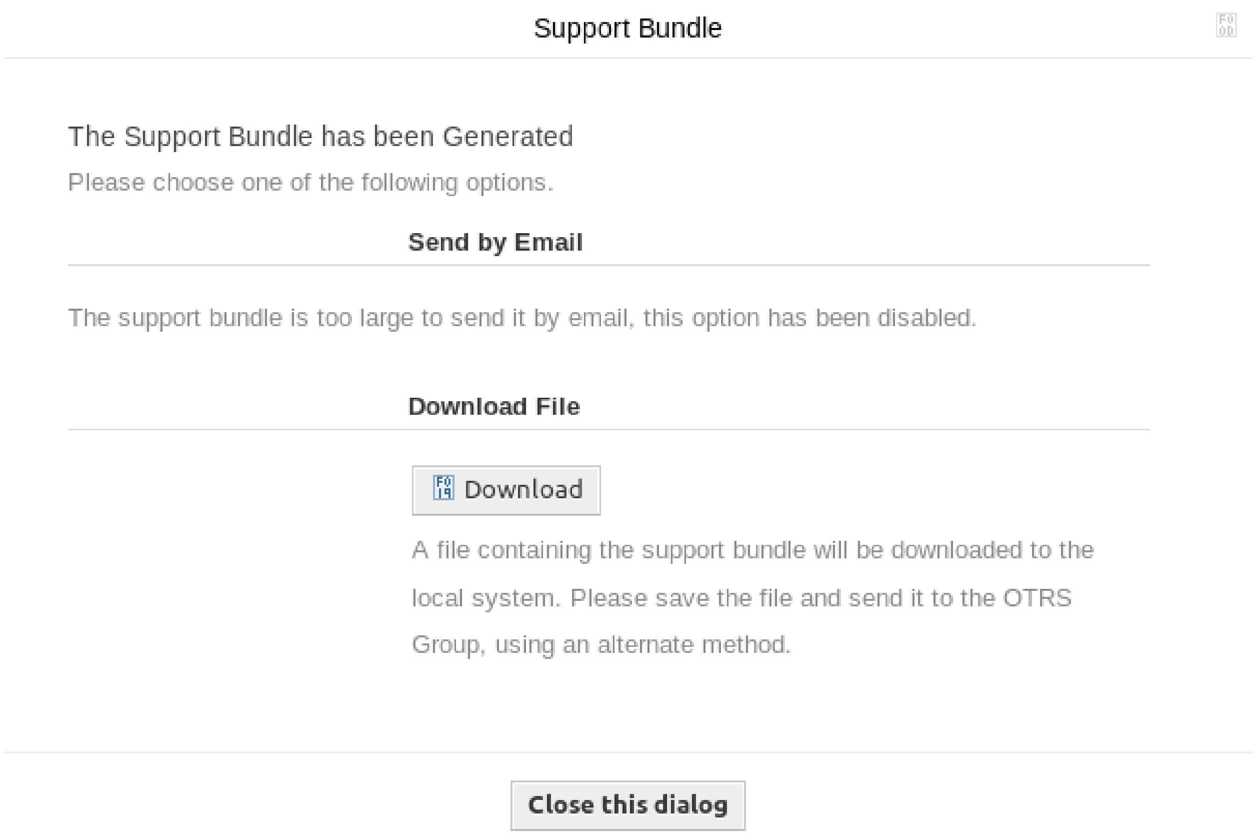
1. Click on the *Show transmitted data* button in the left sidebar.
2. Review the *System Registration Data* and *Support Data* in the newly opened screen.

To manually trigger the support data sending:

1. Click on the *Send Update* button in the left sidebar.

To generate a support bundle:

1. Click on the *Generate Support Bundle* button in the left sidebar.
2. Download the generated support bundle.
3. Open it with an archive manager and review the content.



Support Bundle

The Support Bundle has been Generated

Please choose one of the following options.

Send by Email

The support bundle is too large to send it by email, this option has been disabled.

Download File

A file containing the support bundle will be downloaded to the local system. Please save the file and send it to the OTRS Group, using an alternate method.

Fig. 4: Download Support Bundle Dialog

8.2.2 Collected Data

The screen contains several sections. Each section has some entries with a traffic light, that indicates the following:

- Gray LED means information, just displays a value.
- Green LED means OK, the entry has a good value.
- Yellow LED means notification, you have to check the value, but it is not an error.
- Red LED means error, you have to do something to solve the issue.

Cloud Services Section

This section displays information about OTRS cloud services.

Available SMS
This entry shows information about your available SMS messages. If they are getting low, the LED changes to red.

Database Section

This section displays information about the database used by OTRS.

Outdated Tables
Display the outdated database tables. Green LED means, there are no outdated tables.

Table Presence
Display whether all needed tables exist in the database or not.

Client Connection Charset
Display the character set for the client connection. It must be `utf8`.

Server Database Charset
Display the character set of the database server. It must be `utf8`.

Table Charset
Display the character set of the database table. It must be `utf8`.

InnoDB Log File Size
Display the log file size for InnoDB driver. It must be at least `512 MB`.

Invalid Default Values
Display the invalid default values. Green LED means, there are no invalid default values.

Maximum Query Size
Display the maximum size of a database query. It must be at least `1024 MB`.

Database Size
Display the size of database. This is just an information.

Default Storage Engine
Display the default storage engine of the database. It must be `InnoDB`.

Table Storage Engine
Display the storage engine of the database tables. It must be `InnoDB`.

Database Version
Display the database driver version. Green LED means, the version is high enough.

Document Search Section

This section displays information about document search and the used cluster.

Cluster
The name of the used cluster.

Cluster Health Details
Display some internal variables of the used cluster.

Indices Health
Display information about indices.

Indices Size
Display the size of each index.

Node Health
Display information about the used node.

Operating System Section

This section displays information about the running operating system and installed software components.

Environment Dependencies
Display information about environment dependencies.

OTRS Disk Partition
Display the disk partition to where OTRS is installed.

Information Disk Partitions Usage
Display the used space per disk partitions.

Distribution
Display the distribution name of the operating system.

Kernel Version
Display the kernel version of the operating system.

System Load
Display the system load of the operating system. The system load should be at maximum the number of CPUs the system has (e.g. a load of 8 or less on a system with 8 CPUs is OK).

Perl Version
Display the version of Perl.

Free Swap Space (%)
Display the free swap space as percentages. There should be more than 60% free swap space.

Used Swap Space (MB)
Display the used swap space in megabytes. There should be no more than 200 MB swap space used.

OTRS Section

This section displays information about the OTRS instance.

Article Search Index Status
Display information about indexed articles.

Articles Per Communication Channel
Display the number of articles per communication channels.

Communication Log
Display aggregated information about communications.

Communication Log Account Status (last 24 hours)
Display information about communication log account status in the last 24 hours.

Concurrent Users Details
Display information about the logged in users at the same time separated by hourly.

Concurrent Users
Display information about the number of maximum logged in users in the same time.

Config Settings
Display some important configuration settings from system configurations.

Daemon
Display whether the OTRS daemon is running or not.

Database Records
Display the main OTRS object and the related number of records in the database.

Default Admin Password
Green LED means, that the default administrator password was changed.

Email Sending Queue
Display the number of emails that are queued for sending.

FQDN (Domain Name)
Display the fully qualified domain name set in system configuration setting FQDN.

File System Writable
Display whether the file system is writable or not.

Legacy Configuration Backups
Green LED means, there are no legacy configuration backup files found.

Package Installation Status
Green LED means, that all packages are installed correctly.

Package Framework Version Status
Green LED means, that the OTRS framework version is suitable for the installed packages.

Package Verification Status
Green LED means, that all installed packages are verified by the OTRS Group.

Package List
Display the list of installed packages.

Push Events Status
Display the status of the push events per day.

Session Config Settings
Display the maximum allowed sessions per agents and customers.

Spooled Emails
Display the number of emails that are in the sending pool.

SystemID
Display the system identifier set in system configuration setting SystemID.

Invalid Users with Locked Tickets
Display the number of users, who are set to invalid, but have some ticket locked for him.

Open Tickets
Display the number of open tickets in the system. You will not have performance trouble until you have about 60,000 open tickets in your system.

Ticket Search Index Module
Display the ticket search index module set in system configuration setting Ticket::SearchIndex::ForceUnfilteredStorage.

Time Settings
Display timezone information for OTRS, for the calendars and for users.

UI - Agent Skin Usage
Display the used skins per agents.

UI - Agent Theme Usage
Display the used theme on the agent interface.

UI - Special Statistics
Display some statistics about personal modifications like using favorites, custom menu ordering, etc.

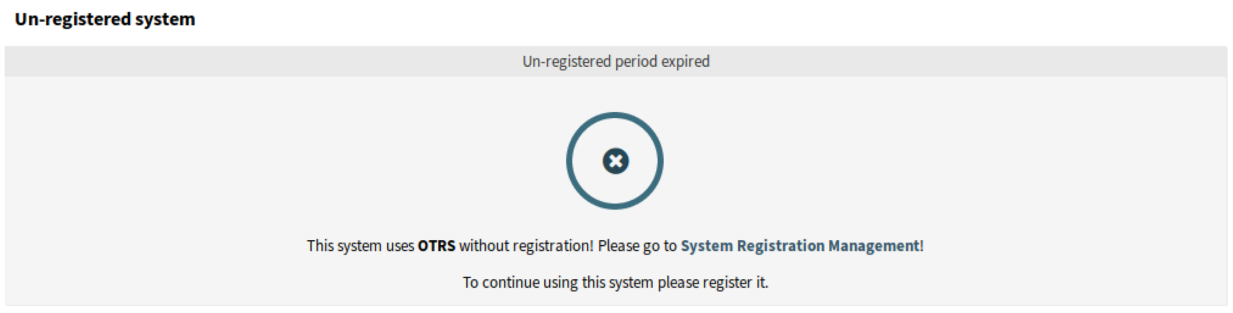
OTRS Version
Display the version number of OTRS.

WebSocket Connection
Display connection status for WebSocket connections.

8.3 System Registration

If the free trial or the registration period is expired, the system will lock itself. In these cases you have to register your system or you have to extend the registration.

Use this screen to register your system with the OTRS Group. The registration screen is available in the *System Registration* module of the *OTRS Group Services* group.



Un-registered system

Un-registered period expired

This system uses **OTRS** without registration! Please go to **System Registration Management!**

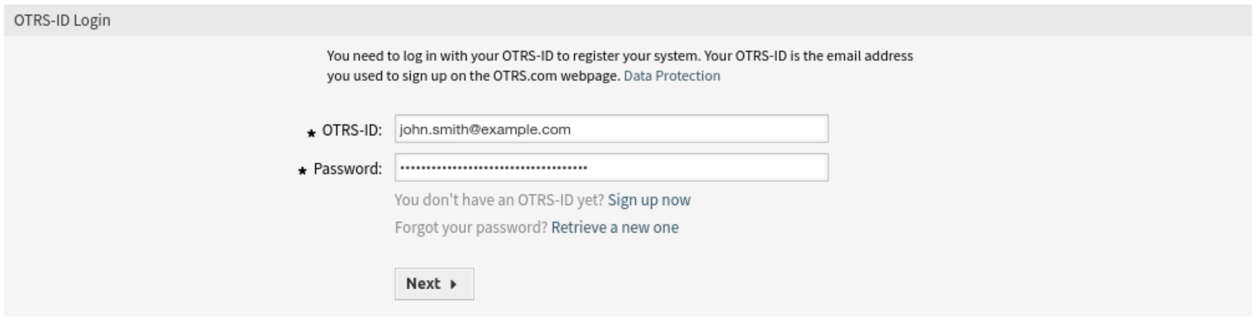
To continue using this system please register it.

Fig. 5: Un-Registred System Screen

8.3.1 Manage System Registration

To register your system:

1. Obtain an OTRS ID. You have to register in the OTRS Portal. After the registration you will get your OTRS ID.
2. Go to the *System Registration* module of the *OTRS Group Services* group, and enter your OTRS ID and your password.



OTRS-ID Login

You need to log in with your OTRS-ID to register your system. Your OTRS-ID is the email address you used to sign up on the OTRS.com webpage. Data Protection

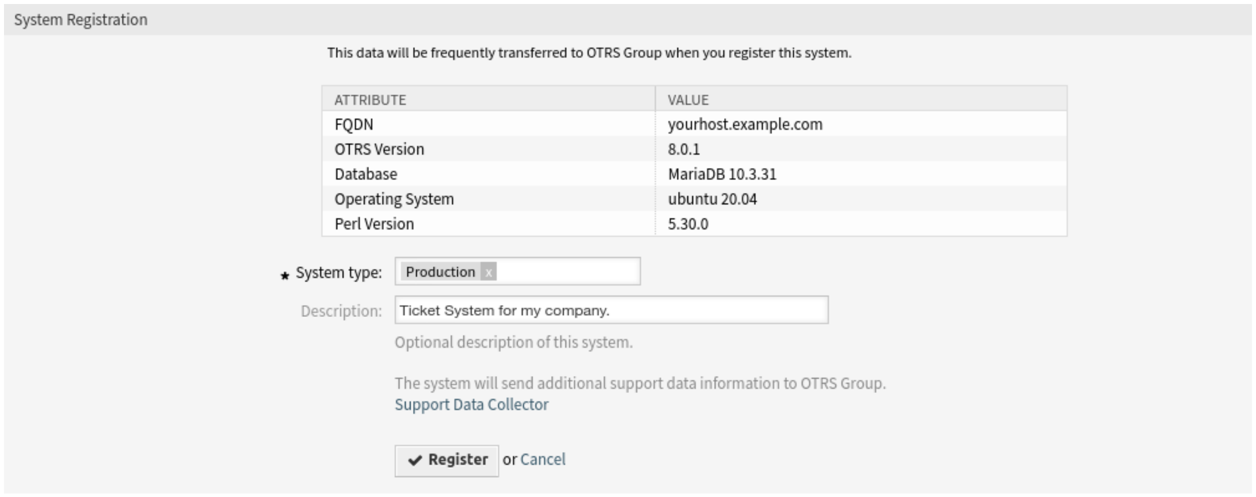
★ OTRS-ID:	john.smith@example.com
★ Password:	••••••••••••••••••••••••••••••••••••

You don't have an OTRS-ID yet? Sign up now

Forgot your password? Retrieve a new one

Fig. 6: System Registration - Add OTRS ID

3. Click on the *Next* button.
4. Select the system type and enter a description.



System Registration

This data will be frequently transferred to OTRS Group when you register this system.

ATTRIBUTE	VALUE
FQDN	yourhost.example.com
OTRS Version	8.0.1
Database	MariaDB 10.3.31
Operating System	ubuntu 20.04
Perl Version	5.30.0

★ System type:	Production
Description:	Ticket System for my company.

Optional description of this system.

The system will send additional support data information to OTRS Group.
Support Data Collector

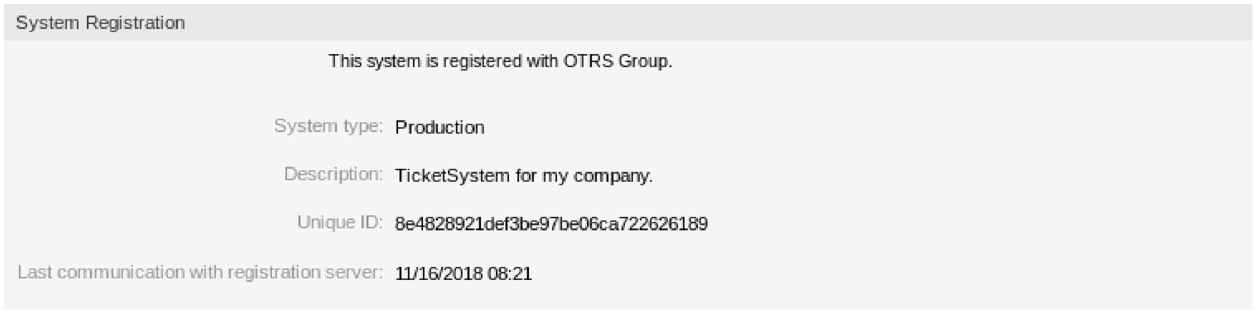
Fig. 7: System Registration - Select System Type

5. Click on the *Register* button.
6. Check your registration.

To edit the system registration:

1. Click on the *Edit details* button in the left sidebar.
2. Modify the system type and the description.
3. Click on the *Update* button.

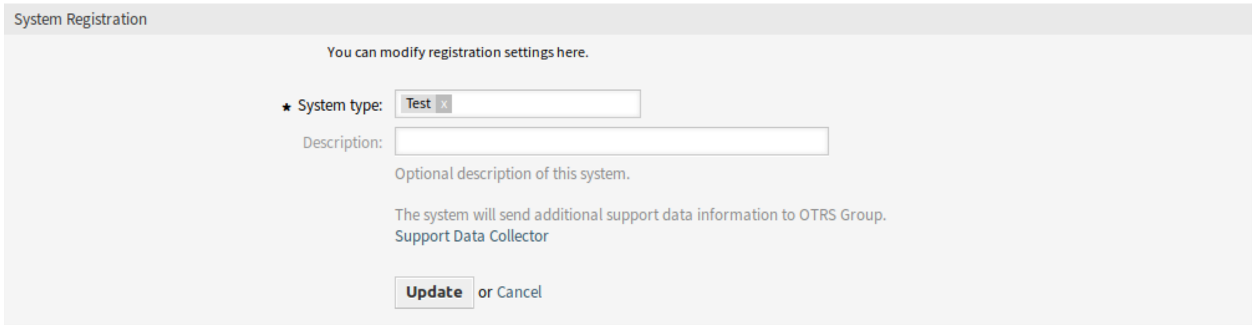
To show the transmitted data:


System Registration

This system is registered with OTRS Group.

System type:	Production
Description:	TicketSystem for my company.
Unique ID:	8e4828921def3be97be06ca722626189
Last communication with registration server:	11/16/2018 08:21

Fig. 8: Registered System


System Registration

You can modify registration settings here.

★ System type:	Test
Description:	

Optional description of this system.

The system will send additional support data information to OTRS Group.
Support Data Collector

Fig. 9: Edit System Registration Screen

1. Click on the *Show transmitted data* button in the left sidebar.
2. Review the data in JSON format.

See also:

Detailed explanation of the fields is available in the *Support Data Collector* chapter.

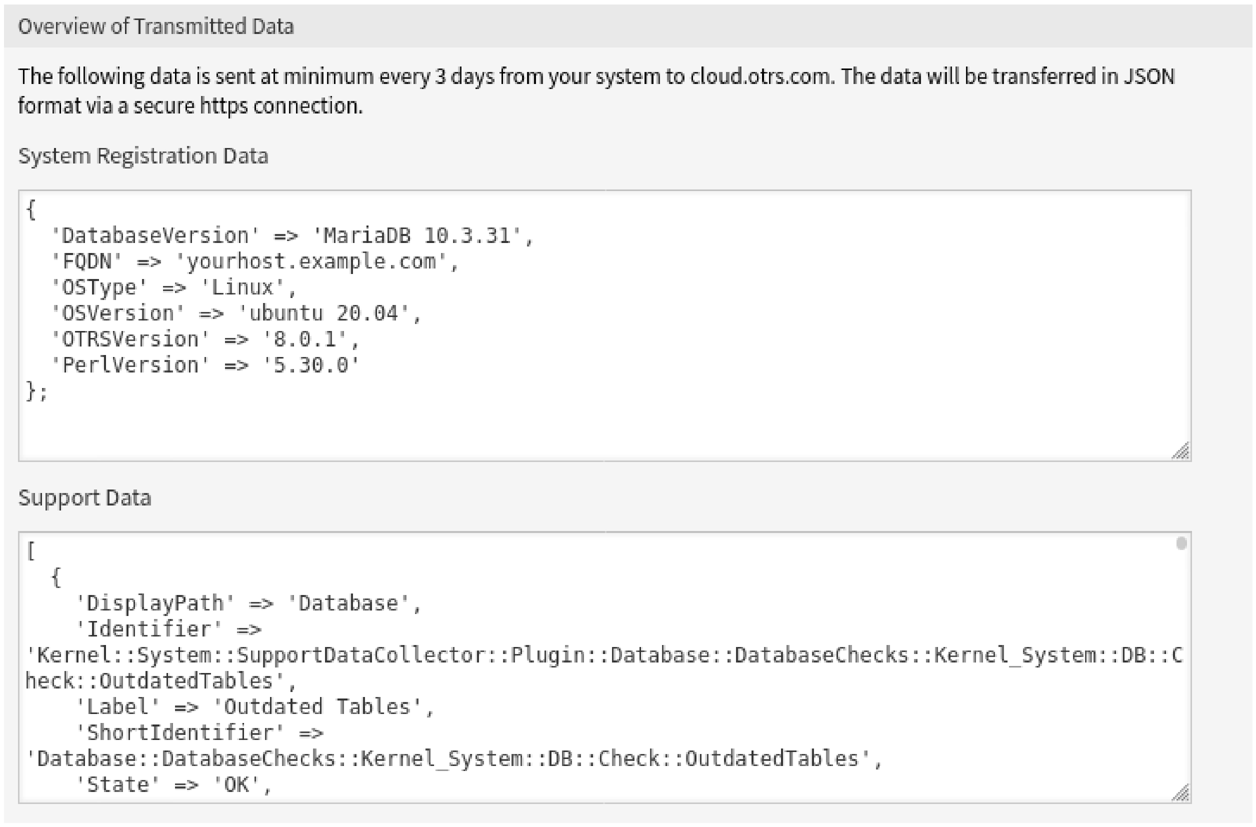
To overview the registered system:

1. Click on the *Overview of registered systems* button in the left sidebar.
2. Log in to the OTRS Portal.
3. Click on the OTRS Systems menu item.
4. Review the list of registered systems.

To deregister a system:

1. Click on the *Deregister system* button in the left sidebar.

Note: You can' t deregister your system if you' re using the **STORM powered by OTRS™** or having a valid service contract.



Overview of Transmitted Data

The following data is sent at minimum every 3 days from your system to cloud.otrs.com. The data will be transferred in JSON format via a secure https connection.

System Registration Data

```
{
  'DatabaseVersion' => 'MariaDB 10.3.31',
  'FQDN' => 'yourhost.example.com',
  'OSType' => 'Linux',
  'OSVersion' => 'ubuntu 20.04',
  'OTRSVersion' => '8.0.1',
  'PerlVersion' => '5.30.0'
};
```

Support Data

```
[
  {
    'DisplayPath' => 'Database',
    'Identifier' =>
'Kernel::System::SupportDataCollector::Plugin::Database::DatabaseChecks::Kernel_System::DB::C
heck::OutdatedTables',
    'Label' => 'Outdated Tables',
    'ShortIdentifier' =>
'Database::DatabaseChecks::Kernel_System::DB::Check::OutdatedTables',
    'State' => 'OK',
```

Fig. 10: Overview of Transmitted Data Screen

CHAPTER

NINE

CHAT

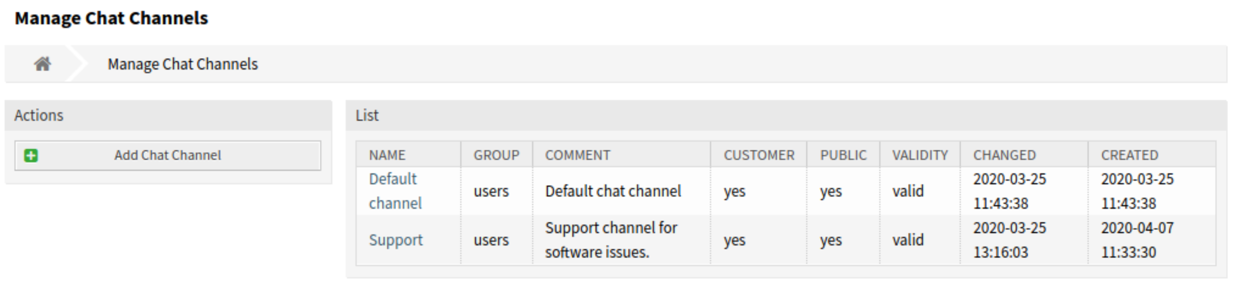
The chat feature allows the agents to provide real time communication with each other, with customer users or with public users. The chat feature is enabled by default.

9.1 Chat Channels

Being able to offer chat possibilities to customers is a must-have for many organizations. Depending on the amount of customer chat requests and the organization's structure, it must be possible to group chat requests.

OTRS offers chat channels with different permissions per channel, so it is, e.g. possible to have different chat channels for registered contract customers and public prospects.

Use this screen to add chat channels to the system. A fresh OTRS installation already contains a chat channel by default. The chat channel management screen is available in the *Chat Channels* module of the *Chat* group.



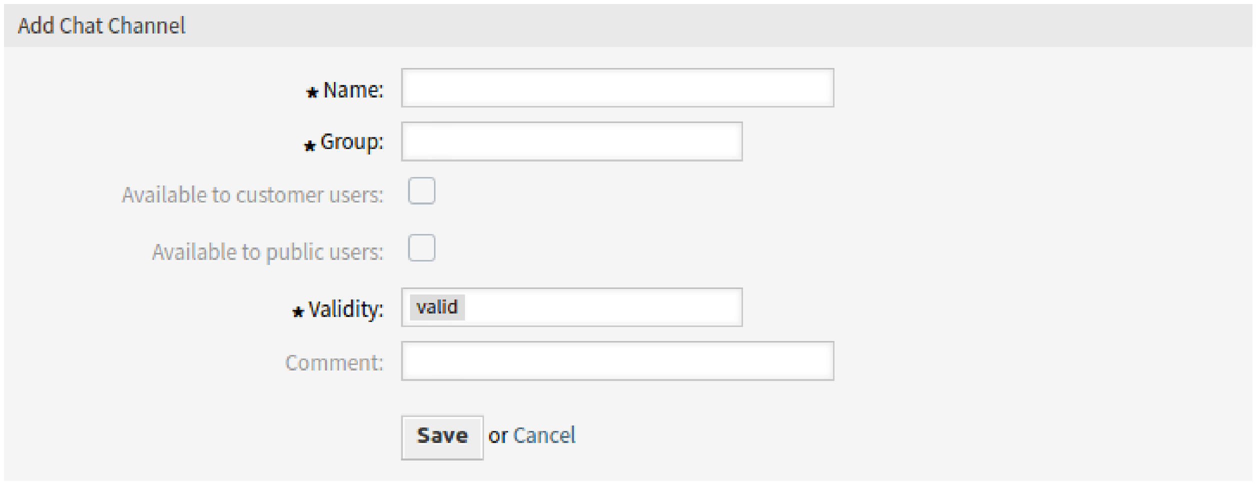
NAME	GROUP	COMMENT	CUSTOMER	PUBLIC	VALIDITY	CHANGED	CREATED
Default channel	users	Default chat channel	yes	yes	valid	2020-03-25 11:43:38	2020-03-25 11:43:38
Support	users	Support channel for software issues.	yes	yes	valid	2020-03-25 13:16:03	2020-04-07 11:33:30

Fig. 1: Chat Channel Management Screen

9.1.1 Manage Chat Channels

To add a chat channel:

1. Click on the *Add Chat Channel* button in the left sidebar.
2. Fill in the required fields.
3. Click on the *Save* button.


Add Chat Channel

★ Name:	
★ Group:	
Available to customer users:	☐
Available to public users:	☐
★ Validity:	valid
Comment:	

Fig. 2: Add Chat Channel Screen

Warning: Chat channels can not be deleted from the system. They can only be deactivated by setting the *Validity* option to *invalid* or *invalid-temporarily*.

To edit a chat channel:

1. Click on a chat channel in the list of chat channels.
2. Modify the fields.
3. Click on the *Save* or *Save and finish* button.

9.1.2 Chat Channel Settings

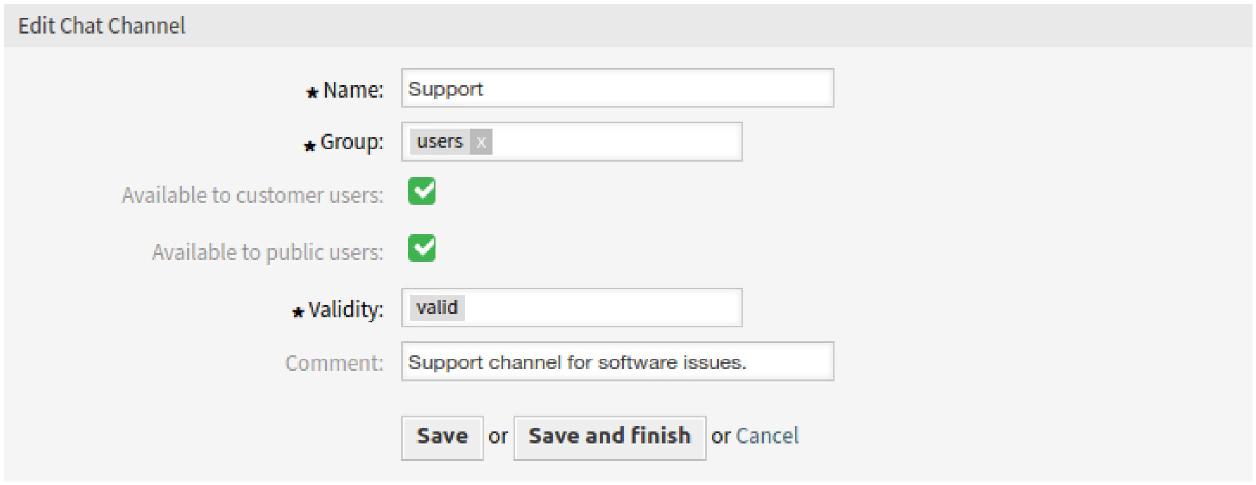
The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

Name *
The name of this resource. Any type of characters can be entered to this field including uppercase letters and spaces. The name will be displayed in the overview table.

Group *
Select which *Groups* can access the chat channel.

Available to customer users
Select the checkbox if you want to display the chat channel for customer users.

Available to public users
Select the checkbox if you want to display the chat channel for public users.


Edit Chat Channel

★ Name:	Support
★ Group:	users
Available to customer users:	✓
Available to public users:	✓
★ Validity:	valid
Comment:	Support channel for software issues.

Fig. 3: Edit Chat Channel Screen

Validity *
: Set the validity of this resource. Each resource can be used in OTRS only, if this field is set to *valid*. Setting this field to *invalid* or *invalid-temporarily* will disable the use of the resource.

Comment
: Add additional information to this resource. It is recommended to always fill this field as a description of the resource with a full sentence for better clarity, because the comment will be also displayed in the overview table.

9.1.3 Manage Chat Support

Note: The chat feature is enabled by default.

To enable or disable the chat feature:

1. Go to the *System Configuration* screen.
2. Search for the setting `ChatEngine::Active`.
3. Enable or disable the setting.

When the chat feature is enabled, the *Default channel* is automatically created if it does not exist.

To review all chat related settings:

1. Go to the *System Configuration* screen.
2. Search for the term `ChatEngine` in the search box.
3. Review the settings.

See also:

Agents need to have at least *ro* permissions to the groups set up in the following settings:

- `ChatEngine::PermissionGroup::ChatReceivingAgents`
- `ChatEngine::PermissionGroup::ChatStartingAgents`

9.2 Chat Integration

You can easily integrate the public chat into a website. Just copy the code snippets from here. If you would like to make any adaptations, please modify the fields in the configuration widget. All changes are applied immediately to the snippet, so you can make your changes and copy the snippet to your clipboard afterwards. Please note that changes made on this page are not remembered.

Use this screen to generate the chat integration code. The chat integration configuration and snippet generator is available in the *Chat Integration* module of the *Chat* group.

To configure the chat and generate code snippet:

1. Select a pre-selected chat channel in the *Pre-selected channel* field.
2. Select a primary color for the chat design.
3. Customize the texts in the *Texts* section.

   Note: All changes made in the *Configuration* tab will not be saved. Every time you open this dialog, all fields will be reset to default values.

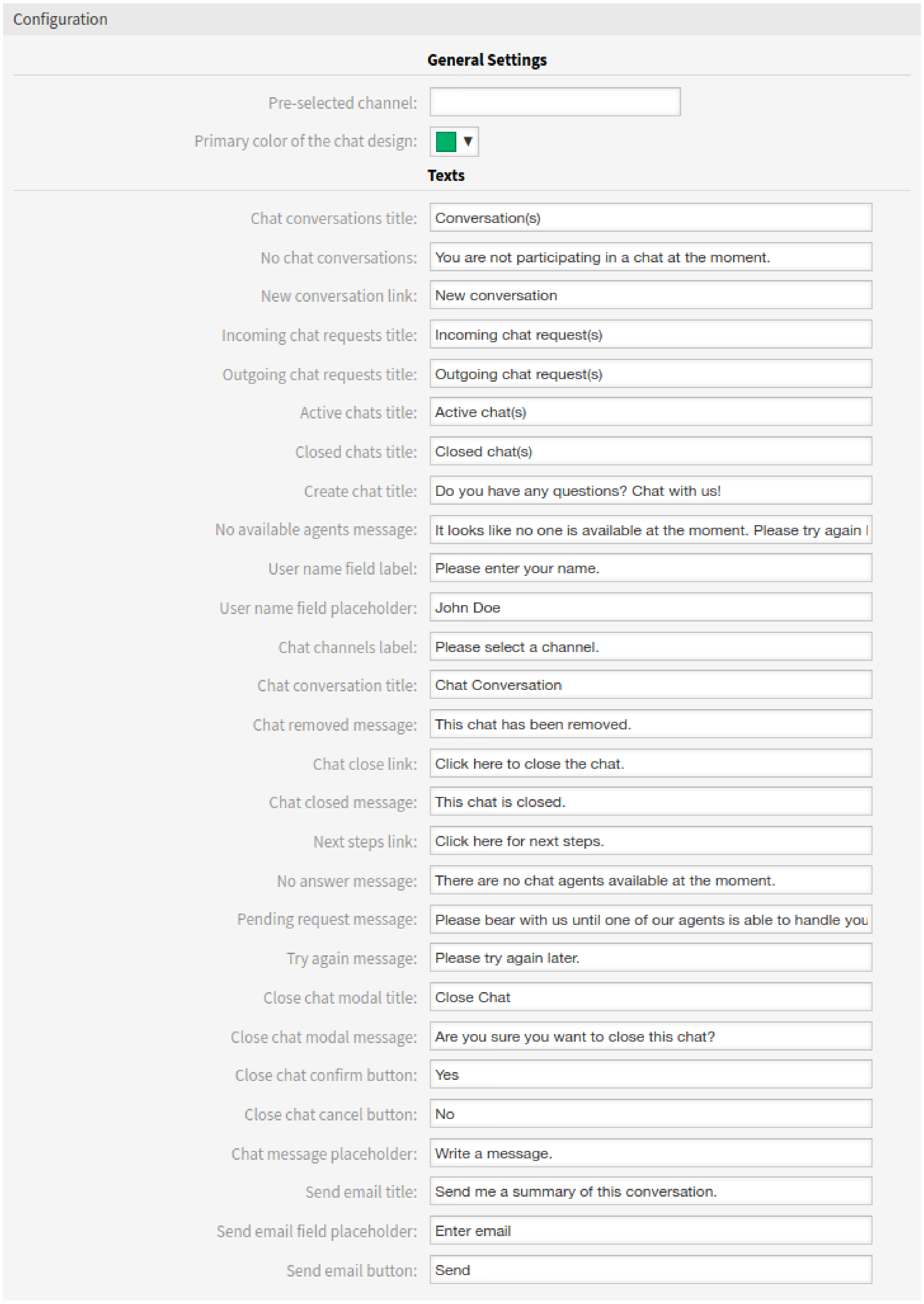
4. You can check the result in the *Preview* widget.

   Note: The preview uses the real chat module. Other agents have to be available for chatting to preview all features.

5. Copy the code snippet from the *Integration Code* widget and paste it into your website right before the `</body>` element.

If mixed content warning is displayed in the browser console, an administrator has to check that the system configuration setting `HttpType` is properly set. The website must run on the same protocol for chat widget to work.

For example, if the website is running OTRS on SSL, the system configuration option must be set to `https`.


Configuration

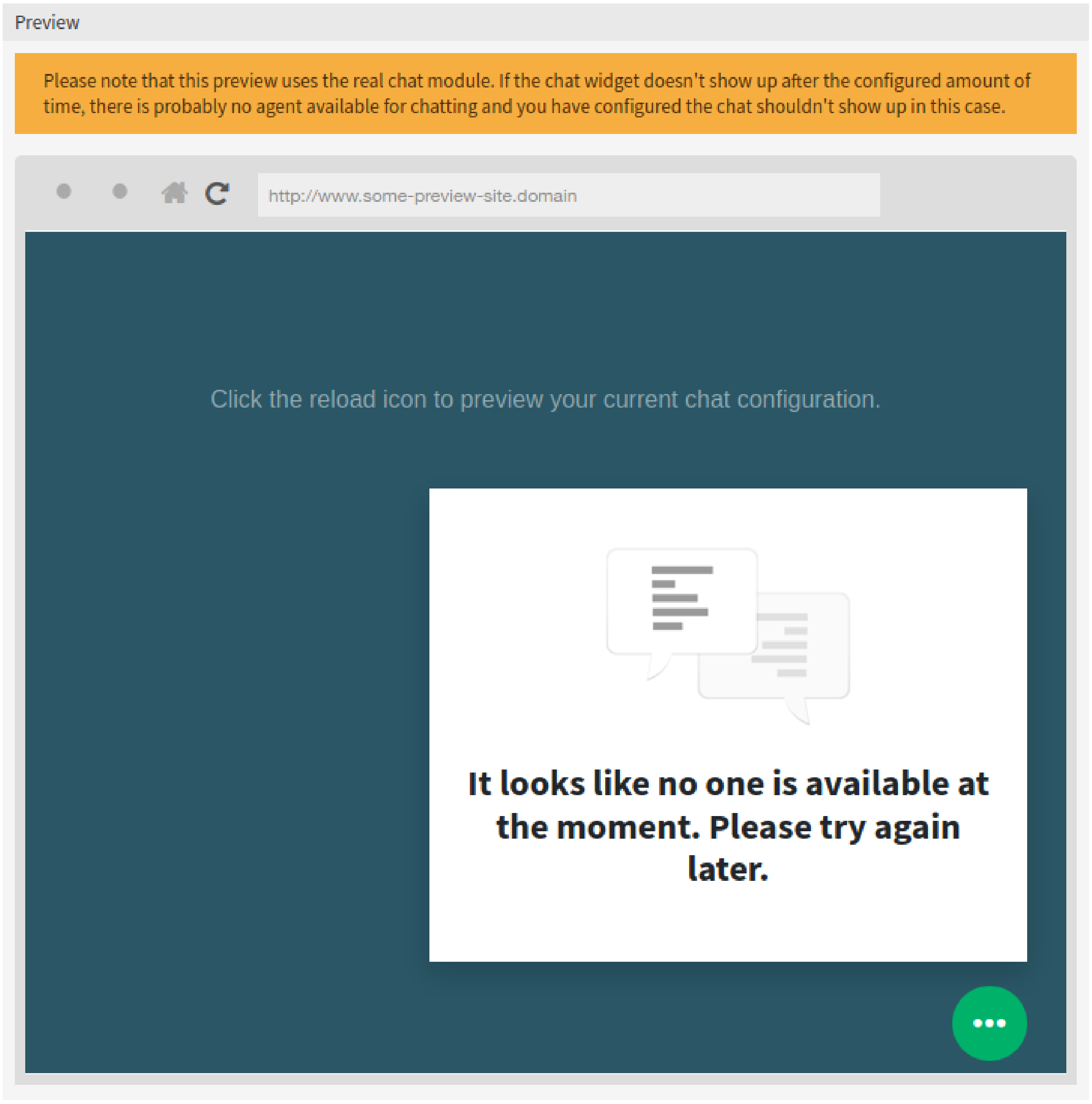
General Settings

Pre-selected channel:	
Primary color of the chat design:	🟩 ▼

Texts

Chat conversations title:	Conversation(s)
No chat conversations:	You are not participating in a chat at the moment.
New conversation link:	New conversation
Incoming chat requests title:	Incoming chat request(s)
Outgoing chat requests title:	Outgoing chat request(s)
Active chats title:	Active chat(s)
Closed chats title:	Closed chat(s)
Create chat title:	Do you have any questions? Chat with us!
No available agents message:	It looks like no one is available at the moment. Please try again l
User name field label:	Please enter your name.
User name field placeholder:	John Doe
Chat channels label:	Please select a channel.
Chat conversation title:	Chat Conversation
Chat removed message:	This chat has been removed.
Chat close link:	Click here to close the chat.
Chat closed message:	This chat is closed.
Next steps link:	Click here for next steps.
No answer message:	There are no chat agents available at the moment.
Pending request message:	Please bear with us until one of our agents is able to handle you
Try again message:	Please try again later.
Close chat modal title:	Close Chat
Close chat modal message:	Are you sure you want to close this chat?
Close chat confirm button:	Yes
Close chat cancel button:	No
Chat message placeholder:	Write a message.
Send email title:	Send me a summary of this conversation.
Send email field placeholder:	Enter email
Send email button:	Send

Fig. 4: Chat Integration –Configuration


Preview

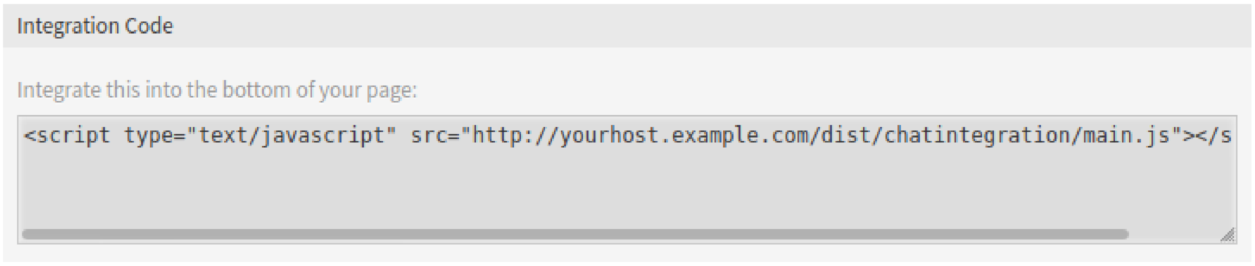
Please note that this preview uses the real chat module. If the chat widget doesn't show up after the configured amount of time, there is probably no agent available for chatting and you have configured the chat shouldn't show up in this case.

http://www.some-preview-site.domain

Click the reload icon to preview your current chat configuration.

It looks like no one is available at the moment. Please try again later.

Fig. 5: Chat Integration –Preview


Integration Code

Integrate this into the bottom of your page:

```
<script type="text/javascript" src="http://yourhost.example.com/dist/chatintegration/main.js"></s
```

Fig. 6: Chat Integration –Integration Code

CHAPTER

TEN

ADMINISTRATION

Any system requires configuration. Configuring a system should be an easy task and the tools for configuration fit-for-purpose.

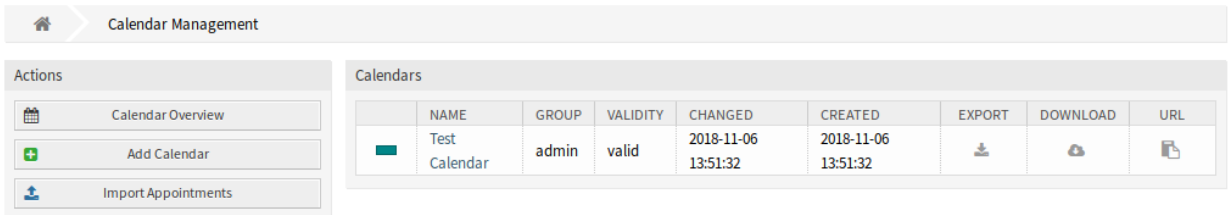
OTRS offers several administration tools to configure, monitor, control and extend OTRS.

10.1 Calendars

When working with customers, resource planning and scheduling can be a complex task. Appointments enable you to meet your customers where and whenever needed.

OTRS supports this requirement with calendars. Calendars allow management of appointments and resources inside the ticket system. You can connect your tickets to scheduled tasks and make them available to all users to see. This feature adds transparency to show your teams workload and prevent users from promising resources which are not available.

Use this screen to manage calendars in the system. A fresh OTRS installation contains no calendars by default. The calendar management screen is available in the *Calendars* module of the *Administration* group.



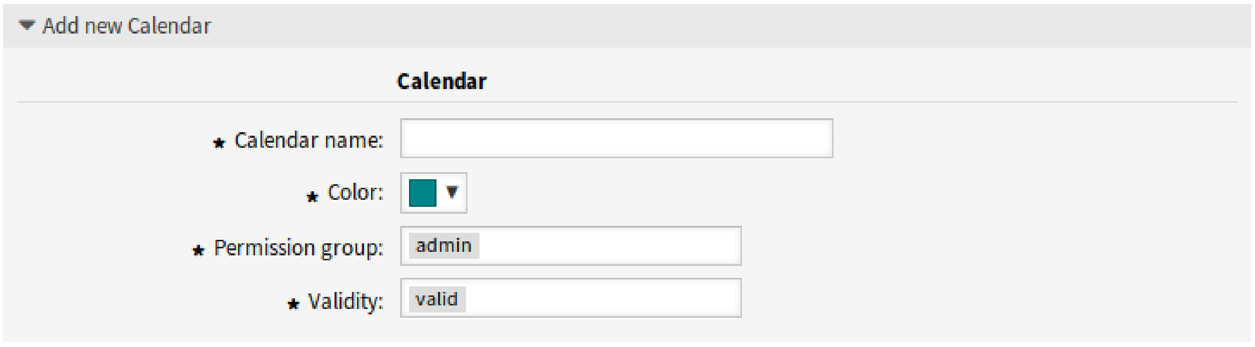
	NAME	GROUP	VALIDITY	CHANGED	CREATED	EXPORT	DOWNLOAD	URL
	Test Calendar	admin	valid	2018-11-06 13:51:32	2018-11-06 13:51:32			

Fig. 1: Calendar Management Screen

10.1.1 Manage Calendars

To add a new calendar:

1. Click on the *Add Calendar* button in the left sidebar.
2. Fill in the required fields.
3. Click on the *Save* button.



Add new Calendar

Calendar

★ Calendar name:	
★ Color:	🟦 ▼
★ Permission group:	admin
★ Validity:	valid

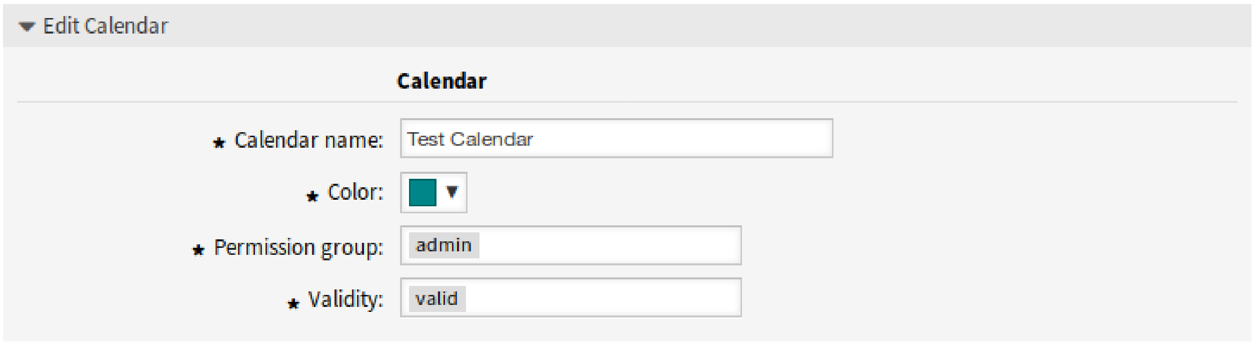
Fig. 2: Add New Calendar Screen

Warning: Calendars can not be deleted from the system. They can only be deactivated by setting the *Validity* option to *invalid* or *invalid-temporarily*.

Warning: The maximum number of 50 *valid* calendars should not be exceeded. Exceeding this limit may affect the system performance.

To edit a calendar:

1. Click on a calendar in the list of calendars.
2. Modify the fields.
3. Click on the *Save* or *Save and finish* button.



Edit Calendar

Calendar

★ Calendar name:	Test Calendar
★ Color:	🟦 ▼
★ Permission group:	admin
★ Validity:	valid

Fig. 3: Edit Calendar Screen

To export a calendar:

1. Click on the export icon in the list of calendars.
2. Choose a location in your computer to save the `Export_Calendar_CalendarName.yml` file.

To import calendars:

1. Click on the *Browse*··· button in the left sidebar.
2. Select a previously exported `.yml` file.
3. Click on the *Overwrite existing entities* checkbox, if you would like to overwrite the existing calendars.

4. Click on the *Import Calendar* button.

Note: If several calendars are added to the system, use the filter box to find a particular calendar by just typing the name to filter.

10.1.2 Calendar Settings

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

General Calendar Settings

Calendar name *
The name of this resource. Any type of characters can be entered to this field including uppercase letters and spaces. The name will be displayed in the overview table.

Color *
The calendar color that will be displayed in the calendar overview screens.

To change the calendar color, just select a new color from the color palette. You can chose from the pre-selected colors or define other colors by choosing it from the color selector or typing the hexadecimal value.

Permission group *
Select which *Groups* can access the calendar.

Depending on the group field, the system will allow users the access to the calendar according to their permission level.

- Read only: users can see and export all appointments in the calendar.
- Move into: users can modify appointments in the calendar, but without changing the calendar selection.
- Create: users can create and delete appointments in the calendar.
- Read/write: users can manage the calendar itself.

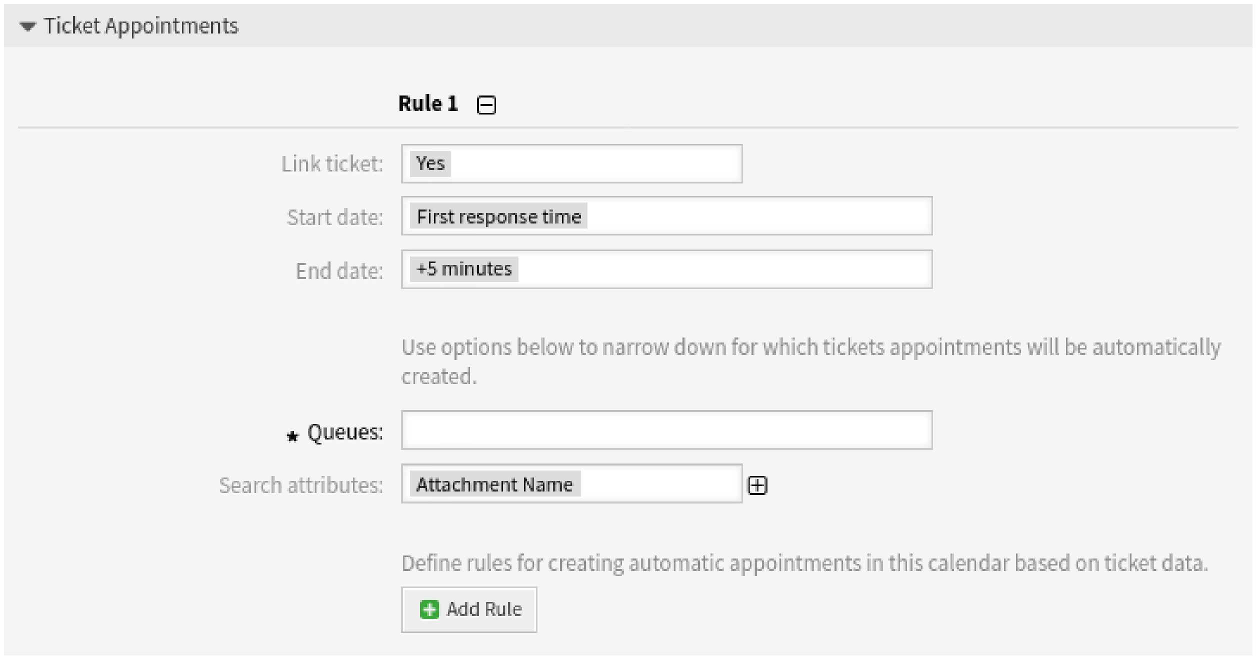
Validity *
Set the validity of this resource. Each resource can be used in OTRS only, if this field is set to *valid*. Setting this field to *invalid* or *invalid-temporarily* will disable the use of the resource.

Calendar Ticket Appointments

Define rules for creating automatic appointments in this calendar based on ticket data. To add a new rule, click on the *Add Rule* button.

Link ticket
If set to *Yes*, a clickable link to the ticket will be displayed in the title of the appointment and the created appointment will be linked to the ticket. If set to *No*, the ticket number will be displayed as plain text in the title of the appointment and the appointment will not be linked to the ticket.

Start date
Select a start date for the ticket appointment.


Ticket Appointments

Rule 1

Link ticket: Yes

Start date: First response time

End date: +5 minutes

Use options below to narrow down for which tickets appointments will be automatically created.

* Queues:

Search attributes: Attachment Name

Define rules for creating automatic appointments in this calendar based on ticket data.

Fig. 4: Calendar Settings - Ticket Appointments

End date
Select the end date for the ticket appointment.

Queues *
Select one or more queues to narrow down for which tickets appointments will be automatically created.

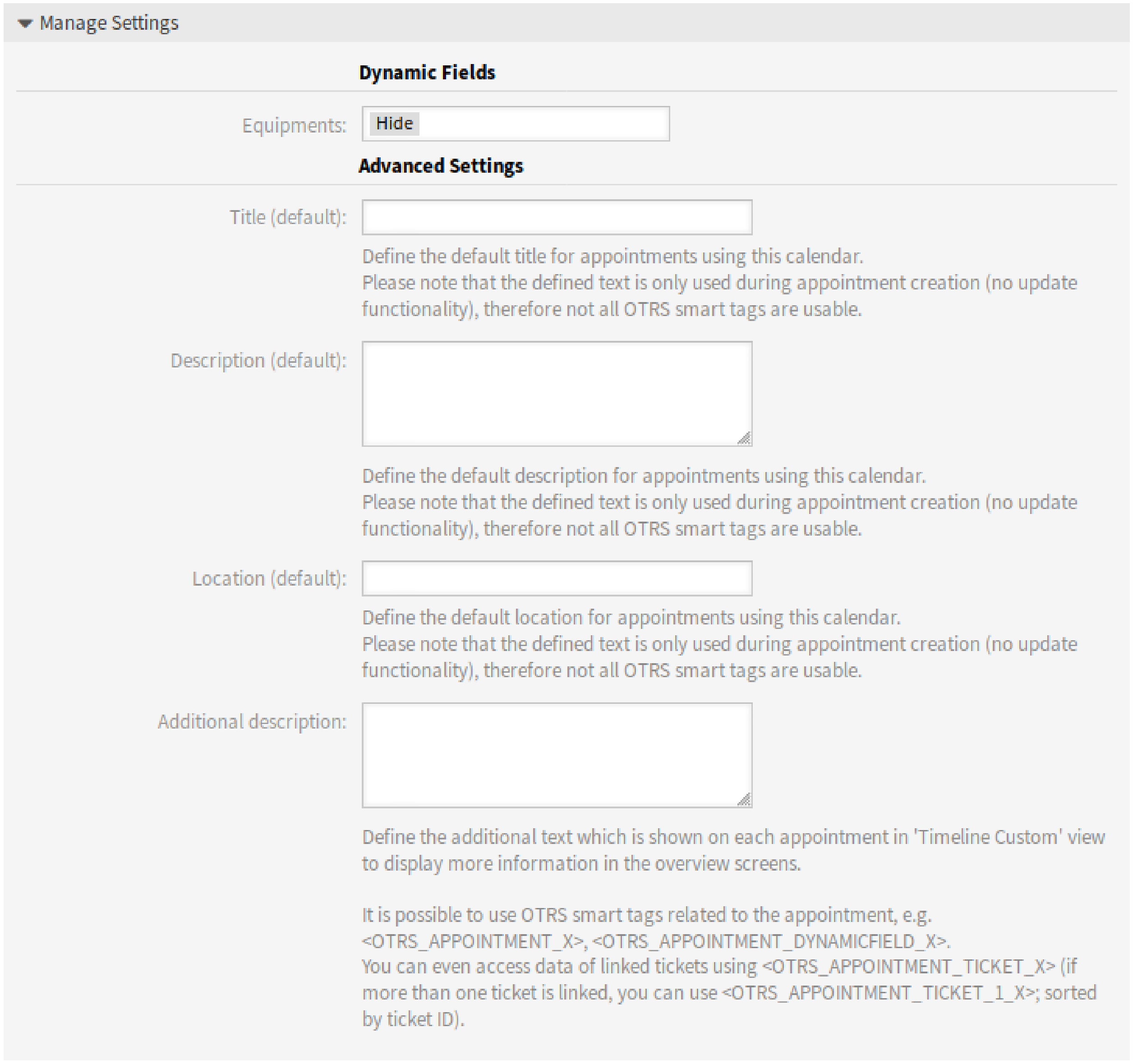
Search attributes
Additional search attributes can be added for further filtering by selecting an attribute and clicking on the button.

By default, these events can trigger an update of a calendar appointment:

- `TicketSLAUpdate`
- `TicketQueueUpdate`
- `TicketStateUpdate`
- `TicketCreate`
- `ArticleCreate`
- `TicketPendingTimeUpdate`
- `TicketDynamicFieldUpdate_.*`

Manage Settings

Use this section to add default values to appointments.



Manage Settings

Dynamic Fields

Equipments: Hide

Advanced Settings

Title (default):

Define the default title for appointments using this calendar.
Please note that the defined text is only used during appointment creation (no update functionality), therefore not all OTRS smart tags are usable.

Description (default):

Define the default description for appointments using this calendar.
Please note that the defined text is only used during appointment creation (no update functionality), therefore not all OTRS smart tags are usable.

Location (default):

Define the default location for appointments using this calendar.
Please note that the defined text is only used during appointment creation (no update functionality), therefore not all OTRS smart tags are usable.

Additional description:

Define the additional text which is shown on each appointment in 'Timeline Custom' view to display more information in the overview screens.

It is possible to use OTRS smart tags related to the appointment, e.g. <OTRS_APPOINTMENT_X>, <OTRS_APPOINTMENT_DYNAMICFIELD_X>.
You can even access data of linked tickets using <OTRS_APPOINTMENT_TICKET_X> (if more than one ticket is linked, you can use <OTRS_APPOINTMENT_TICKET_1_X>; sorted by ticket ID).

Fig. 5: Manage Settings Widget

Each appointment dynamic fields are listed in this section. This feature controls only the visibility of the dynamic fields. Possible options:

Hide
The dynamic field is hidden in the forms and in the appointments.

Show
The dynamic field is shown in the forms and in the appointments, but it can have an empty value.

Show as mandatory
The dynamic field is shown in the forms and in the appointments, and it does not allow an empty value.

Use form configuration
The display of the dynamic field will follow the current form configuration, that is defined in the following system configuration settings:

```
Forms###AgentFrontend::Calendar::AppointmentCreate::Properties
Forms###AgentFrontend::Calendar::AppointmentUpdate::Properties
```

In case the *Hide*, *Show* or *Show as mandatory* options are used, form configuration will be effectively ignored, and the dynamic field configuration from the calendar will be applied.

Note: This section is visible only, if *Dynamic Fields* for appointments are added to the system.

Title (default)
Define the default title for appointments using this calendar.

Description (default)
Define the default description for appointments using this calendar.

Location (default)
Define the default location for appointments using this calendar.

Additional description
Define the additional text which is shown on each appointment in *Timeline Custom* view to display more information in the overview screens. It is possible to show attributes of the appointments, as well as stored dynamic fields. Even the access to linked ticket data is possible.

It is possible to use OTRS smart tags related to the appointment, e.g. `<OTRS_APPOINTMENT_X>`, `<OTRS_APPOINTMENT_DYNAMICFIELD_X>`. You can even access data of linked tickets using `<OTRS_APPOINTMENT_TICKET_X>`. If more than one ticket is linked, you can use `<OTRS_APPOINTMENT_TICKET_1_X>`. Tickets are sorted by ticket ID.

Note: The appointment is not created at this time and not all values are there (e.g. no dynamic field values can be used). The defined texts are only used during appointment creation (no update functionality), therefore not all OTRS smart tags are usable.

The default values are populated whenever the calendar is chosen and the title, description or location is empty.

Behind the input fields for title, description and location, there is a refresh button if default values are specified. This refresh button could be used to re-apply the stored default values if for example dynamic fields are entered in the screen.

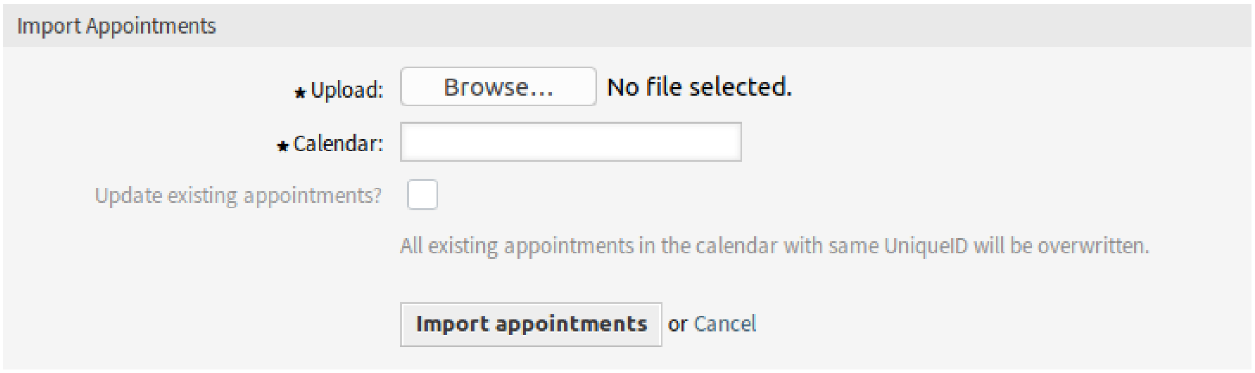
Warning: The already entered data will be overwritten using the refresh buttons.

10.1.3 Import Appointments

If at least one calendar have been added to the system, it is possible to import some appointments into the calendar.

To import some appointments:

1. Click on the *Import Appointments* button in the left sidebar.
2. Upload an iCal file and select a calendar.
3. Click on the *Import appointments* button.


Import Appointments

⋆ Upload:	Browse… No file selected.
⋆ Calendar:	
Update existing appointments?	☐
	All existing appointments in the calendar with same UniqueID will be overwritten.

Fig. 6: Import Appointments Screen

Upload *
Click on the *Browse*··· button, and select a valid iCal (`.ics`) file to upload.

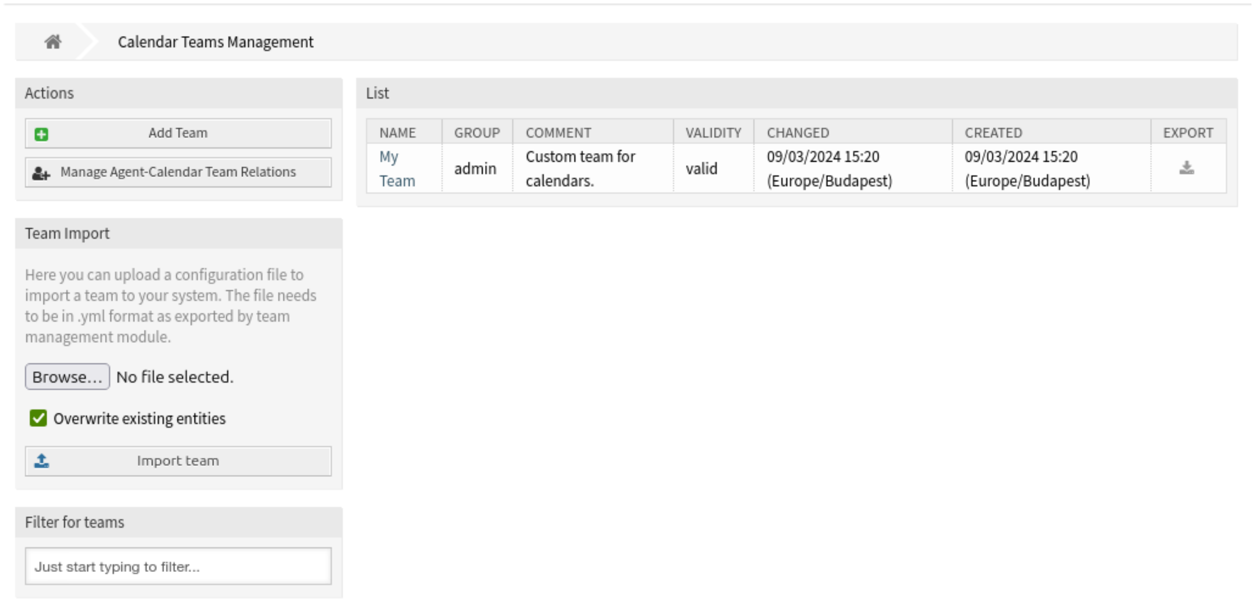
Calendar *
Select an available calendar.

> **Note:** If desired calendar is not listed here, please make sure that you have at least *create* permissions.

Update existing appointments?
If checked, all existing appointments in the calendar with same `UniqueID` will be overwritten.

10.2 Calendar Teams

Use this screen to manage calendar teams. The calendar teams management screen is available in the *Calendar Teams* module of the *Administration* group. Additionally, it is also accessible from the *Agents Calendar Teams* screen.


Calendar Teams Management
Actions
Add Team
Manage Agent-Calendar Team Relations
Team Import
Browse… No file selected.
Overwrite existing entities
Import team
Filter for teams
Just start typing to filter...
List

Here you can upload a configuration file to import a team to your system. The file needs to be in .yml format as exported by team management module.

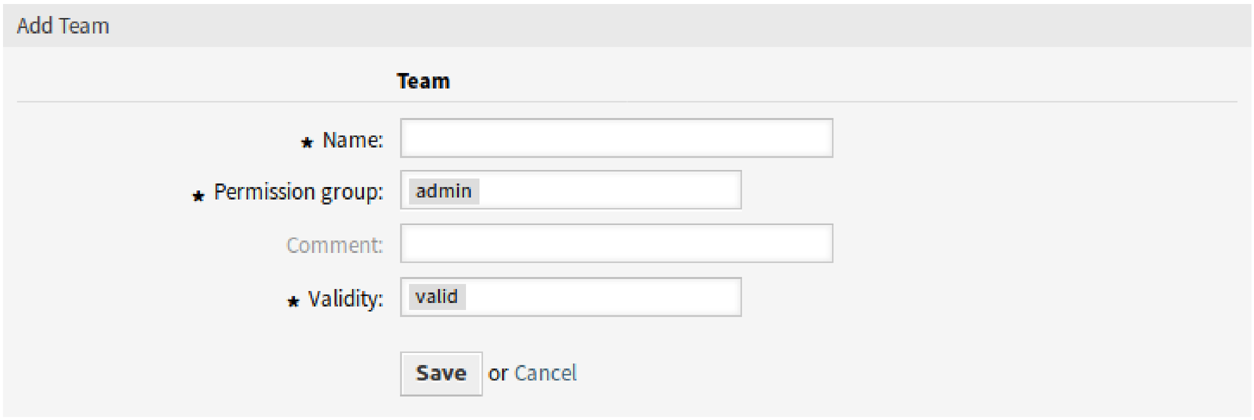
NAME	GROUP	COMMENT	VALIDITY	CHANGED	CREATED	EXPORT
My Team	admin	Custom team for calendars.	valid	09/03/2024 15:20 (Europe/Budapest)	09/03/2024 15:20 (Europe/Budapest)	

Fig. 7: Calendar Teams Management Screen

10.2.1 Manage Calendar Teams

To create a new team:

1. Click on the *Add Team* button in the left sidebar.
2. Fill in the required fields.
3. Click on the *Save* button.



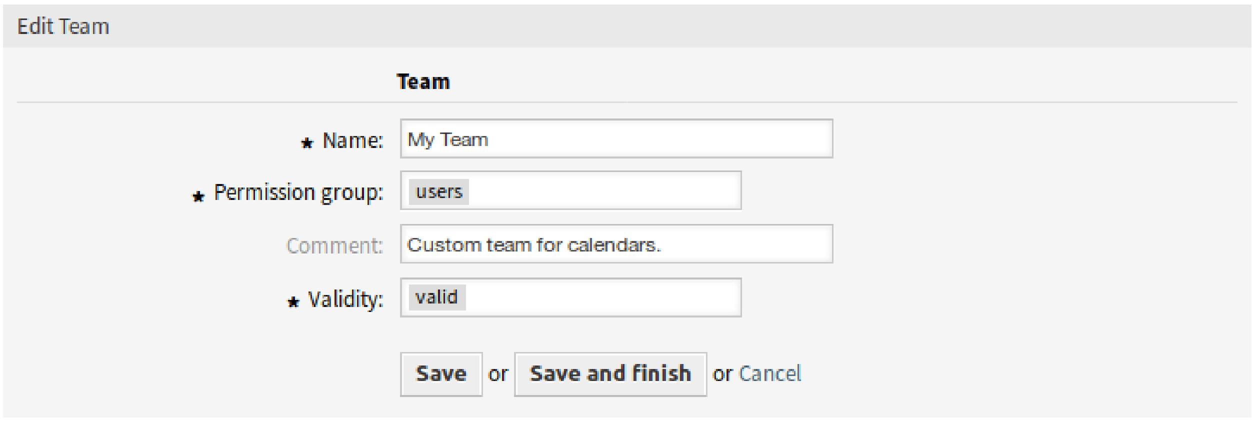
⋆ Name:	
⋆ Permission group:	admin
Comment:	
⋆ Validity:	valid

Fig. 8: Add Team Screen

Warning: Teams can not be deleted from the system. They can only be deactivated by setting the *Validity* option to *invalid* or *invalid-temporarily*.

To edit a team:

1. Click on a team in the list of teams.
2. Modify the fields.
3. Click on the *Save* or *Save and finish* button.


Edit Team

Team

★ Name:	My Team
★ Permission group:	users
Comment:	Custom team for calendars.
★ Validity:	valid

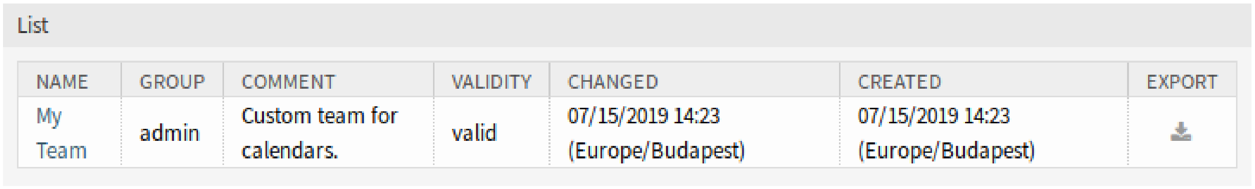
Fig. 9: Edit Team Screen

To manage team agents:

1. Click on the *Manage Agent-Calendar Team Relations* button.
2. Assign agents to teams and vice versa in the *Agents Calendar Teams* screen.

To export a team:

1. Click on the export icon in the last column of the overview table.
2. Choose a location in your computer to save the `Export_Team_Team_name.yml` file.


List

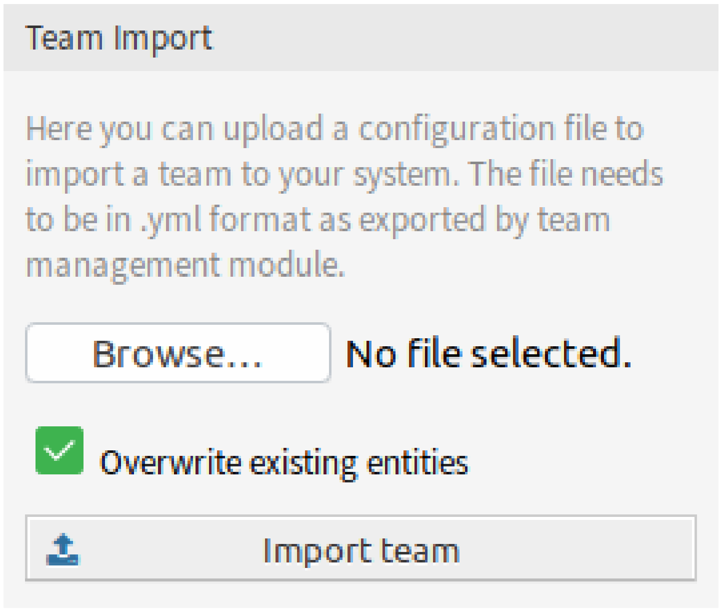
NAME	GROUP	COMMENT	VALIDITY	CHANGED	CREATED	EXPORT
My Team	admin	Custom team for calendars.	valid	07/15/2019 14:23 (Europe/Budapest)	07/15/2019 14:23 (Europe/Budapest)	

Fig. 10: Export Team Screen

To import a team:

1. Click on the *Browse*… button of the *Team Import* widget in the left sidebar.
2. Select a previously exported `.yml` file.
3. Click on the *Overwrite existing entities* checkbox, if you would like to overwrite the existing teams.
4. Click on the *Import team* button.

Note: If several teams are added to the system, use the filter box to find a particular team by just typing the name to filter.


Team Import

Here you can upload a configuration file to import a team to your system. The file needs to be in .yml format as exported by team management module.

Fig. 11: Team Import Widget

10.2.2 Team Settings

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

Name *
: The name of this resource. Any type of characters can be entered to this field including uppercase letters and spaces. The name will be displayed in the overview table.

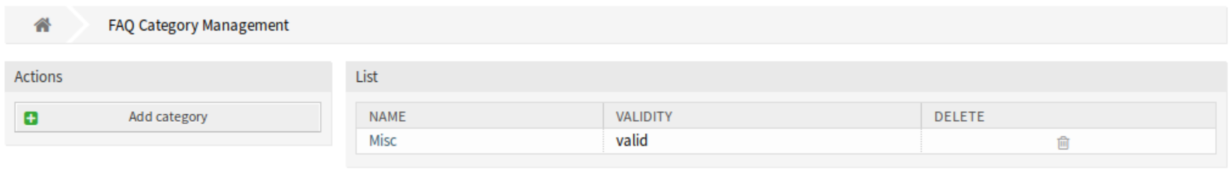
Permission group *
: Agent groups that can manage the team. Full access to this group is required for anyone to be able to manage the team and its agents, as well as assign them as resources to appointments.

Comment
: Add additional information to this resource. It is recommended to always fill this field as a description of the resource with a full sentence for better clarity, because the comment will be also displayed in the overview table.

Validity *
: Set the validity of this resource. Each resource can be used in OTRS only, if this field is set to *valid*. Setting this field to *invalid* or *invalid-temporarily* will disable the use of the resource.

10.3 FAQ Category

Use this screen to manage categories available in the knowledge base articles. A fresh OTRS installation already contains a category by default. The category management screen is available in the *FAQ Category* module of the *Administration* group.



NAME	VALIDITY	DELETE
Misc	valid	🗑

Fig. 12: FAQ Category Management Screen

10.3.1 Manage FAQ Categories

To add a category:

1. Click on the *Add category* button in the left sidebar.
2. Fill in the required fields.
3. Click on the *Submit* button.

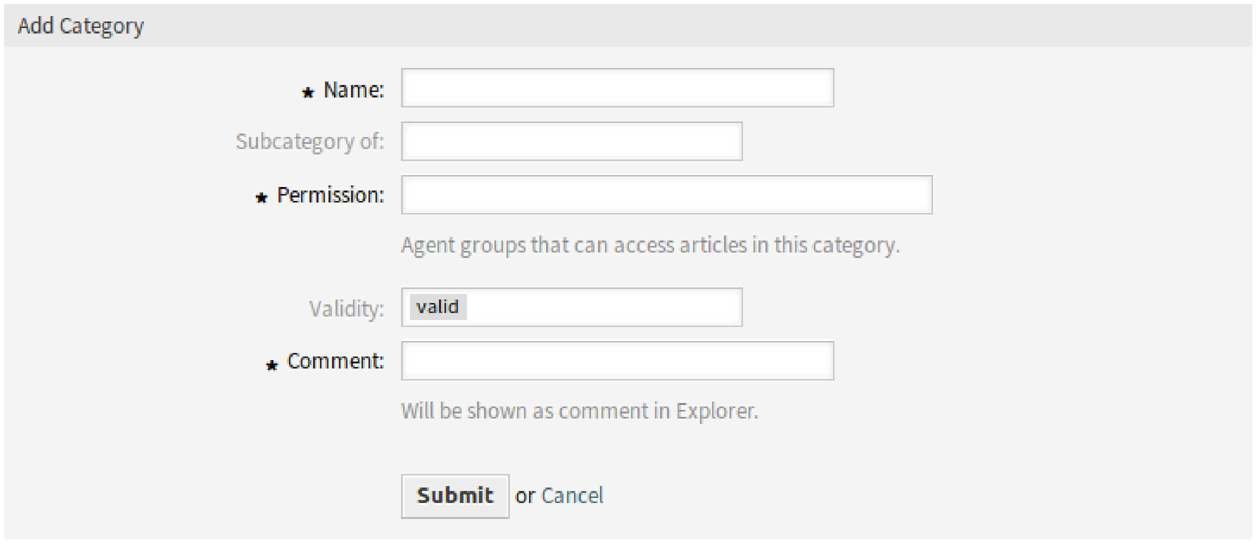

Add Category
★ Name:
Subcategory of:
★ Permission:
Agent groups that can access articles in this category.
Validity: valid
★ Comment:
Will be shown as comment in Explorer.
Submit or Cancel

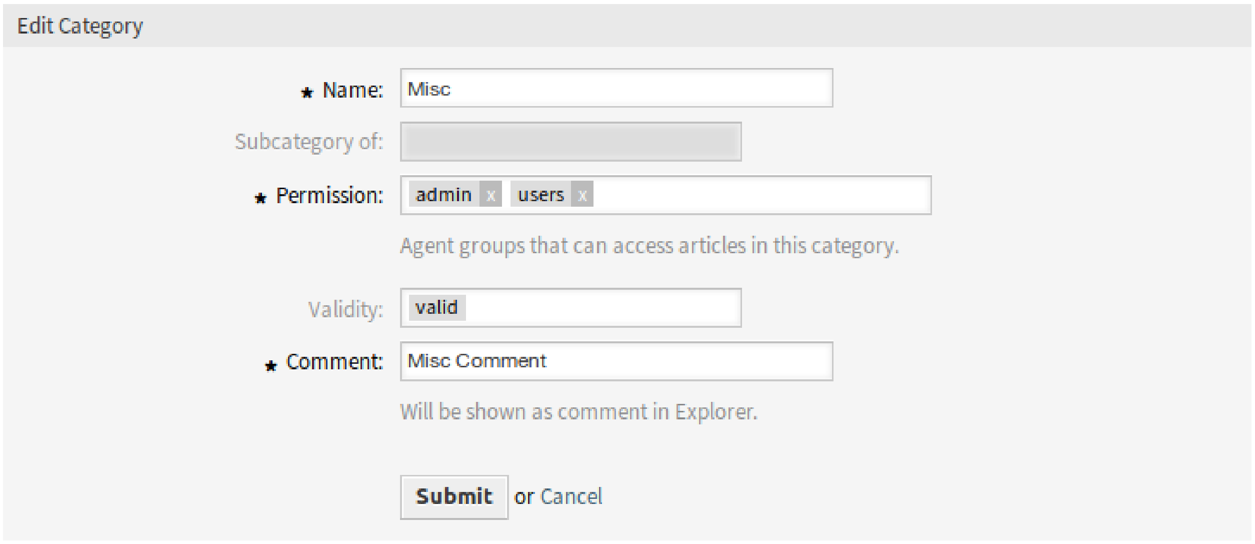
Fig. 13: Add Category Screen

To edit a category:

1. Click on a category in the list of categories.
2. Modify the fields.
3. Click on the *Submit* button.

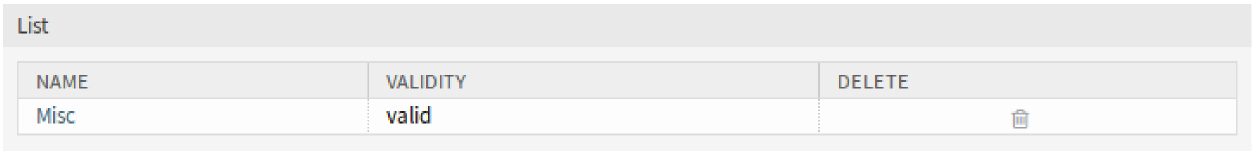
To delete a category:

1. Click on the trash icon in the list of categories.
2. Click on the *Yes* button in the confirmation dialog.


Edit Category

★ Name:	Misc
Subcategory of:	
★ Permission:	admin x users x
	Agent groups that can access articles in this category.
Validity:	valid
★ Comment:	Misc Comment
	Will be shown as comment in Explorer.

Fig. 14: Edit Category Screen


List

NAME	VALIDITY	DELETE
Misc	valid	🗑

Fig. 15: Delete Category Screen

10.3.2 FAQ Category Settings

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

Name *
The name of this resource. Any type of characters can be entered to this field including uppercase letters and spaces. The name will be displayed in the overview table.

Subcategory of
It is possible to add the new category under an existing one as sub-category. This will be displayed as *Parent Category::Child Category*.

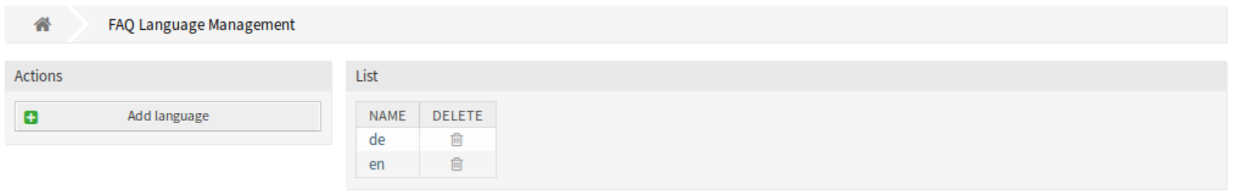
Permission *
Agent groups that can access articles in this category.

Validity
Set the validity of this resource. Each resource can be used in OTRS only, if this field is set to *valid*. Setting this field to *invalid* or *invalid-temporarily* will disable the use of the resource.

Comment *
Add additional information to this resource. It is recommended to always fill this field as a description of the resource with a full sentence for better clarity, because the comment will be also displayed in the overview table.

10.4 FAQ Language

Use this screen to manage languages available in the knowledge base articles. A fresh OTRS installation already contains some languages by default. The language management screen is available in the *FAQ Language* module of the *Administration* group.



NAME	DELETE
de	🗑
en	🗑

Fig. 16: FAQ Language Management Screen

10.4.1 Manage FAQ Languages

To add a language:

1. Click on the *Add language* button in the left sidebar.
2. Fill in the required field.
3. Click on the *Submit* button.

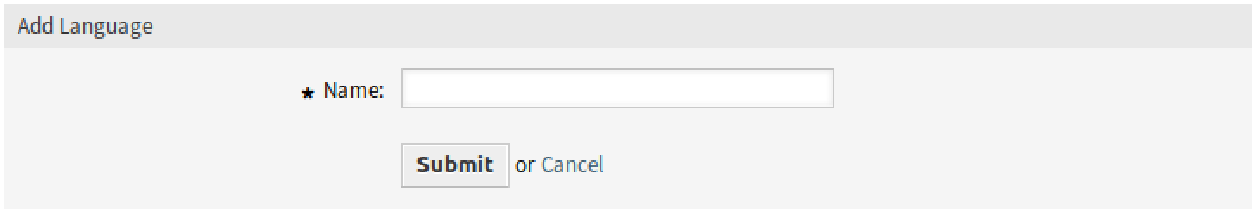
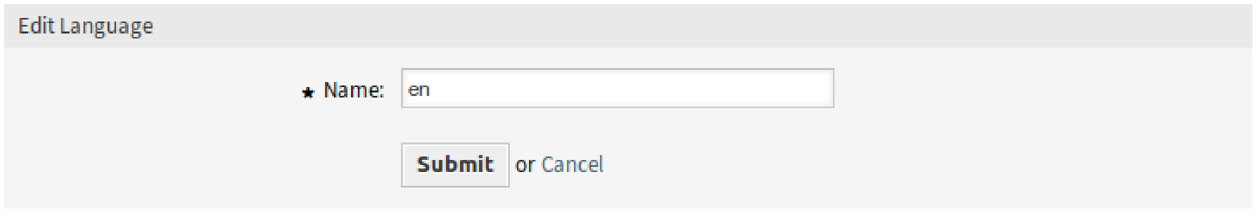


Fig. 17: Add Language Screen

To edit a language:

1. Click on a language in the list of languages.
2. Modify the field.
3. Click on the *Submit* button.

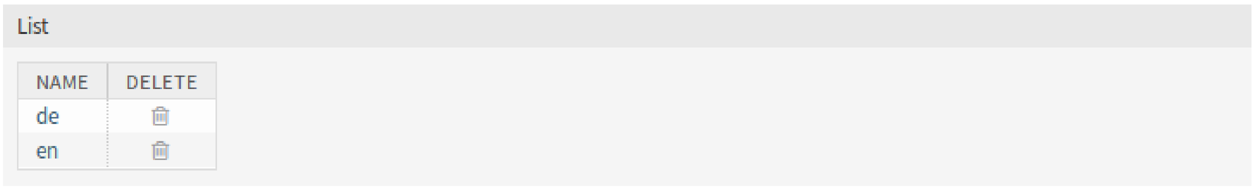


★ Name: en

Fig. 18: Edit Language Screen

To delete a language:

1. Click on the trash icon in the list of languages.
2. Click on the *Yes* button in the confirmation dialog.


List

NAME	DELETE
de	🗑
en	🗑

Fig. 19: Delete Language Screen

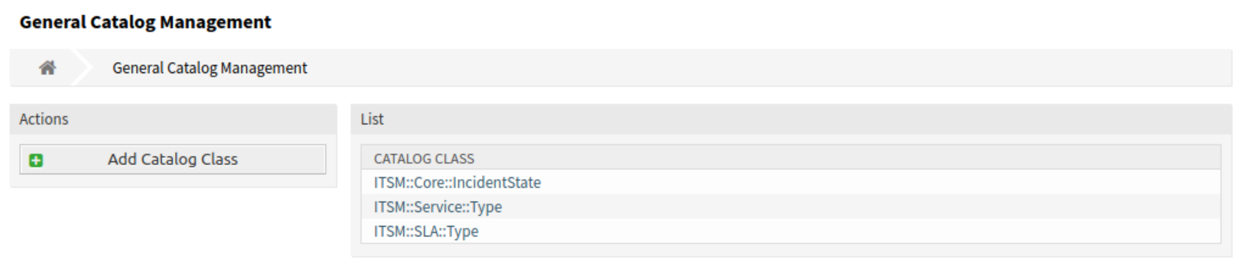
10.4.2 FAQ Language Settings

The following setting is available when adding or editing this resource. The fields marked with an asterisk are mandatory.

Name *
The ISO 639-1 code of a language.

10.5 General Catalog

Use this screen to add catalog classes and items to the system. The general catalog management screen is available in the *General Catalog* module of the *Administration* group.



CATALOG CLASS
ITSM::Core::IncidentState
ITSM::Service::Type
ITSM::SLA::Type

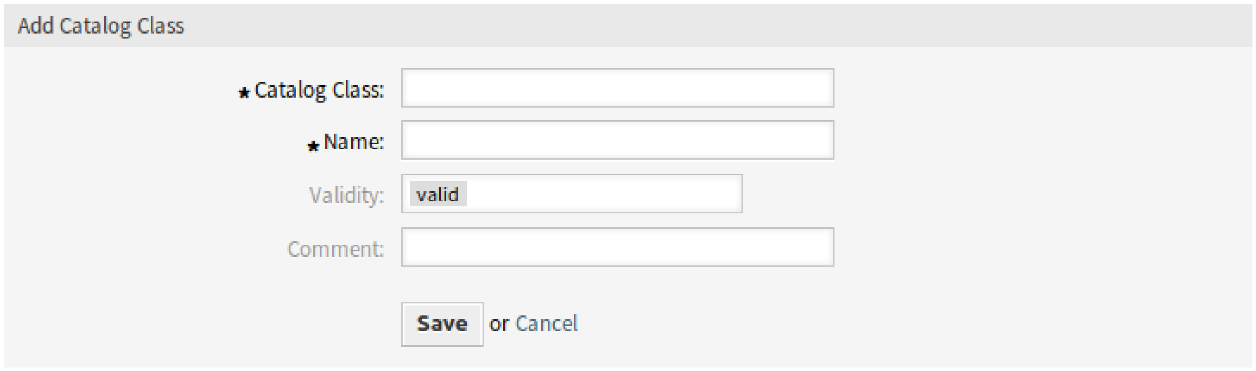
Fig. 20: General Catalog Management Screen

10.5.1 Manage General Catalog

To add a catalog class:

1. Click on the *Add Catalog Class* button in the left sidebar.
2. Fill in the required fields.
3. Click on the *Save* button.

Warning: Catalog classes can not be deleted from the system. They can only be deactivated by setting the *Validity* option to *invalid* or *invalid-temporarily*.


Add Catalog Class

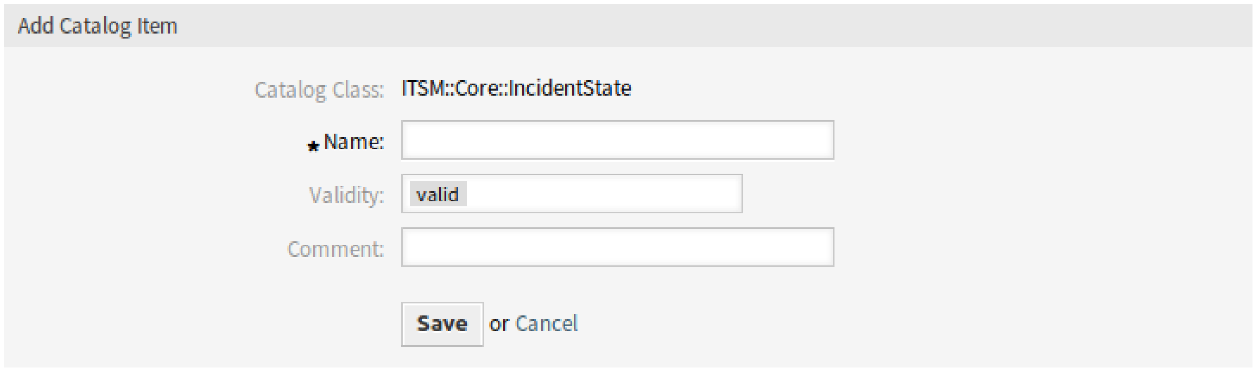
★ Catalog Class:	
★ Name:	
Validity:	valid
Comment:	

Fig. 21: Add Catalog Class Screen

> **Warning:** The maximum number of 20 *valid* catalog classes should not be exceeded. Exceeding this limit may affect the system performance.

To add a catalog item:

1. Select a catalog class in the list of catalog classes.
2. Click on the *Add Catalog Item* button in the left sidebar.
3. Fill in the required fields.
4. Click on the *Save* button.


Add Catalog Item

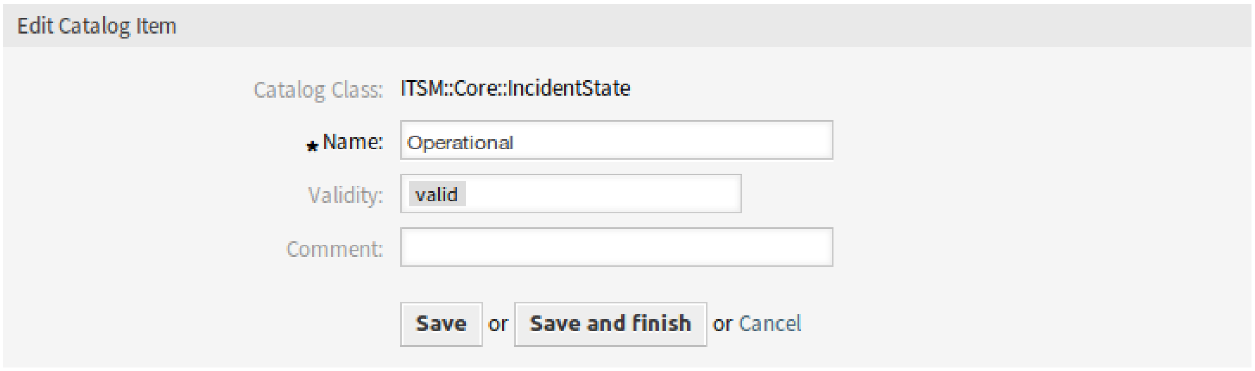
Catalog Class:	ITSM::Core::IncidentState
★ Name:	
Validity:	valid
Comment:	

Fig. 22: Add Catalog Item Screen

> **Warning:** Catalog items can not be deleted from the system. They can only be deactivated by setting the *Validity* option to *invalid* or *invalid-temporarily*.

To edit a catalog item:

1. Select a catalog class in the list of catalog classes.
2. Select a catalog item in the list of catalog items.
3. Modify the fields.
4. Click on the *Save* or *Save and finish* button.


Edit Catalog Item

Catalog Class:	ITSM::Core::IncidentState
* Name:	Operational
Validity:	valid
Comment:	

Fig. 23: Edit Catalog Item Screen

10.5.2 Catalog Class Settings

The following settings are available when adding this resource. The fields marked with an asterisk are mandatory.

Catalog Class *
The name of the catalog class. The catalog class will be displayed in the overview table of catalog classes.

Name *
The name of the catalog item to be added to the class. Any type of characters can be entered to this field including uppercase letters and spaces. The name will be displayed in the overview table of catalog items.

Validity *
Set the validity of this resource. Each resource can be used in OTRS only, if this field is set to *valid*. Setting this field to *invalid* or *invalid-temporarily* will disable the use of the resource.

Comment
Add additional information to this resource. It is recommended to always fill this field as a description of the resource with a full sentence for better clarity.

10.5.3 Catalog Item Settings

The following settings are available when adding this resource. The fields marked with an asterisk are mandatory.

Catalog Class
The name of the catalog class. This is read only in this screen.

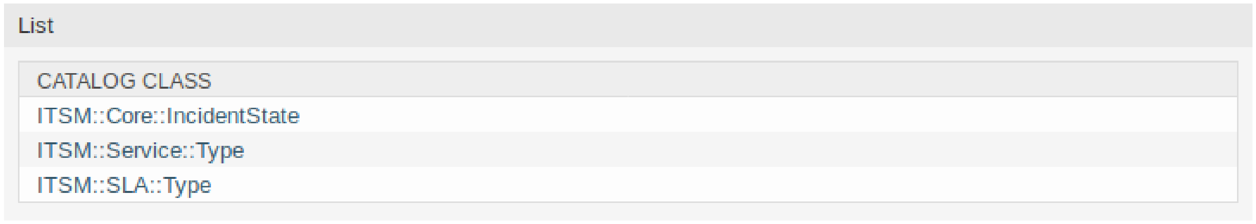
Name *
The name of the catalog item to be added to the class. Any type of characters can be entered to this field including uppercase letters and spaces. The name will be displayed in the overview table of catalog items.

Validity *
Set the validity of this resource. Each resource can be used in OTRS only, if this field is set to *valid*. Setting this field to *invalid* or *invalid-temporarily* will disable the use of the resource.

Comment
Add additional information to this resource. It is recommended to always fill this field as a description of the resource with a full sentence for better clarity.

10.5.4 Classes

The package adds some classes to the *General Catalog*.



List

CATALOG CLASS
ITSM::Core::IncidentState
ITSM::Service::Type
ITSM::SLA::Type

Fig. 24: General Catalog Class List Screen

ITSM::Core::IncidentState
This class contains incident states.

ITSM::Service::Type
This class contains service types.

ITSM::SLA::Type
This class contains service level agreement types.

10.6 Package Manager

Modular systems can be extended by adding additional software packages to the framework. Administrators need an easy way to see which features are installed in which version and for sure to add, update and remove packages.

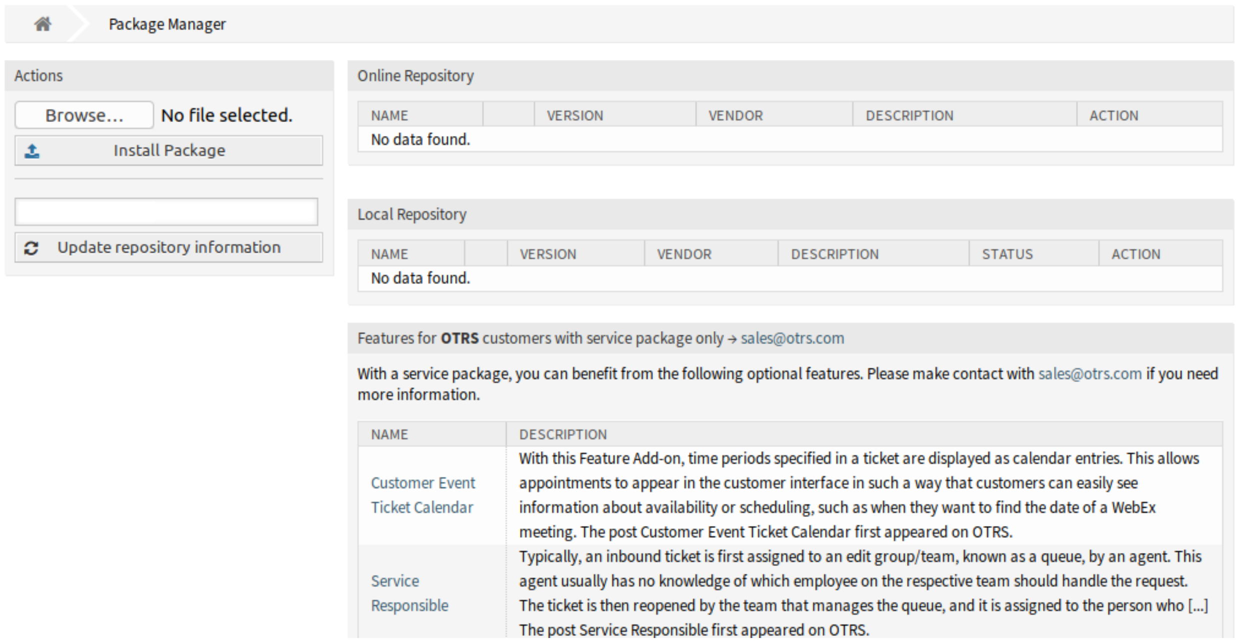
OTRS uses a package manager to perform all package-related activities as mentioned above in the graphical interface.

Note: This feature is only available to *On-Premise* customers. If you are a *Managed* customer, this feature is taken care of by the *Customer Solutions Team* in **OTRS**. Please contact us via support@otrs.com or in the OTRS Portal.

See also:

To see a list of installed modules, you may also see the *Support Data Collector*.

Use this screen to install and manage packages that extend the functionality of OTRS. The package manager screen is available in the *Package Manager* module of the *Administration* group.



Online Repository

NAME		VERSION	VENDOR	DESCRIPTION	ACTION
No data found.					

Local Repository

NAME		VERSION	VENDOR	DESCRIPTION	STATUS	ACTION
No data found.						

Features for **OTRS** customers with service package only → sales@otrs.com

With a service package, you can benefit from the following optional features. Please make contact with sales@otrs.com if you need more information.

NAME	DESCRIPTION
Customer Event Ticket Calendar	With this Feature Add-on, time periods specified in a ticket are displayed as calendar entries. This allows appointments to appear in the customer interface in such a way that customers can easily see information about availability or scheduling, such as when they want to find the date of a WebEx meeting. The post Customer Event Ticket Calendar first appeared on OTRS.
Service Responsible	Typically, an inbound ticket is first assigned to an edit group/team, known as a queue, by an agent. This agent usually has no knowledge of which employee on the respective team should handle the request. The ticket is then reopened by the team that manages the queue, and it is assigned to the person who [...] The post Service Responsible first appeared on OTRS.

Fig. 25: Package Manager Screen

10.6.1 Manage Packages

Warning: The installation of packages which are not verified by the OTRS Group is not possible by default.

See also:

You can activate the installation of not verified packages in the system configuration setting Package::AllowNotVerifiedPackages.

Install Packages

To install a package from online repository:

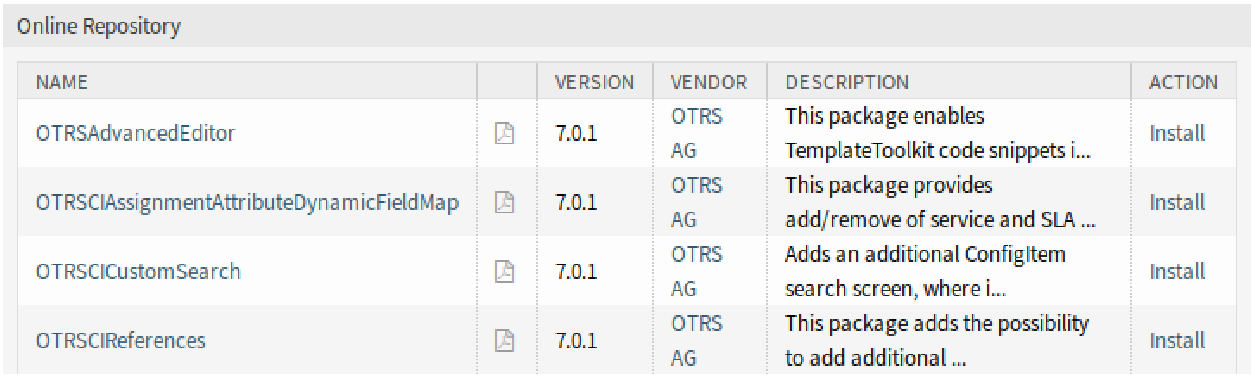
1. Select an online repository from the drop-down in the left sidebar.
2. Click on the *Update repository information* button to refresh the available package list.
3. Select a package from the *Online Repository* widget and click on the *Install* in the last column.
4. Follow the installation instructions.
5. After installation, the package is displayed in the *Local Repository* widget.

See also:

The repository list can be changed in system configuration setting Package::RepositoryList.

To install a package from file:

1. Click on the *Browse*··· button in the left sidebar.

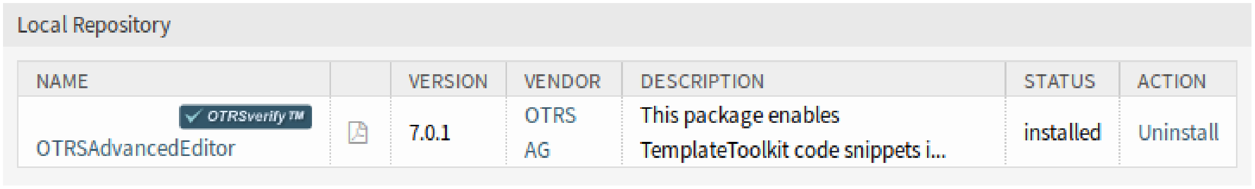


Online Repository

NAME		VERSION	VENDOR	DESCRIPTION	ACTION
OTRSAdvancedEditor		7.0.1	OTRS AG	This package enables TemplateToolkit code snippets i...	Install
OTRSCIAssignmentAttributeDynamicFieldMap		7.0.1	OTRS AG	This package provides add/remove of service and SLA ...	Install
OTRSCICustomSearch		7.0.1	OTRS AG	Adds an additional ConfigItem search screen, where i...	Install
OTRSCIReferences		7.0.1	OTRS AG	This package adds the possibility to add additional ...	Install

Fig. 26: Online Repository Widget

2. Select an `.opm` file from your local file system.
3. Click on the *Install Package* button.
4. Follow the installation instructions.
5. After installation, the package is displayed in the *Local Repository* widget.



Local Repository

NAME		VERSION	VENDOR	DESCRIPTION	STATUS	ACTION
OTRSverify™ OTRSAdvancedEditor		7.0.1	OTRS AG	This package enables TemplateToolkit code snippets i...	installed	Uninstall

Fig. 27: Local Repository Widget

Update Packages

To update a package from online repository:

1. Check the available packages in the *Online Repository* widget if there is *Update* in the *Action* column.
2. Click on the *Update* link.
3. Follow the update instructions.
4. After updating, the package is displayed in the *Local Repository* widget.

To update a package from file:

1. Click on the *Browse…* button in the left sidebar.
2. Select an `.opm` file which is newer than the installed package.
3. Click on the *Install Package* button.
4. Follow the update instructions.
5. After updating, the package is displayed in the *Local Repository* widget.

To update all packages:

1. Click on the *Update all installed packages* button in the left sidebar.

2. Follow the update instructions.
3. After updating, the package is displayed in the *Local Repository* widget.

This feature reads the information of all defined package repositories and determines if there is a new version for every installed package in the system and calculates the correct order to update the packages respecting all other package dependencies, even if new versions of existing packages require new packages not yet installed in the system.

Note: If there are packages installed that do not have a corresponding repository defined in the system, they can not be updated by this feature and will be marked as failed (due to the missing on-line repository).

Reinstall Packages

If at least one of the package files are modified locally, the package manager marks the package as broken, and need to reinstall.

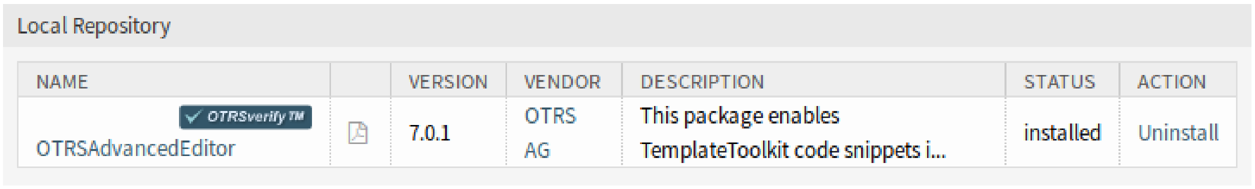
To reinstall a package:

1. Select the package from the *Local Repository* widget that are marked for reinstall.
2. Click on the *Reinstall* link in the *Action* column.
3. Follow the installation instructions.

Uninstall Packages

To uninstall a package:

1. Select the package from the *Local Repository* widget.
2. Click on the *Uninstall* link in the *Action* column.
3. Follow the uninstall instructions.



Local Repository

NAME		VERSION	VENDOR	DESCRIPTION	STATUS	ACTION
OTRSverify™ OTRSAdvancedEditor		7.0.1	OTRS AG	This package enables TemplateToolkit code snippets i...	installed	Uninstall

Fig. 28: Local Repository Widget

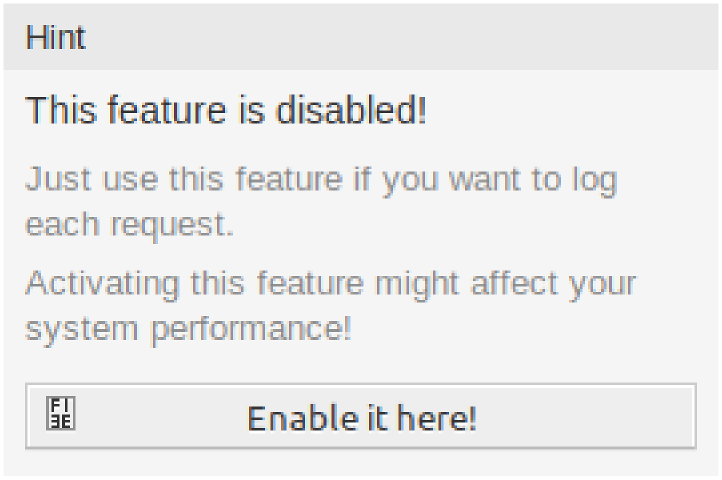
10.7 Performance Log

Performance is always a crucial topic for web-based applications. Administrators need to have the possibility to see which activities use which time to execute over time to identify potential increases over time and take proper measures.

OTRS supports this requirement with the performance log. The performance log can, when it is activated, log activities and display various activity types and their min/max/average response time and a number of requests for different time frames.

Use this screen to view the performance log of OTRS. The log overview screen is available in the *Performance Log* module of the *Administration* group.

Note: To be able to see performance log in OTRS, you have to enabled its setting first.



Hint

This feature is disabled!

Just use this feature if you want to log each request.

Activating this feature might affect your system performance!

Fig. 29: Enable Performance Log Support

If the performance log is enabled, OTRS collects all the information about requests and responses in an overview table.

Clicking on an entry will show the details.

10.8 Session Management

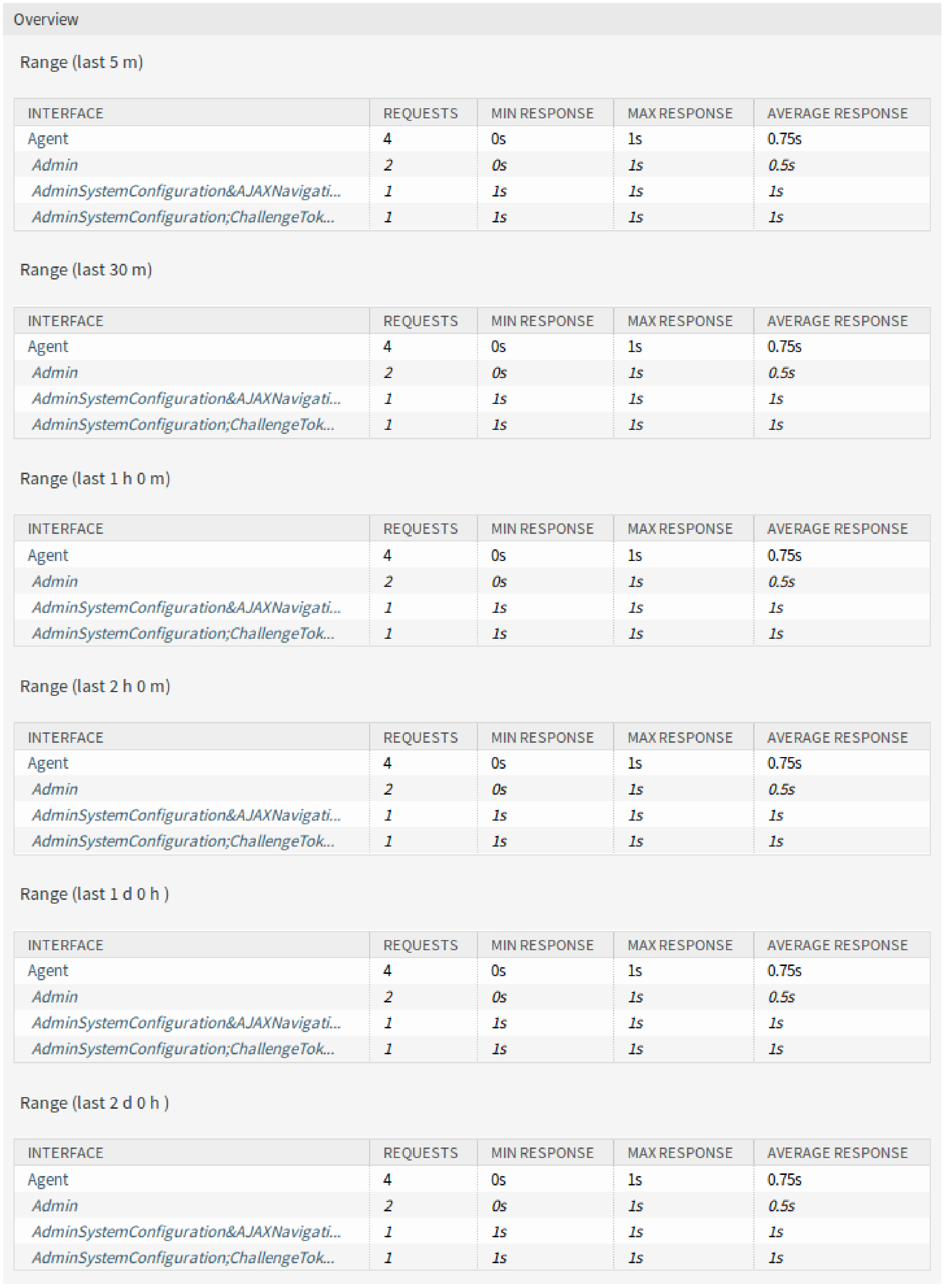
Administrators of web-based applications need to have access to the information who' s connected to the system and, if required, delete unwanted sessions.

OTRS offers the session management to quickly get an overview of agent and customer sessions, unique agents and customers currently logged in and the ability to kill sessions with just a mouse click.

Use this screen to manage logged in user sessions in the system. The session management screen is available in the *Session Management* module of the *Administration* group.

The first widget lists sessions and tokens. Sessions are used for logins to the administrator interface, while logins to the agent interface, to the external interface or to the generic interface use tokens.

The widget *List of tokens (one time only)* shows special one time tokens used for the password recovery and two-factor setup. These tokens are short-lived and can be used for only a single action.



Overview

Range (last 5 m)

INTERFACE	REQUESTS	MIN RESPONSE	MAX RESPONSE	AVERAGE RESPONSE
Agent	4	0s	1s	0.75s
Admin	*2*	*0s*	*1s*	*0.5s*
AdminSystemConfiguration&AJAXNavigati...	*1*	*1s*	*1s*	*1s*
AdminSystemConfiguration;ChallengeTok...	*1*	*1s*	*1s*	*1s*

Range (last 30 m)

INTERFACE	REQUESTS	MIN RESPONSE	MAX RESPONSE	AVERAGE RESPONSE
Agent	4	0s	1s	0.75s
Admin	*2*	*0s*	*1s*	*0.5s*
AdminSystemConfiguration&AJAXNavigati...	*1*	*1s*	*1s*	*1s*
AdminSystemConfiguration;ChallengeTok...	*1*	*1s*	*1s*	*1s*

Range (last 1 h 0 m)

INTERFACE	REQUESTS	MIN RESPONSE	MAX RESPONSE	AVERAGE RESPONSE
Agent	4	0s	1s	0.75s
Admin	*2*	*0s*	*1s*	*0.5s*
AdminSystemConfiguration&AJAXNavigati...	*1*	*1s*	*1s*	*1s*
AdminSystemConfiguration;ChallengeTok...	*1*	*1s*	*1s*	*1s*

Range (last 2 h 0 m)

INTERFACE	REQUESTS	MIN RESPONSE	MAX RESPONSE	AVERAGE RESPONSE
Agent	4	0s	1s	0.75s
Admin	*2*	*0s*	*1s*	*0.5s*
AdminSystemConfiguration&AJAXNavigati...	*1*	*1s*	*1s*	*1s*
AdminSystemConfiguration;ChallengeTok...	*1*	*1s*	*1s*	*1s*

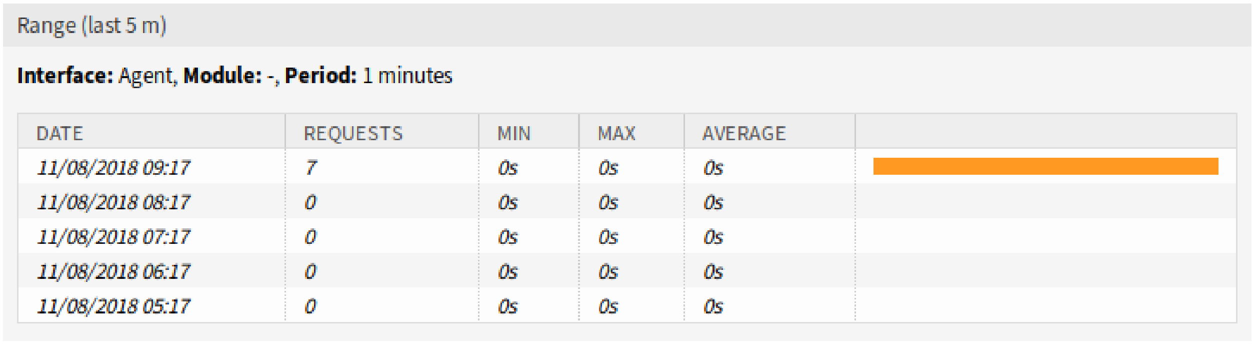
Range (last 1 d 0 h)

INTERFACE	REQUESTS	MIN RESPONSE	MAX RESPONSE	AVERAGE RESPONSE
Agent	4	0s	1s	0.75s
Admin	*2*	*0s*	*1s*	*0.5s*
AdminSystemConfiguration&AJAXNavigati...	*1*	*1s*	*1s*	*1s*
AdminSystemConfiguration;ChallengeTok...	*1*	*1s*	*1s*	*1s*

Range (last 2 d 0 h)

INTERFACE	REQUESTS	MIN RESPONSE	MAX RESPONSE	AVERAGE RESPONSE
Agent	4	0s	1s	0.75s
Admin	*2*	*0s*	*1s*	*0.5s*
AdminSystemConfiguration&AJAXNavigati...	*1*	*1s*	*1s*	*1s*
AdminSystemConfiguration;ChallengeTok...	*1*	*1s*	*1s*	*1s*

Fig. 30: Performance Log Screen

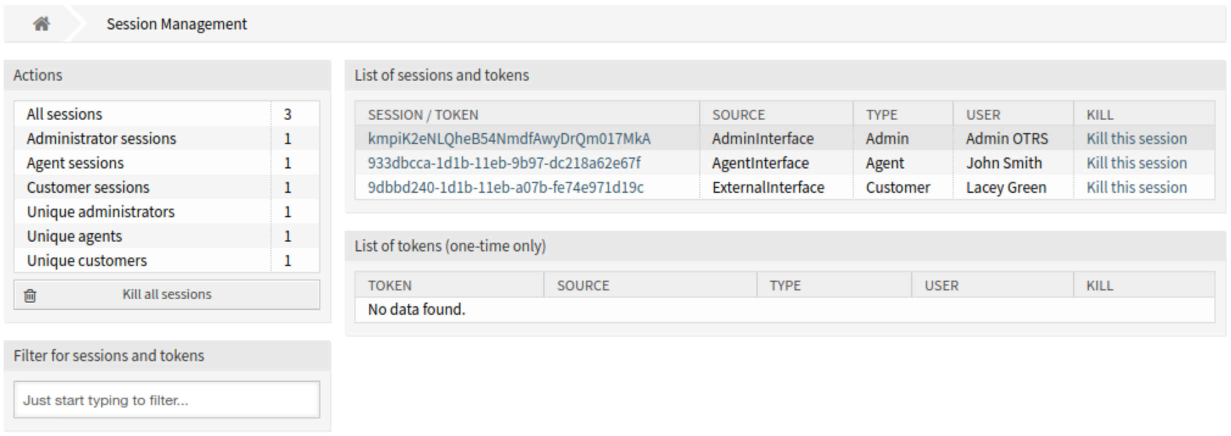


Range (last 5 m)

Interface: Agent, **Module:** -, **Period:** 1 minutes

DATE	REQUESTS	MIN	MAX	AVERAGE	
11/08/2018 09:17	*7*	*0s*	*0s*	*0s*	🟧
11/08/2018 08:17	*0*	*0s*	*0s*	*0s*	
11/08/2018 07:17	*0*	*0s*	*0s*	*0s*	
11/08/2018 06:17	*0*	*0s*	*0s*	*0s*	
11/08/2018 05:17	*0*	*0s*	*0s*	*0s*	

Fig. 31: Performance Log Details Screen



Session Management

Actions

All sessions	3
Administrator sessions	1
Agent sessions	1
Customer sessions	1
Unique administrators	1
Unique agents	1
Unique customers	1

Filter for sessions and tokens

List of sessions and tokens

SESSION / TOKEN	SOURCE	TYPE	USER	KILL
kmpiK2eNLQheB54NmdfAwyDrQm017MkA	AdminInterface	Admin	Admin OTRS	Kill this session
933dbcca-1d1b-11eb-9b97-dc218a62e67f	AgentInterface	Agent	John Smith	Kill this session
9dbbd240-1d1b-11eb-a07b-fe74e971d19c	ExternalInterface	Customer	Lacey Green	Kill this session

List of tokens (one-time only)

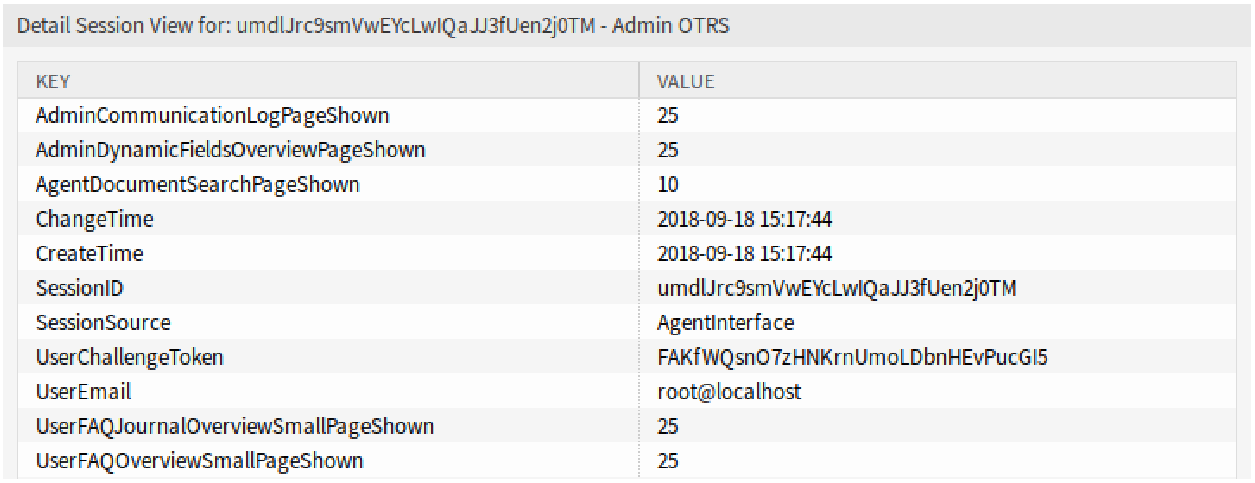
TOKEN	SOURCE	TYPE	USER	KILL
No data found.				

Fig. 32: Session Management Screen

10.8.1 Manage Sessions

To see a logged in user session:

1. Select a logged in user from the list of sessions.
2. Click on the token.
3. See the details.



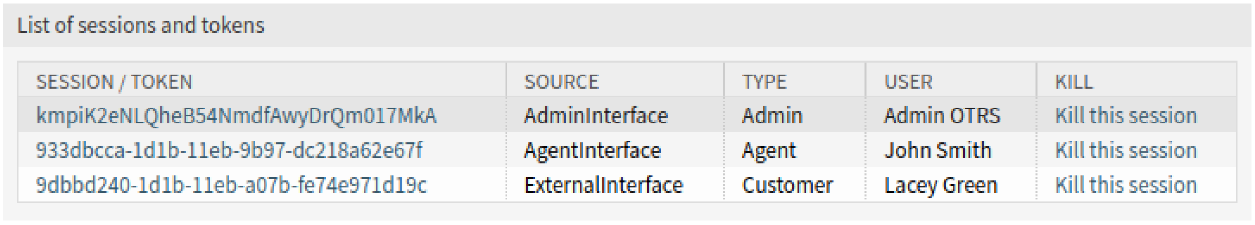
Detail Session View for: umdlJrc9smVwEYcLwlQaJJ3fUen2j0TM - Admin OTRS

KEY	VALUE
AdminCommunicationLogPageShown	25
AdminDynamicFieldsOverviewPageShown	25
AgentDocumentSearchPageShown	10
ChangeTime	2018-09-18 15:17:44
CreateTime	2018-09-18 15:17:44
SessionID	umdlJrc9smVwEYcLwlQaJJ3fUen2j0TM
SessionSource	AgentInterface
UserChallengeToken	FAKfWQsnO7zHNKrnUmoLDbnHEvPucGI5
UserEmail	root@localhost
UserFAQJournalOverviewSmallPageShown	25
UserFAQOverviewSmallPageShown	25

Fig. 33: Session Management Details Screen

To kill a session:

1. Select a logged in user from the list of sessions.
2. Click on the *Kill this session* link in the *Kill* column.



List of sessions and tokens

SESSION / TOKEN	SOURCE	TYPE	USER	KILL
kmpiK2eNLQheB54NmdfAwyDrQm017MkA	AdminInterface	Admin	Admin OTRS	Kill this session
933dbcca-1d1b-11eb-9b97-dc218a62e67f	AgentInterface	Agent	John Smith	Kill this session
9dbbd240-1d1b-11eb-a07b-fe74e971d19c	ExternalInterface	Customer	Lacey Green	Kill this session

Fig. 34: Session Kill Screen

Note: The current administrator session is displayed with a darker gray row background. This allows you to identify your own session in the list easier.

Warning: Clicking the *Kill this session* link removes the session immediately without confirmation. The unsaved work of the user will be lost!

To kill all sessions:

1. Click on the *Kill all sessions* button in the left sidebar.

Warning: Clicking the *Kill all session* link removes all sessions or tokens immediately without confirmation. The unsaved work of the users will be lost!

Note: If several users are logged in to the system, use the filter box to find a particular session by just typing the name to filter.

10.9 SQL Box

Note: This feature is not enabled by default. Activate the system configuration setting Frontend::Module###AdminSelectBox to enable this feature.

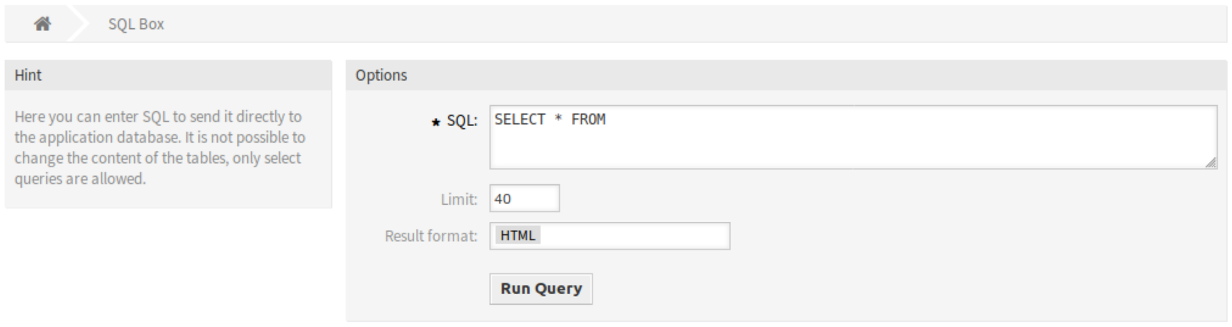
Note: This feature is only available to *On-Premise* customers. If you are a *Managed* customer, this feature is taken care of by the *Customer Solutions Team* in **OTRS**. Please contact us via support@otrs.com or in the OTRS Portal.

In a ticket system, it is usually possible to have statistics that show a summarized view of ticket information when needed. Sometimes, it is however required to access the database directly to have even more individual reports, allow external statistic tools to query information from the system or perform in-depth analysis of a ticket behavior.

Direct access to the database requires access to the command line which an administrator may not have. In addition to the username and password for the command line access, which is not given by all organizations, the username and password for the database are needed. These hurdles can prevent an administrator from using the database for more complex searches and operations.

OTRS offers application administrators the SQL Box in the graphical interface. It allows read access to the database. All results can be seen in the GUI or exported to CSV/Excel files.

Use this screen to query SQL statements in the system. The SQL box screen is available in the *SQL Box* module of the *Administration* group.



Here you can enter SQL to send it directly to the application database. It is not possible to change the content of the tables, only select queries are allowed.

* SQL: SELECT * FROM

Limit: 40

Result format: HTML

Fig. 35: SQL Box Screen

10.9.1 Query SQL statements

Note: The SQL statements entered here are sent directly to the application database. By default, it is not possible to change the content of the tables, only SELECT queries are allowed.

Warning: It is possible to modify the application database via SQL box. To do this, you have to enabled the system configuration setting AdminSelectBox::AllowDatabaseModification. Activate it to your own risk!

To execute an SQL statement:

1. Enter the SQL statement into the SQL box.
2. Select the result format.
3. Click on the *Run Query* button.

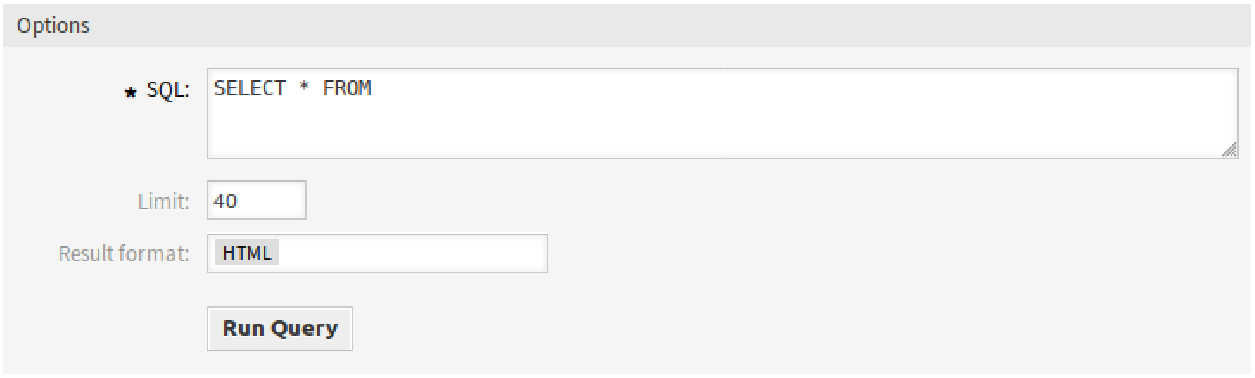


Fig. 36: SQL Box Widget

10.9.2 SQL Settings

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

SQL *
The SQL statement to be queried.

Limit
Enter a number to limit the result to at most this number of rows. Leaving this field empty means there is no limit.

Note: Don't use `LIMIT` inside the SQL statement. Always use this field to limit the number of results.

Result format
The format of the SQL statement result.

HTML
The results are visible below the SQL box in a new widget.

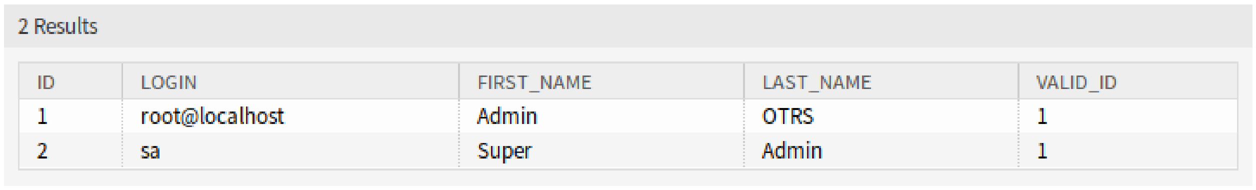
CSV
The results can be downloaded in comma separated plain text format.

Excel
The results can be downloaded as Microsoft Excel spreadsheet.

10.9.3 SQL Examples

To list some information about agents and output the results as HTML:

```
SELECT id, login , first_name, last_name, valid_id FROM users
```


2 Results

ID	LOGIN	FIRST_NAME	LAST_NAME	VALID_ID
1	root@localhost	Admin	OTRS	1
2	sa	Super	Admin	1

Fig. 37: SQL Box Result

To list all tables, you need to leave empty the *Limit* field and run the following query:

```
SHOW TABLES
```

To show the structure of the `users` table, you need to limit the results to 1 and run the following query (see the table header for the columns):

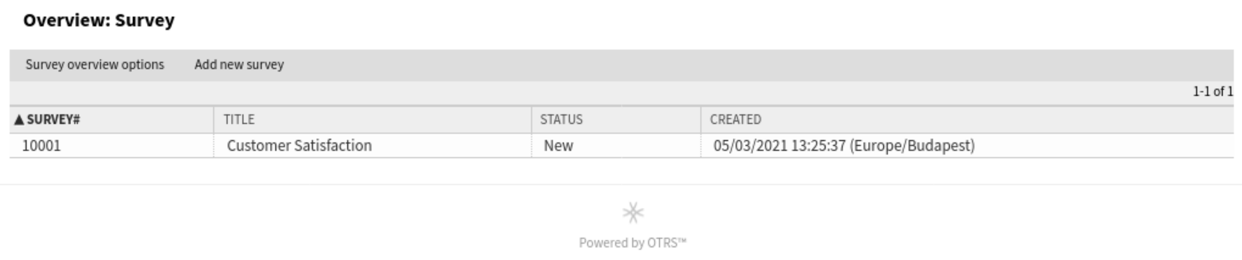
```
SELECT * FROM users
```

10.10 Surveys

This module enables you to create surveys, add questions to a created survey, set a survey live in order to send requests, modify existing surveys to a certain extent and view statistics of submitted survey answers as well as the answers themselves.

The surveys can be sent to customer users of closed tickets via email, but the survey is also accessible from the external interface. The results are displayed as graphical statistics of completed surveys.

Use this screen to add surveys to the system. The survey management screen is available in the *Surveys* module of the *Administration* group.


Overview: Survey

▲ SURVEY#	TITLE	STATUS	CREATED
10001	Customer Satisfaction	New	05/03/2021 13:25:37 (Europe/Budapest)

Fig. 38: Survey Overview Screen

10.10.1 Manage Surveys

To add a survey:

1. Click on the *Add new survey* button in the header bar.
2. Fill in the required fields.
3. Click on the *Create* button.
4. Click on the *Edit Questions* button in the header bar.
5. Add some questions and click on the *Add* button.
6. Click on the *Close this window* button to go back.

Warning: Surveys can not be deleted from the system. They can only be deactivated by setting the *Validity* option to *invalid*.

To edit a survey:

1. Click on a survey in the list of surveys.
2. Click on the *Edit General Information* button in the header bar.
3. Modify the fields.
4. Click on the *Update* button.

To filter the surveys displayed in the overview table:

1. Click on the *Survey overview options* button in the header bar.
2. Define some filters or date restrictions.
3. Click on the *Submit* button.

To make the survey available to use:

1. Click on a survey in the list of surveys.
2. Change the status of the survey.

Warning: As soon as the survey has the state *Master* or *Valid* the questions cannot be edited anymore.

10.10.2 Survey Settings

The following settings are available when adding or editing this resource. The fields marked with an asterisk are mandatory.

Title *
: The name of this resource. Any type of characters can be entered to this field including uppercase letters and spaces. The name will be displayed in the overview table.

Introduction *
: This is the summary of the survey. It is displayed in the external interface together with the title.

Sender *
: This is the email address which is set as sender of the email sent to the users.

Subject *
The subject of the email sent to the users.

Body *
The body text of the message.

Queues
Select the queues for which the new survey can be created. If none is selected, it will be valid for every queue.

Ticket Types
Select the ticket types for which the new survey can be created. If none is selected, it will be valid for every ticket type.

See also:

This field is not displayed by default. It can be enabled in the system configuration.

Services
Select the services for which the new survey can be created. If none is selected, it will be valid for every service.

See also:

This field is not displayed by default. It can be enabled in the system configuration.

Internal Description *
Add additional information to this resource. It is recommended to always fill this field as a description of the resource with a full sentence for better clarity, because the comment will be also displayed in the overview table.

Customer conditions
It is possible to define conditions based on customer user attributes, e.g. all customer users with a specified comment would not get any survey emails. Each attribute could have at least one condition. A condition contains a checkbox value for negation and a value field (input type text or multiple select).

See also:

This field is not displayed by default. It can be enabled in the system configuration.

10.10.3 Add Questions

As soon as a new survey is created, the user can add questions to the survey by clicking on the *Edit Questions* menu item in the displayed survey. A new window will pop up where the user can add questions.

Five types of questions are selectable. These are:

- *Yes/No* questions: a user can answer by selecting *Yes* or *No* from a drop-down field.
- *Radio (List)* questions: a user can answer by selecting exactly one and just one of the possible answers.
- *Checkbox (List)* questions: a user can answer by selecting none to all listed answers.
- *Textarea* questions: a user can write answers by typing regular text.
- *Net Promoter Score* questions: a user can answer by selecting exactly one and just one of the possible scores.
- *Embedded* question: a user can answer directly in the email received. The possible answers can be images embedded in the email that is sent out. Only one embedded question can be added to a survey. The embedded question will not be shown in the external interface.

For each answer type you can choose if you want a customer user to be required to answer this question or if a customer user can ignore the question.

As soon as questions of type *Radio (List)*, *Checkbox (List)* or *Net Promoter Score* got added, the question will be shown in the *Survey Questions* overview. By clicking on the question you can add answers, change the answer order by clicking on the up or down icon or delete the answer by clicking on the trash icon.

Questions of type *Radio (List)* and *Net Promoter Score* needs to have at least two answers to be valid. Furthermore, questions of type *Checkbox (List)* needs to have at least one answer to be valid.

After all questions and possible answers have been added, you can leave the *Edit Questions* area by clicking on the *Close this window* button.

10.10.4 Start Sending Survey Requests

On the survey detail page, you can select *Master* from the *- Change Status -* drop-down. As soon as the survey has the state *Master* the survey is set live.

Note: Only one survey can have the state *Master*.

If a ticket is closed, the customer user receives an email invitation to fill the survey by clicking a link inside the invitation email.

10.10.5 Survey Results Graph

Select a survey that is already submitted to the customer users and scroll down. You can see the *Survey Results Graph* widget. This displays statistically a summary of questions like *Yes/No*, *Radio (List)* and *Checkbox (List)*.

Questions where answers were configured as required will have one vote for each submitted survey by a customer user.

Questions where answers were not configured as required are optional for the customer users. If customer users have not selected any of the *Radio (List)*, *Yes/No* or *Checkbox (List)* answers or have not entered anything in a *Textarea* answer, it can happen that only 10 or 20 answers appear in a statistic, although the survey was sent by 200 customer users.

Statistics for required *Checkbox (List)* questions have at least one checked value for each answer submitted. However, since multiple selections are possible, there may be more answers than the total number of surveys submitted.

Example: 200 surveys submitted by customer users, 350 answers on one *Checkbox (List)* question.

Statistics for not required questions may have less answers than the total amount of submitted surveys (answer is optional).

10.10.6 Statistics Details

When you click on a survey that customer users have already submitted, you can see the following details in the *Stats Details* section:

- The time when the survey invitation was sent to the customer user.
- The time when the customer user submitted the filled survey.
- The associated ticket as a link.
- A magnifying glass showing the votes selected for that customer user.

In this area, you will find the answers to each question that the customer user has selected or typed in. Especially *Textarea* answers are of interest here.

10.10.7 Modifying Survey

As soon as a survey had the status *Master*, the modification possibilities are limited. Therefore, make sure that the number of questions and answers is configured correctly before changing the status to *Master*.

Warning: In a survey of type *Master* you can not add or delete questions any more.

In a survey of type *Master* the general information (title, introduction, sender etc.), the question text, the answer text, as well as the order of the answers can be changed.

You can change the *Answer Required* property of a question too, but this may just make sense as long as there are no submitted customer surveys. Changing the *Answer Required* question to *No* makes the answer optional, so the statistic for this question may not have an answer for each submitted customer survey any more. Changing the *Answer Required* property to *Yes* will require the customer users of the future survey to give a vote for this question, but will not add votes for already submitted surveys. So statistics will render rather useless too.

10.11 System Configuration

Modern systems have many ways to configure their behavior. Some use configuration files edited on the command line, and some use a graphical interface (and save the information to configuration files in the background), yet others use a database. Maintaining changes and auditing can sometimes be an issue, as it is not always clear who made a change. Making bulk changes is not always possible, and rolling back changes a chore.

OTRS uses a comfortable graphical interface to configure the system. All changes to the default system configuration are stored in the database and can be audited (who changed a setting and when, what was the old and what is the new value) and rolled back to a previous state in case of misconfiguration.

Comfortable search allows finding the needed settings quickly and easily.

Use this screen to manage the system configuration settings. OTRS brings about 1750 configuration settings. The system configuration management screen is available in the *System Configuration* module of the *Administration* group.

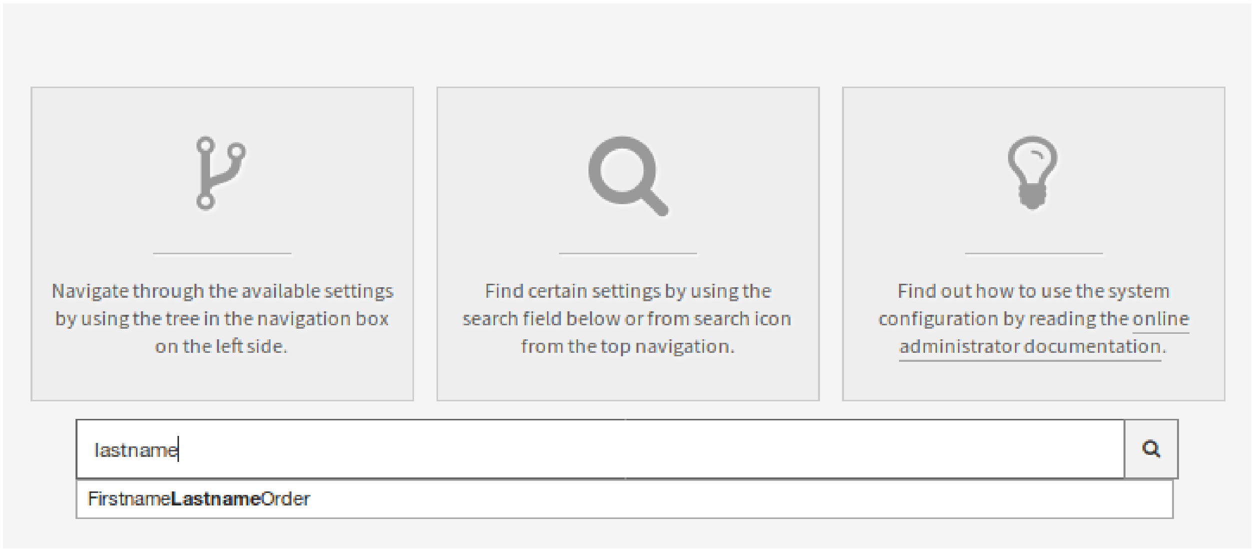
10.11.1 Manage System Configurations

Note: For security reasons, the configuration settings for the database connection cannot be changed in the graphical user interface of the system configuration. These have to be set manually in `Kernel/Config.pm`.

To modify a system configuration, you need to do several steps. The following example shows you, how to find, modify, deploy and reset the system configuration `FirstnameLastnameOrder`.

1. Find the system configuration by entering a search term `lastname` into to search box.

   With the full-text search, all configuration settings can be scanned for one or more keywords. The full-text search not only searches through the names of the configuration settings, but also the descriptions and values. This allows an element to be found easily even if its name is unknown.


Navigate through the available settings by using the tree in the navigation box on the left side.

Find certain settings by using the search field below or from search icon from the top navigation.

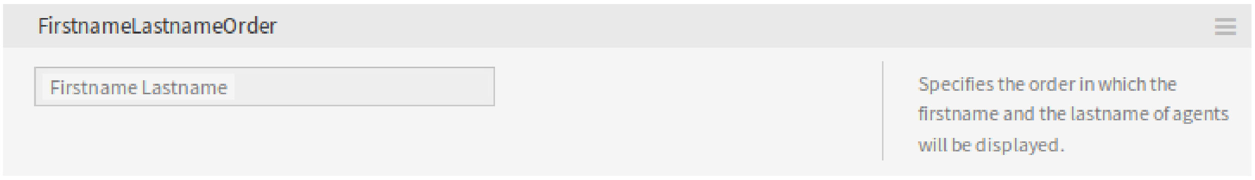
Find out how to use the system configuration by reading the online administrator documentation.

lastname

FirstnameLastnameOrder

Fig. 39: System Configuration - Search For Setting

2. Select the setting from the search results.


FirstnameLastnameOrder

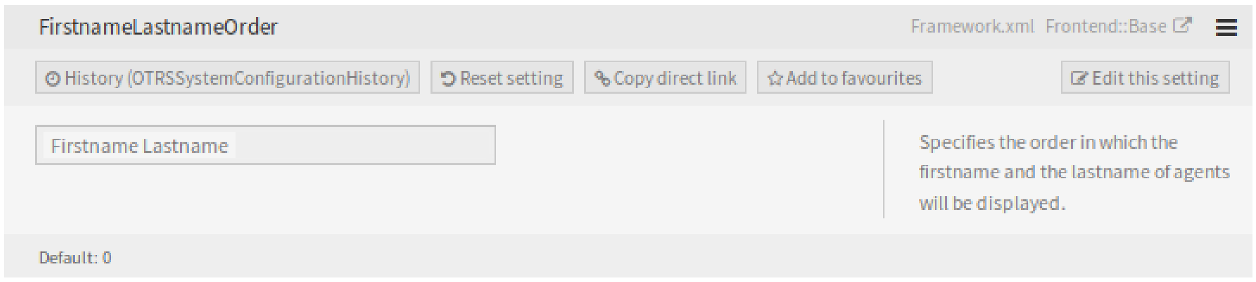
Firstname Lastname

Specifies the order in which the firstname and the lastname of agents will be displayed.

Fig. 40: System Configuration - Setting Found

3. Click on the header of the widget to see the options.
4. Hover the mouse over the widget body to see the *Edit this setting* button.
5. Click on the *Edit this setting* button to activate the edit mode. In edit mode the widget gets an orange border on the left.

   Note: If a setting is currently edited by another administrator, it is not possible to get access to the edit mode for that setting until the other administrator finished their work.



Specifies the order in which the firstname and the lastname of agents will be displayed.

Default: 0

Fig. 41: System Configuration - Setting Expanded

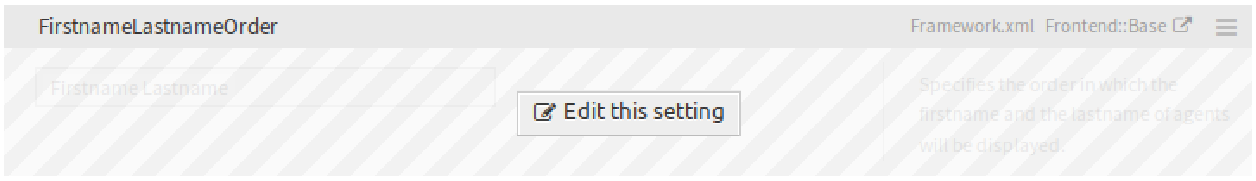
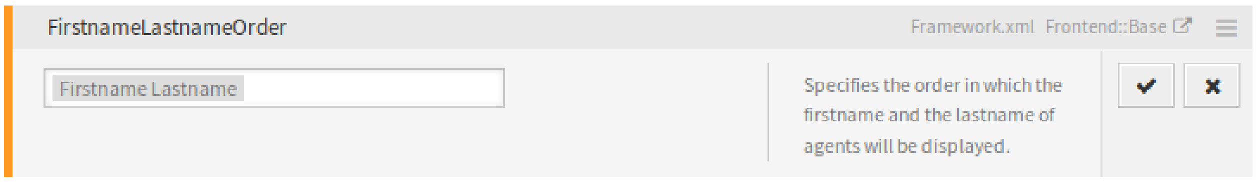

FirstnameLastnameOrder
Framework.xml Frontend::Base
Firstname Lastname
Edit this setting
Specifies the order in which the firstname and the lastname of agents will be displayed.

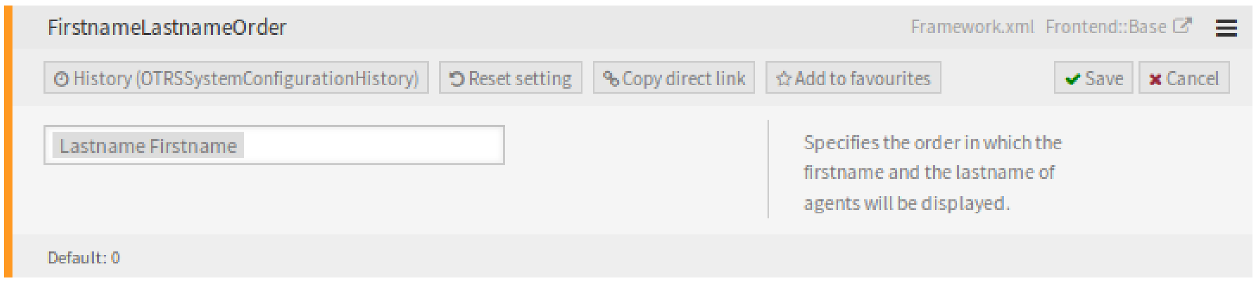
Fig. 42: System Configuration - Setting Hovered

6. Change the value of the setting. Editing can be cancelled by clicking the *Cancel* button on the right or hitting the *Escape* key on your keyboard. When editing is cancelled, all changes made during the current editing session are discarded.
7. Click on the *Save* button. If the modification is saved, the widget gets a green border on the left.
8. Go back and click on the *Deployment* button in the left sidebar. You are also notified in the notification bar, that you have undeployed settings.
9. Review the changes.
10. You can click on the button in the top right corner to compare the changes side-by-side.
11. Click on the *Deploy selected changes* button. If several settings are changed, it is possible to deploy only the selected settings.
12. Add a deployment comment, that explain for other administrators, what is changed and why. Use full sentence here.
13. Go back and search again the term `lastname` to find the modified setting. The widget has a gray border on the left to indicate, this setting is modified.
14. To reset the setting, click on the header of the widget to see the options. Then click on the *Reset setting* button.



Specifies the order in which the firstname and the lastname of agents will be displayed.

Fig. 43: System Configuration - Setting Clicked


FirstnameLastnameOrder

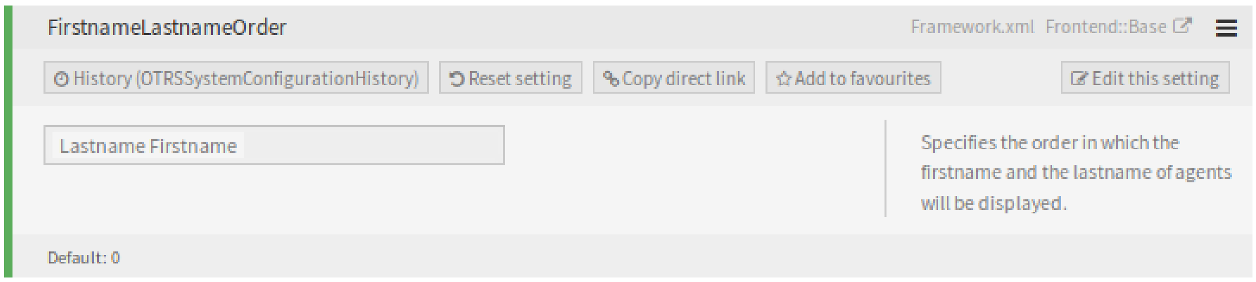
Framework.xml Frontend::Base

Lastname Firstname

Specifies the order in which the firstname and the lastname of agents will be displayed.

Default: 0

Fig. 44: System Configuration - Setting Modified


FirstnameLastnameOrder

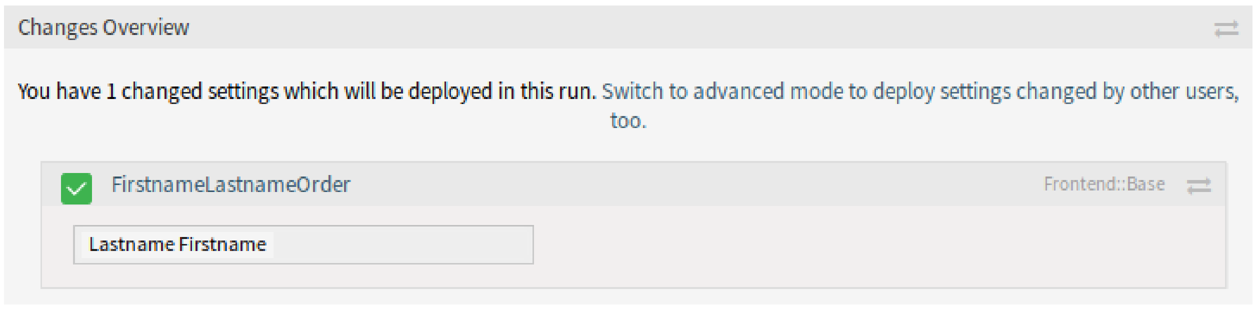
Framework.xml Frontend::Base

Lastname Firstname

Specifies the order in which the firstname and the lastname of agents will be displayed.

Default: 0

Fig. 45: System Configuration - Setting Saved

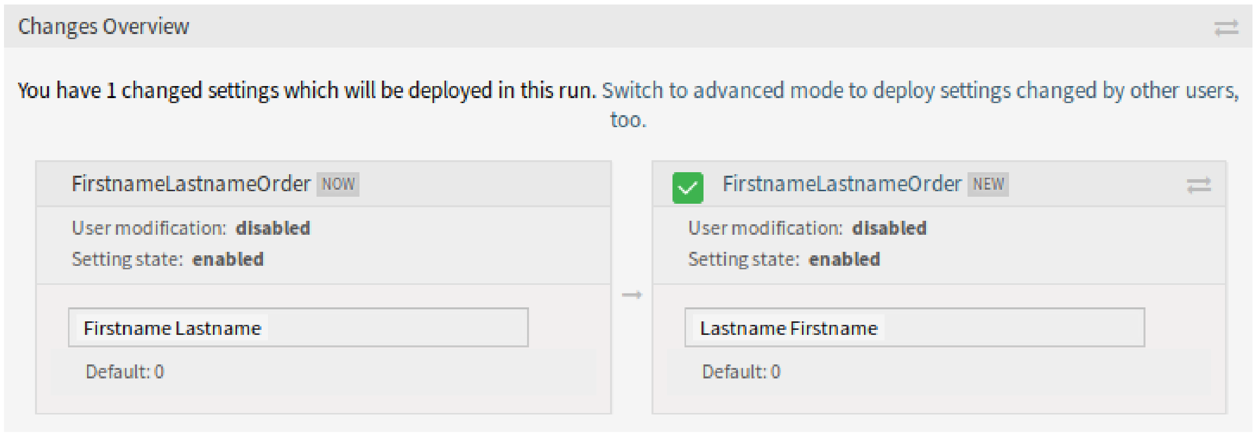

Changes Overview

You have 1 changed settings which will be deployed in this run. Switch to advanced mode to deploy settings changed by other users, too.

FirstnameLastnameOrder Frontend::Base

Lastname Firstname

Fig. 46: System Configuration - Setting Changes


Changes Overview

You have 1 changed settings which will be deployed in this run. Switch to advanced mode to deploy settings changed by other users, too.

FirstnameLastnameOrder NOW

User modification: **disabled**

Setting state: **enabled**

Firstname Lastname

Default: 0

FirstnameLastnameOrder NEW

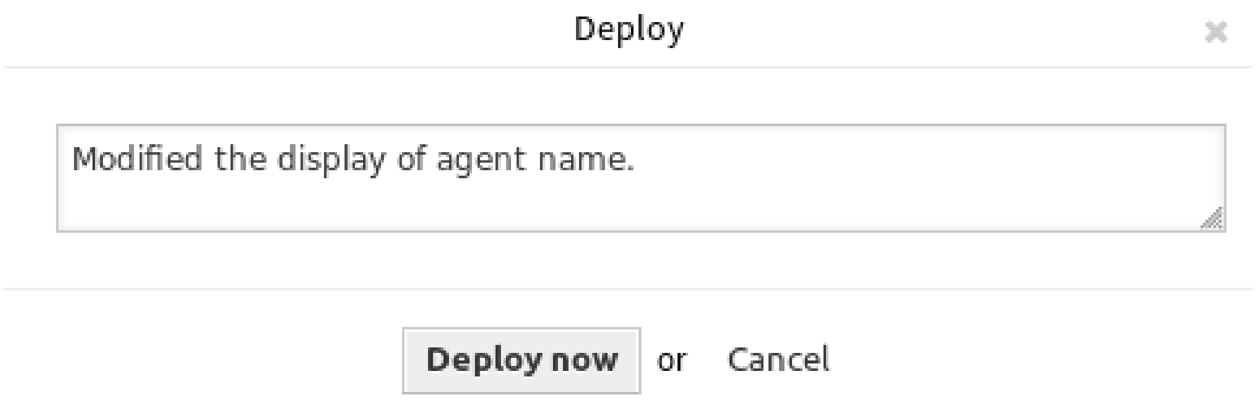
User modification: **disabled**

Setting state: **enabled**

Lastname Firstname

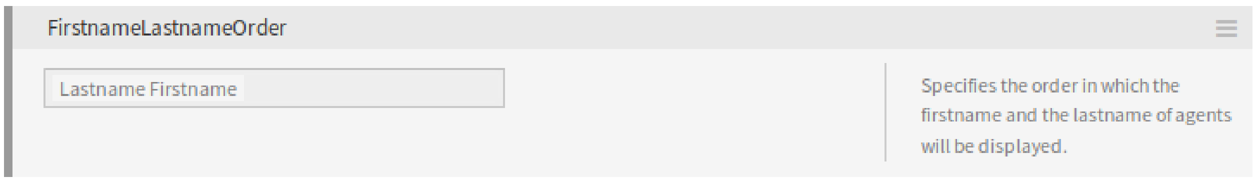
Default: 0

Fig. 47: System Configuration - Setting Different



Modified the display of agent name.

Fig. 48: System Configuration - Deploy Setting

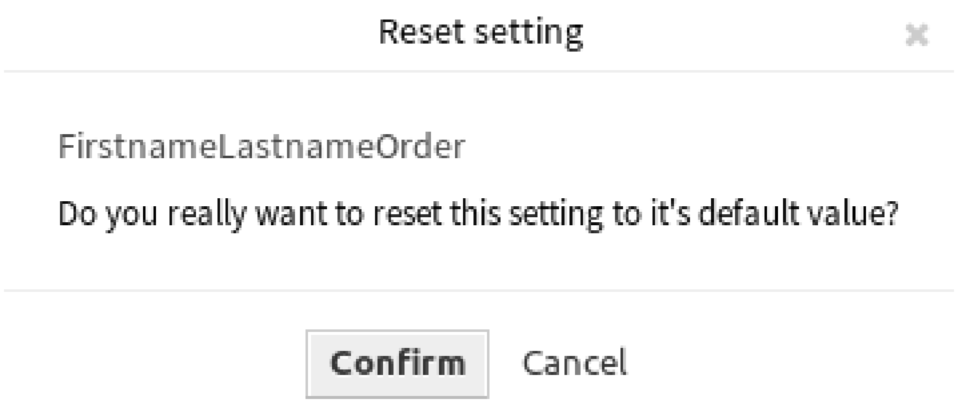


FirstnameLastnameOrder

Lastname Firstname

Specifies the order in which the firstname and the lastname of agents will be displayed.

Fig. 49: System Configuration - Setting Deployed



FirstnameLastnameOrder

Do you really want to reset this setting to it's default value?

Fig. 50: System Configuration - Reset Setting

15. Click on the *Confirm* button.
16. Deploy the settings.

10.11.2 Using The Navigation Tree

Each configuration setting is classified by a category and a navigation group. Navigation groups are individual elements in the main navigation tree. By selecting one of these navigation entries, all settings assigned to the selected group will be shown. As long as no extensions are installed, the category selection is automatically hidden, but as soon as a package is installed which brings its own configuration settings (such as ITSM modules or Survey), the category selection will be revealed. Selecting a category makes the main navigation tree show only the navigation groups belonging to the selected category.

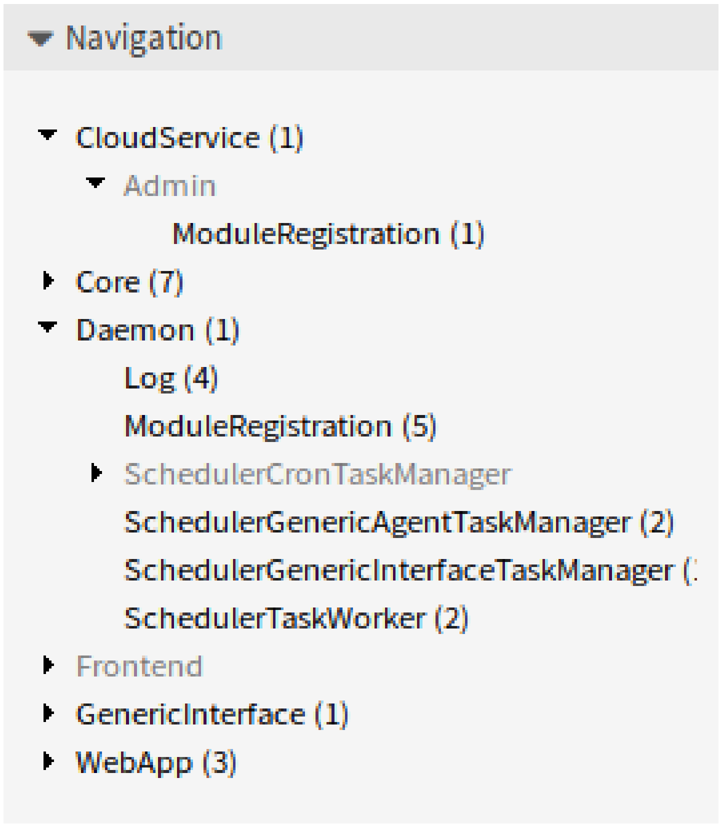
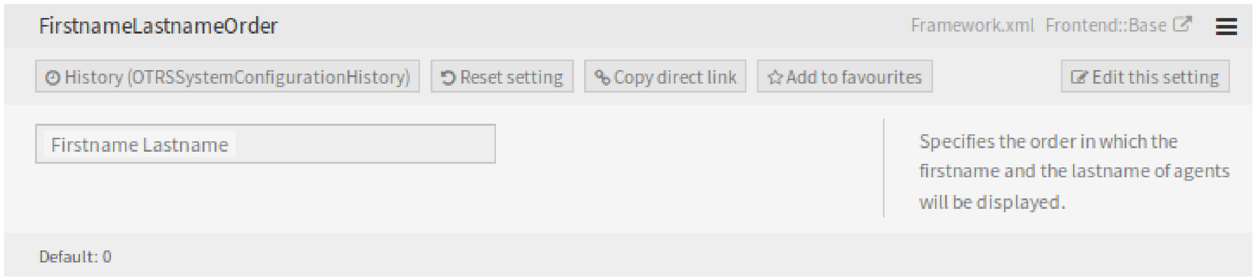


Fig. 51: System Configuration Navigation Tree

To expand an element, click on the arrow next to it. The number between the parentheses indicates how many settings belongs to this element. If an element has no number, this element is only a wrapper category. It doesn’t have settings, it has only sub-categories.

Using the navigation tree results the same as search for a setting. If you would like to see for a setting to which group belongs to, expand it by clicking on the header of the widget.

For example `FirstnameLastnameOrder` can be found in *Frontend* → *Base*.



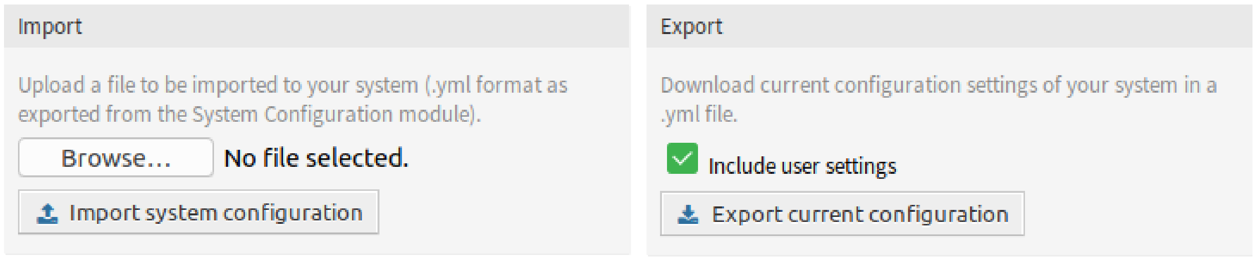
Specifies the order in which the firstname and the lastname of agents will be displayed.

Default: 0

Fig. 52: System Configuration - Setting Expanded

10.11.3 Import And Export System Configurations

Click on the *Import & Export* button in the left sidebar to access the import-export screen.



Upload a file to be imported to your system (.yml format as exported from the System Configuration module).

Download current configuration settings of your system in a .yml file.

Fig. 53: System Configuration - Import and Export

To export the system configurations:

1. Click on the *Export current configuration* button in the *Export* widget.
2. Save the `Export_Current_System_Configuration.yml` file to your local file system.
3. Rename the file to a more descriptive name.

To import the system configurations:

1. Click on the *Browse*… button in the *Import* widget.
2. Select a previously exported `.yml` file.
3. Click on the *Import system configuration* button.

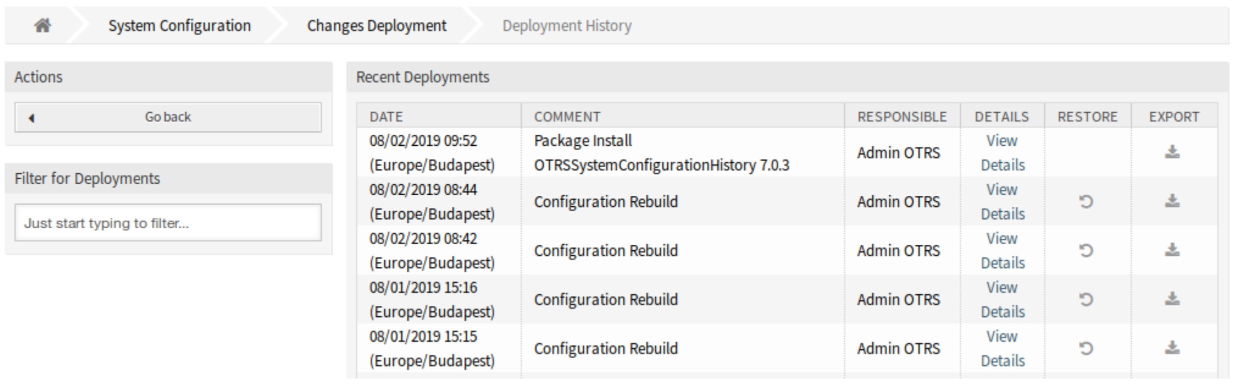
Note: This feature is only available to *On-Premise* customers. If you are a *Managed* customer, this feature is taken care of by the *Customer Solutions Team* in **OTRS**. Please contact us via support@otrs.com or in the OTRS Portal.

10.11.4 Deployment History

Previous revisions can be viewed and restored individually. This makes it easy to reset a setting, reducing errors and downtime. In addition, changes are auditable and revision-proof –the system registers which changes have been made by whom.

To see the deployment history:

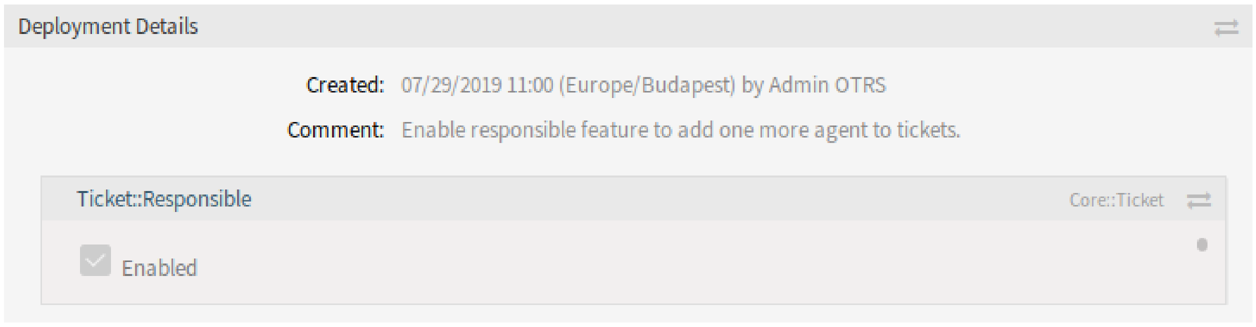
1. Go to the *System Configuration* screen.
2. Click on the *Deployment* button.
3. Click on the *History* button.



DATE	COMMENT	RESPONSIBLE	DETAILS	RESTORE	EXPORT
08/02/2019 09:52 (Europe/Budapest)	Package Install OTRSSystemConfigurationHistory 7.0.3	Admin OTRS	View Details		⤓
08/02/2019 08:44 (Europe/Budapest)	Configuration Rebuild	Admin OTRS	View Details	↺	⤓
08/02/2019 08:42 (Europe/Budapest)	Configuration Rebuild	Admin OTRS	View Details	↺	⤓
08/01/2019 15:16 (Europe/Budapest)	Configuration Rebuild	Admin OTRS	View Details	↺	⤓
08/01/2019 15:15 (Europe/Budapest)	Configuration Rebuild	Admin OTRS	View Details	↺	⤓

Fig. 54: Deployment History Screen

Every deployment can be further inspected by clicking on *View Details* link next to it. Details screen can be used to compare the setting with its previous value, before the deployment took place.



Created: 07/29/2019 11:00 (Europe/Budapest) by Admin OTRS

Comment: Enable responsible feature to add one more agent to tickets.

Ticket::Responsible

✓ Enabled

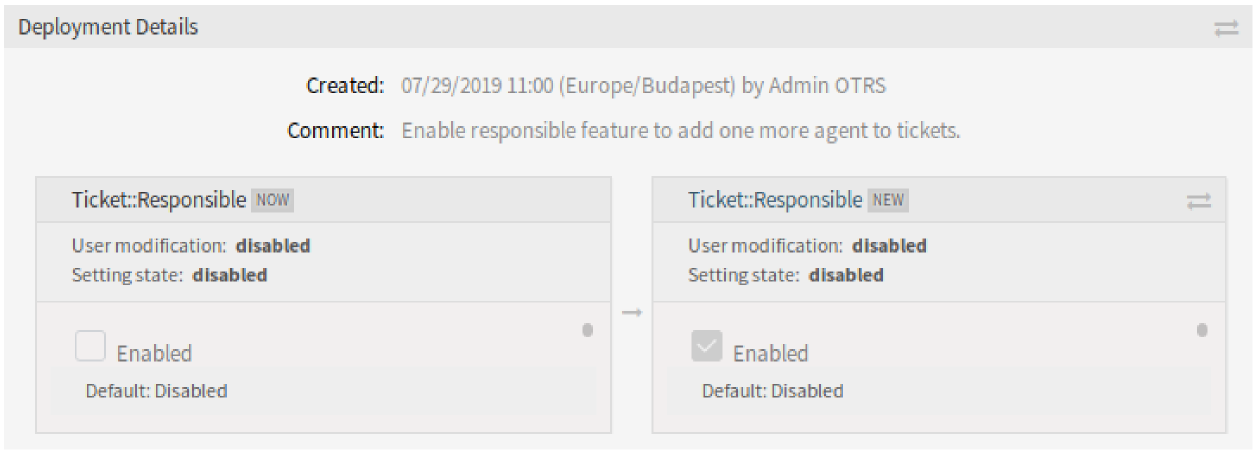
Fig. 55: Deployment Details Screen

It is possible to compare the old and the new settings side-by-side by clicking on the two arrows button.

Additionally, older deployments (every one before current state) can be restored with a simple click. By restoring a deployment, all settings will be reverted to the value they had at the time of the deployment in question.

Finally, deployments can be exported with click on the export button. User will be presented with a download of a YML file that contains changed settings. This YML file can be later restored via *Import & Export* screen in the system configuration.

Note: If several history entries are displayed in the history, use the filter box to find a particular log entry by


Deployment Details

Created: 07/29/2019 11:00 (Europe/Budapest) by Admin OTRS

Comment: Enable responsible feature to add one more agent to tickets.

Ticket::Responsible NOW

User modification: **disabled**
Setting state: **disabled**

☐ Enabled
Default: Disabled

Ticket::Responsible NEW

User modification: **disabled**
Setting state: **disabled**

✓ Enabled
Default: Disabled

Fig. 56: Deployment Details Difference

just typing the name to filter.

10.11.5 Setting History

Specific setting history can be accessed via *History* button in the setting widget. This button opens a screen of all values setting had over different deployments. Information like name of the user that made the change and time of change is displayed, along with a useful comparison tool.

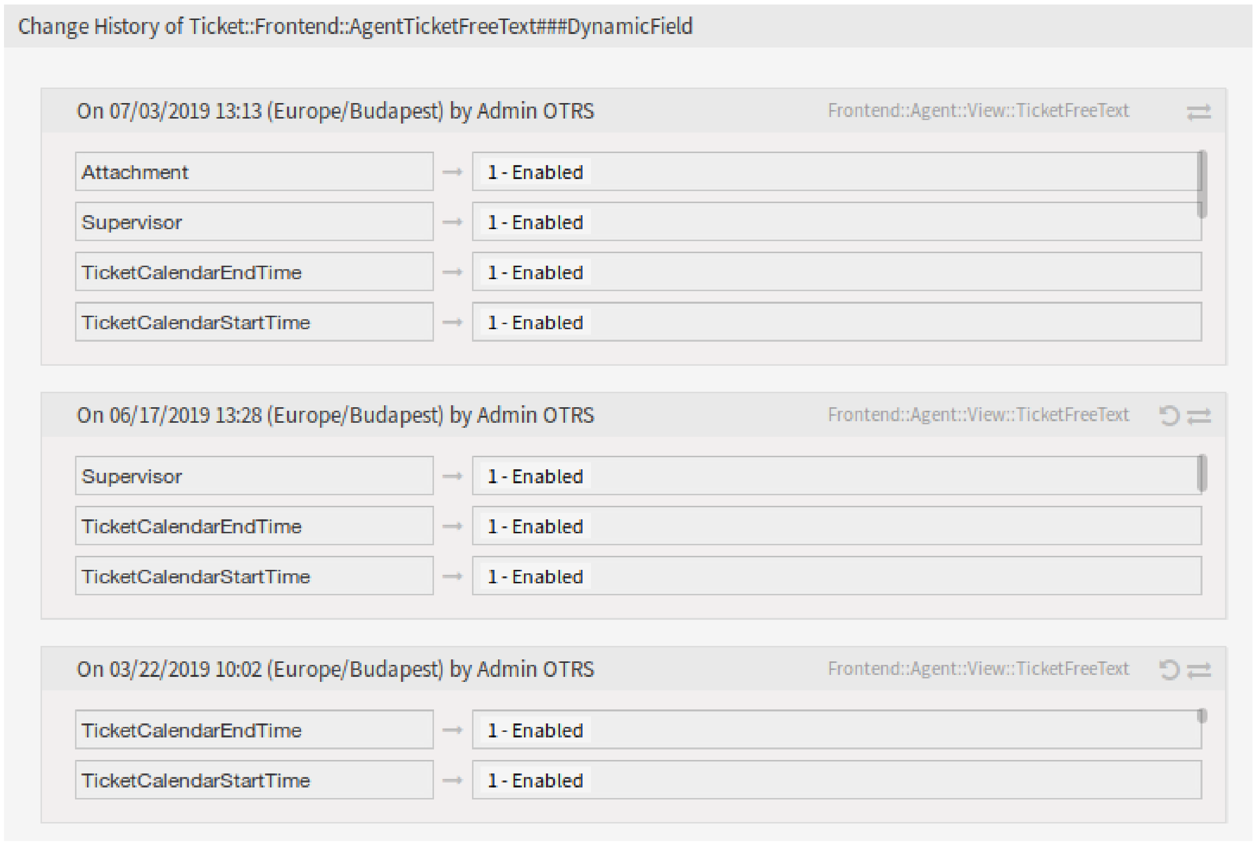
To see the setting history:

1. Go to the *System Configuration* screen.
2. Search for a system configuration setting.
3. Open the setting for editing and click on its header to reveal the buttons.
4. Click on the *History* button.

Every historical setting value can be restored by clicking on the *Restore* button in top right corner of the widget.

10.11.6 Business Object Configuration

The following section contains a description of the business object configuration, including the business object lists, business object detail views, business cards and forms.



Change History of Ticket::Frontend::AgentTicketFreeText###DynamicField

On 07/03/2019 13:13 (Europe/Budapest) by Admin OTRS — Frontend::Agent::View::TicketFreeText

Attachment	1 - Enabled
Supervisor	1 - Enabled
TicketCalendarEndTime	1 - Enabled
TicketCalendarStartTime	1 - Enabled

On 06/17/2019 13:28 (Europe/Budapest) by Admin OTRS — Frontend::Agent::View::TicketFreeText

Supervisor	1 - Enabled
TicketCalendarEndTime	1 - Enabled
TicketCalendarStartTime	1 - Enabled

On 03/22/2019 10:02 (Europe/Budapest) by Admin OTRS — Frontend::Agent::View::TicketFreeText

TicketCalendarEndTime	1 - Enabled
TicketCalendarStartTime	1 - Enabled

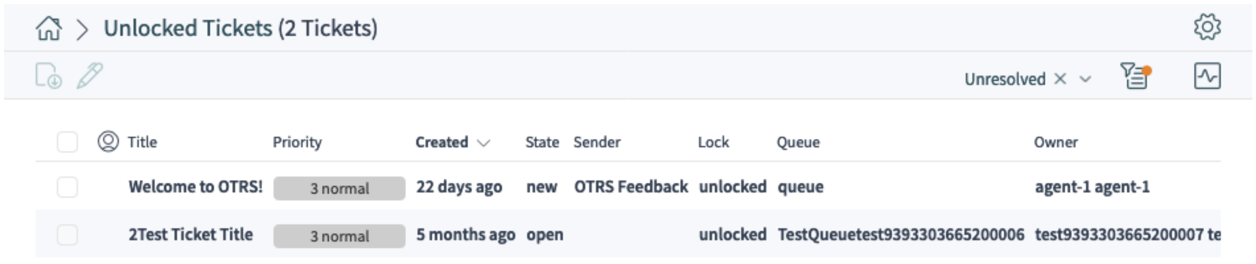
Fig. 57: Change History Screen

Business Object Lists

The business object lists provide a tabular view of items with support for configurable columns, sorting and filtering. These lists can be used in many contexts, including as stand-alone screens, widgets, actions, etc.

The default configuration of business object lists can be defined in several places, depending on the screen or element where they are used.

`AgentFrontend::<BusinessObjectListType>::<SlugName>###DefaultConfig`
Standalone or static list screens which are usable via their own URL.



Unlocked Tickets (2 Tickets)

Unresolved

Title	Priority	Created	State	Sender	Lock	Queue	Owner
Welcome to OTRS!	3 normal	22 days ago	new	OTRS Feedback	unlocked	queue	agent-1 agent-1
2Test Ticket Title	3 normal	5 months ago	open		unlocked	TestQueuetest9393303665200006	test9393303665200007 te

Fig. 58: An Example of the *Unlocked Tickets* Screen

The following system configuration settings are relevant:

- AgentFrontend::KnowledgeBaseArticleList::Static###DefaultConfig
- AgentFrontend::KnowledgeBaseArticleList::Added###DefaultConfig
- AgentFrontend::KnowledgeBaseArticleList::Updated###DefaultConfig
- AgentFrontend::KnowledgeBaseArticleList::Rated###DefaultConfig
- AgentFrontend::KnowledgeBaseArticleList::Custom1###DefaultConfig
- AgentFrontend::KnowledgeBaseArticleList::Custom2###DefaultConfig
- AgentFrontend::KnowledgeBaseArticleList::Custom3###DefaultConfig
- AgentFrontend::KnowledgeBaseArticleList::Custom4###DefaultConfig
- AgentFrontend::KnowledgeBaseArticleList::Custom5###DefaultConfig
- AgentFrontend::TicketList::Static###DefaultConfig
- AgentFrontend::TicketList::Unresolved###DefaultConfig
- AgentFrontend::TicketList::Unlocked###DefaultConfig
- AgentFrontend::TicketList::Reminders###DefaultConfig
- AgentFrontend::TicketList::Escalations###DefaultConfig
- AgentFrontend::TicketList::Created###DefaultConfig
- AgentFrontend::TicketList::Closed###DefaultConfig
- AgentFrontend::TicketList::Queues###DefaultConfig
- AgentFrontend::TicketList::Legacyservice###DefaultConfig
- AgentFrontend::TicketList::Legacystatus###DefaultConfig
- AgentFrontend::TicketList::Legacyescalations###DefaultConfig
- AgentFrontend::TicketList::Legacyresponsible###DefaultConfig

- AgentFrontend::TicketList::Legacywatcher###DefaultConfig
- AgentFrontend::TicketList::Custom1###DefaultConfig
- AgentFrontend::TicketList::Custom2###DefaultConfig
- AgentFrontend::TicketList::Custom3###DefaultConfig
- AgentFrontend::TicketList::Custom4###DefaultConfig
- AgentFrontend::TicketList::Custom5###DefaultConfig

`BusinessObjectListType`

Type of the business object list to use.

Possible values for the `BusinessObjectListType` key:

```
TicketList
KnowledgeBaseArticleList
```

`SlugName`

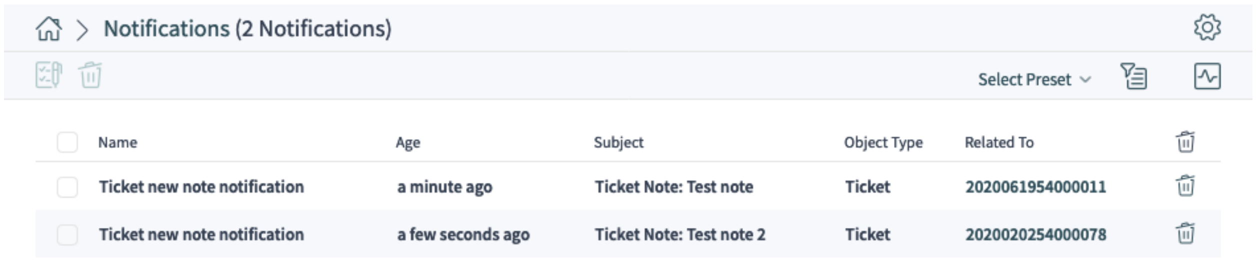
Determines the search engine-friendly URL part via which the screen is available.

Note: Currently only single word search engine-friendly URL parts are supported. Also note that the first character of the search engine-friendly URL part will always be converted to lower case variant, for example:

```
AgentFrontend::TicketList::Static => /agent/tickets/static
AgentFrontend::TicketList::Unresolved => agent/tickets/unresolved
AgentFrontend::TicketList::Reminders => agent/tickets/reminders
AgentFrontend::KnowledgeBaseArticleList::Added => agent/knowledge-base-
↪articles/added
```

`AgentFrontend::WebNotificationList###DefaultConfig`

The default configuration for the *Web Notifications* list.



Name	Age	Subject	Object Type	Related To
Ticket new note notification	a minute ago	Ticket Note: Test note	Ticket	2020061954000011
Ticket new note notification	a few seconds ago	Ticket Note: Test note 2	Ticket	2020020254000078

Fig. 59: An Example of the *Web Notifications* List Screen

Link to the reference of the system configuration:

- AgentFrontend::WebNotificationList###DefaultConfig

`AgentFrontend::Search###001-Framework`

The default configuration for the *Search Results* list.

Link to the reference of the system configuration:

- AgentFrontend::Search###001-Framework
- AgentFrontend::Search::Widget###AppointmentList

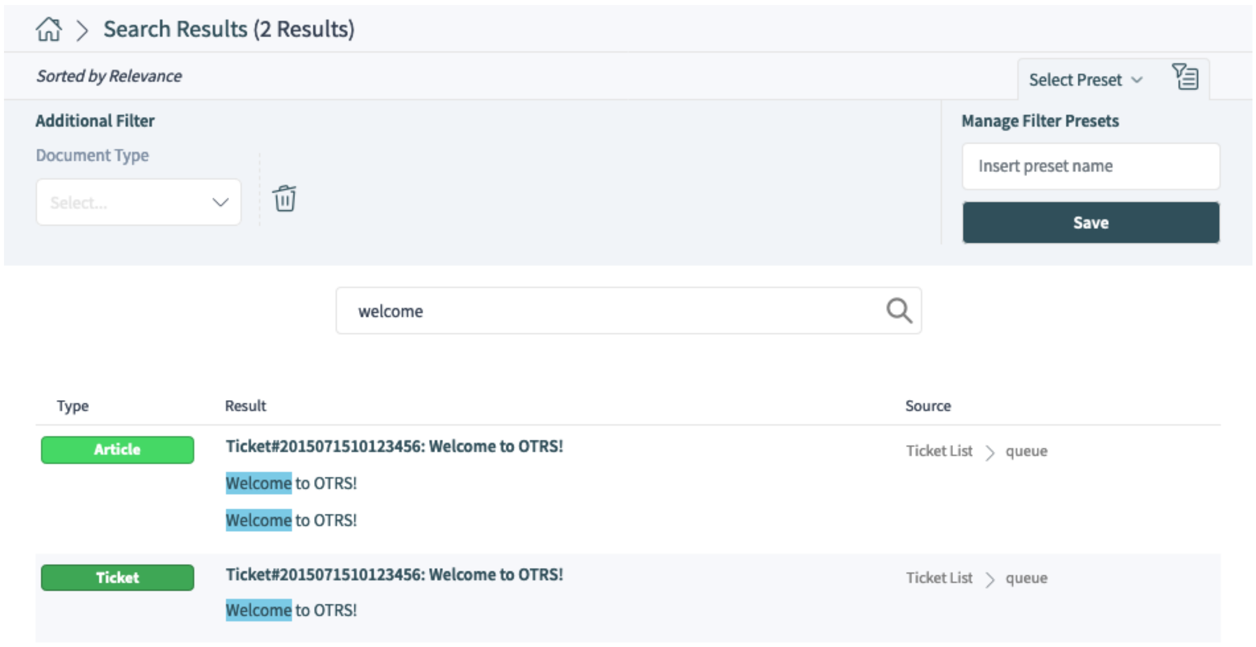


Fig. 60: An Example of the *Search Results* List Screen

- AgentFrontend::Search::Widget###ArticleList
- AgentFrontend::Search::Widget###AttachmentList
- AgentFrontend::Search::Widget###KnowledgeBaseArticleList
- AgentFrontend::Search::Widget###TicketList

`AgentFrontend::TicketList::ArticlePreview###DefaultConfig`

The default configuration for the *Article Preview* in the *Ticket* list.

Link to the reference of the system configuration:

- AgentFrontend::TicketList::ArticlePreview###DefaultConfig

`AgentFrontend::*###DefaultConfig`

The default configuration for the inline business object lists. Inline list tables are shown as part of several actions like *Merge Ticket*, *Link Objects* or *Append to Ticket*.

The following system configuration settings are relevant:

- AgentFrontend::Merge::Ticket###DefaultConfig
- AgentFrontend::LinkObject::Ticket###DefaultConfig
- AgentFrontend::LinkObject::KnowledgeBaseArticle###DefaultConfig
- AgentFrontend::LinkObject::CalendarAppointment###DefaultConfig
- AgentFrontend::Chat::AppendToTicket###DefaultConfig

`AgentFrontend::*AddressBookList*###DefaultConfig`

The default configuration for the *Customer User* and *Customer* address book screens. The address books allow contextual adding of customer and customer user records to target form fields.

The following system configuration settings are relevant:

- AgentFrontend::CustomerUserAddressBookList::Email###DefaultConfig

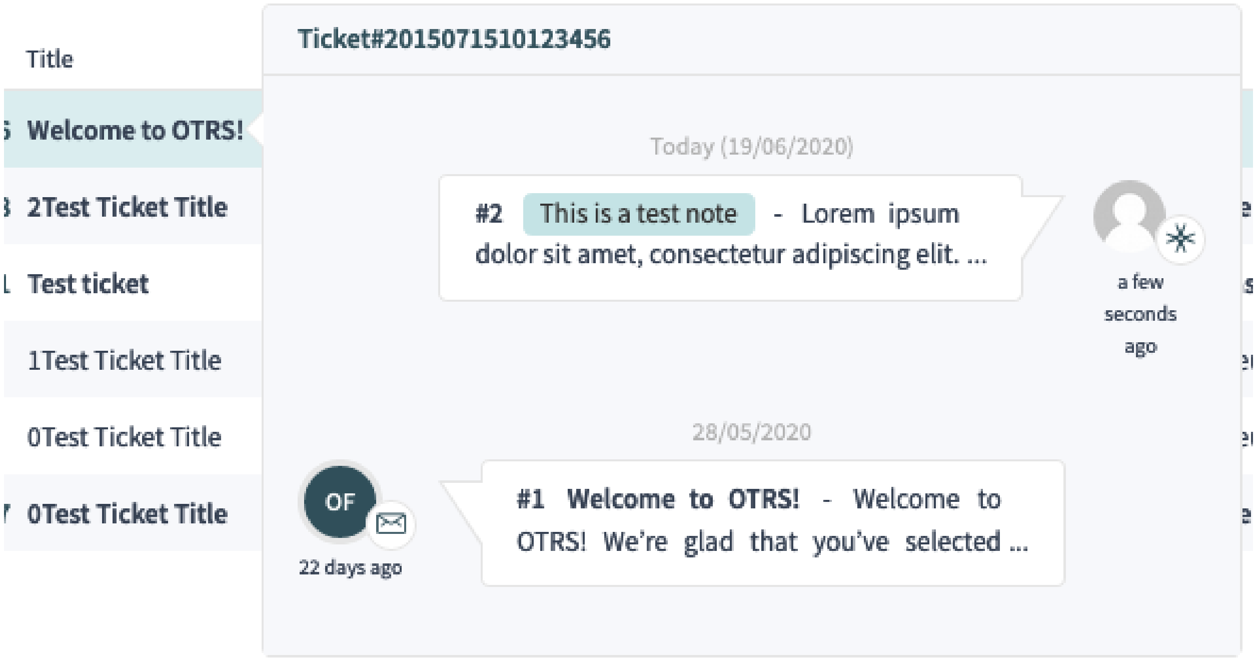
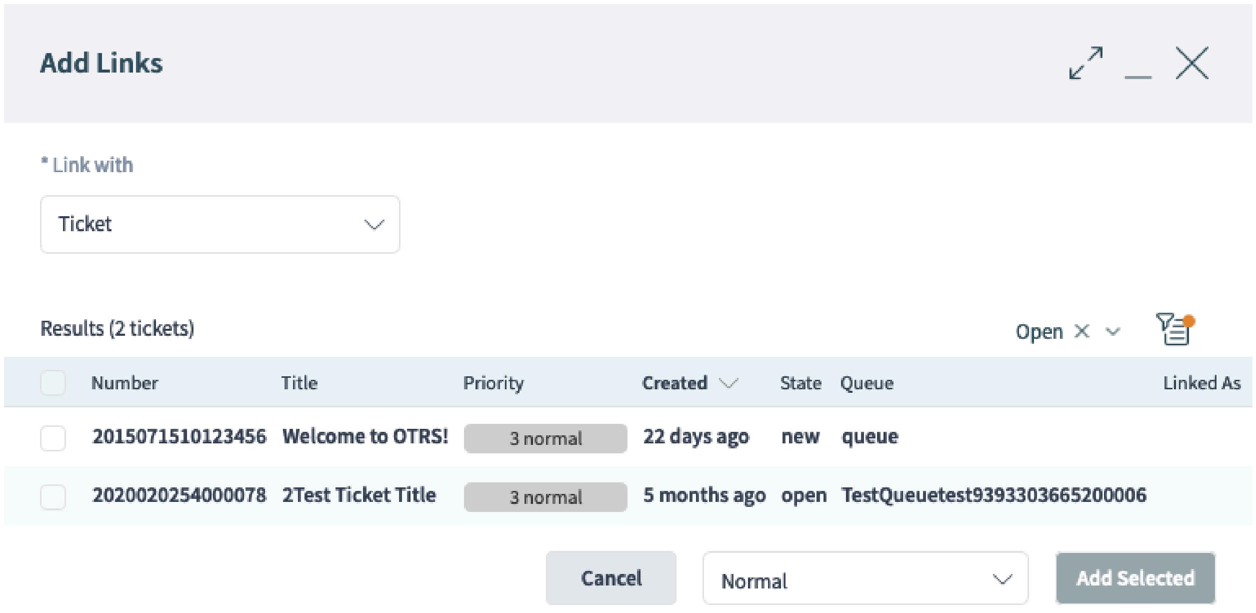
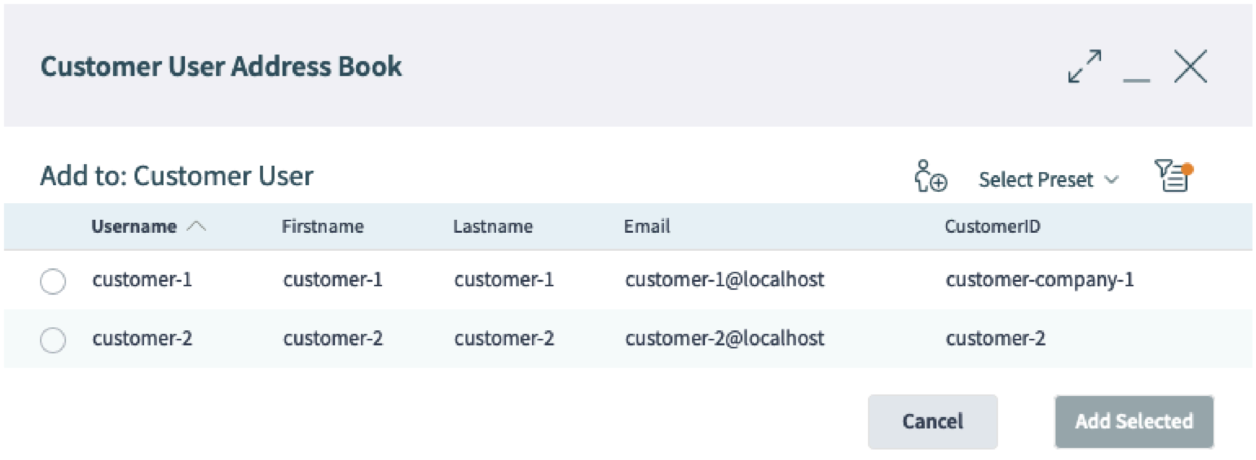


Fig. 61: An Example of the *Article Preview* in the Ticket List Screen



	Number	Title	Priority	Created	State	Queue	Linked As
	2015071510123456	Welcome to OTRS!	3 normal	22 days ago	new	queue	
	2020020254000078	2Test Ticket Title	3 normal	5 months ago	open	TestQueuetest9393303665200006	

Fig. 62: An Example of the *Link Objects* Action in the Ticket Detail View



Add to: Customer User

	Username	Firstname	Lastname	Email	CustomerID
○	customer-1	customer-1	customer-1	customer-1@localhost	customer-company-1
○	customer-2	customer-2	customer-2	customer-2@localhost	customer-2

Fig. 63: An Example of the *Customer User Address Book* List Screen

- AgentFrontend::CustomerUserAddressBookList::SMS###DefaultConfig
- AgentFrontend::CustomerCompanyAddressBookList###DefaultConfig

Each business object list setting has a `Config` key which contains its configuration.

See also:

The detailed explanation of the possible `Config` key values can be found in the *Business Object List YAML Configuration* chapter.

YAML Configuration Basics

Before we delve too deeply into the business object configuration, it is important to explain the YAML syntax which is heavily used for some of the setting values.

YAML (a recursive acronym for *YAML Ain' t Markup Language*) is a **human-readable** data-serialization language. It is commonly used for configuration files, but it can describe data structures of arbitrary complexity.

Although YAML supports many of them, only a few data types are used in the context of OTRS configuration:

Scalar

This is the most basic data type; it could be a number, a string, or a boolean expression. Sometimes it is also referred to as a **value**.

```
Number: 42
String1: Foo
String2: 'Foo bar'
String3: "Foo bar Xyzzy"
Boolean1: 1
Boolean2: 0
```

Note: Scalar data type is never used on its own, but it is always part of another higher-level structure.

List

List contains an arbitrary number of items which can be of any data type. Sometimes it is also referred to as an **array**.

Each item in a list is divided from others by a separator in the form of a minus sign followed by a single space.

```
List1:
- Item1
- Item2
- Item3
```

It is recommended that list items are indented via a minus sign followed by a single space. This style is used throughout this guide and default configuration.

Warning: In YAML, the number of spaces is important. The number of spaces used for indentation must always be consistent, otherwise unexpected structures or syntax errors can occur.

Note: To define an empty list, you can use the square bracket syntax:

```
EmptyList: []
```

Object

Object is a data type consisting of key/value pairs, separated by the colon punctuation mark. Sometimes it is also referred to as a **hash**.

Object **key** is always on the left side of the colon punctuation mark, while the **value** is always on its right side (or indented below):

```
Object1:
  Key1: Value1
  Key2: Value2
  Key3: Value3
```

It is recommended that object key/value pairs are indented with two spaces. This style is used throughout this guide and default configuration.

Warning: In YAML, the number of spaces is important. The number of spaces used for indentation must always be consistent, otherwise unexpected structures or syntax errors can occur.

Note: To define an empty object, you can use the curly brackets syntax:

```
EmptyObject: {}
```

To signal that a text uses YAML syntax, it is recommended to add a **document header** to its first line that consists of three consecutive minus signs, followed by the line break. By doing this consistently, it becomes very easy to identify YAML structures when mixed with other configurations.

With the three basic data types explained above, it is possible to create more complex structures. For example, an **array of hashes**:

```
---
Array:
```

(continues on next page)

(continued from previous page)

```
- Key1: A
  Key2: B
  Key3: C
- Key1: 1
  Key2: 2
  Key3: 3
- Key1: I
  Key2: II
  Key3: III
```

Note: In a list that contains multiple object keys, only the first key must be prefixed with the minus sign followed by a space. Each subsequent key must be prefixed with two spaces.

The following example illustrates the usage of a **hash of arrays**:

```
---
Hash:
  Key:
    Subkey1:
    - Item1
    - Item2
    - Item3
    Subkey2:
    - Item4
    - Item5
    - Item6
```

Note: If the object keys are separated by other structures, it is important to keep them on the same indentation level. Since they are all siblings, they must be prefixed by the same number of spaces.

It is also possible to mix the data types that result in more complex data structures:

```
---
Key1:
- Subkey1: Value1
- Subkey2: Value2
Key2: Value3
Key3:
  Subkey3:
  - Value4
  - Value5
```

YAML supports comments in its code, so it is possible to include additional text that will be ignored during parsing. This is useful in case users need to describe certain parts of the structure, perhaps for future reference, or to quickly ignore part of the code without actually removing it.

Comments can be marked with `#`. Any following text will be ignored.

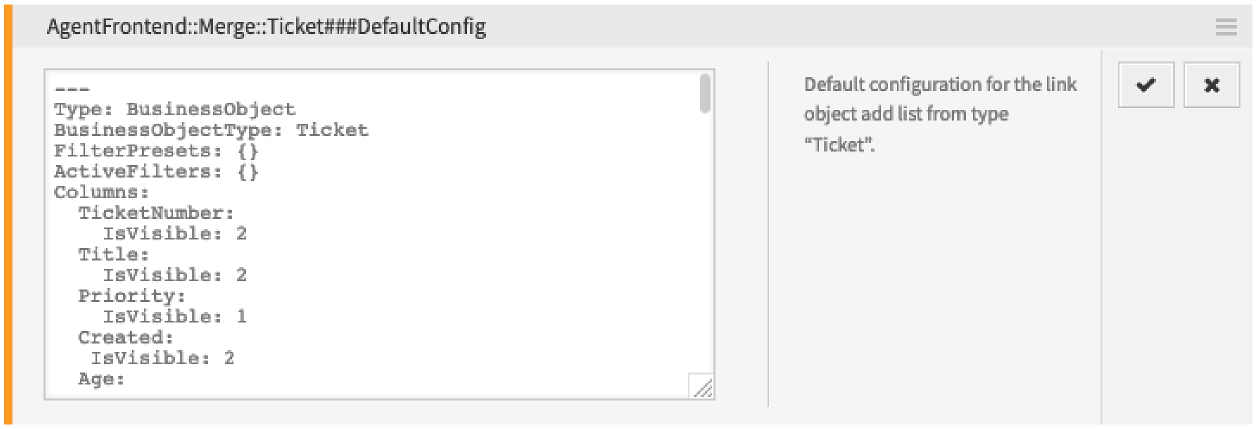
```
---
# This is a comment
Key: Value # This is another comment
# Key2: Value 2
```

In the example above, the last line will be ignored too since it starts with `#`.

If you actually want to use the pound sign as part of your string of values, simply enclose the complete string in quotes.

```
---
String: 'Comments start with # character'
```

When you edit an OTRS system setting with the YAML value type, a suitable text editor will be displayed.

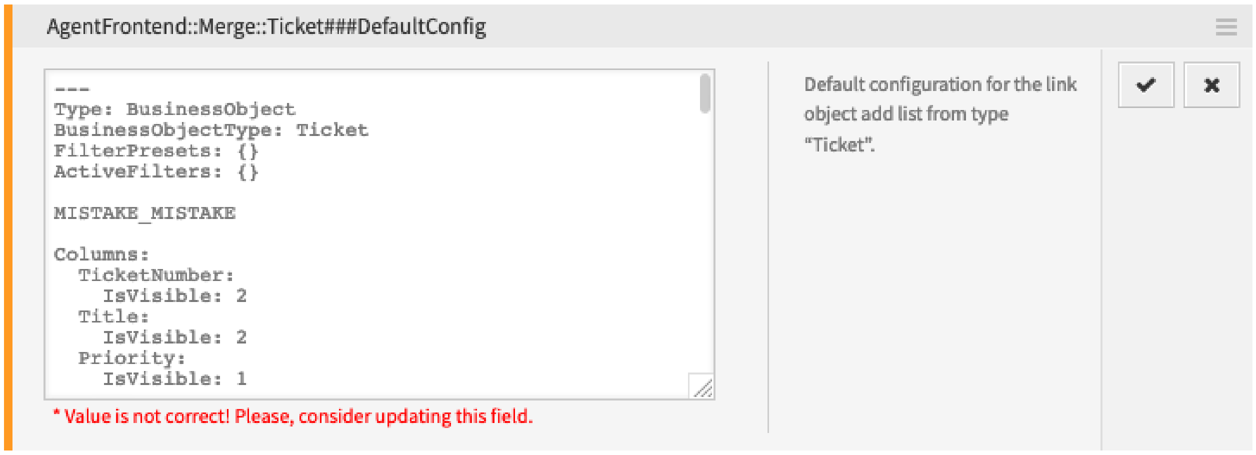

AgentFrontend::Merge::Ticket###DefaultConfig

```
---
Type: BusinessObject
BusinessObjectType: Ticket
FilterPresets: {}
ActiveFilters: {}
Columns:
  TicketNumber:
    IsVisible: 2
  Title:
    IsVisible: 2
  Priority:
    IsVisible: 1
  Created:
   IsVisible: 2
  Age:
```

Default configuration for the link object add list from type "Ticket".

Fig. 64: System Setting with a YAML Value

When saving the setting, the YAML structure will be checked for validity. In case of an error, it is possible to change and correct the structure.


AgentFrontend::Merge::Ticket###DefaultConfig

```
---
Type: BusinessObject
BusinessObjectType: Ticket
FilterPresets: {}
ActiveFilters: {}

MISTAKE_MISTAKE

Columns:
  TicketNumber:
    IsVisible: 2
  Title:
    IsVisible: 2
  Priority:
    IsVisible: 1
```

* Value is not correct! Please, consider updating this field.

Default configuration for the link object add list from type "Ticket".

Fig. 65: Error in a YAML Structure

It is important to understand that only the syntax of the structure is checked during validation. A deep validation will not occur. It is your responsibility as an administrator to ensure that the configuration you specify is correct.

Warning: The most common errors that occur are related to indentation. You have to ensure that your indentations are consistent, especially the indentations of the different data types. For example, the following two structures are considered different.

```
---
Object:
   Key:
      Subkey: Value
---
Object:
   Key:
   Subkey: Value
```

Since the `Subkey` in the second example is not aligned properly within its parent `Key`, it is considered to be its sibling part and not a child. In this case the actual value of the `Key` will be empty.

Business Object List YAML Configuration

A business object list YAML configuration can have many parameters. The following example shows how to construct a business object list in general.

```
Type: BusinessObject
BusinessObjectType: Ticket
ScreenTitle: <Some Screen Title>
Changeable: 1
FilterPresets:
  "<Displayed Filter Preset Name>":
    <FilterName1>:
      Value: <Value>
    <FilterName2>:
      Value:
      - <Value1>
      - <Value2>
    <FilterName3>:
      Value:
        <Parameter1>: <Value1>
        <Parameter2>: <Value2>
  ...
ActiveFilters:
  <FilterName1>:
    Value: <Value>
  <FilterName2>:
    Value:
    - <Value1>
    - <Value2>
  <FilterName3>:
    Value:
      <Parameter1>: <Value1>
      <Parameter2>: <Value2>
      ...
  ...
AllowGETConfig:
- <ConfigurationName1>
- <ConfigurationName2>
- ...
Columns:
  <ColumnName1>:
    IsVisible: 0|1|2
    IsInlineEditable: 0|1
  ...
```

(continues on next page)

(continued from previous page)

```
DefaultColumnOrder:
- <ColumnName1>
- <ColumnName2>
- ...
HideAvailableFilters:
- <FilterName1>
- <FilterName2>
- ...
ItemsPerPage: 10
Limit: 1000
SortBy:
- Column: <ColumnName>
  Direction: Down|Up
- ...
AvailableDynamicFieldFilters:
- <FilterName1>
- <FilterName2>
- ...
```

The following YAML keys and values can be used for business object lists.

`Type`
Defines the type of the setting. This is an internal value which is always set to `BusinessObject` and must not be changed.

> **Warning:** Changing this value might result in a broken system configuration.

`BusinessObjectType`
Defines the business object type of the business object list. This is an internal value which is specific for the current list type and must not be changed.

> **Warning:** Changing this value might result in a broken system configuration.

`ScreenTitle`
Defines the title of the business object list.

> **Note:** In some places, you may notice `# translatable` comments that are displayed next to the text values. This marker is only used for the internal collection of translatable strings and has no effect in the changed settings. Adding this comment to modified values will not result in the actual text translation.
>
> The visible text in the configuration can be translated into other languages with custom language files. For more information, see the *Custom Language File* chapter.

`Changeable`
Defines whether the business object list can be personalized by the user in the front end. If turned off, the view configuration will not be available and also filter presets for the view cannot be saved. The value can be `1` (on) or `0` (off). The assumed default is off when the key is not used in the definition.

`FilterPresets`
Defines the pre-defined filters used for the business object list. Multiple filters can be defined here.

The filter value can be a string, array or a hash. The following YAML snippet shows examples for all types.

```
FilterPresets:
  Closed:
    StateType:
      Value: Closed
  Locked:
    LockIDs:
      Value:
      - 2
  "Reminder Reached":
    TicketPending_DateTimeRelative:
      Value:
        Format: minute
        Point: 1
        Start: Before
```

See also:

The list of possible filter names can be found in the *Filter Names* chapter.

`ActiveFilters`

Defines the filters that are active when displaying the business object list. Multiple filters can be added here; each is defined by the filter name. The filter value can be a string, array or a hash. The following YAML snippet shows examples for both.

```
ActiveFilters:
  StateType:
    Value: Closed
  TicketClose_DateTimeRelative:
    Value:
      Start: Before
      Point: 1
      Format: minute
```

See also:

The list of possible filter names can be found in the *Filter Names* chapter.

`AllowGETConfig`

Defines what parameters can be changed via the URL with the `Config` URL query parameter:

```
AllowGETConfig:
- VisibleColumns
- SortBy
- ActiveFilters
- FilterPresets
- ItemsPerPage
- FilterPresetSelected
```

The list should contain names of other configuration keys for which definition will be allowed when passed via the correct URL query parameter to the view.

See also:

For an example and more information, please see the section about *Custom URL Support*.

`Columns`

Defines the columns that are available for the business object list in the front end.

`IsVisible`
Defines whether the column is not visible (`0`), not visible by default but the agent can make it visible (`1`) or visible by default (`2`).

`IsInlineEditable`
Defines whether the value in the column is inline editable (`1`) or not (`0`).

See also:

The list of possible column names can be found in the *Column Names* chapter.

`DefaultColumnOrder`
Defines the default column order in the business object list.

`HideAvailableFilters`
Defines the filters which are not available for the agents in the front end, either in the view configuration or in the filter presets.

`ItemsPerPage`
Defines the number of objects per page that are displayed by default and the number of new objects that are loaded when the agent scrolls down the list. If omitted, it takes the default value of `10`. Although there is no upper limit, it is not advisable to set the value greater than `100` for performance reasons.

`Limit`
Defines the maximum amount of displayed objects in a business object list. Most of the list types in the system are limited to an upper boundary of 10,000 objects for performance reasons.

`SortBy`
Defines the sorting criteria and the sorting order of the objects in the business object list.

`AvailableDynamicFieldFilters`
Defines the list of dynamic fields by name which are available as filters.

Business Object Detail Views

The business object detail views can be organized into several families.

The following settings define the default column layout configuration for the detail view screens.

The following system configuration settings are relevant:

- AgentFrontend::CustomerCompanyDetailView###001-Framework
- AgentFrontend::CustomerUserDetailView###001-Framework
- AgentFrontend::KnowledgeBaseArticleDetailView###001-Framework
- AgentFrontend::TicketDetailView###001-Framework

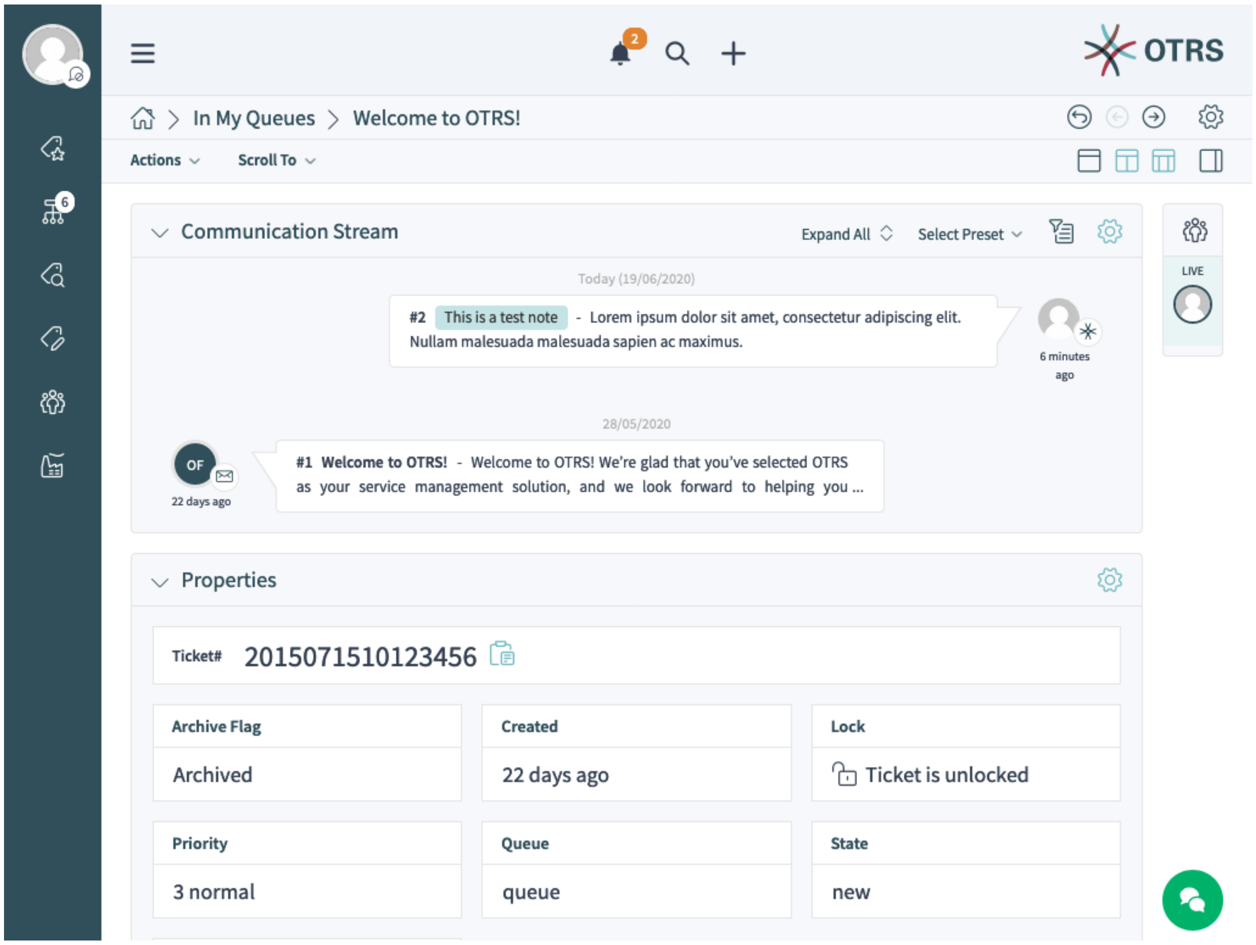
The following settings define the default column layout configuration for the business object overview screens.

The following system configuration settings are relevant:

- AgentFrontend::CalendarOverview###001-Framework
- AgentFrontend::Dashboard###001-Framework
- AgentFrontend::StatisticReportOverview###001-Framework

The following settings define the default column layout configuration for the business object create screens.

The following system configuration settings are relevant:



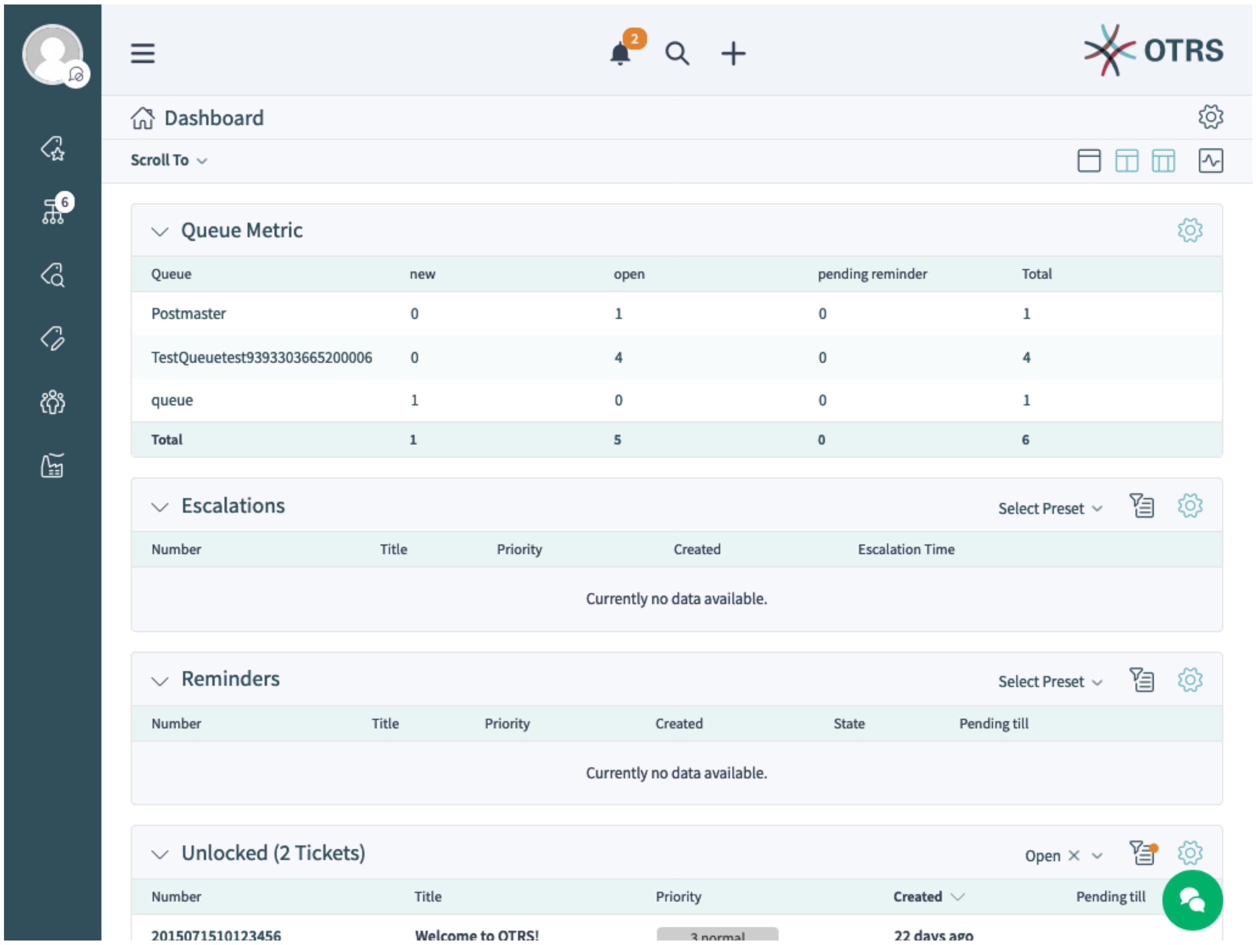
#2 This is a test note - Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam malesuada malesuada sapien ac maximus.

#1 Welcome to OTRS! - Welcome to OTRS! We're glad that you've selected OTRS as your service management solution, and we look forward to helping you ...

Ticket# 2015071510123456

Archive Flag	Created	Lock
Archived	22 days ago	Ticket is unlocked
Priority	Queue	State
3 normal	queue	new

Fig. 66: An Example of the *Ticket* Detail View



Queue Metric

Queue	new	open	pending reminder	Total
Postmaster	0	1	0	1
TestQueuetest9393303665200006	0	4	0	4
queue	1	0	0	1
Total	**1**	**5**	**0**	**6**

Escalations

Select Preset

Number	Title	Priority	Created	Escalation Time

Currently no data available.

Reminders

Select Preset

Number	Title	Priority	Created	State	Pending till

Currently no data available.

Unlocked (2 Tickets)

Open

Number	Title	Priority	Created	Pending till
2015071510123456	Welcome to OTRS!	3 normal	22 days ago	

Fig. 67: An Example of the *Dashboard* Screen

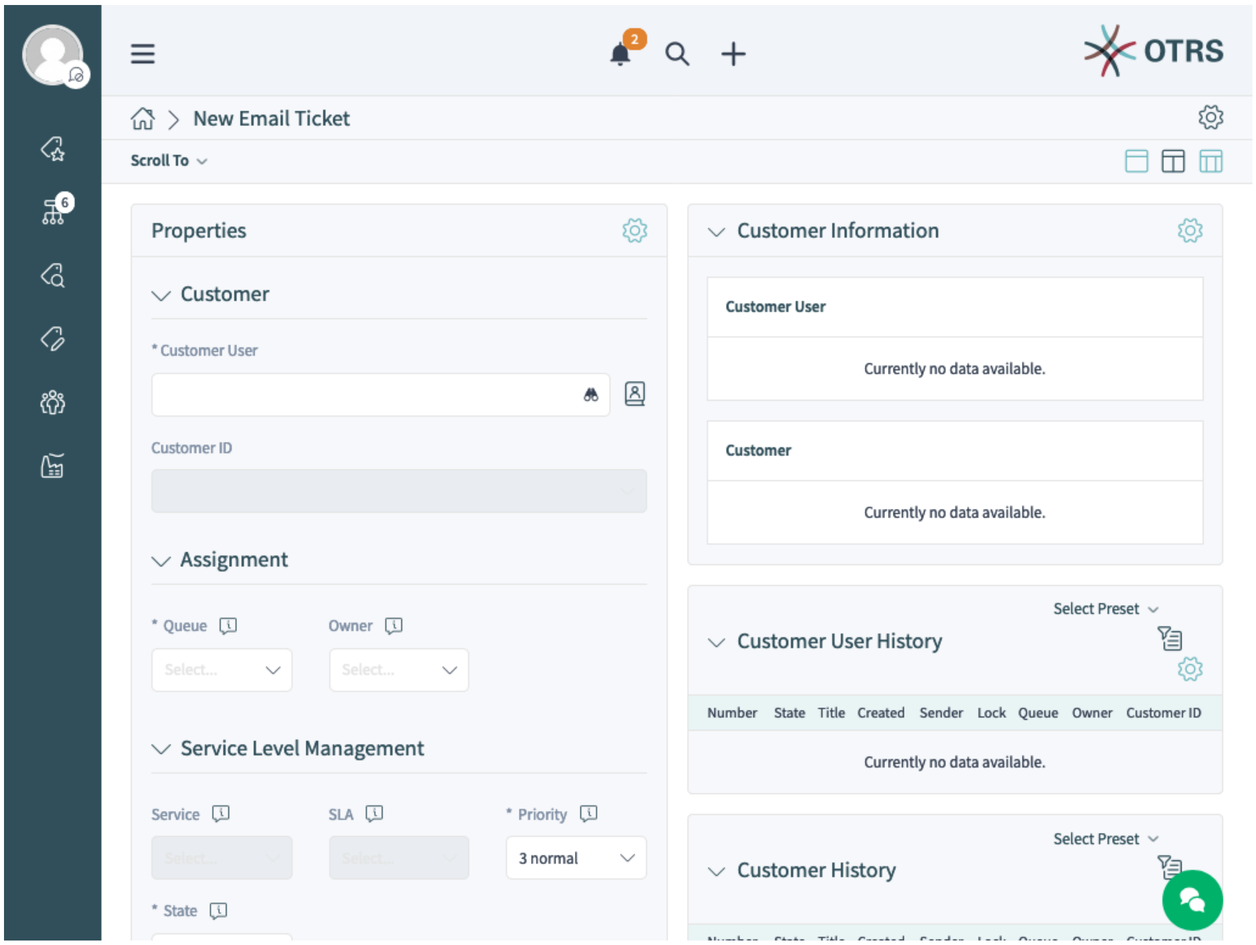


Fig. 68: An Example of the *Create Email Ticket* Screen

- AgentFrontend::CustomerCompanyCreate###001-Framework
- AgentFrontend::CustomerUserCreate###001-Framework
- AgentFrontend::KnowledgeBaseArticleCreate###001-Framework
- AgentFrontend::StatisticCreateUpdateView###001-Framework
- AgentFrontend::StatisticReportCreateUpdateView###001-Framework
- AgentFrontend::TicketCreate::Email###001-Framework
- AgentFrontend::TicketCreate::Phone###001-Framework
- AgentFrontend::TicketCreate::Process###001-Framework
- AgentFrontend::TicketCreate::SMS###001-Framework

The following settings define the custom column layout configurations for adding additional widgets to a specific view. By default, these are empty settings. If additional widgets are added to them, they will be merged with the remaining settings for the corresponding view.

The following system configuration settings are relevant:

- AgentFrontend::CalendarOverview###100-Custom
- AgentFrontend::CustomerCompanyCreate###100-Custom
- AgentFrontend::CustomerCompanyDetailView###100-Custom
- AgentFrontend::CustomerUserCreate###100-Custom
- AgentFrontend::CustomerUserDetailView###100-Custom
- AgentFrontend::Dashboard###100-Custom
- AgentFrontend::KnowledgeBaseArticleCreate###100-Custom
- AgentFrontend::KnowledgeBaseArticleDetailView###100-Custom
- AgentFrontend::StatisticCreateUpdateView###100-Custom
- AgentFrontend::StatisticReportCreateUpdateView###100-Custom
- AgentFrontend::StatisticReportOverview###100-Custom
- AgentFrontend::TicketCreate::Email###100-Custom
- AgentFrontend::TicketCreate::Phone###100-Custom
- AgentFrontend::TicketCreate::Process###100-Custom
- AgentFrontend::TicketCreate::SMS###100-Custom
- AgentFrontend::TicketDetailView###100-Custom

Each of the above settings has the same key structure:

`Type`

Type of the business object screen.

Possible values for the `Type` key:

```
BusinessObjectCreate
BusinessObjectDetailView
BusinessObjectOverview
```

`BusinessObjectType`

Business object type behind the screen.

Possible values for the `BusinessObjectType` key:

```
CustomerCompany
CustomerUser
Dashboard
KnowledgeBaseArticle
Statistic
StatisticReport
Ticket
```

`ColumnLayout`

The YAML configuration for different column views.

See also:

The detailed explanation of the `ColumnLayout` key can be found in the *Business Object Detail View YAML Configuration* chapter.

Business Object Detail View YAML Configuration

The YAML structure defines the content for the one, two or three column views. The definition contains the information about which widgets are displayed per default in each view.

```
ActiveColumnLayout: OneColumn
OneColumn:
  1:
  - Name: <WidgetName1>
  - Name: <WidgetName2>
  ...
TwoColumns:
  1:
  - Name: <WidgetName1>
  - Name: <WidgetName2>
  ...
  2:
  - Name: <WidgetName3>
  - Name: <WidgetName4>
  ...
ThreeColumns:
  1:
  - Name: <WidgetName1>
  - Name: <WidgetName2>
  ...
  2:
  - Name: <WidgetName3>
  - Name: <WidgetName4>
  ...
  3:
  - Name: <WidgetName5>
  - Name: <WidgetName6>
  ...
StripeSidebar:
- Name: StripePeople
...
```

Note: The sidebar is only available for business objects of type `Ticket`.

`ActiveColumnLayout`
Controls which column layout should be loaded by default. Possible values are: `OneColumn`, `TwoColumns` or `ThreeColumns`.

Available widget names for specific column layouts depend on the specific business object screen.

`CalendarOverview`
Possible widget names for the *Calendar Overview* screen.

```
AppointmentsToday
AppointmentsThisWeek
AppointmentsThisMonth
```

`CustomerCompanyCreate`
Possible widget names for the *Add Customer* screen.

```
CreateProperties
```

`CustomerCompanyDetailView`
Possible widget names for the *Customer* detail view screen.

```
CustomerInformation
CustomerUserList
EscalatedTickets
OpenTickets
ReminderTickets
TicketList
```

`CustomerUserCreate`
Possible widget names for the *Add Customer User* screen.

```
CreateProperties
```

`CustomerUserDetailView`
Possible widget names for the *Customer User* detail view screen.

```
CustomerInformation
EscalatedTickets
OpenTickets
ReminderTickets
TicketList
```

`Dashboard`
Possible widget names for the *Dashboard* screen.

```
CalendarView
CustomerList
CustomerUserList
DashboardIframe
DashboardImage
DashboardPeople
EscalatedTickets
KnowledgeBaseArticleList
News
```

(continues on next page)

(continued from previous page)

```
OpenTickets
QueueOverview
RecentlyUpdatedKnowledgeBaseArticles
ReminderTickets
TicketList
UnlockedTickets
```

`KnowledgeBaseArticleCreate`
Possible widget names for the *Add Knowledge Base Article* screen.

```
CreateProperties
```

`KnowledgeBaseArticleDetailView`
Possible widget names for the *Knowledge Base Article* detail view screen.

```
KBAAttachments
KBAItemField1
KBAItemField2
KBAItemField3
KBAItemField4
KBAItemField5
KBAItemField6
KBALinkedObjects::CalendarAppointment
KBALinkedObjects::KnowledgeBaseArticle
KBALinkedObjects::Ticket
KBAProperties
KBARating
People
```

`StatisticCreateUpdateView`
Possible widget names for the *Create Statistic* and *Edit Statistic* screens.

```
CreateUpdateProperties
```

`StatisticReportCreateUpdateView`
Possible widget names for the *Create Report* and *Edit Report* screens.

```
CreateUpdateProperties
```

`StatisticReportOverview`
Possible widget names for the *Statistics and Reports* overview screen.

```
StatisticLists
StatisticMetrics
StatisticReportList
StatisticStatic
```

`TicketCreate::Email`
Possible widget names for the *Create Email Ticket* screen.

```
CreateProperties
CustomerHistory
CustomerInformation
CustomerUserHistory
```

`TicketCreate::Phone`
Possible widget names for the *Create Phone Ticket* screen.

```
CreateProperties
CustomerHistory
CustomerInformation
CustomerUserHistory
```

`TicketCreate::SMS`
Possible widget names for the *Create SMS Ticket* screen.

```
CreateProperties
CustomerHistory
CustomerInformation
CustomerUserHistory
```

`TicketCreate::Process`
Possible widget names for the *Create Process Ticket* screen.

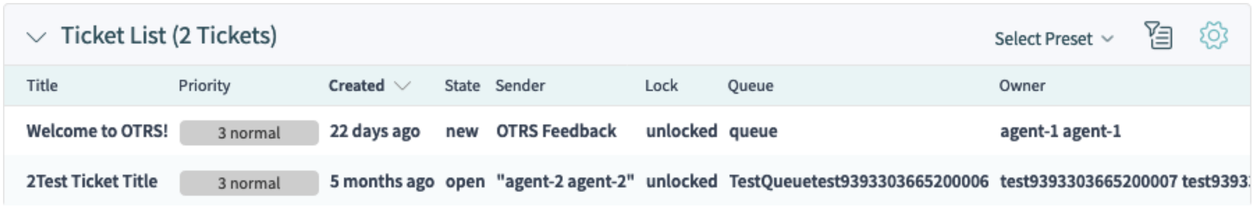
```
CreatePropertiesProcess
ProcessInformation
```

`TicketDetailView`
Possible widget names for the ticket detail view screen.

```
Attachments
BusinessProcessInformation
CommunicationCompact
CommunicationStream
CustomerInformation
FormDrafts
LinkedObjects::CalendarAppointment
LinkedObjects::KnowledgeBaseArticle
LinkedObjects::Ticket
People
Properties
```

Widget Type Definitions

Each widget type provides its own definition that will be inherited by any widget that uses it. For example, let’ s take a look how this definition looks for one of the dashboard widget types:



Title	Priority	Created	State	Sender	Lock	Queue	Owner
Welcome to OTRS!	3 normal	22 days ago	new	OTRS Feedback	unlocked	queue	agent-1 agent-1
2Test Ticket Title	3 normal	5 months ago	open	"agent-2 agent-2"	unlocked	TestQueuetest9393303665200006	test9393303665200007 test9393

Fig. 69: An Example of the *Ticket List* Widget Type in the *Dashboard* Screen

The configuration of this widget type is located in the AgentFrontend::Dashboard::WidgetType###TicketList system configuration setting in YAML format.

A widget type definition contains the following general YAML keys:

```
Type: BusinessObject
BusinessObjectType: Ticket
ActiveFilters: {}
FilterPresets: {}
Columns: {}
Collapsed: 0
Hidden: 0
```

`Type`
Defines whether the related widget type handles a business object or other data. `BusinessObject` is currently the only supported type, but this might be extended by installed packages.

`BusinessObjectType`
This key defines the type of object the configuration is valid for. Based on this type, the front end and the back end will handle this object differently. New business object types may also be added by installed packages.

`ActiveFilters`
This key defines the default filters that are active when displaying the business object list. It contains a hash of filters. A single filter also has a hash structure that normally contains a filter value. The value can be a string, array or hash for special filters like dates.

```
ActiveFilters:
  <FilterName1>:
    Value:
      <Value>
  <FilterName2>:
     Value:
       <Value>
```

The following example illustrates two active filters which are showing only tickets that were closed recently:

```
ActiveFilters:
  StateType:
    Value: Closed
  TicketClose_DateTimeRelative:
    Value:
      Start: Before
      Point: 1
      Format: minute
```

See also:

The list of possible filter names can be found in the *Filter Names* chapter.

`FilterPresets`
A hash of default filter preset names with a defined set of filters.

```
FilterPresets:
  "<Filter Preset Name 1>":
      <FilterName1>:
         Value: <Value>
  "<Filter Preset Name 2>":
      <FilterName2>:
         Value: <Value>
```

The following example illustrates three separate filter presets: locked, unlocked and unread tickets.

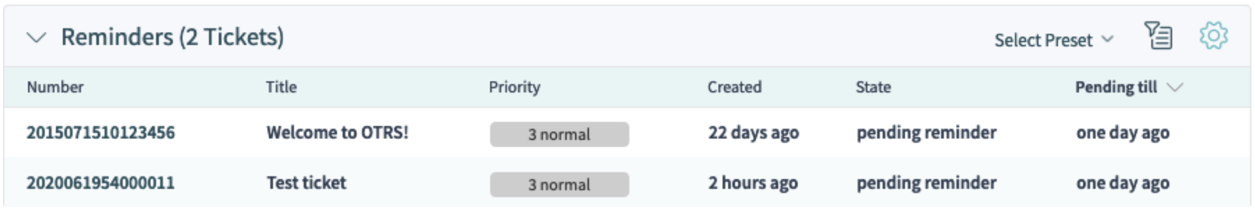
```
FilterPresets:
  Locked:
    LockIDs:
      Value:
      - 2
  Unlocked:
    LockIDs:
      Value:
      - 1
  Unread:
    AgentTicketFlagSeen:
      Value: Unread
```

See also:

The list of possible filter names can be found in the *Filter Names* chapter.

Widget Definitions

Each widget provides its own definition that will be merged with the configuration of the inherited widget type. For example, let' s take a look at what this definition looks like for one of the widgets in the customer business object detail view:



Number	Title	Priority	Created	State	Pending till
2015071510123456	Welcome to OTRS!	3 normal	22 days ago	pending reminder	one day ago
2020061954000011	Test ticket	3 normal	2 hours ago	pending reminder	one day ago

Fig. 70: An Example of the *Reminder Tickets* Widget in the *Customer* Detail View

The configuration of this widget is located in the AgentFrontend::CustomerCompanyDetailView::Widget###ReminderTickets system configuration setting in YAML format.

A widget definition contains the following general YAML keys:

```
Title: Reminders # Translatable
Active: 1
IsVisible: 1
IsAlwaysPresent: 0
IsDuplicatable: 1
Config: {}
```

Active
Defines whether the widget is active.

Collapsed
Defines whether the form group is collapsed by default.

Config
This key contains the complete default configuration of a widget. If left empty, the configuration of the widget type is used.

Hidden
Defines whether the widget is hidden.

`IsVisible`
Defines whether the widget is visible per default.

`IsAlwaysPresent`
Defines whether the widget can be removed from a view by the user.

`IsDuplicatable`
Defines whether the widget can be placed in a view multiple times.

`Title`
Defines the displayed title of a widget in the header section.

Note: In some places, you may notice `# translatable` comments that are displayed next to the text values. This marker is only used for the internal collection of translatable strings and has no effect in the changed settings. Adding this comment to modified values will not result in the actual text translation.

The visible text in the configuration can be translated into other languages with custom language files. For more information, see the *Custom Language File* chapter.

If the widget contains a business object list:

`Columns`
A hash with column names that are displayed in the business object list.

```
Columns:
  <ColumnName1>:
     IsVisible: <Value>
  <ColumnName2>:
     IsVisible: <Value>
```

Possible values for the `IsVisible` key:

- `0` = not available
- `1` = available but not visible
- `2` = available and visible

See also:

The list of possible column names can be found in the *Column Names* chapter.

`ArticleDynamicFields`
Contains a list of dynamic fields to be displayed in the article preview. The dynamic fields must be added **without** the prefix `DynamicField_`.

```
ArticleDynamicFields:
  - <FieldName1>
  - <FieldName2>
  ...
```

`ArticleViewType`
Defines whether the articles should be displayed collapsed or expanded by default.

```
ArticleViewType: collapsed
```

`DefaultColumnOrder`
An array of column names that defines the default column order in the business object list.

```
DefaultColumnOrder:
- <ColumnName1>
- <ColumnName2>
...
```

DefaultFilterPresetFields
Defines the default filter preset fields and their values.

```
DefaultFilterPresetFields:
  <FilterName1>:
    Value: '<Value>'
  <FilterName2>:
    Value: '<Value>'
```

CountPolling
This determines whether the organizer item queries the current number of its objects in the background and displays them in a speech bubble next to the icon.

```
CountPolling: ShowNumberFoundItems
```

Possible values for the `CountPolling` key:

- `0` = do not show number
- `ShowNumberFoundItems` = show number of found tickets

Header

This key contains the definition of information fields to be displayed in the header section of the agent business cards.

```
Header:
  Properties:
  - Name: Avatar
    IsVisible: 1
```

HideAvailableFilters
This key contains a list of filter names that are not present in the list of available filters in the widget configuration.

```
HideAvailableFilters:
- <FilterName1>
- <FilterName2>
- <FilterName3>
```

The following example hides filters for the ticket customer, text search and ticket type:

```
HideAvailableFilters:
- CustomerID
- Fulltext
- TypeIDs
```

Identifier
Defines the property that will be used as a full-width card in the *Properties* widget. The property must be a unique identifier of the business object. If defined this property can be configured separately from other cards.

```
Identifier:
  Name: TicketNumber
  IsVisible: 0
```

InitialLimit
Defines the limit of displayed agents in the stripe sidebar in the collapsed state.

ItemsPerPage
A number that defines the number of items per page or load.

Limit
A number that defines the maximum amount of displayed items.

LastUsedFilterPreset
Defines the last used filter preset of a widget. This will be automatically applied when the widget is loaded.

```
LastUsedFilterPreset: "Reminder Reached"
```

Properties
The properties are special containers (cards) within a widget that contain detailed information about a specific value of the associated business object. The cards have the option to allow editing of the associated value directly, such as setting a new queue or the owner of a ticket. The properties list in the configuration contains the name and default visibility state of every property.

```
Properties:
- Name: ArchiveFlag
  IsVisible: 1
- Name: Created
  IsVisible: 1
- Name: CustomerTickets
  IsVisible: 1
- Name: Lock
  IsVisible: 1
  IsInlineEditable: 0
- Name: Watch
  IsVisible: 1
  IsInlineEditable: 0
```

Possible values for the `IsVisible` key:

- `0` = not available
- `1` = available but not visible
- `2` = available and visible

Possible values for the `IsInlineEditable` key:

- `0` = not editable
- `1` = editable

ShowPropertyOnEmpty
Defines whether the customer or customer user business card is displayed in the customer information widget when it does not contain a value.

```
ShowPropertyOnEmpty: 1
```

Note: This key is supported only by the customer and customer user business cards and cannot be used for regular property cards.

`SortBy`

Defines the sort order of the business object list. To sort by multiple columns simultaneously, add each sort column as a separate element in the configuration.

```
SortBy:
- Column: Created
  Direction: Down
- Column: Priority
  Direction: Up
```

See also:

The list of sortable columns per business object type can be found in the *Column Names* chapter. Look for the (sortable) marker next to the column name.

Note: Only certain business object types support the multi-sorting feature, please see *Column Names* chapter for more information.

Forms

The configurable forms can be used in many screens, including ticket and article actions. The form configuration will define the fields and properties displayed for each action. Here is the form of *Add Note* action as an example.

The configuration of this form is located in the Forms###AgentFrontend::Ticket::Action::Note system configuration setting in YAML format.

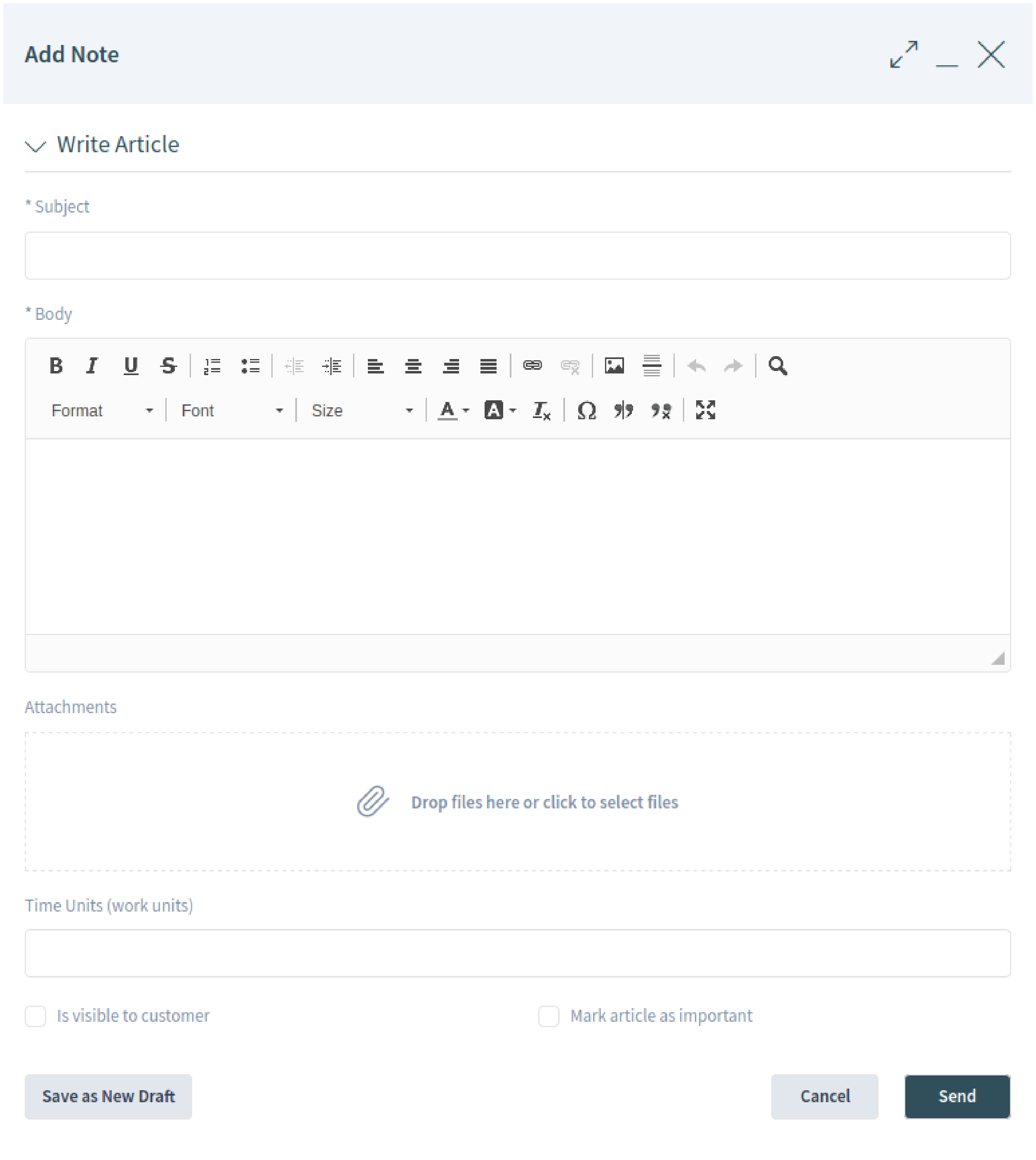
In general a form can contain a list of fields and form groups. Both are optional and can be added multiple times.

```
---
- <Field>
- ...
- <FormGroup>
- ...
```

The field is the basic element of a form. A field can have multiple properties.

```
- Name: <InternalName>
  Label: <Displayed Label>
  Config:
    <Parameter>:
    - <value>
    - ...
  Default: <default value>
  Description: <some description>
  Disabled: 1
  Hidden: 1
  Hint: <some hint>
  Required: 1
```

(continues on next page)



* Subject

* Body

Attachments

Time Units (work units)

Is visible to customer

Mark article as important

Fig. 71: Form of *Add Note* Action

(continued from previous page)

```
  Props:
    MaxLength: 20
```

The following keys and values can be used for the fields.

Warning: The keys marked with the * (mandatory) symbol are mandatory. Skipping them might result in broken configuration and/or non-functional features.

If a non-mandatory key is missing from the form field configuration, it will assume the built-in default behavior.

Warning: Most forms and their fields are subject to existing *Access Control Lists (ACL)*. These rules have the final word on which values can be used in the fields and which will overwrite the form configuration.

Name *

Defines the internal name of the field. This key has pre-defined values for each form.

See also:

The list of possible field names can be found in the *Form Fields* chapter.

The dynamic fields can also be added to the form by using the `DynamicField_` prefix and the name of the dynamic field. For example, if there is a dynamic field with the name `TestDropdown`, you need to use `DynamicField_TestDropdown` in the form configuration.

Label

Defines the label of the field displayed above the input element. If omitted, the default label is displayed for the field specified with `Name`.

Note: In some places, you may notice `# translatable` comments that are displayed next to the text values. This marker is only used for the internal collection of translatable strings and has no effect in the changed settings. Adding this comment to modified values will not result in the actual text translation.

The visible text in the configuration can be translated into other languages with custom language files. For more information, see the *Custom Language File* chapter.

Config

Defines the values that can be selected in a drop-down list. The values depend on the field specified with `Name`.

```
- Name: StateID
  Config:
    StateType:
    - open
    - pending auto
    - pending reminder
    - closed
  Default: 4 # open
```

Note: The `Config` key is not applicable for all fields. It is only supported for specific fields. If the key is missing from the default configuration, it is most likely not supported by that field.

Default
Defines the default value for the field. If the form field refers to an ID, the default value will be an ID from the appropriate database table.

In the example above, the state type `open` has the `id=4` in the `ticket_state` table.

Note: The `Default` key must refer to a constant value. Substituting it with search terms, regular expressions or similar constructs is currently not supported.

Description
Defines the description of the field. The description is displayed next to the label as a bubble icon and shows a tooltip when the mouse is hovered.

Disabled
Defines whether the field is displayed in a disabled state. A disabled field means that the field is read-only. The field value will, however, be sent by submitting the form.

Hidden
Defines whether the field will be hidden. A hidden field is still part of the form and its value will be sent by submitting the form, but it will not be displayed in the user interface.

Hint
Defines explanation text for the field which is displayed below the input element as help text.

Required
Defines whether the field is mandatory. A mandatory field cannot be empty.

Props
Props are like traits or additional behavior of the front end components, and normally they are used in the source code by the developers, but the configurable form feature now allows modifying their values.

See also:

For the complete list of props, see the API documentation of the design system.

The fields can be grouped into form groups. The form groups can contain other form groups or fields. The form groups can be displayed in one, two or three column layouts.

```
- Label: <Displayed Label>
  Collapsible: 1
  Collapsed: 1
  Fields:
  - <Field>
  - ...
  - <FormGroup>
  - ...
  - ColumnLayout: N # Repeated N-times (2 or 3)
    Fields:
    - <Field>
    - ...
    - <FormGroup>
    - ...
```

The following keys and values can be used for the form groups.

Warning: The keys marked with the * symbol are mandatory. Skipping them might result in broken configuration and/or non-functional features.

`Label` *
: Defines the label of the form group.

`Collapsible`
: Defines whether the form group is collapsible.

`Collapsed`
: Defines whether the form group is collapsed by default.

`Fields` *
: Lists the fields and form groups that belong to this form group. There is no limitation on how many fields can be added or how many form groups can be nested.

  If the form group has no fields and acts only as placeholder, then this key has to be defined as an empty list.

```yaml
- Label: Placeholder Form Group
  Fields: []
```

`ColumnLayout`
: Defines the grid in the form component. The fields are displayed in one column layout by default. This key can be used only for the two or three column layouts. The width of the columns are the same, 50%-50% for two column layout and 33%-33%-33% for three column layout.

```yaml
- Label: Two Column Example
  - ColumnLayout: 2
    Fields:
    - Name: IsVisibleForCustomer
  - ColumnLayout: 2
    Fields:
    - Name: MarkAsImportant
```

Business Cards

Business cards are used throughout the system to display additional information about agent users.

The configuration of this business card is located in the AgentFrontend::BusinessCard::User system configuration setting in YAML format.

`AdditionalProperties`
: This key is used to show additional user fields in the agent business card. Each field is referenced by its `Name` key. It is possible to customize its `DisplayName` and control visibility via `IsVisible` flag (1 shows it, 0 hides it).

`Contact`
: This key defines contact options in agent business cards. Each property item is referencing a user field with some contact information via the `Name` key (i.e. `UserEmail`), an icon to show the user (regular weight only!) and can be made visible via the `IsVisible:  1` key. By default, clicking on these contact icons will copy the value of the user field to the clipboard.

  Alternatively, the `Link` key can be specified to show a URL instead. Each `Link` key supports the TemplateToolkit syntax for replacements, and will receive all of the field values of the user via the `Data` variable.

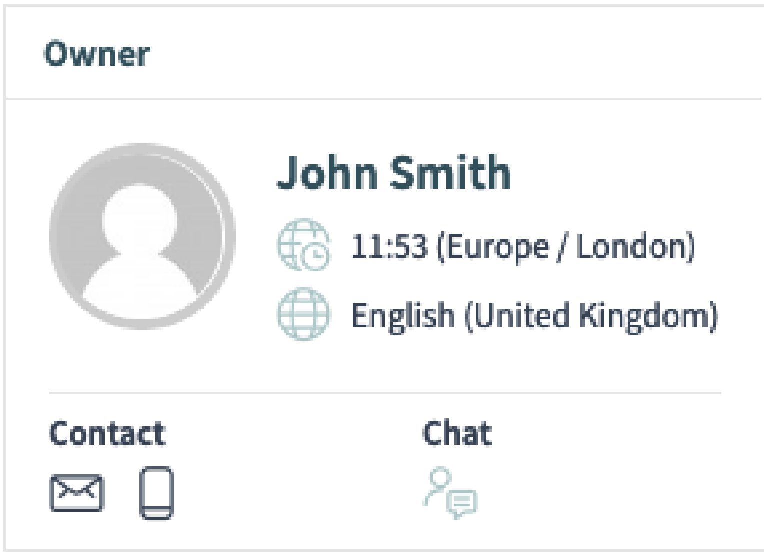


Fig. 72: An Example of the Ticket *Owner* Business Card

`Chat`
This key adds an icon to call the chat function directly from a business card.

`Header`
This key defines the information shown in the header of a business card. Each field is referenced by its `Name` key. It is only possible to control the visibility via `IsVisible` flag (1 shows it; 0 hides it).

10.11.7 Business Object Reference

This chapter lists all possible values for specific business object configurations. The values can be different in each setting.

Column Names

The columns can be referenced by their name. This section lists the possible column names for specific business object types.

Note: The columns marked with the symbol are sortable. The default limit for the multi-sorting feature is a maximum of three columns at a time, unless explicitly mentioned differently.

`ChatRequest`
Possible column names:

- `Action`
- `Channel`
- `CreateTime`
- `Description`
- `RequesterName`

- `RequesterType`
- `Type`

Note: This business object type does not support the multi-sorting feature. Sorting is supported only by a single column at a time.

`CustomerCompany`

Possible column names depend on the available fields in the `CustomerCompany###Map` configuration array. For the default database back end, these include:

- `CustomerCompanyCity`
- `CustomerCompanyComment`
- `CustomerCompanyCountry`
- `CustomerCompanyName`
- `CustomerCompanyStreet`
- `CustomerCompanyURL`
- `CustomerCompanyZIP`
- `CustomerID`
- `ValidID`

Additional column names which are always available:

- `ClosedTickets`
- `Edit`
- `OpenTickets`

`CustomerUser`

Possible column names depend on the available fields in the `CustomerUser###Map` and the `CustomerCompany###Map` configuration arrays. For the default database back ends, these include:

- `CustomerCompanyCity`
- `CustomerCompanyComment`
- `CustomerCompanyCountry`
- `CustomerCompanyName`
- `CustomerCompanyStreet`
- `CustomerCompanyURL`
- `CustomerCompanyValidID`
- `CustomerCompanyZIP`
- `CustomerID`
- `UserCity`
- `UserComment`
- `UserCountry`
- `UserCustomerID`

- `UserEmail`
- `UserFax`
- `UserFirstname`
- `UserLastname`
- `UserLogin`
- `UserMobile`
- `UserPassword`
- `UserPhone`
- `UserStreet`
- `UserTitle`
- `UserZip`
- `ValidID`

Additional column names which are always available:

- `Chat`
- `ClosedTickets`
- `CreateTicket`
- `Edit`
- `OpenTickets`
- `SwitchToCustomer`

`FormDraft`

Possible column names:

- `Delete`
- `Saved`
- `Title`
- `Type`

`KnowledgeBaseArticle`

Possible column names:

- `Approved`
- `Category`
- `Changed`
- `ChangedBy`
- `ContentType`
- `Created`
- `CreatedBy`
- `Helpful`
- `Language`
- `NotHelpful`

- `Number`
- `State`
- `Title`
- `Valid`

`KnowledgeBaseArticleAttachment`
Possible column names:

- `ContentType`
- `CreateTime`
- `Download`
- `Filename`
- `Filesize`
- `Preview`

Note: This business object type does not support the multi-sorting feature. Sorting is supported only by a single column at a time.

`Search`
Possible column names:

- `Result`
- `Source`
- `Type`

`Statistic`
Possible column names:

- `Changed`
- `Created`
- `ObjectName`
- `ObjectType`
- `StatNumber`
- `StatType`
- `Title`
- `Valid`

Note: This business object type does not support the multi-sorting feature. Sorting is supported only by a single column at a time.

`StatisticReport`
Possible column names:

- `ChangeTime`
- `CreateTime`

- `CronDefinition`
- `Description`
- `Language`
- `Name`
- `Valid`

Note: This business object type does not support the multi-sorting feature. Sorting is supported only by a single column at a time.

`Ticket`
Possible column names:

- `Age`
- `ArticleTree`
- `Changed`
- `Created`
- `CreatedBy`
- `CustomerCompanyName`
- `CustomerID`
- `CustomerName`
- `CustomerUserID`
- `EscalationResponseTime`
- `EscalationSolutionTime`
- `EscalationTime`
- `EscalationUpdateTime`
- `Lock`
- `Owner`
- `PendingTime`
- `Priority`
- `Queue`
- `Responsible`
- `Sender`
- `Service`
- `SLA`
- `State`
- `Subject`
- `TicketNumber`
- `Title`

- `Type`
- `Watch`

`TicketArticle`

Possible column names:

- `ArticleNumber`
- `ArticleProperties`
- `Attachment`
- `Channel`
- `CreateTime`
- `Direction`
- `Sender`
- `Status`
- `Subject`

Note: This business object type does not support the multi-sorting feature. Sorting is supported only by a single column at a time.

`TicketAttachment`

Possible column names:

- `ContentType`
- `CreateTime`
- `Direction`
- `Download`
- `Filename`
- `Filesize`
- `Preview`

Note: This business object type does not support the multi-sorting feature. Sorting is supported only by a single column at a time.

`WebNotification`

Possible column names:

- `CreateTime`
- `Name`
- `ObjectReference`
- `ObjectType`
- `Subject`

Filter Names

Filters can be referenced by their names. This section lists the possible filter names for specific business object types. Each filter has a value type displayed next to it that can be used to quickly jump to the reference.

See also:

The list of possible filter value types can be found in the *Filter Value Types* chapter.

Note: The filters marked with the × (multiple) symbol support array values for multiple matches.

`ChatRequest`
The following filter names and values can be used for the *Chat Request* business object lists.

`ChannelIDs`: ***array*** ×
Defines the IDs of the chat channels.

`Create_DateTimeRange`: ***absolute date***
Defines the absolute range of the chat request creation time.

`Create_DateTimeRelative`: ***relative date***
Defines the relative range of the chat request creation time.

`RequesterType`: ***array*** ×
Defines the type of the user that initiated the chat request.

`User`
The user is an agent.

`Customer`
The user is a customer user.

`Public`
The user is anonymous (public user).

`Type`: ***array*** ×
Defines the type of the chat request.

`Invitation`
The invitation to a chat initiated by another agent user.

`Request`
A chat request initiated by a customer user or a public user.

`CustomerCompany`
The following filter names and values can be used for the *Customer* business object lists.

`CustomerCompanyCity`: ***string***
Defines the text when searching for the customer city. Wildcard * can be used. The search is case insensitive.

`CustomerCompanyComment`: ***string***
Defines the text when searching for the customer comment. Wildcard * can be used. The search is case insensitive.

`CustomerCompanyCountry`: ***string***
Defines the text when searching for the customer country. Wildcard * can be used. The search is case insensitive.

`CustomerCompanyName`: *string*
: Defines the text when searching for the customer name. Wildcard * can be used. The search is case insensitive.

`CustomerCompanyStreet`: *string*
: Defines the text when searching for the customer street. Wildcard * can be used. The search is case insensitive.

`CustomerCompanyURL`: *string*
: Defines the text when searching for the customer URL. Wildcard * can be used. The search is case insensitive.

`CustomerCompanyZIP`: *string*
: Defines the text when searching for the customer postcode. Wildcard * can be used. The search is case insensitive.

`CustomerID`: *string*
: Defines the text when searching for the customer ID. Wildcard * can be used. The search is case insensitive.

`ValidID`: *array* ×
: Defines the IDs of the customer validity. By default, these include:

- `1` = valid
- `2` = invalid
- `3` = invalid-temporarily

`CustomerUser`
The following filter names and values can be used for the *Customer User* business object lists.

`CustomerCompany_CustomerCompanyCity`: *string*
: Defines the text when searching for the customer city. Wildcard * can be used. The search is case insensitive.

`CustomerCompany_CustomerCompanyComment`: *string*
: Defines the text when searching for the customer comment. Wildcard * can be used. The search is case insensitive.

`CustomerCompany_CustomerCompanyCountry`: *string*
: Defines the text when searching for the customer country. Wildcard * can be used. The search is case insensitive.

`CustomerCompany_CustomerCompanyName`: *string*
: Defines the text when searching for the customer name. Wildcard * can be used. The search is case insensitive.

`CustomerCompany_CustomerCompanyStreet`: *string*
: Defines the text when searching for the customer street. Wildcard * can be used. The search is case insensitive.

`CustomerCompany_CustomerCompanyURL`: *string*
: Defines the text when searching for the customer URL. Wildcard * can be used. The search is case insensitive.

`CustomerCompany_CustomerCompanyZIP`: *string*
: Defines the text when searching for the customer postcode. Wildcard * can be used. The search is case insensitive.

`CustomerCompany_ValidID`: *array* ×
: Defines the IDs of the customer validity. By default, these include:

- `1` = valid
- `2` = invalid
- `3` = invalid-temporarily

`CustomerID`: *string*
: Defines the text when searching for the related customer ID. Wildcard * can be used. The search is case insensitive.

`UserCity`: *string*
: Defines the text when searching for the customer user city. Wildcard * can be used. The search is case insensitive.

`UserComment`: *string*
: Defines the text when searching for the customer user comment. Wildcard * can be used. The search is case insensitive.

`UserCountry`: *string*
: Defines the text when searching for the customer user country. Wildcard * can be used. The search is case insensitive.

`UserCustomerID`: *string*
: Defines the text when searching for the customer user ID. Wildcard * can be used. The search is case insensitive.

`UserEmail`: *string*
: Defines the text when searching for the customer user email address. Wildcard * can be used. The search is case insensitive.

`UserFax`: *string*
: Defines the text when searching for the customer user fax number. Wildcard * can be used. The search is case insensitive.

`UserFirstname`: *string*
: Defines the text when searching for the customer user first name. Wildcard * can be used. The search is case insensitive.

`UserLastname`: *string*
: Defines the text when searching for the customer user last name. Wildcard * can be used. The search is case insensitive.

`UserLogin`: *string*
: Defines the text when searching for the customer user login name (username). Wildcard * can be used. The search is case insensitive.

`UserMobile`: *string*
: Defines the text when searching for the customer user mobile number. Wildcard * can be used. The search is case insensitive.

`UserPhone`: *string*
: Defines the text when searching for the customer user phone number. Wildcard * can be used. The search is case insensitive.

`UserStreet`: *string*
: Defines the text when searching for the customer user street. Wildcard * can be used. The search is case insensitive.

`UserTitle`: *string*
: Defines the text when searching for the customer user title. Wildcard * can be used. The search is case insensitive.

UserZip: *string*
Defines the text when searching for the customer user postcode. Wildcard * can be used. The search is case insensitive.

ValidID: *array* ×
Defines the IDs of the customer user validity. By default, these include:

- `1` = valid
- `2` = invalid
- `3` = invalid-temporarily

KnowledgeBaseArticle
The following filter names and values can be used for the *Knowledge Base Article* business object lists.

Approved: *boolean*
Defines whether the article has been approved.

CategoryIDs: *array* ×
Defines the IDs of the article categories.

CreatedUserIDs: *array* ×
Defines the IDs of the agents who created the article.

ItemChange_DateTimeRange: *absolute date*
Defines the absolute range of the article change time.

ItemChange_DateTimeRelative: *relative date*
Defines the relative range of the article change time.

ItemCreate_DateTimeRange: *absolute date*
Defines the absolute range of the article create time.

ItemCreate_DateTimeRelative: *relative date*
Defines the relative range of the article create time.

Keyword: *string*
Defines the text when searching for the article keyword. Wildcard * can be used. The search is case insensitive.

LanguageIDs: *array* ×
Defines the IDs of the article language. By default, these include:

- `en`
- `de`

LastChangedUserIDs: *array* ×
Defines the IDs of the agents who changed the article last.

Number: *string*
Defines the text when searching for the article number. Wildcard * can be used.

Rate: *comparison*
Compares the article rating in percentage against the supplied value.

StateIDs: *array* ×
Defines the IDs of the article state. By default, these include:

- `1` = internal (agent)
- `2` = external (customer)

- `3` = public (all)

`Title`: *string*
Defines the text when searching for the article title. Wildcard * can be used. The search is case insensitive.

`ValidIDs`: *array* ×
Defines the IDs of the article validity. By default, these include:

- `1` = valid
- `2` = invalid
- `3` = invalid-temporarily

`Votes`: *comparison*
Compares the number of votes for the article against the supplied value.

`What`: *string*
Defines the text when searching through the article full text. Wildcard * can be used. The search is case insensitive.

`KnowledgeBaseArticleAttachment`
The following filter names and values can be used for the *Knowledge Base Article Attachment* business object lists.

`Create_DateTimeRange`: *absolute date*
Defines the absolute range of the attachment create time.

`Create_DateTimeRelative`: *relative date*
Defines the relative range of the attachment create time.

`Filename`: *string*
Defines the text when searching for the attachment filename. Wildcard * and regular expressions can be used. The search is case insensitive.

`Type`: *string*
Defines the text when searching for the attachment MIME type. Wildcard * and regular expressions can be used. The search is case insensitive.

`Search`
The following filter names and values can be used for the *Search* business object lists.

`Appointment Calendar`: *string*
Defines the calendar in which to search for the appointment. Please set it to a string value formatted as:

`CalendarID#`
Where `#` is the ID of the calendar, i.e. `CalendarID1`.

Warning: This filter is applicable only when the filter `DocumentType` is defined and set to the value `Appointment`. Otherwise, it will be ignored.

`Appointment Schedule`: *string*
Defines the type of the appointment.

`AllDay`
The appointment is an all day appointment (i.e. it does not have the time component).

`TimeFrame`
The appointment is a time frame appointment (i.e. it has the time component).

Warning: This filter is applicable only when the filter `DocumentType` is defined and set to the value `Appointment`. Otherwise, it will be ignored.

`DocumentType`: *string*
Defines the type of document to be searched. By default, these include:

- `Appointment` = Calendar Appointments
- `FAQ` = Knowledge Base Articles
- `Ticket` = Tickets

`FAQ FAQ Categories`: *string*
Defines the knowledge base article category for which you are searching. Please set it to a string value formatted as:

`CategoryID#`
Where `#` is the ID of the category, i.e. `CategoryID1`.

Warning: This filter is applicable only when the filter `DocumentType` is defined and set to the value `FAQ`. Otherwise, it will be ignored.

`FAQ FAQ Languages`: *string*
Defines the knowledge base article language for which you are searching. Please set it to a string value formatted as:

`LanguageID#`
Where `#` is the ID of the language, i.e. `LanguageID1`.

Warning: This filter is applicable only when the filter `DocumentType` is defined and set to the value `FAQ`. Otherwise, it will be ignored.

`FAQ Has attachments`: *string*
Defines the presence of knowledge base article attachments.

`HasAttachments`
The article has at least one attachment.

Warning: This filter is applicable only when the filter `DocumentType` is defined and set to the value `FAQ`. Otherwise, it will be ignored.

`FAQ State`: *string*
Defines the knowledge base article state for which you are searching. Please set it to a string value formatted as:

`StateID#`
Where `#` is the ID of the state, i.e. `StateID1`.

Warning: This filter is applicable only when the filter `DocumentType` is defined and set to the value `FAQ`. Otherwise, it will be ignored.

`Ticket Has attachments`: *string*
Defines the presence of the ticket article attachments.

`HasAttachments`
The ticket article has at least one attachment.

> **Warning:** This filter is applicable only when the filter `DocumentType` is defined and set to the value `Ticket`. Otherwise, it will be ignored.

`Ticket Owner`: *string*
Defines the ID of the agent who is the ticket owner. Please set it to a string value formatted as:

`OwnerID#`
Where `#` is the ID of the agent, i.e. `OwnerID1`.

> **Warning:** This filter is applicable only when the filter `DocumentType` is defined and set to the value `Ticket`. Otherwise, it will be ignored.

`Ticket Priority`: *string*
Defines the ID of the ticket priority. Please set it to a string value formatted as:

`PriorityID#`
Where `#` is the ID of the priority, i.e. `PriorityID1`.

> **Warning:** This filter is applicable only when the filter `DocumentType` is defined and set to the value `Ticket`. Otherwise, it will be ignored.

`Ticket Queue`: *string*
Defines the ID of the ticket queue. Please set it to a string value formatted as:

`QueueID#`
Where `#` is the ID of the queue, i.e. `QueueID1`.

> **Warning:** This filter is applicable only when the filter `DocumentType` is defined and set to the value `Ticket`. Otherwise, it will be ignored.

`Ticket Responsible`: *string*
Defines the ID of the agent who is the ticket responsible. Please set it to a string value formatted as:

`ResponsibleID#`
Where `#` is the ID of the agent, i.e. `ResponsibleID1`.

> **Warning:** This filter is applicable only when the filter `DocumentType` is defined and set to the value `Ticket`. Otherwise, it will be ignored.

`Ticket Sender type`: *string*
Defines the sender of the ticket articles.

SenderTypeAgent
The sender of the ticket article is an agent user.

SenderTypeCustomer
The sender of the ticket article is a customer user.

SenderTypeSystem
The sender of the ticket article is a system user.

> **Warning:** This filter is applicable only when the filter `DocumentType` is defined and set to the value `Ticket`. Otherwise, it will be ignored.

Ticket Service Level Agreement (SLA): *string*
Defines the ID of the ticket SLA. Please set it to a string value formatted as:

SLAID#
Where `#` is the ID of the SLA, i.e. `SLAID1`.

> **Warning:** This filter is applicable only when the filter `DocumentType` is defined and set to the value `Ticket`. Otherwise, it will be ignored.

Ticket Service: *string*
Defines the ID of the ticket service. Please set it to a string value formatted as:

ServiceID#
Where `#` is the ID of the service, i.e. `ServiceID1`.

> **Warning:** This filter is applicable only when the filter `DocumentType` is defined and set to the value `Ticket`. Otherwise, it will be ignored.

Ticket State: *string*
Defines the ID of the ticket state. Please set it to a string value formatted as:

StateID#
Where `#` is the ID of the state, i.e. `StateID1`.

> **Warning:** This filter is applicable only when the filter `DocumentType` is defined and set to the value `Ticket`. Otherwise, it will be ignored.

Ticket Visible to customer: *string*
Defines whether the ticket articles are visible to the customer user.

IsVisibleForCustomer
Ticket article is visible to the customer user.

> **Warning:** This filter is applicable only when the filter `DocumentType` is defined and set to the value `Ticket`. Otherwise, it will be ignored.

Statistic
The following filter names and values can be used for the *Statistic* business object lists.

`StatNumber`: *string*
: Defines the text when searching for the statistic number. Wildcard * can be used.

`StatType`: *array* ×
: Defines the statistic type for which you are searching.

  `static`
  : The statistic is of the static type.

  `dynamic`
  : The statistic is of the dynamic type.

`Title`: *string*
: Defines the text when searching for the statistic name. Wildcard * can be used. The search is case insensitive.

`ObjectName`: *array* ×
: Defines the statistic object name to be searched. By default, these include:

  - `Ticketlist`
  - `TicketAccountedTime`
  - `TicketAccumulation`
  - `TicketSolutionResponseTime`
  - `FAQAccess`
  - `StateAction`

`ObjectType`: *array* ×
: Defines the statistic object type to be searched.

  `DynamicList`
  : The object type is a list.

  `DynamicMatrix`
  : The object type is a matrix.

  `Static`
  : The object type is static.

`Valid`: *array* ×
: Defines the IDs of the statistic validity. By default, these include:

  - `1` = valid
  - `0` = invalid

`StatisticReport`
: The following filter names and values can be used for the *Report* business object lists.

  `Name`: *string*
  : Defines the text when searching for the report name. Wildcard * can be used. The search is case insensitive.

  `Description`: *string*
  : Defines the text when searching for the report description. Wildcard * can be used. The search is case insensitive.

  `LanguageID`: *string*
  : Defines the report language code, i.e. `de`.

`ValidID`: *array* ×
Defines the IDs of the report validity. By default, these include:

- `1` = valid
- `2` = invalid
- `3` = invalid-temporarily

`Ticket`
The following filter names and values can be used for the *Ticket* business object lists.

`AgentCreator`: *boolean*
Defines whether the ticket was created by the current agent.

`AgentInvolved`: *boolean*
Defines whether the current agent is involved in the ticket.

`AgentOwner`: *boolean*
Defines whether the current agent is the ticket owner while the ticket is locked.

`AgentQueues`: *boolean*
Defines whether the ticket is in one of the subscribed queues of the current agent. Agents can subscribe to queues via their *My Queues* personal preference.

`AgentResponsible`: *boolean*
Defines whether the current agent is the ticket responsible.

`AgentServices`: *boolean*
Defines whether the ticket has one of the subscribed services of the current agent. Agents can subscribe to services via their *My Services* personal preference.

`AgentTicketFlagSeen`: *string*
Defines whether the current agent has read the ticket.

`Unread`
All articles are unread by the current agent.

`Read`
All articles are read by the current agent.

`AgentWatcher`: *boolean*
Defines whether the current agent is watching the ticket.

`ArticleCreate_DateTimeRange`: *absolute date*
Defines the absolute range of the ticket article creation time.

`ArticleCreate_DateTimeRelative`: *relative date*
Defines the relative range of the ticket article creation time.

`Chat_ChatterName`: *string*
Defines the text when searching for the chat article participants. Wildcard * can be used. The search is case insensitive.

`Chat_MessageText`: *string*
Defines the text when searching for the chat article message. Wildcard * can be used. The search is case insensitive.

`CreatedQueueIDs`: *array* ×
Defines the IDs of the queues where the ticket was originally created.

`CreatedUserIDs`: *array* ×
Defines the IDs of the agents who created the ticket.

`CustomerID`: *string*
Defines the text when searching for the ticket customer ID field. Use this filter for a complex search. Wildcard * can be used. The search is case insensitive.

`CustomerIDRaw`: *string*
Defines the text when searching for the ticket customer ID field. Use this filter for an exact match. The search is case insensitive.

`CustomerUserID`: *string*
Defines whether the ticket is accessible to a specific customer user in the external interface. Supports only exact match. The search is case insensitive.

`CustomerUserLogin`: *string*
Defines the text when searching for the ticket customer user ID field. Use this filter for the complex search. Wildcard * can be used. The search is case insensitive.

`CustomerUserLoginRaw`: *string*
Defines the text when searching for the ticket customer user ID field. Use this filter for an exact match. The search is case insensitive.

`Fulltext`: *string*
Defines the text when searching for the ticket article full text. Wildcard * can be used. The search is case insensitive.

`LockIDs`: *array* ×
Defines the IDs of the ticket lock types. By default, these include:

- `1` = unlock
- `2` = lock
- `3` = tmp_lock

`MIMEBase_AttachmentName`: *string*
Defines the text when searching for the article attachment names. Wildcard * can be used. The search is case insensitive.

`MIMEBase_Bcc`: *string*
Defines the text when searching for the article blind carbon copy field. Wildcard * can be used. The search is case insensitive.

`MIMEBase_Body`: *string*
Defines the text when searching for the article body. Wildcard * can be used. The search is case insensitive.

`MIMEBase_Cc`: *string*
Defines the text when searching for the article carbon copy field. Wildcard * can be used. The search is case insensitive.

`MIMEBase_From`: *string*
Defines the text when searching for the article sender field. Wildcard * can be used. The search is case insensitive.

`MIMEBase_Subject`: *string*
Defines the text when searching for the article subject field. Wildcard * can be used. The search is case insensitive.

`MIMEBase_To`: *string*
Defines the text when searching for the article recipient field. Wildcard * can be used. The search is case insensitive.

`OwnerIDs`: *array* ×
Defines the IDs of the agents who are the ticket owners.

`PriorityIDs`: *array* ×
Defines the IDs of the ticket priorities.

`QueueIDs`: *array* ×
Defines the IDs of the ticket queues.

`ResponsibleIDs`: *array* ×
Defines the IDs of the agents who are the ticket responsibles.

`SearchInArchive`: *string*
Defines how the search behaves related to the ticket archive status.

`ArchivedTickets`
Search in archived tickets only.

`NotArchivedTickets`
Search in unarchived tickets only.

`AllTickets`
Search in all tickets.

`ServiceIDs`: *array* ×
Defines the IDs of the ticket services.

`SLAIDs`: *array* ×
Defines the IDs of the ticket SLAs.

`SMS_PhoneNumbers`: *string*
Defines the text when searching for the SMS article recipient phone numbers. Wildcard * can be used. The search is case insensitive.

`SMS_Text`: *string*
Defines the text when searching for the SMS article body. Wildcard * can be used. The search is case insensitive.

`SMS_TransactionNumbers`: *string*
Defines the text when searching for the SMS article transaction numbers. Wildcard * can be used. The search is case insensitive.

`StateIDs`: *array* ×
Defines the IDs of the ticket states.

`StateType`: *string*
Defines the ticket state category.

`Open`
Search in open tickets only.

`Closed`
Search in closed tickets only.

Warning: Despite its name, this filter does not apply to the actual *ticket state types*. `Open` and `Closed` are not valid state types. They are group constructs included for compatibility reasons. If a ticket is in a state which has any of the state types defined in the Ticket::ViewableStateType setting, it will be considered as open. Otherwise, it is considered as closed.

`StateTypeIDs`: *array* ×
: Defines the IDs of the ticket state types.

`TicketChange_DateTimeRange`: *absolute date*
: Defines the absolute range of the ticket change time.

`TicketChange_DateTimeRelative`: *relative date*
: Defines the relative range of the ticket change time.

`TicketClose_DateTimeRange`: *absolute date*
: Defines the absolute range of the ticket close time.

`TicketClose_DateTimeRelative`: *relative date*
: Defines the relative range of the ticket close time.

`TicketCreate_DateTimeRange`: *absolute date*
: Defines the absolute range of the ticket create time.

`TicketCreate_DateTimeRelative`: *relative date*
: Defines the relative range of the ticket create time.

`TicketEscalation_DateTimeRange`: *absolute date*
: Defines the absolute range of the ticket escalation time.

`TicketEscalation_DateTimeRelative`: *relative date*
: Defines the relative range of the ticket escalation time.

`TicketLastChange_DateTimeRange`: *absolute date*
: Defines the absolute range of the ticket last change time.

`TicketLastChange_DateTimeRelative`: *relative date*
: Defines the relative range of the ticket last change time.

`TicketNumber`: *string*
: Defines the text when searching for the ticket number. Wildcard * can be used.

`TicketPending_DateTimeRange`: *absolute date*
: Defines the absolute range of the ticket pending time.

`TicketPending_DateTimeRelative`: *relative date*
: Defines the relative range of the ticket pending time.

`Title`: *string*
: Defines the text when searching for the ticket title. Wildcard * can be used. The search is case insensitive.

`TypeIDs`: *array* ×
: Defines the text when searching for the ticket type.

`WatchUserIDs`: *array* ×
: Defines the IDs of the agents who are watching the ticket.

`TicketArticle`

The following filter names and values can be used for the *Ticket Article* business object lists.

`CommunicationChannelID`: *array*

`IsVisibleForCustomer`: *string*

`SenderTypeID`: *array*

`TicketAttachment`

The following filter names and values can be used for the *Ticket Attachment* business object lists.

`Article`: *string*

`Create_DateTimeRange`: *absolute date*

`Create_DateTimeRelative`: *relative date*

`Direction`: *array* ×

`Filename`: *string*

`Type`: *string*

`WebNotification`
The following filter names and values can be used for the *Web Notification* business object lists.

`Name`: *array* ×

`Subject`: *string*

`ObjectTypes`: *array* ×

`Seen`: *array* ×

`ObjectReferences`: *array* ×

Filter Value Types

Every filter supports exactly one kind of a value type. Depending on this type, the following structures should be used to define it in the YAML configuration.

Boolean
A simple value that is considered either true or false.

```
<FilterName1>:
  Value: 1
<FilterName2>:
  Value: 0
```

1
The filter is active (turned on).

0
The filter is inactive (turned off).

String
A text value, normally used when searching. Check specific filters for more details.

```
<FilterName1>:
  Value: simple
<FilterName2>:
  Value: 'String with spaces or special characters like @$#!+*'
```

Array
An array value normally used to search for multiple values at the same time.

```
<FilterName1>:
  Value:
  - 1
  - 2
  - 3
<FilterName2>:
  Value:
```

(continues on next page)

(continued from previous page)

```
  - 4
  - 5
```

Absolute date

The time range value between two absolute dates.

```
<FilterName1>:
  Value:
    Start: '2020-01-01 00:00:00'
    End: '2020-02-01 00:00:00'
<FilterName2>:
  Value:
    Start: '2020-02-01 00:00:00'
    End: '2020-03-01 00:00:00'
```

`Start`

Absolute timestamp in ISO format for the start of the range (i.e. `2020-01-01 00:00:00`). The value must be supplied in the current system time zone (configurable via the OTRSTimeZone system setting).

`End`

Absolute timestamp in ISO format for the end of the range (i.e. `2020-02-01 00:00:00`). The value must be supplied in the current system time zone (configurable via the OTRSTimeZone system setting).

Relative date

The time range value relative to the current time.

```
<FilterName1>:
  Value:
    Start: Last
    Point: 1
    Format: month
<FilterName2>:
  Value:
    Start: Before
    Point: 3
    Format: day
```

`Start`

The type of the relative range.

Possible values for the `Start` key:

- `Last` = within the last…
- `Before` = more than …ago

`Point`

Integer value of the time unit (i.e. `3`).

`Format`

Time unit of the relative range.

Possible values for the `Format` key:

```
minute
hour
```

(continues on next page)

(continued from previous page)

```
day
week
month
year
```

Comparison

The comparison with the supplied value.

```
<FilterName1>:
  Value:
    Type: Equals
    Value: 25
<FilterName2>:
  Value:
    Start: GreaterThan
    Value: 50
```

`Type`

The type of the comparison.

Possible values for the `Type` key:

- `Equals` = equals …
- `GreaterThan` = greater than …
- `GreaterThanEquals` = greater than or equals …
- `SmallerThan` = smaller than …
- `SmallerThanEquals` = smaller than or equals …

`Value`

Integer value to compare against (i.e. `25`).

Form Fields

The form fields can be defined via their `Name` key. This section lists the values of the `Name` key for the specific settings.

See also:

Form fields can be made mandatory by setting their *Required* key to `1`.

Possible field names for ticket actions:

`Forms###AgentFrontend::Ticket::Action::Close`

Configurable form for the *Close Ticket* action.

Link to the reference of the system configuration:

- Forms###AgentFrontend::Ticket::Action::Close

Possible values for the `Name` key of the form field:

```
AccountedTime
AddMessage
Attachments
Body
```

(continues on next page)

(continued from previous page)

```
CustomerID
CustomerUserID
DynamicField
HistoryComment
HistoryType
InformAgent
InvolvedAgent
IsVisibleForCustomer
MarkAsImportant
Messages
OwnerID
PriorityID
QueueID
RelevantKnowledge
ResponsibleID
SenderType
ServiceID
SLAID
StandardTemplateID
StateID
Subject
Title
TypeID
```

`Forms###AgentFrontend::Ticket::Action::Customer`

Configurable form for the *Change Customer* action.

Link to the reference of the system configuration:

- Forms###AgentFrontend::Ticket::Action::Customer

Possible values for the `Name` key of the form field:

```
CustomerUserID
CustomerID
```

`Forms###AgentFrontend::Ticket::Action::EmailOutbound`

Configurable form for the *Send Email Outbound* action.

Link to the reference of the system configuration:

- Forms###AgentFrontend::Ticket::Action::EmailOutbound

Possible values for the `Name` key of the form field:

```
AccountedTime
Attachments
Bcc
Body
Cc
CustomerID
CustomerUserID
DynamicField
EmailSecurity
From
HistoryComment
HistoryType
InformAgent
```

(continues on next page)

(continued from previous page)

```
InvolvedAgent
IsVisibleForCustomer
MarkAsImportant
Messages
OwnerID
PriorityID
QueueID
RelevantKnowledge
ResponsibleID
SenderType
ServiceID
Signature
SLAID
StandardTemplateID
StateID
Subject
Title
To
TypeID
```

`Forms###AgentFrontend::Ticket::Action::FreeText`

Configurable form for the *Change Free Fields* action.

Link to the reference of the system configuration:

- Forms###AgentFrontend::Ticket::Action::FreeText

Possible values for the `Name` key of the form field:

```
AccountedTime
AddMessage
Attachments
Body
CustomerID
CustomerUserID
DynamicField
HistoryComment
HistoryType
InformAgent
InvolvedAgent
IsVisibleForCustomer
MarkAsImportant
Messages
OwnerID
PriorityID
QueueID
RelevantKnowledge
ResponsibleID
SenderType
ServiceID
SLAID
StandardTemplateID
StateID
Subject
Title
TypeID
```

`Forms###AgentFrontend::Ticket::Action::Merge`

Configurable form for the *Merge Ticket* action.

Link to the reference of the system configuration:

- Forms###AgentFrontend::Ticket::Action::Merge

Possible values for the `Name` key of the form field:

```
AddMessage
Body
DynamicField
From
HistoryComment
HistoryType
IsVisibleForCustomer
MarkAsImportant
Messages
RelevantKnowledge
SenderType
Subject
To
```

`Forms###AgentFrontend::Ticket::Action::Move`

Configurable form for the *Move Ticket* action.

Link to the reference of the system configuration:

- Forms###AgentFrontend::Ticket::Action::Move

Possible values for the `Name` key of the form field:

```
AccountedTime
AddMessage
Attachments
Body
CustomerID
CustomerUserID
DynamicField
HistoryComment
HistoryType
InformAgent
InvolvedAgent
IsVisibleForCustomer
MarkAsImportant
Messages
OwnerID
PriorityID
QueueID
RelevantKnowledge
ResponsibleID
SenderType
ServiceID
SLAID
StandardTemplateID
StateID
Subject
Title
TypeID
```

`Forms###AgentFrontend::Ticket::Action::Note`

Configurable form for the *Add Note* action.

Link to the reference of the system configuration:

- Forms###AgentFrontend::Ticket::Action::Note

Possible values for the `Name` key of the form field:

```
AccountedTime
Attachments
Body
CustomerID
CustomerUserID
DynamicField
HistoryComment
HistoryType
InformAgent
InvolvedAgent
IsVisibleForCustomer
MarkAsImportant
Messages
OwnerID
PriorityID
QueueID
RelevantKnowledge
ResponsibleID
SenderType
ServiceID
SLAID
StandardTemplateID
StateID
Subject
Title
TypeID
```

`Forms###AgentFrontend::Ticket::Action::Owner`

Configurable form for the *Change Owner* action.

Link to the reference of the system configuration:

- Forms###AgentFrontend::Ticket::Action::Owner

Possible values for the `Name` key of the form field:

```
AccountedTime
AddMessage
Attachments
Body
CustomerID
CustomerUserID
DynamicField
HistoryComment
HistoryType
InformAgent
InvolvedAgent
IsVisibleForCustomer
MarkAsImportant
Messages
OwnerID
PriorityID
QueueID
RelevantKnowledge
```

(continues on next page)

(continued from previous page)

```
ResponsibleID
SenderType
ServiceID
SLAID
StandardTemplateID
StateID
Subject
Title
TypeID
```

`Forms###AgentFrontend::Ticket::Action::Pending`

Configurable form for the *Set Pending Time* action.

Link to the reference of the system configuration:

- Forms###AgentFrontend::Ticket::Action::Pending

Possible values for the `Name` key of the form field:

```
AccountedTime
AddMessage
Attachments
Body
CustomerID
CustomerUserID
DynamicField
HistoryComment
HistoryType
InformAgent
InvolvedAgent
IsVisibleForCustomer
MarkAsImportant
Messages
OwnerID
PriorityID
QueueID
RelevantKnowledge
ResponsibleID
SenderType
ServiceID
SLAID
StandardTemplateID
StateID
Subject
Title
TypeID
```

`Forms###AgentFrontend::Ticket::Action::PhoneCallInbound`

Configurable form for the *Add Phone Call Inbound* action.

Link to the reference of the system configuration:

- Forms###AgentFrontend::Ticket::Action::PhoneCallInbound

Possible values for the `Name` key of the form field:

```
AccountedTime
Attachments
```

(continues on next page)

(continued from previous page)

```
Body
CustomerID
CustomerUserID
DynamicField
HistoryComment
HistoryType
InformAgent
InvolvedAgent
IsVisibleForCustomer
MarkAsImportant
Messages
OwnerID
PriorityID
QueueID
RelevantKnowledge
ResponsibleID
SenderType
ServiceID
SLAID
StandardTemplateID
StateID
Subject
Title
TypeID
```

`Forms###AgentFrontend::Ticket::Action::PhoneCallOutbound`

Configurable form for the *Add Phone Call Outbound* action.

Link to the reference of the system configuration:

- Forms###AgentFrontend::Ticket::Action::PhoneCallOutbound

Possible values for the `Name` key of the form field:

```
AccountedTime
Attachments
Body
CustomerID
CustomerUserID
DynamicField
HistoryComment
HistoryType
InformAgent
InvolvedAgent
IsVisibleForCustomer
MarkAsImportant
Messages
OwnerID
PriorityID
QueueID
RelevantKnowledge
ResponsibleID
SenderType
ServiceID
SLAID
StandardTemplateID
StateID
```

(continues on next page)

(continued from previous page)

```
Subject
Title
TypeID
```

`Forms###AgentFrontend::Ticket::Action::Priority`

Configurable form for the *Change Priority* action.

Link to the reference of the system configuration:

- Forms###AgentFrontend::Ticket::Action::Priority

Possible values for the `Name` key of the form field:

```
AccountedTime
AddMessage
Attachments
Body
CustomerID
CustomerUserID
DynamicField
HistoryComment
HistoryType
InformAgent
InvolvedAgent
IsVisibleForCustomer
MarkAsImportant
Messages
OwnerID
PriorityID
QueueID
RelevantKnowledge
ResponsibleID
SenderType
ServiceID
SLAID
StandardTemplateID
StateID
Subject
Title
TypeID
```

`Forms###AgentFrontend::Ticket::Action::Responsible`

Configurable form for the *Change Responsible* action.

Link to the reference of the system configuration:

- Forms###AgentFrontend::Ticket::Action::Responsible

Possible values for the `Name` key of the form field:

```
AccountedTime
AddMessage
Attachments
Body
CustomerID
CustomerUserID
DynamicField
HistoryComment
```

(continues on next page)

(continued from previous page)

```
HistoryType
InformAgent
InvolvedAgent
IsVisibleForCustomer
MarkAsImportant
Messages
OwnerID
PriorityID
QueueID
RelevantKnowledge
ResponsibleID
SenderType
ServiceID
SLAID
StandardTemplateID
StateID
Subject
Title
TypeID
```

Forms###AgentFrontend::Ticket::Action::SmsOutbound

Configurable form for the *Send SMS Outbound* action.

Link to the reference of the system configuration:

- Forms###AgentFrontend::Ticket::Action::SmsOutbound

Possible values for the `Name` key of the form field:

```
AccountedTime
Body
CustomerID
CustomerUserID
DynamicField
FlashMessage
HistoryComment
HistoryType
InformAgent
InvolvedAgent
IsVisibleForCustomer
MarkAsImportant
Messages
OwnerID
PriorityID
QueueID
RelevantKnowledge
ResponsibleID
SenderType
ServiceID
Signature
SLAID
StandardTemplateID
StateID
Title
To
TypeID
```

Possible field names for article actions:

`Forms###AgentFrontend::TicketArticle::Action::Forward`

Configurable form for the *Forward via Email* action.

Link to the reference of the system configuration:

- Forms###AgentFrontend::TicketArticle::Action::Forward

Possible values for the `Name` key of the form field:

```
AccountedTime
Attachments
Bcc
Body
Cc
DynamicField
EmailSecurity
From
HistoryComment
HistoryType
IsVisibleForCustomer
MarkAsImportant
Messages
RelevantKnowledge
SenderType
StandardTemplateID
StateID
Subject
To
```

`Forms###AgentFrontend::TicketArticle::Action::Redirect`

Configurable form for the *Redirect via Email* action.

Link to the reference of the system configuration:

- Forms###AgentFrontend::TicketArticle::Action::Redirect

Possible values for the `Name` key of the form field:

```
AddMessage
Body
From
HistoryType
Messages
RedirectTo
StateID
Subject
To
```

`Forms###AgentFrontend::TicketArticle::Action::Reply`

Configurable form for the *Reply via Email* action.

Link to the reference of the system configuration:

- Forms###AgentFrontend::TicketArticle::Action::Reply

Possible values for the `Name` key of the form field:

```
AccountedTime
Attachments
Bcc
Body
```

(continues on next page)

(continued from previous page)

```
Cc
DynamicField
EmailSecurity
From
HistoryComment
HistoryType
IsVisibleForCustomer
MarkAsImportant
Messages
RelevantKnowledge
SenderType
StandardTemplateID
StateID
Subject
To
```

Forms###AgentFrontend::TicketArticle::Action::ReplyAll

Configurable form for the *Reply to All via Email* action.

Link to the reference of the system configuration:

- Forms###AgentFrontend::TicketArticle::Action::ReplyAll

Possible values for the `Name` key of the form field:

```
AccountedTime
Attachments
Bcc
Body
Cc
DynamicField
EmailSecurity
From
HistoryComment
HistoryType
IsVisibleForCustomer
MarkAsImportant
Messages
RelevantKnowledge
SenderType
StandardTemplateID
StateID
Subject
To
```

Forms###AgentFrontend::TicketArticle::Action::ReplyToNote

Configurable form for the *Reply via Note* action.

Link to the reference of the system configuration:

- Forms###AgentFrontend::TicketArticle::Action::ReplyToNote

Possible values for the `Name` key of the form field:

```
AccountedTime
Attachments
AutoInvolvedAgents
Body
```

(continues on next page)

(continued from previous page)

```
CustomerID
CustomerUserID
DynamicField
HistoryComment
HistoryType
InformAgent
InvolvedAgent
IsVisibleForCustomer
MarkAsImportant
Messages
OwnerID
PriorityID
QueueID
RelevantKnowledge
ResponsibleID
SenderType
ServiceID
SLAID
StandardTemplateID
StateID
Subject
Title
TypeID
```

`Forms###AgentFrontend::TicketArticle::Action::ReplyViaSms`

Configurable form for the *Reply via SMS* action.

Link to the reference of the system configuration:

- Forms###AgentFrontend::TicketArticle::Action::ReplyViaSms

Possible values for the `Name` key of the form field:

```
AccountedTime
Body
CustomerID
CustomerUserID
DynamicField
FlashMessage
HistoryComment
HistoryType
InformAgent
InvolvedAgent
IsVisibleForCustomer
MarkAsImportant
Messages
OwnerID
PriorityID
QueueID
RelevantKnowledge
ResponsibleID
SenderType
ServiceID
Signature
SLAID
StandardTemplateID
StateID
```

(continues on next page)

(continued from previous page)

```
Title
To
TypeID
```

`Forms###AgentFrontend::TicketArticle::Action::Split`

Configurable form for the *Split Article* action.

Link to the reference of the system configuration:

- Forms###AgentFrontend::TicketArticle::Action::Split

Possible values for the `Name` key of the form field:

```
LinkAs
Messages
ProcessID
Target
```

Possible field names for the ticket create properties forms:

`Forms###AgentFrontend::TicketCreate::Email::CreateProperties`

Configurable form for the *Properties* widget of the *New Email Ticket* screen.

Link to the reference of the system configuration:

- Forms###AgentFrontend::TicketCreate::Email::CreateProperties

Possible values for the `Name` key of the form field:

```
AccountedTime
Attachments
Bcc
Body
Cc
CustomerID
CustomerUserID
DynamicField
EmailSecurity
HistoryComment
HistoryType
IsVisibleForCustomer
LinkTicketID
LinkType
OwnerID
PriorityID
QueueID
RelevantKnowledge
ResponsibleID
SenderType
ServiceID
Signature
SLAID
StandardTemplateID
StateID
Subject
To
TypeID
```

`Forms###AgentFrontend::TicketCreate::Phone::CreateProperties`

Configurable form for the *Properties* widget of the *New Phone Ticket* screen.

Link to the reference of the system configuration:

- Forms###AgentFrontend::TicketCreate::Phone::CreateProperties

Possible values for the `Name` key of the form field:

```
AccountedTime
Attachments
Body
CustomerID
CustomerUserID
DynamicField
From
HistoryComment
HistoryType
IsVisibleForCustomer
LinkTicketID
LinkType
OwnerID
PendingDate
PriorityID
QueueID
RelevantKnowledge
ResponsibleID
SenderType
ServiceID
SLAID
StandardTemplateID
StateID
Subject
To
TypeID
```

`Forms###AgentFrontend::TicketCreate::SMS::CreateProperties`

Configurable form for the *Properties* widget of the *New SMS Ticket* screen.

Link to the reference of the system configuration:

- Forms###AgentFrontend::TicketCreate::SMS::CreateProperties

Possible values for the `Name` key of the form field:

```
AccountedTime
Body
CustomerID
CustomerUserID
DynamicField
FlashMessage
HistoryComment
HistoryType
IsVisibleForCustomer
OwnerID
PendingDate
PriorityID
QueueID
RelevantKnowledge
ResponsibleID
Sender
SenderType
ServiceID
```

(continues on next page)

(continued from previous page)

```
SLAID
StandardTemplateID
StateID
Subject
To
TypeID
```

Possible field names for knowledge base article actions:

Forms###AgentFrontend::KnowledgeBaseArticleCreate::Properties

Configurable form for the *Properties* widget of the *Create Knowledge Base Article* screen.

Link to the reference of the system configuration:

- Forms###AgentFrontend::KnowledgeBaseArticleCreate::Properties

Possible values for the `Name` key of the form field:

```
Approved
Attachments
Category
DynamicFields
Field1
Field2
Field3
Field4
Field5
Field6
Keywords
Language
State
Title
Valid
```

Forms###AgentFrontend::KnowledgeBaseArticleUpdate::Properties

Configurable form for the *Properties* widget of the *Edit Knowledge Base Article* screen.

Link to the reference of the system configuration:

- Forms###AgentFrontend::KnowledgeBaseArticleUpdate::Properties

Possible values for the `Name` key of the form field:

```
Approved
Attachments
Category
DynamicFields
Field1
Field2
Field3
Field4
Field5
Field6
Keywords
Language
State
Title
Valid
```

Possible field names for customer actions:

`Forms###AgentFrontend::CustomerCompanyCreate::Properties`

Configurable form for the *Properties* widget of the *Create Customer* screen.

Link to the reference of the system configuration:

- Forms###AgentFrontend::CustomerCompanyCreate::Properties

Possible values for the `Name` key depend on the available fields in the `CustomerCompany###Map` configuration array. For the default database back end, these include:

```
CustomerCompanyCity
CustomerCompanyComment
CustomerCompanyCountry
CustomerCompanyName
CustomerCompanyStreet
CustomerCompanyURL
CustomerCompanyZIP
CustomerID
ValidID
```

Additional column names which are always available:

```
DataSource
```

Warning: In case *Multiple Customer User Back Ends* are configured, all possible fields must be present in the form configuration in order to be able to modify them. Both configurations must be kept in sync, otherwise it will not be possible to modify the customer records.

`Forms###AgentFrontend::CustomerCompanyUpdate::Properties`

Configurable form for the *Properties* widget of the *Edit Customer* screen.

Link to the reference of the system configuration:

- Forms###AgentFrontend::CustomerCompanyUpdate::Properties

Possible values for the `Name` key depend on the available fields in the `CustomerCompany###Map` configuration array. For the default database back end, these include:

```
CustomerCompanyCity
CustomerCompanyComment
CustomerCompanyCountry
CustomerCompanyName
CustomerCompanyStreet
CustomerCompanyURL
CustomerCompanyZIP
CustomerID
ValidID
```

Warning: In case *Multiple Customer User Back Ends* are configured, all possible fields must be present in the form configuration in order to be able to modify them. Both configurations must be kept in sync, otherwise it will not be possible to modify the customer records.

Possible field names for customer user actions:

`Forms###AgentFrontend::CustomerUserCreate::Properties`

Configurable form for the *Properties* widget of the *Create Customer User* screen.

Link to the reference of the system configuration:

- Forms###AgentFrontend::CustomerUserCreate::Properties

Possible values for the `Name` key depend on the *Customer User Back Ends* configuration. For the default database back end, these include:

```
UserCity
UserComment
UserCountry
UserCustomerID
UserEmail
UserFax
UserFirstname
UserLastname
UserLogin
UserMobile
UserPassword
UserPhone
UserStreet
UserTitle
UserZip
ValidID
```

For most customer user preference modules registered under `CustomerPersonalPreference###*` namespace, it is also possible to show dedicated form fields. By default, these include:

```
Preference_Language
Preference_LoginForbidden
Preference_PGP
Preference_SMIME
Preference_TimeZone
Preference_TwoFactor
```

Additional column names which are always available:

```
DataSource
```

Warning: In case *Multiple Customer User Back Ends* are configured, all possible fields must be present in the form configuration in order to be able to modify them. Both configurations must be kept in sync, otherwise it will not be possible to modify the customer user records.

`Forms###AgentFrontend::CustomerUserUpdate::Properties`

Configurable form for the *Properties* widget of the *Edit Customer User* screen.

Link to the reference of the system configuration:

- Forms###AgentFrontend::CustomerUserUpdate::Properties

Possible values for the `Name` key depend on the *Customer User Back Ends* configuration. For the default database back end, these include:

```
UserCity
UserComment
UserCountry
UserCustomerID
UserEmail
```

(continues on next page)

(continued from previous page)

```
UserFax
UserFirstname
UserLastname
UserLogin
UserMobile
UserPassword
UserPhone
UserStreet
UserTitle
UserZip
ValidID
```

For most customer user preference modules registered under `CustomerPersonalPreference###*` namespace, it is also possible to show dedicated form fields. By default, these include:

```
Preference_Language
Preference_LoginForbidden
Preference_PGP
Preference_SMIME
Preference_TimeZone
Preference_TwoFactor
```

Warning: In case *Multiple Customer User Back Ends* are configured, all possible fields must be present in the form configuration in order to be able to modify them. Both configurations must be kept in sync, otherwise it will not be possible to modify the customer user records.

Possible field names for calendar appointment actions:

Forms###AgentFrontend::Calendar::AppointmentCreate::Properties
Configurable form for the *Add Appointment* action.

Link to the reference of the system configuration:

- Forms###AgentFrontend::Calendar::AppointmentCreate::Properties

Possible values for the `Name` key of the form field:

```
AllDay
CalendarID
Description
EndTime
Location
Notification
Recurrence
ResourceID
StartTime
TeamID
TicketPlugin
Title
```

Forms###AgentFrontend::Calendar::AppointmentUpdate::Properties
Configurable form for the *Edit Appointment* action.

Link to the reference of the system configuration:

- Forms###AgentFrontend::Calendar::AppointmentUpdate::Properties

Possible values for the `Name` key of the form field:

```
AllDay
CalendarID
Description
EndTime
Location
Notification
Recurrence
ResourceID
StartTime
TeamID
TicketPlugin
Title
```

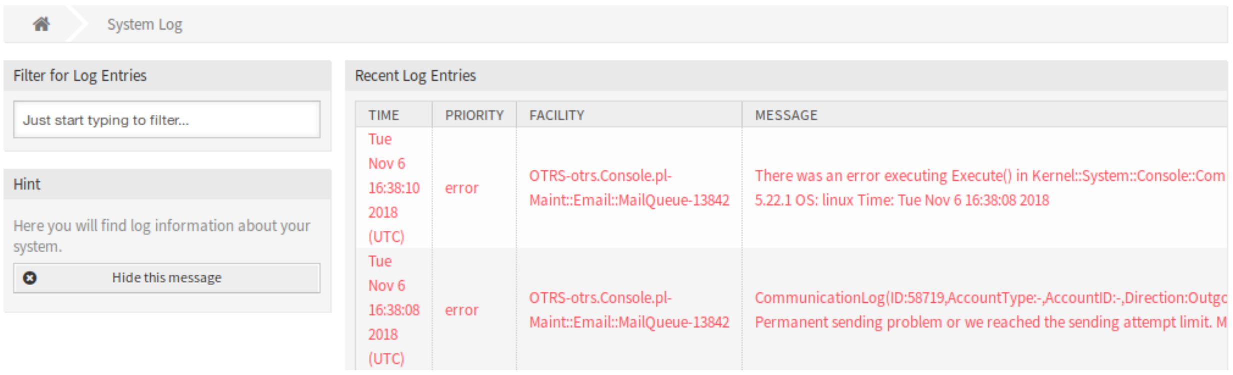
10.12 System Log

Professional systems log their activities in one or more log files to help administrators when troubleshooting or to get an overview of what is going on in their system on a detailed level.

These logs are usually not available to application administrators without a certain level of permissions, and skills, on the operating system.

OTRS allows application administrators to access the system log comfortably by using the graphical interface without the need to have access to the server's command shell. The administrator can decide which level of logging is needed, to make sure that the log files are not unnecessarily filled.

Use this screen to view log entries of OTRS. The log overview screen is available in the *System Log* module of the *Administration* group.


System Log
Filter for Log Entries
Just start typing to filter...
Hint
Here you will find log information about your system.
Hide this message
Recent Log Entries

TIME	PRIORITY	FACILITY	MESSAGE
Tue Nov 6 16:38:10 2018 (UTC)	error	OTRS-otrs.Console.pl-Maint::Email::MailQueue-13842	There was an error executing Execute() in Kernel::System::Console::Com 5.22.1 OS: linux Time: Tue Nov 6 16:38:08 2018
Tue Nov 6 16:38:08 2018 (UTC)	error	OTRS-otrs.Console.pl-Maint::Email::MailQueue-13842	CommunicationLog(ID:58719,AccountType:-,AccountID:-,Direction:Outgc Permanent sending problem or we reached the sending attempt limit. M

Fig. 73: System Log Screen

Each line in the log contains a timestamp, the log priority, the system component and the log entry itself.

Note: If several log entries are displayed in the log, use the filter box to find a particular log entry by just typing the name to filter.

To adjust the logging settings of the system:

1. Go to the *System Configuration* screen.

2. Navigate to *Core* → *Log* in the navigation tree.
3. Review the settings.

10.13 Translations

Agents and customer users can work more efficiently if the graphical user interface of the software speaks in their native language. International companies can also benefit with translation support and let the users choose which language they would like to use in the graphical user interface.

OTRS comes with translations of the built-in resources, but the resources created or changed after deployment has no translation entries. Administrators have to provide translations for all resources which have been added during the customization phase of the software.

There are two kind of usage of the custom translations:

1. To add translation to system resources created or changed after the installation.
2. To customize the existing translation.

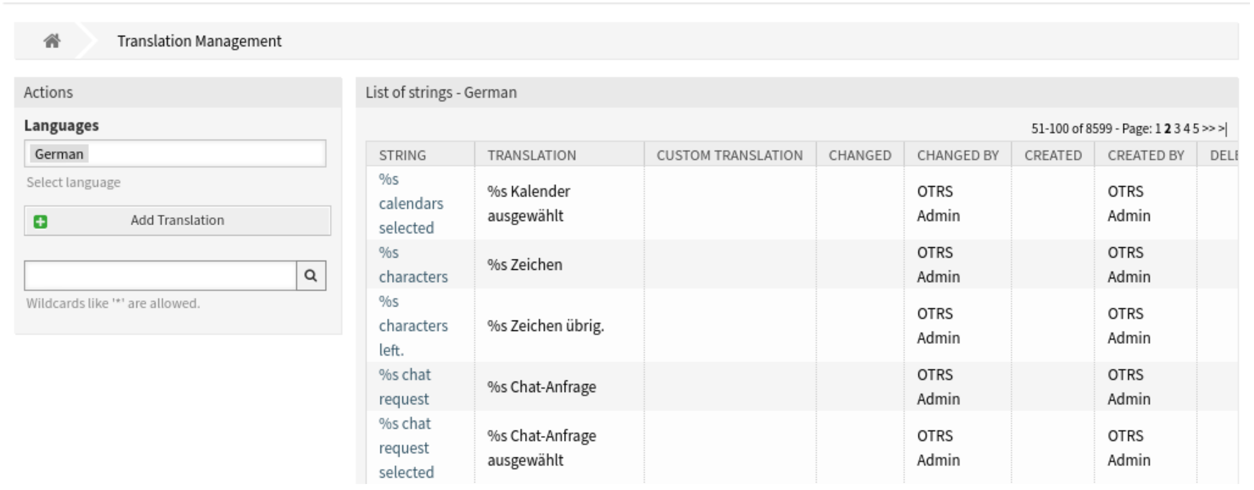
During the configuration phase of the system the created or modified resources can contain new translatable strings like names, descriptions or any other attributes of the following resources:

- *Calendars*
- *FAQ Category*
- *System Configuration*
- *Dynamic Fields*
- *Process Management*
- *Priorities*
- *Services*
- *Service Level Agreements*
- *States*
- *Types*
- *Customers*
- *Customer Users*

It is recommended to always use English texts for the resources above and translate them using custom translations even if the system is designed to be used in a specific language.

When the resources have been added to the system, you have to collect the translatable strings manually and you have to add them to the list of custom translations.

Use this screen to manage translations in the system. The translation management screen is available in the *Translations* module of the *Administration* group.



STRING	TRANSLATION	CUSTOM TRANSLATION	CHANGED	CHANGED BY	CREATED	CREATED BY	DELE
%s calendars selected	%s Kalender ausgewählt			OTRS Admin		OTRS Admin	
%s characters	%s Zeichen			OTRS Admin		OTRS Admin	
%s characters left.	%s Zeichen übrig.			OTRS Admin		OTRS Admin	
%s chat request	%s Chat-Anfrage			OTRS Admin		OTRS Admin	
%s chat request selected	%s Chat-Anfrage ausgewählt			OTRS Admin		OTRS Admin	

Fig. 74: Translation Management Screen

10.13.1 Manage Translations

To add a new translation:

1. Select the target language in the left sidebar.
2. Click on the *Add Translation* button.
3. Enter the English text and its custom translation.
4. Click on the *Save* button.

To edit a translation:

1. Click on a translation in the list of translations.
2. Enter the custom translation.
3. Click on the *Save* or *Save and finish* button.

You can use the *Filter only customized translations* option to see only the customized translations.

To export translations:

1. Click on the *Export translations* button in the left sidebar.
2. Choose a location in your computer to save the `Export_Translations.csv` file.

You can use the *Built-in translations* option to export all translatable strings. This results a huge .csv file. Use this option only if you would like to change the existing translations of the system.

To import translations:

1. Click on the *Browse…* button in the left sidebar.
2. Select a previously exported `.csv` file.
3. Click on the *Import translations* button.

The file needs to be in .csv format, UTF-8 encoded, having `;` as a separator and values encapsulated with `"`. List of available languages and language codes can be viewed in the `DefaultUsedLanguages` system configuration setting.

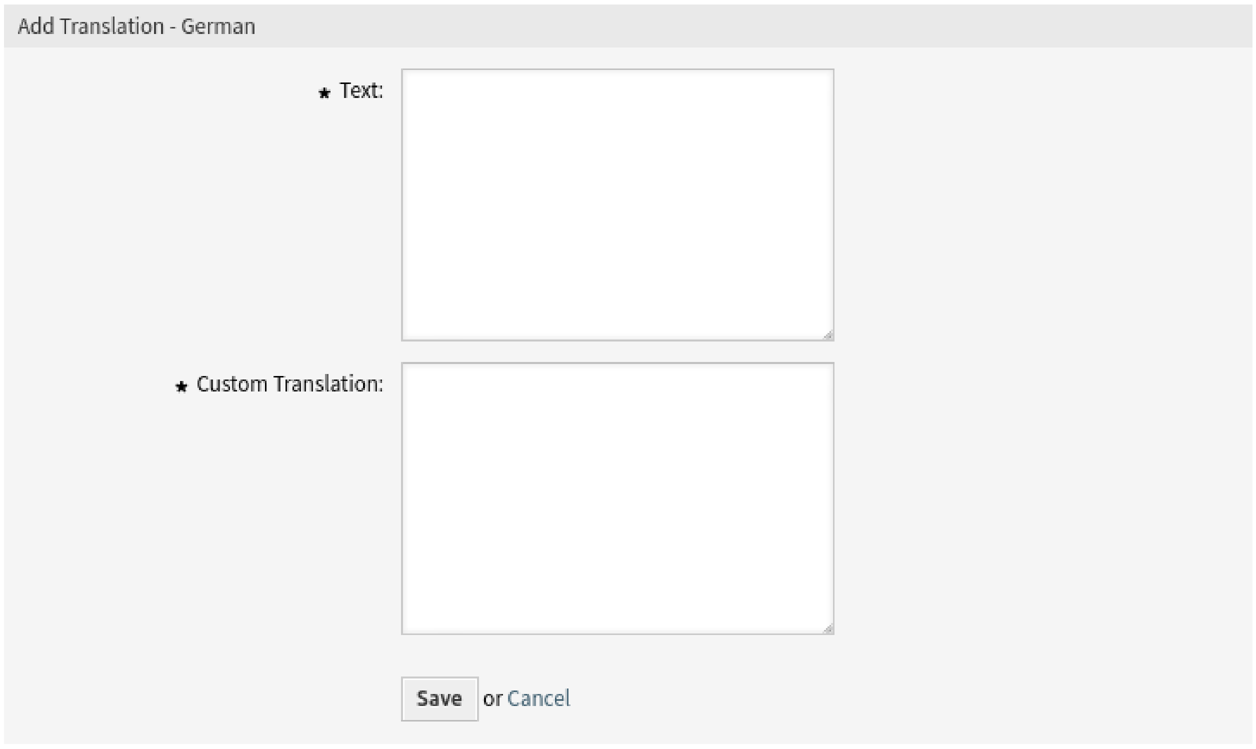
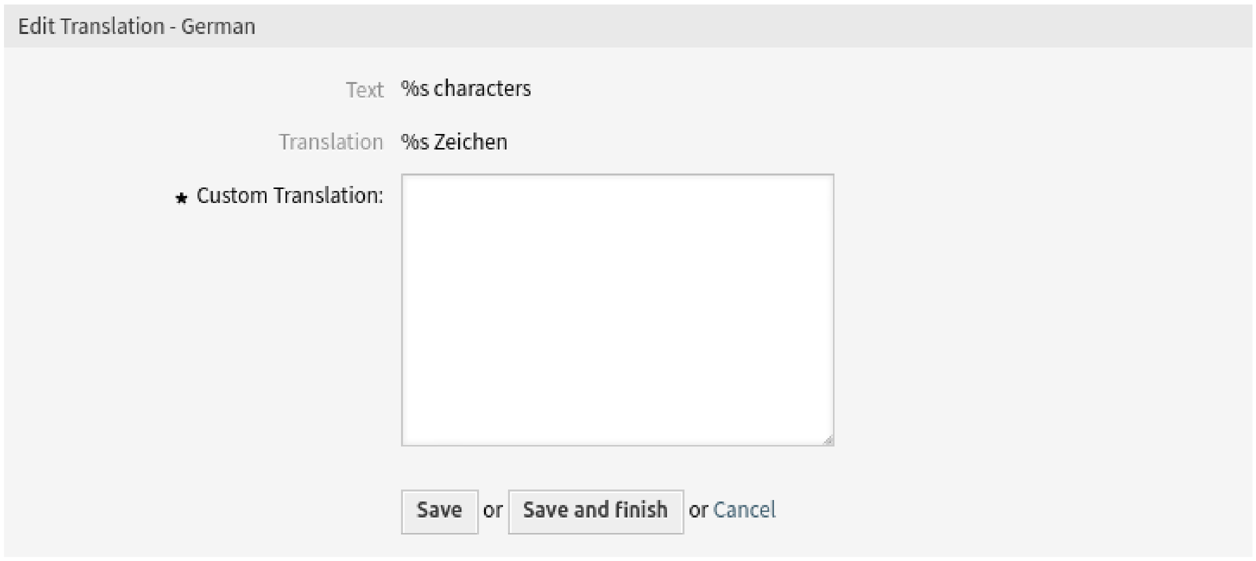


Fig. 75: Add New Translation Screen


Edit Translation - German

Text %s characters

Translation %s Zeichen

Fig. 76: Edit Translation Screen

Custom translations overwrite the built-in translations. This feature can be used to change the default translations of the system.

It is highly recommend to change only those translations which are specific to your company, because custom translations take place only in the target system. Correcting wrong translations like fixing typos or grammar issues should be done on the Translations Portal where translations are synced before each release and will be part of the next version of OTRS.

Note: Source strings can contain placeholders (`%s` characters) and markups (`<html>` or `` `Markdown` `` format). Make sure to copy these special characters without changing them.

Note: Use the search box to find a particular translation.

CHAPTER

ELEVEN

TUTORIALS

Any system requires configuration. Configuring a system should be an easy task and the tools for configuration fit-for-purpose.

OTRS offers several administration tools to configure, monitor, control and extend OTRS.

11.1 Agent Email Interface

This feature makes it possible for an agent to work with tickets via emails without having to use the web agent interface of **OTRS**.

Reply to any ticket notification and insert for example the following command anywhere in the body of the mail between two special command tags, like this:

```
<OTRS_CMD> close </OTRS_CMD>
```

Switch to the ticket from the ticket notification and compare the ticket state. The ticket should have the state *closed*.

Commands can be combined (comma separated) if it makes sense to combine them.

```
<OTRS_CMD> send, nocc </OTRS_CMD>
```

The following commands exist:

- `send` –Sends an email to the customer, including the carbon copy and blind carbon copy recipients (`send` also locks the ticket).
- `nocc` –If used together with `send`, the carbon copy and blind carbon copy recipients will be excluded.
- `lock` –Locks the ticket.
- `unlock` –Unlocks the ticket.
- `get` –Gets the last customer article and sends it to the agent.
- `note` –Adds an internal note to the ticket.
- `close` –Closes the ticket (also unlocks the ticket).

The functionality of this feature is restricted to agents known to the system, but it does not use advanced security mechanisms such as digital signatures and the like.

Warning: An attacker can impersonate the agent by sending a mail with the address of the agent. Activate it to your own risk!

This feature is not enabled by default, it must be configured by an administrator first.

The agent interface address should be set to the same as in setting *Core → Email → NotificationSenderEmail*.

To use this feature:

1. Go to the *System Configuration* screen.
2. Search for the setting `NotificationSenderEmail`.
3. Change the email address if needed, and remember it.
4. Search for the setting `PostMaster::PreFilterModule###999-AgentEmailInterface`.
5. Change the value of key `AgentInterfaceAddress` to the same as set in `NotificationSenderEmail`.

11.2 Archiving Tickets

OTRS can be used as an audit-proof system. In this case deleting closed tickets may not be a good idea. Therefore a feature has been implemented that allows you to archive tickets.

Tickets in *closed* states that match certain criteria can be marked as *archived*. These tickets are not displayed in ticket lists and not accessed by generic agent jobs. The system itself does not have to deal with a huge amount of tickets any longer as only the latest tickets are taken into consideration. This can result in a huge performance gain on large systems.

To use the archive feature:

1. Go to the *System Configuration* screen.
2. Search for the setting `Ticket::ArchiveSystem` and enable the setting.
3. Go to the *Generic Agent* screen.
4. Click on the *Add job* button in the left sidebar.
5. Provide a name in the *Job Settings* section, and select proper options to schedule this job.
6. Define the ticket filters in the *Select Tickets* section. It might be a good idea to only archive those tickets in a closed state that have been closed a few months before.
7. In the *Update/Add Ticket Attributes* section, set the field *Archive selected tickets* to *archive tickets*.
8. Click on the *Save* button.
9. Run the job. The system will display all tickets which will be archived when executing the generic agent job.

Note: Up to 5000 tickets can be modified by running this job manually.

Archived tickets can be still searched by using the search feature in the top of the page but they are not displayed in ticket lists and ticket list widgets. However, these lists and widgets have a filter in their configuration to display the archived tickets, too.

To display the archived tickets in a ticket list or in a ticket list widget:

1. Open the screen configuration or the widget configuration using the gear icon in the top right corner.
2. Add the *Archive search* filter.
3. Select *All tickets* from the list.

11.3 Knowledge Base Article Approval

OTRS has a knowledge base article approval feature. If you activate the approval feature all newly created knowledge base articles also create a new ticket in a predefined queue. The persons who need to approve the knowledge base articles can then act on these tickets and approve the knowledge base articles if they see fit. As long as the article has not been approved the article will not be visible in the external interface.

The approval function can be activated via the `FAQ::ApprovalRequired` system configuration setting. The following system configuration settings may be useful to set this feature properly:

```
FAQ::ApprovalGroup
FAQ::ApprovalRequired
FAQ::ApprovalTicketBody
FAQ::ApprovalTicketDefaultState
FAQ::ApprovalTicketPriority
FAQ::ApprovalTicketSubject
FAQ::ApprovalTicketType
```

The approval message can be defined in the `FAQ::ApprovalTicketBody` setting. You can modify the text if you need, and you can also use OTRS smart tags that will be substituted with their actual values when the approval note is generated. The available OTRS smart tags are listed in the table below.

Name	Description
`<OTRS_FAQ_CATEGORY>`	Category name of the knowledge base article.
`<OTRS_FAQ_CATEGORYID>`	Category ID of the knowledge base article.
`<OTRS_FAQ_LANGUAGE>`	Language of the knowledge base article.
`<OTRS_FAQ_ITEMID>`	Item ID of the knowledge base article.
`<OTRS_FAQ_NUMBER>`	FAQ number of the knowledge base article.
`<OTRS_FAQ_TITLE>`	Title of the knowledge base article.
`<OTRS_FAQ_AUTHOR>`	Author name of the knowledge base article.
`<OTRS_FAQ_STATE>`	State (visibility) of the knowledge base article.

11.4 Creating New Phone Ticket on Incoming Calls

It is possible to integrate OTRS with third-party computer telephony integration (CTI) solutions and make them work in unison. On as simple action as a click on a button in the CTI incoming call screen, the OTRS *New Phone Ticket* screen can be opened with the customer pre-selected by their phone number.

Note: This feature is only available to *On-Premise* customers. If you are a *Managed* customer, this feature is taken care of by the *Customer Solutions Team* in **OTRS**. Please contact us via support@otrs.com or in the OTRS Portal.

The only pre-requisite for the example below is that the used CTI software is able to open a link of a certain format in the web browser.

1. First, edit the customer user configuration file and add the phone number field to `CustomerUserSearchFields` (in our case it is aptly named `phone`):

   ```
   CustomerUserSearchFields => [ 'login', 'first_name', 'last_name', 'customer_id',
   ↪'phone' ],
   ```

   See also:

   For more information on how to provide custom user configuration please see the *Customer User Back Ends* chapter.

   Warning: The customer user configuration is provided as an additional Perl module file. It is your responsibility to check the validity of the supplied file, since any syntax mistakes can lead to a broken system.

   Optionally, you can also add the `phone` field to `CustomerUserListFields` array, if you want to see the telephone number while searching for customer users:

   ```
   CustomerUserListFields => [ 'first_name', 'last_name', 'phone', 'email' ],
   ```

2. Configure your CTI software to open the following link when you answer the phone:

   ```
   https://otrs.example.com/agent/ticket/create/phone?CustomerUserSearch=1&
   ↪CustomerUserID=callernumber
   https://otrs.example.com/agent/ticket/create/phone?CustomerUserSearch=1&
   ↪CustomerUserID=callerloginname
   ```

   For example, for customer user *John Smith* with login name *johnsmith* and telephone number *12345678*:

   ```
   https://otrs.example.com/agent/ticket/create/phone?CustomerUserSearch=1&
   ↪CustomerUserID=12345678
   https://otrs.example.com/agent/ticket/create/phone?CustomerUserSearch=1&
   ↪CustomerUserID=johnsmith
   ```

   Where *otrs.example.com* is the FQDN of the OTRS system in question. Which exact value you use in the end (caller number or login name) depends on what your CTI software can construct the link with.

3. When the ticket create screen is open in OTRS, a search will be conducted for the supplied value. In case only one customer user with that information is found, they will be pre-selected as the ticket customer user, and added to the related fields.

Note: In case you would like to do a search by the customer user email address, first make sure that the `CustomerUserPostMasterSearchFields` key of the customer user configuration is set to an appropriate field. Additionally, you must use an alternative URL query parameter called `CustomerPostMasterSearch=1`, since the email addresses use a different search mechanism. Finally, provide the email address for search as the `CustomerUserID` parameter, but you make sure to URL encode the value because it can contain some unsafe characters.

Note: Mechanism described above works for any of the following ticket create screens:

- New Phone Ticket (default slug: `/agent/ticket/create/phone`)
- New Email Ticket (default slug: `/agent/ticket/create/email`)
- New SMS Ticket (default slug: `/agent/ticket/create/sms`)
- New Process Ticket (default slug: `/agent/ticket/create/process`)

Considering the new process ticket screen consists of a more complex form, you must also provide a correct `ProcessID` parameter in order for the process to be pre-selected. For example:

```
https://otrs.example.com/agent/ticket/create/process?ProcessID=Process-
↪9690ae9ae455d8614d570149b8ab1199&CustomerUserSearch=1&CustomerUserID=12345678
```

11.5 Custom URL Support

Most OTRS front ends support automation via custom URL addresses, whether for specifying configuration parameters, or automating forms. This mechanism can be used to integrate OTRS easily into existing workflows, or just provide a quick pre-configured overview of objects. There is no limit to number of use-cases.

11.5.1 Triggering Actions

Actions in the agent interface can be automatically triggered via a special URL parameter called `Trigger-Action`. The approach outlined below works for any action which is registered for the current screen.

For example, let's trigger the *Change Free Fields* action available in the ticket detail view.

1. Identify the slug name of the view in question. For the ticket detail view and the ticket with ID **1** this is:

   ```
   /agent/ticket/1
   ```

   You can access the ticket via the ticket number which will redirect you to the slug name of the view. If the ticket with ID **1** has the ticket number **2022042710123456** you can use this:

   ```
   /agent/ticket/number/2022042710123456
   ```

2. Identify the `BusinessObjectType` of the screen where the action should be triggered. In our case, for the ticket detail view, this is:

   ```
   Ticket
   ```

3. Go to the *System Configuration* screen and search for the action registration setting in question. In our case this is:

   ```
   AgentFrontend::Ticket::Action###FreeText
   ```

4. Identify the value of the `Component` key in the setting, which is the name of the front end component. In our case this is:

   ```
   TicketFreeText
   ```

5. Construct the action identifier in form of `<BusinessObjectType>::<Component>`. In our case, this is:

```
Ticket::TicketFreeText
```

6. Append a query parameter to the URL of the detail view named `TriggerAction`, and set it to the action identifier, which is the value constructed in the previous step:

```
/agent/ticket/1?TriggerAction=Ticket%3A%3ATicketFreeText
```

Note: In case of custom URL query parameter values, we have to take care to encode all special characters that are normally used in URLs (e.g. double quotes). This process is called "URL encoding", and can be done by any of freely available on-line tools, like the Online Text Tools.

When visiting the URL as constructed above, the action will be automatically triggered:

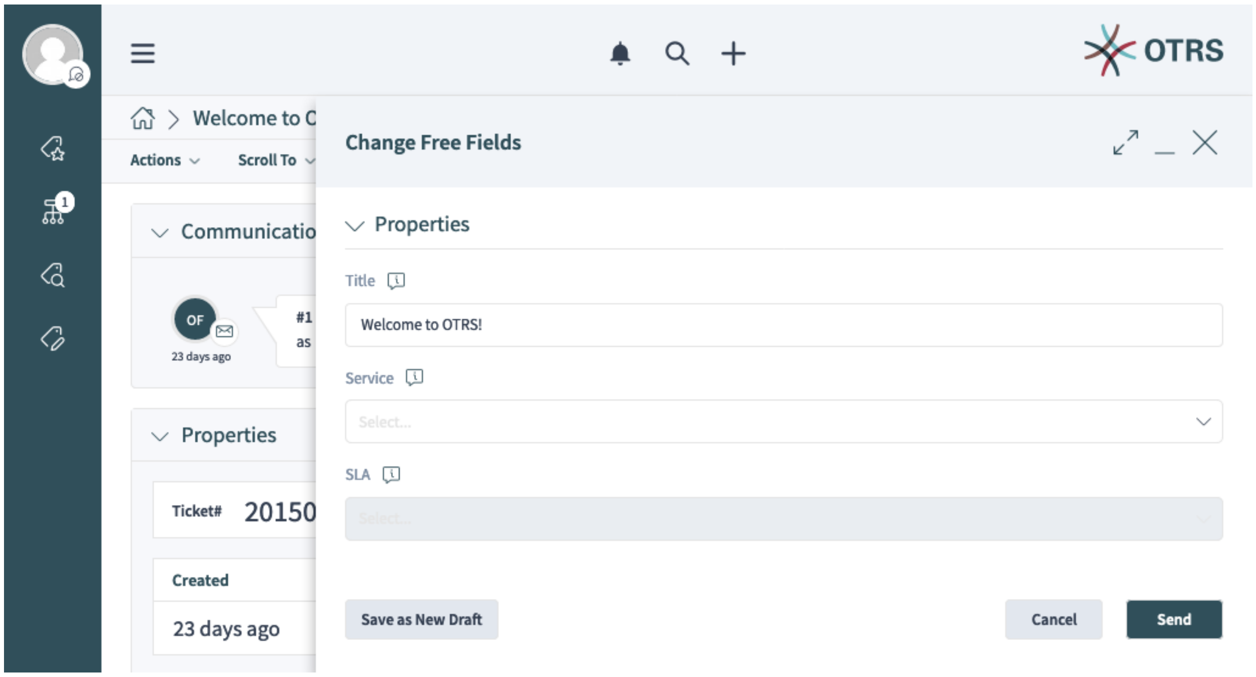


Fig. 1: Automatically Triggered Change Free Fields Action

Note: Please keep in mind that any restrictions applied to the regular screen actions will still be honored. For example, you will not be able to trigger an action which is currently not accessible, e.g. because you don't have permissions to it, or it was filtered out by the ACL rules in effect.

11.5.2 Pre-populating Form Fields

OTRS form fields can be pre-populated with values supplied via URL query parameters. The approach outlined below works for any form in the agent and external interface.

For example, let' s see how we can pre-populate the *Title* field of the *Change Free Fields* action form, which we triggered in the previous example.

Since this action is not initially shown on the screen, we will keep the *trigger action* parameter in place, and just add on to it:

```
/agent/ticket/1?TriggerAction=Ticket%3A%3ATicketFreeText
```

1. Identify the name of the form field in question.

   In our case, we are dealing with the following form:

   ```
   Forms###AgentFrontend::Ticket::Action::FreeText
   ```

   According to *the form fields reference*, field is aptly named:

   ```
   Title
   ```

2. Append a query parameter to the URL of the detail view with the name of the field, and set it to the desired value:

   ```
   /agent/ticket/1?TriggerAction=Ticket%3A%3ATicketFreeText&Title=Some+text
   ```

   Note: Additional URL query parameters can be chained with an ampersand character (`&`).

   Note: In case of custom URL query parameter values, we have to take care to encode all special characters that are normally used in URLs (e.g. double quotes). This process is called "URL encoding" , and can be done by any of freely available on-line tools, like the Online Text Tools.

When visiting the URL as constructed above, the action will be automatically triggered and the field pre-populated:

The approach can be repeated for all other fields in the form, for example, if we would also like to set the *Service* field (field name `ServiceID`) to a hypothetical value **Service 2** with an ID of **2**, we can add it to the existing URL like this:

```
/agent/ticket/1?TriggerAction=Ticket%3A%3ATicketFreeText&Title=Some+text&ServiceID=2
```

When visiting the URL as constructed above, the action will be automatically triggered and both fields pre-populated:

Note: In case of most drop-down fields that contain business objects (e.g. services, queues, etc.), you must always provide the ID of the object, not its label. You can recognize these form fields by the `ID` suffix in their name. You should always do a lookup of the object ID value, and set the field to it.

Note: Please keep in mind that any restrictions applied to the regular action form fields will still be honored. For example, you will not be able to set a value to a disabled field, or select an option which was filtered out

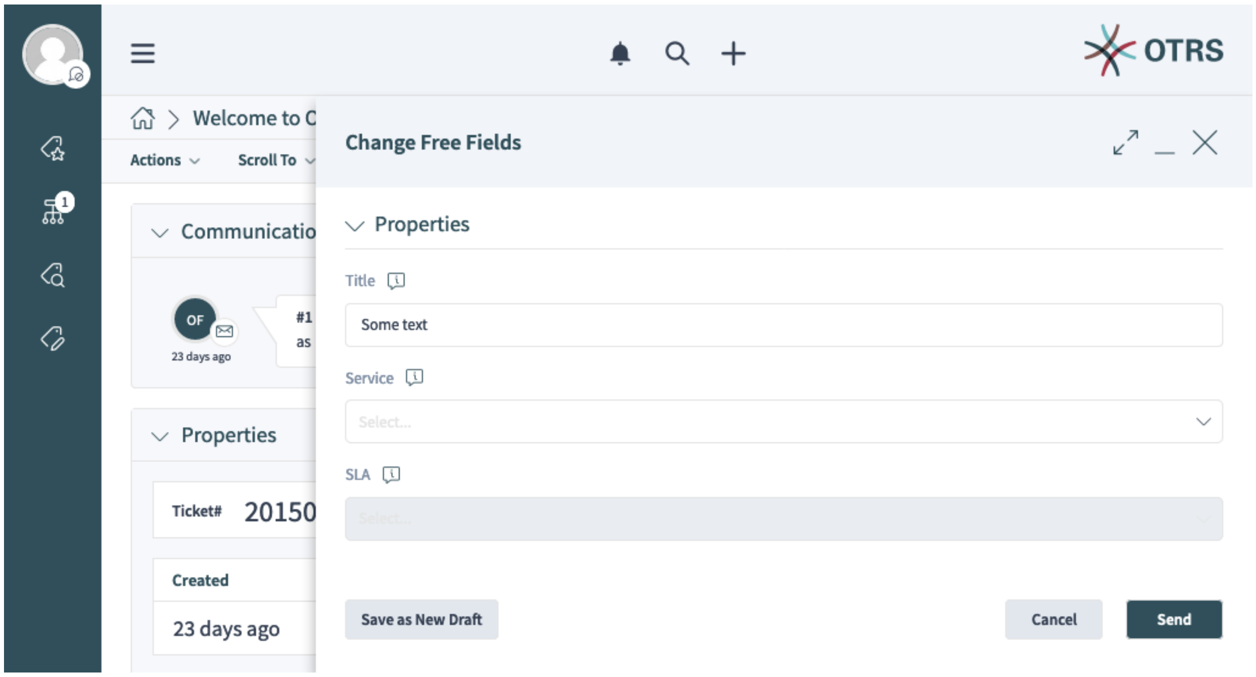


Fig. 2: Pre-populated Title Field of the Change Free Fields Action

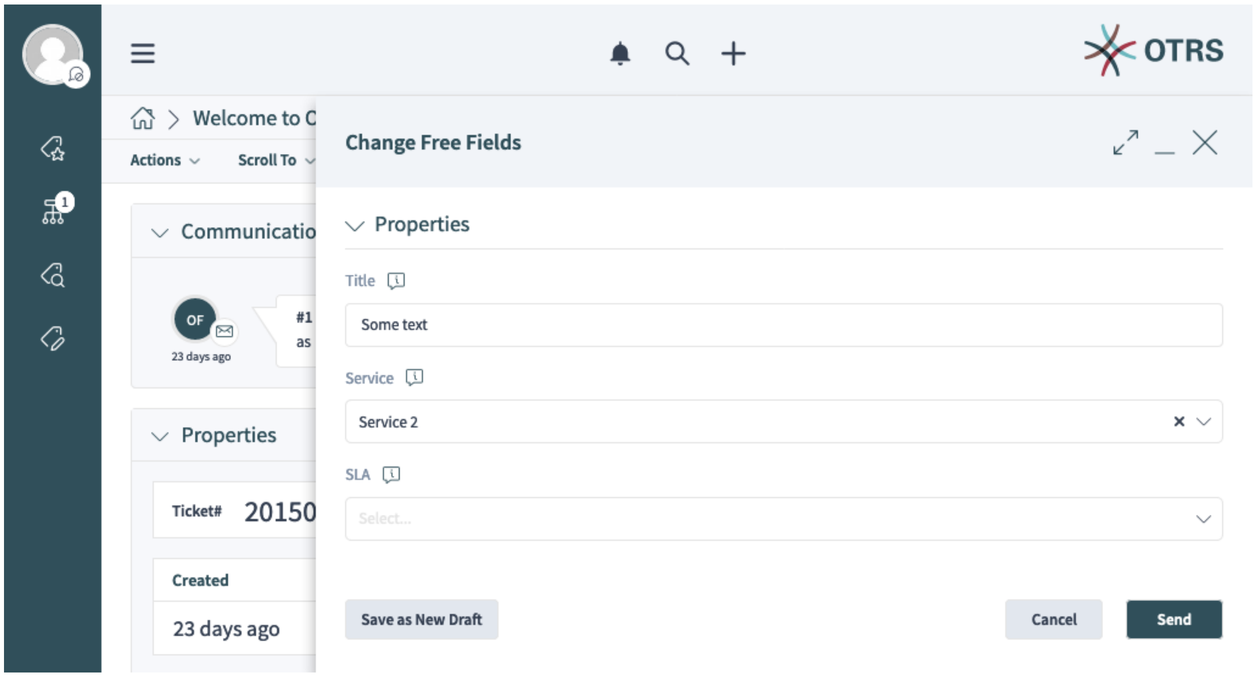


Fig. 3: Pre-populated Title and Service Fields of the Change Free Fields Action

by the ACL rules in effect.

11.5.3 Submitting Actions

Actions in the agent interface can be automatically submitted by via a special URL parameter called `SubmitForm`. The approach outlined below works for any action which is registered for the current screen.

For example, let' s see how we can submit the *Change Free Fields* action form, which we already triggered and pre-populated in previous examples.

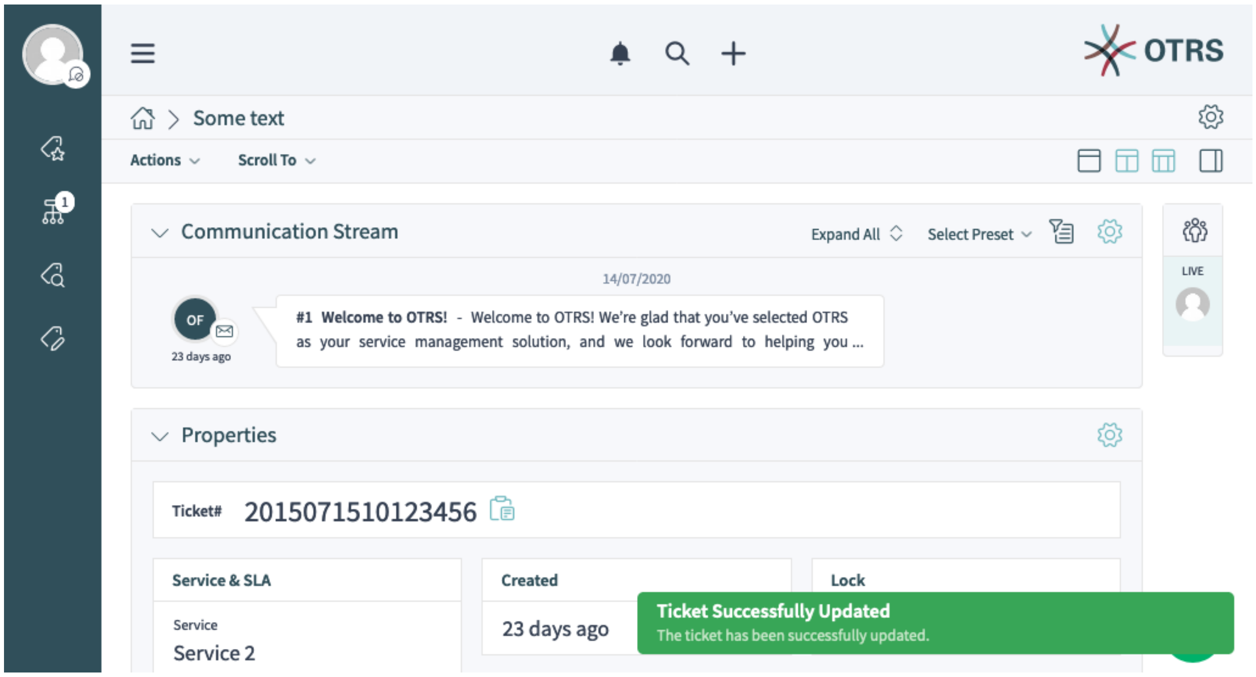
Since this action is not initially shown on the screen, we will keep the *trigger action* parameter in place, as well as some *form field parameters*, and just add on to it:

```
/agent/ticket/1?TriggerAction=Ticket%3A%3ATicketFreeText&Title=Some+text&ServiceID=2
```

Append a query parameter to the URL of the detail view named `SubmitForm` and set **1** as its value:

```
/agent/ticket/1?TriggerAction=Ticket%3A%3ATicketFreeText&Title=Some+text&ServiceID=2&
↪SubmitForm=1
```

When visiting the URL as constructed above, the action will be automatically triggered, both fields pre-populated and the action immediately submitted:



#1 Welcome to OTRS! - Welcome to OTRS! We're glad that you've selected OTRS as your service management solution, and we look forward to helping you ...

Ticket# 2015071510123456

Service 2

23 days ago

Ticket Successfully Updated
The ticket has been successfully updated.

Fig. 4: Automatically Submitted Change Free Fields Action

Note: Please keep in mind that any restrictions applied to the regular action forms will still be honored. For example, you will not be able to submit an invalid form by omitting a mandatory field, or setting a field to an invalid value.

11.5.4 Custom Business Object Lists

All static business object lists in the agent interface support custom configuration via a special URL parameter called `Config`. The approach outlined below works for similar screens which are listed in *the following section*, and for any keys documented in *the YAML reference*.

For example, let's see how we can set an active filter preset for a business object list by carefully constructing a direct URL for the view.

1. Identify the *slug name* of the view in question. We will be using `AgentFrontend::TicketList::Static` which has the slug:

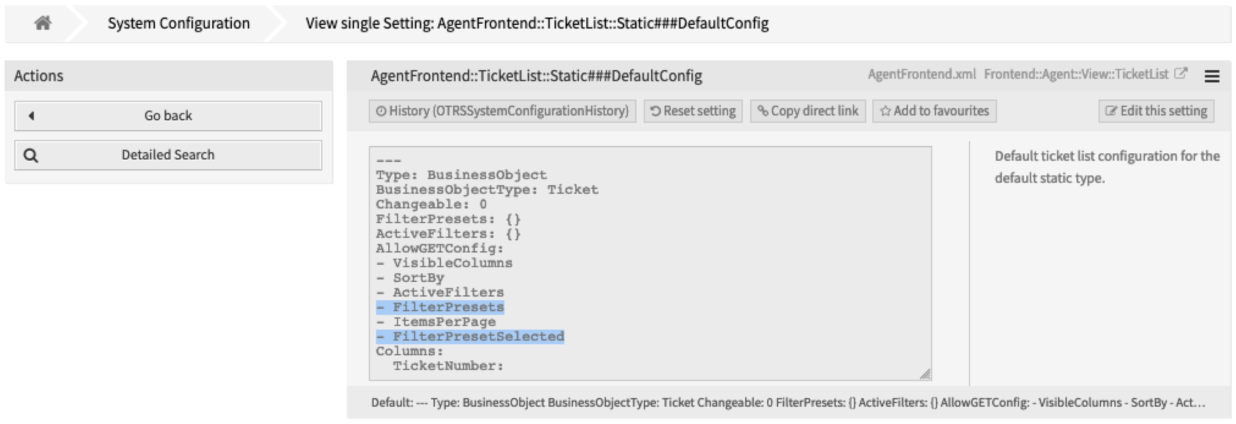
```
/agent/tickets/static
```

2. Go to the *System Configuration* screen and search for the screen configuration setting in question, in our case this is:

```
AgentFrontend::TicketList::Static###DefaultConfig
```

2. Make sure that the key `AllowGETConfig` in the screen configuration contains the properties `FilterPresets` and `FilterPresetSelected`, which we will pass via the query parameter.

   To be able change these keys, they must present in the list, otherwise they will be ignored. For more information, please see the *key reference*.


System Configuration
View single Setting: AgentFrontend::TicketList::Static###DefaultConfig
Actions
Go back
Detailed Search
AgentFrontend::TicketList::Static###DefaultConfig
AgentFrontend.xml Frontend::Agent::View::TicketList
History (OTRSSystemConfigurationHistory)
Reset setting
Copy direct link
Add to favourites
Edit this setting
Default ticket list configuration for the default static type.

```
---
Type: BusinessObject
BusinessObjectType: Ticket
Changeable: 0
FilterPresets: {}
ActiveFilters: {}
AllowGETConfig:
- VisibleColumns
- SortBy
- ActiveFilters
- FilterPresets
- ItemsPerPage
- FilterPresetSelected
Columns:
  TicketNumber:
```

Default: --- Type: BusinessObject BusinessObjectType: Ticket Changeable: 0 FilterPresets: {} ActiveFilters: {} AllowGETConfig: - VisibleColumns - SortBy - Act...

Fig. 5: Screen Configuration with Defined Allowed Keys

3. Construct a valid YAML configuration that defines a user-defined filter preset and sets it as pre-selected.

```
---
FilterPresets:
  "Total Tickets":
    CustomerID:
      Value: my-customer@otrs.com
FilterPresetSelected: "Total Tickets"
```

4. Convert the YAML structure from previous step into JSON syntax, since YAML is unsuitable for passing via parameters. If we do this, we get the following analogous structure.

```
{
   "FilterPresets": {
      "TotalTickets": {
```

(continues on next page)

(continued from previous page)

```
            "CustomerID": {
                "Value": "my-customer@otrs.com"
            }
        }
    },
    "FilterPresetSelected": "Total Tickets"
}
```

4. Remove all the extra white space to compress this configuration into a single line.

```
{"FilterPresets":{"Total Tickets":{"CustomerID":{"Value":"my-customer@otrs.com"}}}
↪,"FilterPresetSelected":"Total Tickets"}
```

5. To pass the configuration to the screen located at `/agent/tickets/static`, we just need to define an URL query parameter named `Config` and set the JSON structure above as its value.

Note: In case of custom URL query parameter values, we have to take care to encode all special characters that are normally used in URLs (e.g. double quotes). This process is called "URL encoding" , and can be done by any of freely available on-line tools, like the Online Text Tools.

```
/agent/tickets/static?Config=%7B%22FilterPresets%22%3A%7B%22Total%20Tickets%22%3A
↪%7B%22CustomerID%22%3A%7B%22Value%22%3A%22my-customer%40otrs.com%22%7D%7D%7D%2C
↪%22FilterPresetSelected%22%3A%22Total%20Tickets%22%7D
```

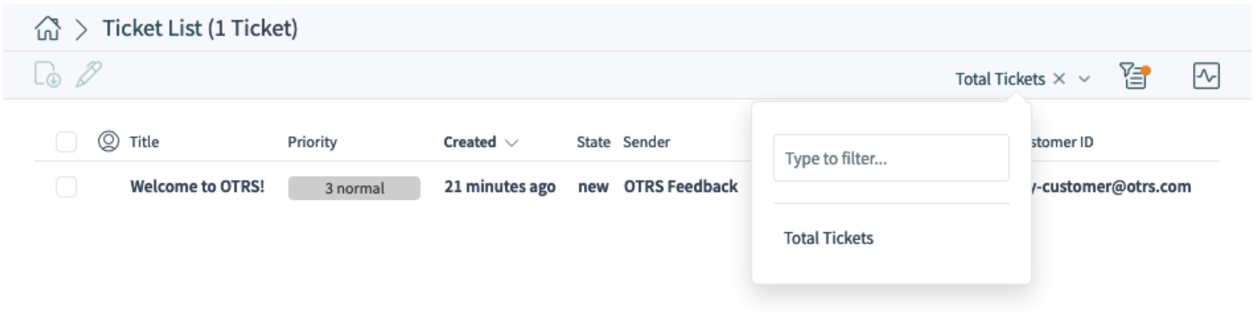


Fig. 6: Example of the Filter Preset Defined via URL Query Parameter

Note: Did you know that you can use many freely available on-line tools to do the conversion steps outlined above? Some of them even support chaining, like the Online YAML Tools for the example above.

11.6 Inline Editing

OTRS provides an inline editing feature to easily modify any of the ticket attributes right in a list table or a property card. This allows for fast contextual edits, which can amplify an already instituted workflow for tickets and make it easier for agent users to do their daily work.

11.6.1 Inline Editing in Business Object Lists

For an example, we will demonstrate how to turn on inline editing for the *State* ticket attribute in the *Recently Created Tickets* list screen.

1. Go to the *System Configuration* screen.
2. Search for the setting `AgentFrontend::TicketList::Created###DefaultConfig`.
3. Edit the setting and change the `IsInlineEditable` flag to `1` under the `State` column.

```
State:
  IsVisible: 2
  IsInlineEditable: 1
```

4. Search for the setting `AgentFrontend::Ticket::InlineEditing::Property###State` and verify the required permission level and lock state on the ticket, in order for the editing to be allowed for the user. By default, this attribute will require `state` permission type and ticket in a locked state.
5. Deploy the modified system configuration.

From now on, the *State* ticket attribute will be editable in a popover which is displayed by hovering the value in the same table column on the *Recently Created Tickets* list screen. By using the provided drop-down field, the ticket state can be changed after clicking on the *Save* button.

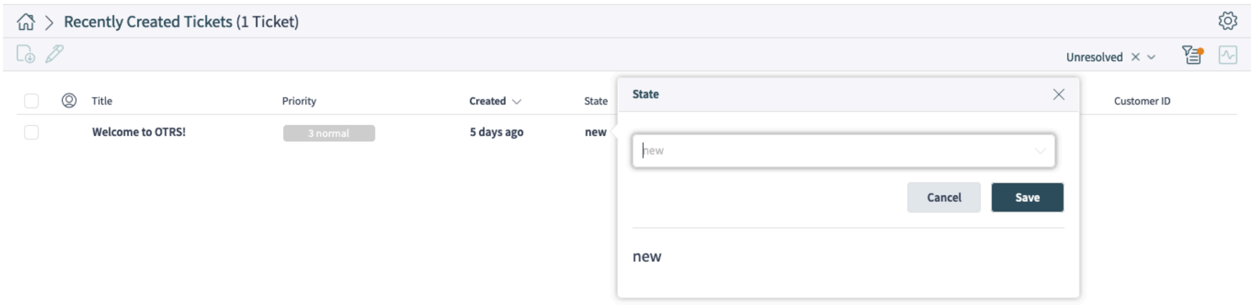


Fig. 7: Ticket State Inline Editing in Business Object List

Warning: Please note that inline editing is only applicable for ticket lists in the organizer items or static screens, and not widgets. Ticket lists in widgets have very limited space, and the column popovers are disabled for them all together.

Note: For granular permission to the inline editing of ticket attributes, you can also employ *Access Control Lists (ACL)*. Each ticket attribute can be targeted by the `AgentFrontend::Ticket::InlineEditing::Property::*` endpoint namespace.

For our example from above, this can be done by the targeting the `AgentFrontend::Ticket::InlineEditing::Property::State` context.

11.6.2 Inline Editing in Property Cards

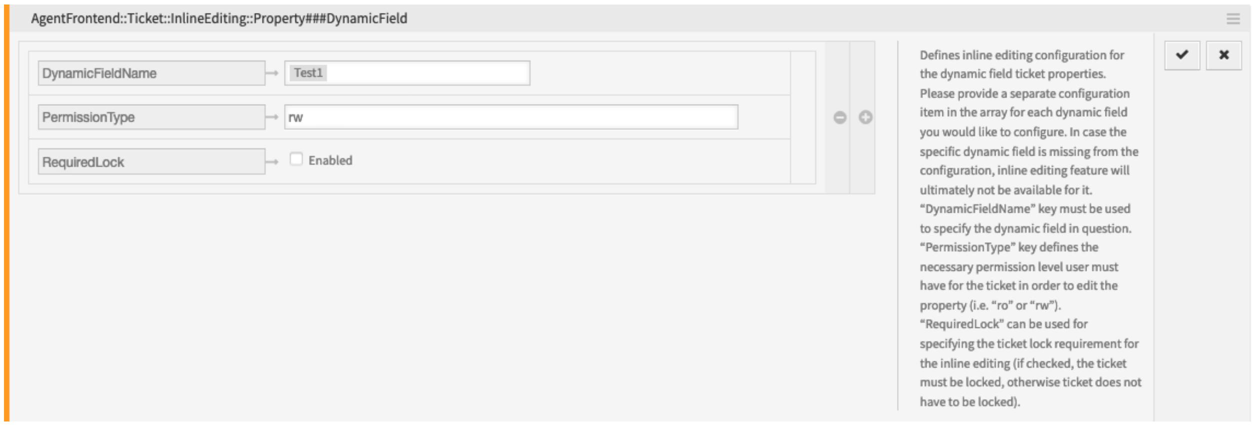
For an example, we will demonstrate how to turn on inline editing for a ticket dynamic field property card in the *Properties* widget on the ticket detail view.

The following example uses a dynamic field named `Test1`. Please make sure that you replace it with the actual name of your dynamic field.

1. Go to the *System Configuration* screen.
2. Search for the setting `AgentFrontend::TicketDetailView::WidgetType###Properties`.
3. Edit the setting and add a property definition under the `Properties` key for the dynamic field in question:

   ```
   Properties:
   - Name: DynamicField_Test1
     IsVisible: 2
     IsInlineEditable: 1
   ```

   Note the `IsInlineEditable` flag which is set to `1` in order to activate inline editing feature for this property card.
4. Search for the setting `AgentFrontend::Ticket::InlineEditing::Property###DynamicField`.
5. Edit the setting and add a configuration for the dynamic field by clicking on the plus button. Choose the name of the dynamic field under the `DynamicFieldName` key. Verify the required permission level and lock state on the ticket, in order for the editing to be allowed for the user. By default, this attribute will require `rw` permission type and ticket in an unlocked state.



AgentFrontend::Ticket::InlineEditing::Property###DynamicField

DynamicFieldName → Test1

PermissionType → rw

RequiredLock → ☐ Enabled

Defines inline editing configuration for the dynamic field ticket properties. Please provide a separate configuration item in the array for each dynamic field you would like to configure. In case the specific dynamic field is missing from the configuration, inline editing feature will ultimately not be available for it. "DynamicFieldName" key must be used to specify the dynamic field in question. "PermissionType" key defines the necessary permission level user must have for the ticket in order to edit the property (i.e. "ro" or "rw"). "RequiredLock" can be used for specifying the ticket lock requirement for the inline editing (if checked, the ticket must be locked, otherwise ticket does not have to be locked).

Fig. 8: Dynamic Field Inline Editing Configuration

5. Deploy the modified system configuration.

From now on, the configured dynamic field property card will be editable in the *Properties* widget of the ticket detail view. An editing icon in the upper right corner of the property card will switch the card in the editing mode. The dynamic field value can then be changed after clicking on the *Save* button.

> **Warning:** Please note that property cards are shown only for defined dynamic field values. If a ticket does not have a dynamic field value set, the property card will not be shown, and, in turn, it will not be possible to edit it inline. Similarly, in case you remove an existing dynamic field value, property card will disappear.

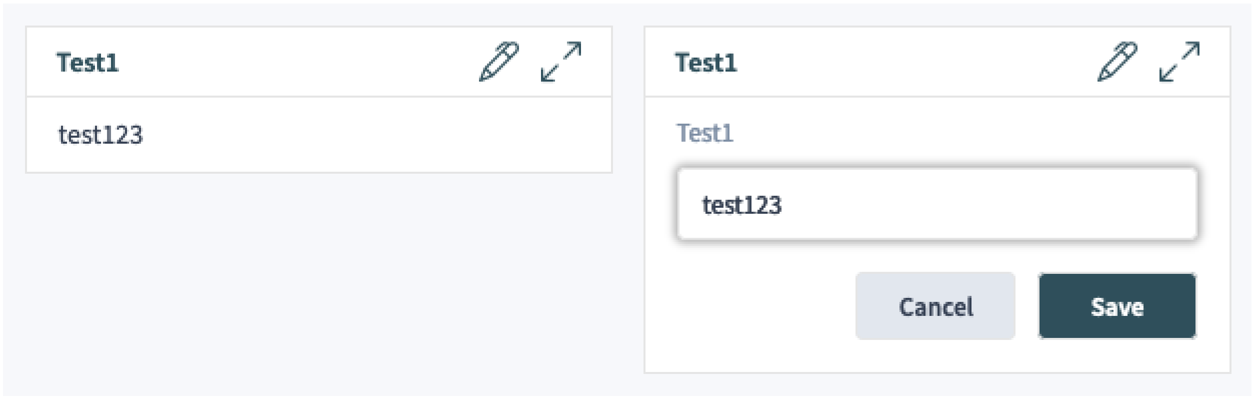


Fig. 9: Dynamic Field Inline Editing in Properties Widget

Note: For granular permission to the inline editing of ticket dynamic field values, you can also employ *Access Control Lists (ACL)*. Each dynamic field can be targeted by the `AgentFrontend::Ticket::InlineEditing::Property::DynamicField_*` endpoint namespace.

For our example from above, this can be done by the targeting the `AgentFrontend::Ticket::InlineEditing::Property::DynamicField_Test1` context.

11.7 External Link Previews

OTRS supports link previews via several features aimed at integrating externally provided content. By having contextual information in the right place, agent' s daily work can be made much more streamlined and straightforward.

11.7.1 Article Meta Filters

Article content can be automatically scanned and relevant information extracted with the help of meta filters.

For example, let' s configure extraction of CVE codes from article preview in the ticket detail view and display direct links to external pages with more information.

1. Go to the *System Configuration* screen, and enable the `AgentFrontend::TicketDetailView::ArticleMeta` master switch for the article meta feature.
2. Search for the setting `AgentFrontend::TicketDetailView::ArticleMetaFilters###0002-Custom` and enable it.
3. Edit the setting, and set its value to the following YAML configuration:

```
---
- Label: CVE Details
  Name: cvedetails
  Target: _blank
  URL: https://www.cvedetails.com/cve/<MATCH1>-<MATCH2>-<MATCH3>
  URLPreview: https://www.cvedetails.com/cve/<MATCH1>-<MATCH2>-<MATCH3>
  RegExp:
```

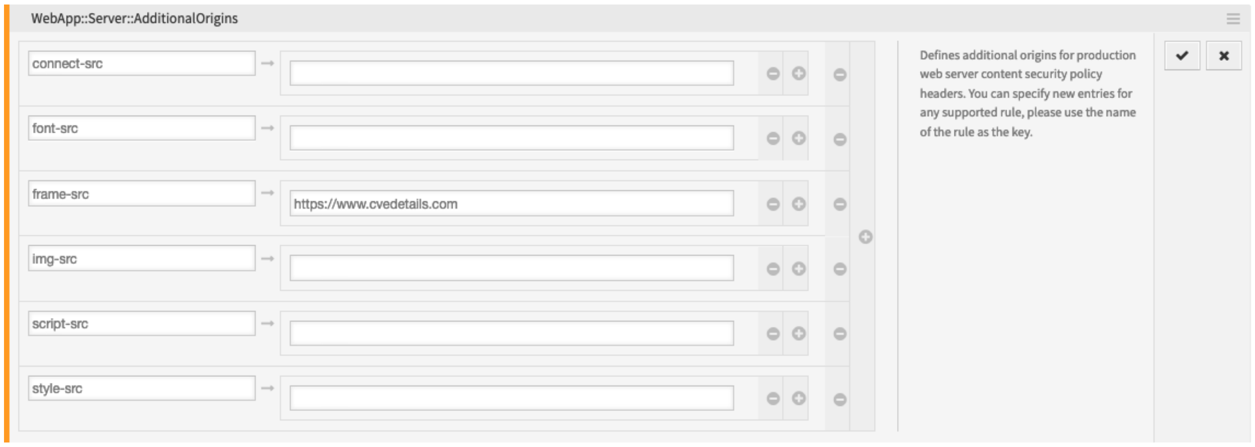
(continues on next page)

(continued from previous page)

```
  - (CVE|CAN)\-(\d{3,4})\-(\d{2,})
  Active: 1
```

4. To configure the Content Security Policy, search for the setting `WebApp::Server::AdditionalOrigins`.
5. Add the following origin under the `frame-src` key, in order to allow these web pages for embedding in the OTRS application:

```
https://www.cvedetails.com
```



WebApp::Server::AdditionalOrigins

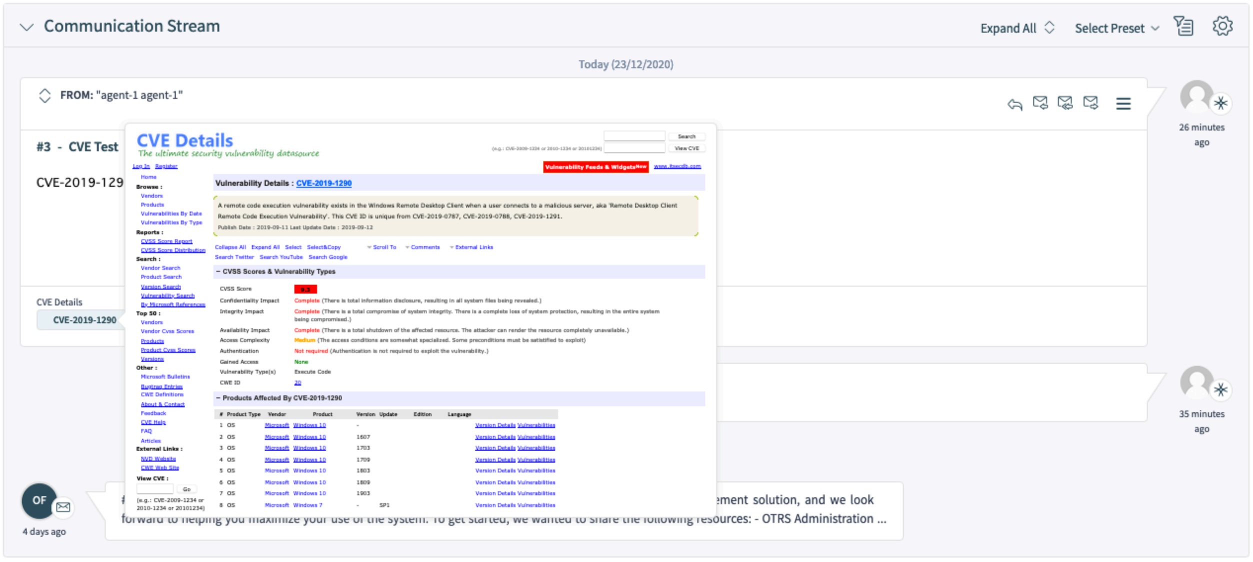
connect-src	
font-src	
frame-src	https://www.cvedetails.com
img-src	
script-src	
style-src	

Defines additional origins for production web server content security policy headers. You can specify new entries for any supported rule, please use the name of the rule as the key.

Fig. 10: Additional Origins for Content Security Policy

6. Deploy the modified system configuration.

To test the feature, you will now need a ticket article that contains at least one valid CVE identifier in its text, for example `CVE-2019-1290`.



#3 - CVE Test

CVE-2019-129

CVE Details
CVE-2019-1290

(e.g.: CVE-2009-1234 or 2010-1234 or 20101234)

Vulnerability Details : CVE-2019-1290

A remote code execution vulnerability exists in the Windows Remote Desktop Client when a user connects to a malicious server, aka 'Remote Desktop Client Remote Code Execution Vulnerability'. This CVE ID is unique from CVE-2019-0787, CVE-2019-0788, CVE-2019-1291.

Publish Date : 2019-09-11 Last Update Date : 2019-09-12

– CVSS Scores & Vulnerability Types

CVSS Score	9.3
Confidentiality Impact	Complete (There is total information disclosure, resulting in all system files being revealed.)
Integrity Impact	Complete (There is a total compromise of system integrity. There is a complete loss of system protection, resulting in the entire system being compromised.)
Availability Impact	Complete (There is a total shutdown of the affected resource. The attacker can render the resource completely unavailable.)
Access Complexity	Medium (The access conditions are somewhat specialized. Some preconditions must be satisfied to exploit)
Authentication	Not required (Authentication is not required to exploit the vulnerability.)
Gained Access	None
Vulnerability Type(s)	Execute Code
CWE ID	20

– Products Affected By CVE-2019-1290

#	Product Type	Vendor	Product	Version	Update	Edition	Language	
1	OS	Microsoft	Windows 10	-				Version Details Vulnerabilities
2	OS	Microsoft	Windows 10	1607				Version Details Vulnerabilities
3	OS	Microsoft	Windows 10	1703				Version Details Vulnerabilities
4	OS	Microsoft	Windows 10	1709				Version Details Vulnerabilities
5	OS	Microsoft	Windows 10	1803				Version Details Vulnerabilities
6	OS	Microsoft	Windows 10	1809				Version Details Vulnerabilities
7	OS	Microsoft	Windows 10	1903				Version Details Vulnerabilities
8	OS	Microsoft	Windows 7	-	SP1			Version Details Vulnerabilities

…ement solution, and we look forward to helping you maximize your use of the system. To get started, we wanted to share the following resources: - OTRS Administration …

Fig. 11: Article Meta Preview

A button will be shown below the preview area in the *Communication Stream* or *Communication Compact* widget, which leads to the external page. Hovering over this button will show a preview popover with the scaled down content of the same page.

11.7.2 Dynamic Field Value

A dynamic field can be configured to show a link preview based on its value. The value can be used as whole or part of the link in question.

The following example uses a dynamic field named `Field1`. Please make sure that you replace it with the actual name of your dynamic field.

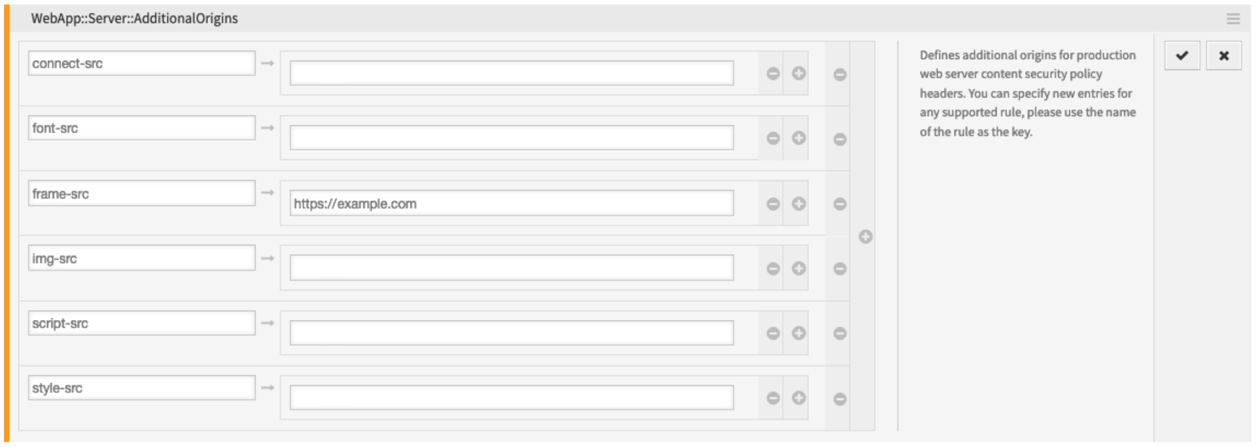
1. Configure the dynamic field to show a link based on its value via *Link for preview* setting. Please see *the setting reference* for more information.

   You can use the following value for testing:

   ```
   https://example.com/handle?query=[% Data.Field1 | uri %]
   ```

2. Configure the display of the dynamic field value in ticket detail view, for example in the *Properties* widget. Please see *the relevant guide* for more information.
3. Go to the *System Configuration* screen.
4. Search for the setting `WebApp::Server::AdditionalOrigins`.
5. Add the following origin under the `frame-src` key, in order to allow these web pages for embedding in the OTRS application:

   ```
   https://example.com
   ```



WebApp::Server::AdditionalOrigins

connect-src	
font-src	
frame-src	https://example.com
img-src	
script-src	
style-src	

Defines additional origins for production web server content security policy headers. You can specify new entries for any supported rule, please use the name of the rule as the key.

Fig. 12: Additional Origins for Content Security Policy

6. Deploy the modified system configuration.

To test the feature, you will now need a ticket with a value set for the dynamic field in question. To set this field via a ticket action, please see *the relevant guide* for more information.

If defined, the ticket dynamic field value will be shown as a property card in the *Properties* widget, in form of a button which leads to the configured link. Hovering over this button will show a preview popover with the scaled down content of the same page.

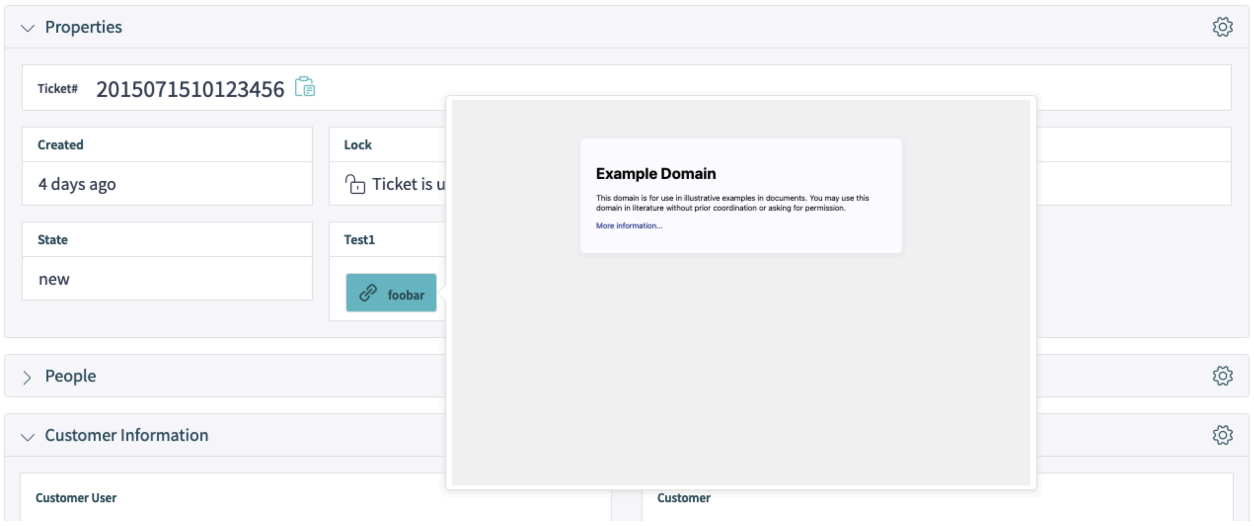


Fig. 13: Dynamic Field Link Preview

11.7.3 Troubleshooting

In case you can not see the preview of the configured link, please note that target web servers might have a configuration in place that prevents them from being embedded in external pages. This is indicated by following or similar errors in the browser console:

```
Load denied by X-Frame-Options: “SAMEORIGIN” from “site”, site does not permit␣
↪cross-origin framing from “site”.
```

```
Firefox prevented this page from loading in this context because the page has an X-
↪Frame-Options policy that disallows it.
```

The blocking is normally done via the `X-Frame-Options` response header which can be set to the prevent the preview on the browser level. If you have the control of the target web server configuration, you might be able to relax this restriction by including the OTRS domain name in the list of allowed origins, or lift the restriction all together.

For more information, please take a look at X-Frame-Options page in the *Mozilla Developer Network*.

Alternatively, the newer `frame-ancestors` directive of the *Content Security Policy* header can be used to block embedding of the target server. This is indicated by following or similar error in the browser console:

```
Refused to display 'https://example.com/...' in a frame because an ancestor violates␣
↪the following Content Security Policy directive: "frame-ancestors 'self'".
```

For more information, please take a look at CSP: frame-ancestors page in the *Mozilla Developer Network*.

11.8 Additional Customer User Attribute

This tutorial explains how to add an additional attribute to the customer user using dynamic fields. Any kind of attribute can be added.

The following example shows how to add an attribute that holds VIP status for the customer user.

11.8.1 VIP Customer User

To use the VIP customer user functionality:

1. Go to the *Dynamic Fields* screen.
2. Add a drop-down dynamic field of type *Customer User*.
   - Name: `CustomerUserVIPStatus`
   - Label: VIP Status
   - Field type: Dropdown
   - Object type: Customer User
   - Possible values: 0 = No VIP , 1 = VIP
   - Default value: No VIP
   - Add empty value: No
   - Translatable values: Yes
   - Show link: leave it empty
3. Go to the *System Configuration* screen.
4. Add the dynamic field to the screens.

```
Agent::Organizer::ItemType###CustomerUserList
Forms###AgentFrontend::CustomerUserCreate::Properties
Forms###AgentFrontend::CustomerUserUpdate::Properties
```

5. Copy the used customer user back end mapping from the `Kernel/Config/Defaults.pm` and paste it into the `Kernel/Config.pm`.
6. Uncomment the dynamic field section in the `Map` array and add the created dynamic field.

```
# Dynamic field example
[ 'DynamicField_CustomerUserVIPStatus', undef, 'CustomerUserVIPStatus', 1, 0,
↪'dynamic_field', undef, 0, undef, undef ],
```

7. Go to the *Access Control Lists (ACL)* screen.
8. Create an ACL that disables the SLA to all customer users. You can import the following ACL as well.

```
- ChangeBy: root@localhost
  ChangeTime: 2021-06-29 11:01:59
  Comment: 'Disable SLA to all customer users.'
  ConfigChange:
    PossibleNot:
      Ticket:
        SLA:
```

(continues on next page)

(continued from previous page)

```
      - 'SLA_Name'
  CreateTime: 2021-06-29 11:01:59
  Description: ''
  ID: 3
  Name: VIP only SLA, disable to all
  StopAfterMatch: 0
  ValidID: 1
```

9. Create an ACL that enables the SLA only to VIP customer users. You can import the following ACL as well.

```
- ChangeBy: root@localhost
  ChangeTime: 2021-06-29 11:03:48
  Comment: 'Enable SLA only to VIP customer users.'
  ConfigChange:
    PossibleAdd:
      Ticket:
        SLA:
        - 'SLA_Name'
  ConfigMatch:
    Properties:
      CustomerUser:
        DynamicField_CustomerUserVIPStatus:
        - '1'
  CreateBy: root@localhost
  CreateTime: 2021-06-29 11:03:48
  Description: ''
  ID: 4
  Name: VIP only SLA, enable to VIPs
  StopAfterMatch: 0
  ValidID: 1
```

Do not forget to change the SLA name in the exported ACLs.

11.9 XSLT Mapping for Outgoing Emails

This feature allows to define XSLT templates for manipulating data mapping of outgoing emails. All email specific data are available for the mapping. Additionally any ticket related data including dynamic fields are provided if an article is sent. The XSLT template must contain at least the following mappings:

```
To / Cc / Bcc
From
Subject
Body
Charset
```

Desired additional email headers are set via the `CustomHeaders` element. Any empty element (for example due to missing mapping data) is rejected.

The difference of origin email data and mapped data is recorded in the *Communication Log* with priority *debug*.

11.9.1 Example XSLT template

This template can be used in the `Sendmail::XSLT::Template` system configuration setting.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml" encoding="utf-8" indent="yes"/>
  <xsl:template match="/RootElement">
    <NewRootElement>
      <To>
        <xsl:value-of select="/RootElement/To" />
      </To>
      <From>
        <xsl:value-of select="/RootElement/From" />
      </From>
      <Subject>
        <xsl:value-of select="/RootElement/Subject" />
      </Subject>
      <MimeType>
        <xsl:value-of select="/RootElement/MimeType" />
      </MimeType>
      <Charset>
        <xsl:value-of select="/RootElement/Charset" />
      </Charset>
      <Body>
        <xsl:value-of select="/RootElement/Body" />
      </Body>
      <CustomHeaders>
        <X-OTRS-Charset>
          <xsl:value-of select="/RootElement/Charset" />
        </X-OTRS-Charset>
        <X-OTRS-From>
          <xsl:value-of select="/RootElement/From" />
        </X-OTRS-From>
        <X-OTRS-TicketNumber>
          <xsl:value-of select="/RootElement/Ticket/TicketNumber" />
        </X-OTRS-TicketNumber>
      </CustomHeaders>
    </NewRootElement>
  </xsl:template>
</xsl:stylesheet>
```

11.10 OTRS Tags

This tutorial gives an overview about OTRS tags that can be used in the screens.

Note: Features can extend the list of OTRS tags. This tutorial explains only those OTRS tags that are shipped with the framework.

11.10.1 OTRS Tag Name Space

OTRS tags are grouped into name spaces. For a quick overview, the following name spaces exist:

```
<OTRS_AGENT_*>
<OTRS_APPOINTMENT_*>
<OTRS_CALENDAR_*>
<OTRS_COMMENT>
<OTRS_CONFIG_*>
<OTRS_CURRENT_*>
<OTRS_CUSTOMER_*>
<OTRS_EMAIL_*>
<OTRS_FAQ_*>
<OTRS_MERGE_TO_TICKET>
<OTRS_NEWPW>
<OTRS_NOTIFICATION_RECIPIENT_*>
<OTRS_OWNER_*>
<OTRS_QUEUE>
<OTRS_REDIRECT_TO>
<OTRS_RESPONSIBLE_*>
<OTRS_TICKET_*>
<OTRS_TOKEN>
```

The * in the end of the tags indicates that some fields can be put there.

11.10.2 OTRS Tag Reference

The following reference lists all possible OTRS tags.

`<OTRS_AGENT_Body>`
: Body text of the latest agent article.

`<OTRS_AGENT_Body[n]>`
: First n lines of the latest agent article.

`<OTRS_AGENT_BODY_RichText>`
: The Rich Text body of the latest agent article.

`<OTRS_AGENT_BODY_RichText[n]>`
: First n lines of the Rich Text body of the latest agent article.

`<OTRS_AGENT_Cc>`
: Carbon copy address of the latest agent article.

`<OTRS_AGENT_ChangeTime>`
: Alias of `<OTRS_CURRENT_ChangeTime>`.

`<OTRS_AGENT_CreateTime>`
: Alias of `<OTRS_CURRENT_CreateTime>`.

`<OTRS_AGENT_Email>`
: Alias of `<OTRS_AGENT_Body>`.

`<OTRS_AGENT_Email[n]>`
: Alias of `<OTRS_AGENT_Body[n]>`.

`<OTRS_AGENT_Note>`
: Alias of `<OTRS_AGENT_Body>`.

<OTRS_AGENT_Note[n]>
: Alias of `<OTRS_AGENT_Body[n]>`.

<OTRS_AGENT_From>
: Sender address of the latest agent article.

<OTRS_AGENT_Subject>
: Subject of the latest agent article.

<OTRS_AGENT_Subject[n]>
: First n characters of the subject of the latest agent article.

<OTRS_AGENT_To>
: Recipient address of the latest agent article.

<OTRS_AGENT_UserEmail>
: Alias of `<OTRS_CURRENT_UserEmail>`.

<OTRS_AGENT_UserFirstname>
: Alias of `<OTRS_CURRENT_UserFirstname>`.

<OTRS_AGENT_UserFullname>
: Alias of `<OTRS_CURRENT_UserFullname>`.

<OTRS_AGENT_UserID>
: Alias of `<OTRS_CURRENT_UserID>`.

<OTRS_AGENT_UserLastname>
: Alias of `<OTRS_CURRENT_UserLastname>`.

<OTRS_AGENT_UserLogin>
: Alias of `<OTRS_CURRENT_UserLogin>`.

<OTRS_AGENT_UserPw>
: Alias of `<OTRS_CURRENT_UserPw>`.

<OTRS_AGENT_ValidID>
: Alias of `<OTRS_CURRENT_ValidID>`.

<OTRS_APPOINTMENT_ALLDAY>
: Returns *Yes* if the appointment is an all day appointment, otherwise returns *No*.

<OTRS_APPOINTMENT_APPOINTMENTID>
: ID of the appointment.

<OTRS_APPOINTMENT_CALENDARID>
: Calendar ID of the appointment.

<OTRS_APPOINTMENT_CHANGEBY>
: Full name of the agent who changed the appointment.

<OTRS_APPOINTMENT_CHANGETIME>
: Date and time when the appointment was changed.

<OTRS_APPOINTMENT_CREATEBY>
: Full name of the agent who created the appointment.

<OTRS_APPOINTMENT_CREATETIME>
: Date and time when the appointment was created.

<OTRS_APPOINTMENT_DESCRIPTION>
: Description of the appointment.

<OTRS_APPOINTMENT_ENDTIME>
End time of the appointment.

<OTRS_APPOINTMENT_LOCATION>
Location of the appointment.

<OTRS_APPOINTMENT_NOTIFICATIONCUSTOMUNIT>
Notification custom unit of the appointment.

<OTRS_APPOINTMENT_NOTIFICATIONCUSTOMUNITCOUNT>
Notification custom unit count of the appointment.

<OTRS_APPOINTMENT_NOTIFICATIONCUSTOMUNITPOINTOFTIME>
Notification custom unit point of time of the appointment.

<OTRS_APPOINTMENT_NOTIFICATIONTEMPLATE>
Notification template of the appointment.

<OTRS_APPOINTMENT_NOTIFICATIONTIME>
Notification time of the appointment.

<OTRS_APPOINTMENT_RECURRENCECOUNT>
Recurrence count of the appointment.

<OTRS_APPOINTMENT_RECURRENCEEXCLUDE>
Recurrence exclude of the appointment.

<OTRS_APPOINTMENT_RECURRENCEFREQUENCY>
Recurrence frequency of the appointment.

<OTRS_APPOINTMENT_RECURRENCEID>
Recurrence ID of the appointment.

<OTRS_APPOINTMENT_RECURRENCEINTERVAL>
Recurrence interval of the appointment.

<OTRS_APPOINTMENT_RECURRENCETYPE>
Recurrence type of the appointment.

<OTRS_APPOINTMENT_RECURRENCEUNTIL>
Recurrence until time of the appointment.

<OTRS_APPOINTMENT_RECURRING>
Returns *Yes* if the appointment is recurring, otherwise returns *No*.

<OTRS_APPOINTMENT_RESOURCEID>
Resource ID of the appointment.

<OTRS_APPOINTMENT_STARTTIME>
Start time of the appointment.

<OTRS_APPOINTMENT_TEAMID>
Team ID of the appointment.

<OTRS_APPOINTMENT_TICKET_CUSTOMER_DATA_ChangeBy>
ID of the agent who changed the customer user of the ticket linked to the appointment.

<OTRS_APPOINTMENT_TICKET_CUSTOMER_DATA_Changed>
Alias of `<OTRS_APPOINTMENT_TICKET_CUSTOMER_DATA_ChangeTime>`.

<OTRS_APPOINTMENT_TICKET_CUSTOMER_DATA_ChangeTime>.
Date and time when the customer user of the ticket linked to the appointment was changed.

<OTRS_APPOINTMENT_TICKET_CUSTOMER_DATA_CreateBy>
ID of the agent who created the customer user of the ticket linked to the appointment.

<OTRS_APPOINTMENT_TICKET_CUSTOMER_DATA_Created>
Alias of `<OTRS_APPOINTMENT_TICKET_CUSTOMER_DATA_CreateTime>`.

<OTRS_APPOINTMENT_TICKET_CUSTOMER_DATA_CreateTime>
Date and time when the customer user of the ticket linked to the appointment was created.

<OTRS_APPOINTMENT_TICKET_CUSTOMER_DATA_CustomerCompanyCity>
City of the company of the customer user of the ticket linked to the appointment.

<OTRS_APPOINTMENT_TICKET_CUSTOMER_DATA_CustomerCompanyComment>
Comment of the company of the customer user of the ticket linked to the appointment.

<OTRS_APPOINTMENT_TICKET_CUSTOMER_DATA_CustomerCompanyCountry>
Country of the company of the customer user of the ticket linked to the appointment.

<OTRS_APPOINTMENT_TICKET_CUSTOMER_DATA_CustomerCompanyName>
Name of the company of the customer user of the ticket linked to the appointment.

<OTRS_APPOINTMENT_TICKET_CUSTOMER_DATA_CustomerCompanyStreet>
Street of the company of the customer user of the ticket linked to the appointment.

<OTRS_APPOINTMENT_TICKET_CUSTOMER_DATA_CustomerCompanyURL>
URL of the company of the customer user of the ticket linked to the appointment.

<OTRS_APPOINTMENT_TICKET_CUSTOMER_DATA_CustomerCompanyZIP>
ZIP code of the company of the customer user of the ticket linked to the appointment.

<OTRS_APPOINTMENT_TICKET_CUSTOMER_DATA_UserCity>
City of the customer user of the ticket linked to the appointment.

<OTRS_APPOINTMENT_TICKET_CUSTOMER_DATA_UserComment>
Comment of the customer user of the ticket linked to the appointment.

<OTRS_APPOINTMENT_TICKET_CUSTOMER_DATA_UserCountry>
Country of the customer user of the ticket linked to the appointment.

<OTRS_APPOINTMENT_TICKET_CUSTOMER_DATA_UserCustomerID>
Customer ID of the customer user of the ticket linked to the appointment.

<OTRS_APPOINTMENT_TICKET_CUSTOMER_DATA_UserEmail>
Email of the customer user of the ticket linked to the appointment.

<OTRS_APPOINTMENT_TICKET_CUSTOMER_DATA_UserFax>
Fax of the customer user of the ticket linked to the appointment.

<OTRS_APPOINTMENT_TICKET_CUSTOMER_DATA_UserFirstname>
First name of the customer user of the ticket linked to the appointment.

<OTRS_APPOINTMENT_TICKET_CUSTOMER_DATA_UserLastname>
Last name of the customer user of the ticket linked to the appointment.

<OTRS_APPOINTMENT_TICKET_CUSTOMER_DATA_UserLogin>
Login name of the customer user of the ticket linked to the appointment.

<OTRS_APPOINTMENT_TICKET_CUSTOMER_DATA_UserMobile>
Cell phone number of the customer user of the ticket linked to the appointment.

<OTRS_APPOINTMENT_TICKET_CUSTOMER_DATA_UserPassword>
Password of the customer user of the ticket linked to the appointment. Masked and replaced by *xxx*.

<OTRS_APPOINTMENT_TICKET_CUSTOMER_DATA_UserPhone>
Phone number of the customer user of the ticket linked to the appointment.

<OTRS_APPOINTMENT_TICKET_CUSTOMER_DATA_UserStreet>
Street of the customer user of the ticket linked to the appointment.

<OTRS_APPOINTMENT_TICKET_CUSTOMER_DATA_UserTitle>
Title of the customer user of the ticket linked to the appointment.

<OTRS_APPOINTMENT_TICKET_CUSTOMER_DATA_UserZip>
ZIP code of the customer user of the ticket linked to the appointment.

<OTRS_APPOINTMENT_TICKET_CUSTOMER_DATA_ValidID>
Validity ID of the customer user of the ticket linked to the appointment.

<OTRS_APPOINTMENT_TICKETAPPOINTMENTRULEID>
Ticket appointment rule ID of the appointment.

<OTRS_APPOINTMENT_TITLE>
Title of the appointment.

<OTRS_APPOINTMENT_TITLE[n]>
First n characters from the title of the appointment.

<OTRS_APPOINTMENT_UNIQUEID>
Unique ID of the appointment.

<OTRS_CALENDAR_CalendarID>
ID of the calendar of the appointment.

<OTRS_CALENDAR_CalendarName>
Name of the calendar of the appointment.

<OTRS_CALENDAR_ChangeBy>
ID of the agent who changed the calendar of the appointment.

<OTRS_CALENDAR_ChangeTime>
Date and time when the calendar of the appointment was changed.

<OTRS_CALENDAR_Color>
Color of the calendar of the appointment in RGB format.

<OTRS_CALENDAR_CreateBy>
ID of the agent who created the calendar of the appointment.

<OTRS_CALENDAR_CreateTime>
Date and time when the calendar of the appointment was created.

<OTRS_CALENDAR_GroupID>
Group ID of the calendar of the appointment.

<OTRS_CALENDAR_TicketAppointments>
Returns an array of the ticket appointments.

<OTRS_CALENDAR_ValidID>
Validation ID of the calendar of the appointment.

<OTRS_COMMENT>
Alias of `<OTRS_CUSTOMER_Body>`.

<OTRS_COMMENT[n]>
Alias of `<OTRS_CUSTOMER_Body[n]>`.

<OTRS_CONFIG_AuthModule::Radius::Password>
Masked and replaced by *xxx*.

<OTRS_CONFIG_Customer::AuthModule::DB::CustomerPassword>
Masked and replaced by *xxx*.

<OTRS_CONFIG_Customer::AuthModule::Radius::Password>
Masked and replaced by *xxx*.

<OTRS_CONFIG_DatabasePw>
Masked and replaced by *xxx*.

<OTRS_CONFIG_PGP::Key::Password>
Masked and replaced by *xxx*.

<OTRS_CONFIG_PublicFrontend::AuthPassword>
Masked and replaced by *xxx*.

<OTRS_CONFIG_SearchUserPw>
Masked and replaced by *xxx*.

<OTRS_CONFIG_SendmailModule::AuthPassword>
Masked and replaced by *xxx*.

<OTRS_CONFIG_UserPw>
Masked and replaced by *xxx*.

<OTRS_CONFIG_X>
Any system configuration value or empty string if the configuration not present. X is the name of the system configuration setting.

<OTRS_CURRENT_ChangeTime>
Change time of the current agent.

<OTRS_CURRENT_CreateTime>
Creation time of the current agent.

<OTRS_CURRENT_UserEmail>
Email of the current agent.

<OTRS_CURRENT_UserFirstname>
First name of the current agent.

<OTRS_CURRENT_UserFullname>
Full name of the current agent.

<OTRS_CURRENT_UserID>
User ID of the current agent.

<OTRS_CURRENT_UserLastname>
Last name of the current agent.

<OTRS_CURRENT_UserLogin>
Login name of the current agent.

<OTRS_CURRENT_UserPw>
Password of the current agent. Masked and replaced by *xxx*.

<OTRS_CURRENT_ValidID>
Validation ID of the current agent.

<OTRS_CUSTOMER_Body>
Body text of the latest customer user article.

<OTRS_CUSTOMER_Body[n]>
First n lines of the latest customer user article.

<OTRS_CUSTOMER_BODY_RichText>
The Rich Text body of the latest customer user article.

<OTRS_CUSTOMER_BODY_RichText[n]>
First n lines of the Rich Text body of the latest customer user article.

<OTRS_CUSTOMER_Cc>
Carbon copy address of the latest customer user article.

<OTRS_CUSTOMER_DATA_UserTitle>
Title of the customer user of the ticket.

<OTRS_CUSTOMER_DATA_UserFirstname>
First name of the customer user of the ticket.

<OTRS_CUSTOMER_DATA_UserLastname>
Last name of the customer user of the ticket.

<OTRS_CUSTOMER_DATA_UserLogin>
Login name of the customer user of the ticket.

<OTRS_CUSTOMER_DATA_UserPassword>
Password of the customer user of the ticket. Masked and replaced by *xxx*.

<OTRS_CUSTOMER_DATA_UserEmail>
Email of the customer user of the ticket.

<OTRS_CUSTOMER_DATA_UserCustomerID>
Customer ID of the customer user of the ticket.

<OTRS_CUSTOMER_DATA_UserPhone>
Phone number of the customer user of the ticket.

<OTRS_CUSTOMER_DATA_UserFax>
Fax number of the customer user of the ticket.

<OTRS_CUSTOMER_DATA_UserMobile>
Cell phone number of the customer user of the ticket.

<OTRS_CUSTOMER_DATA_UserStreet>
Street of the customer user of the ticket.

<OTRS_CUSTOMER_DATA_UserZip>
ZIP code of the customer user of the ticket.

<OTRS_CUSTOMER_DATA_UserCity>
City of the customer user of the ticket.

<OTRS_CUSTOMER_DATA_UserCountry>
Country of the customer user of the ticket.

<OTRS_CUSTOMER_DATA_UserComment>
Comment of the customer user of the ticket.

<OTRS_CUSTOMER_DATA_CustomerCompanyName>
Name of the company of the customer user of the ticket.

<OTRS_CUSTOMER_DATA_CustomerCompanyStreet>
Street of the company of the customer user of the ticket.

<OTRS_CUSTOMER_DATA_CustomerCompanyZIP>
ZIP code of the company of the customer user of the ticket.

<OTRS_CUSTOMER_DATA_CustomerCompanyCity>
City of the company of the customer user of the ticket.

<OTRS_CUSTOMER_DATA_CustomerCompanyCountry>
Country of the company of the customer user of the ticket.

<OTRS_CUSTOMER_DATA_CustomerCompanyURL>
URL of the company of the customer user of the ticket.

<OTRS_CUSTOMER_DATA_CustomerCompanyComment>
Comment of the company of the customer user of the ticket.

<OTRS_CUSTOMER_DATA_CreateTime>
Date and time when the customer user was created.

<OTRS_CUSTOMER_DATA_CreateBy>
ID of the agent who created the customer user.

<OTRS_CUSTOMER_DATA_ChangeTime>
Date and time when the customer user was changed.

<OTRS_CUSTOMER_DATA_ChangeBy>
ID of the agent who changed the customer user.

<OTRS_CUSTOMER_DATA_ValidID>
Validation ID of the customer user.

<OTRS_CUSTOMER_UserTitle>
Alias of `<OTRS_CUSTOMER_DATA_UserTitle>`.

<OTRS_CUSTOMER_UserFirstname>
Alias of `<OTRS_CUSTOMER_DATA_UserFirstname>`.

<OTRS_CUSTOMER_UserLastname>
Alias of `<OTRS_CUSTOMER_DATA_UserLastname>`.

<OTRS_CUSTOMER_UserLogin>
Alias of `<OTRS_CUSTOMER_DATA_UserLogin>`.

<OTRS_CUSTOMER_UserPassword>
Alias of `<OTRS_CUSTOMER_DATA_UserPassword>`.

<OTRS_CUSTOMER_UserEmail>
Alias of `<OTRS_CUSTOMER_DATA_UserEmail>`.

<OTRS_CUSTOMER_UserCustomerID>
Alias of `<OTRS_CUSTOMER_DATA_UserCustomerID>`.

<OTRS_CUSTOMER_UserPhone>
Alias of `<OTRS_CUSTOMER_DATA_UserPhone>`.

<OTRS_CUSTOMER_UserFax>
Alias of `<OTRS_CUSTOMER_DATA_UserFax>`.

<OTRS_CUSTOMER_UserMobile>
Alias of `<OTRS_CUSTOMER_DATA_UserMobile>`.

<OTRS_CUSTOMER_UserStreet>
Alias of `<OTRS_CUSTOMER_DATA_UserStreet>`.

<OTRS_CUSTOMER_UserZip>
Alias of `<OTRS_CUSTOMER_DATA_UserZip>`.

<OTRS_CUSTOMER_UserCity>
Alias of `<OTRS_CUSTOMER_DATA_UserCity>`.

<OTRS_CUSTOMER_UserCountry>
Alias of `<OTRS_CUSTOMER_DATA_UserCountry>`.

<OTRS_CUSTOMER_UserComment>
Alias of `<OTRS_CUSTOMER_DATA_UserComment>`.

<OTRS_CUSTOMER_CustomerCompanyName>
Alias of `<OTRS_CUSTOMER_DATA_CustomerCompanyName>`.

<OTRS_CUSTOMER_CustomerCompanyStreet>
Alias of `<OTRS_CUSTOMER_DATA_CustomerCompanyStreet>`.

<OTRS_CUSTOMER_CustomerCompanyZIP>
Alias of `<OTRS_CUSTOMER_DATA_CustomerCompanyZIP>`.

<OTRS_CUSTOMER_CustomerCompanyCity>
Alias of `<OTRS_CUSTOMER_DATA_CustomerCompanyCity>`.

<OTRS_CUSTOMER_CustomerCompanyCountry>
Alias of `<OTRS_CUSTOMER_DATA_CustomerCompanyCountry>`.

<OTRS_CUSTOMER_CustomerCompanyURL>
Alias of `<OTRS_CUSTOMER_DATA_CustomerCompanyURL>`.

<OTRS_CUSTOMER_CustomerCompanyComment>
Alias of `<OTRS_CUSTOMER_DATA_CustomerCompanyComment>`.

<OTRS_CUSTOMER_CreateTime>
Alias of `<OTRS_CUSTOMER_DATA_CreateTime>`.

<OTRS_CUSTOMER_CreateBy>
Alias of `<OTRS_CUSTOMER_DATA_CreateBy>`.

<OTRS_CUSTOMER_ChangeTime>
Alias of `<OTRS_CUSTOMER_DATA_ChangeTime>`.

<OTRS_CUSTOMER_ChangeBy>
Alias of `<OTRS_CUSTOMER_DATA_ChangeBy>`.

<OTRS_CUSTOMER_ValidID>
Alias of `<OTRS_CUSTOMER_DATA_ValidID>`.

<OTRS_CUSTOMER_Email>
Alias of `<OTRS_CUSTOMER_Body>`.

<OTRS_CUSTOMER_Email[n]>
Alias of `<OTRS_CUSTOMER_Body[n]>`.

<OTRS_CUSTOMER_Note>
Alias of `<OTRS_CUSTOMER_Body>`.

<OTRS_CUSTOMER_Note[n]>
Alias of `<OTRS_CUSTOMER_Body[n]>`.

<OTRS_CUSTOMER_From>
Sender address of the latest customer user article.

<OTRS_CUSTOMER_REALNAME>
Real name of the customer user of the ticket.

<OTRS_CUSTOMER_Subject>
Subject of the latest customer user article.

<OTRS_CUSTOMER_Subject[n]>
First n lines of the subject of the latest customer user article.

<OTRS_CUSTOMER_To>
Recipient address of the latest customer user article.

<OTRS_EMAIL_Date>
Full date and time of the email message based on UTC.

<OTRS_EMAIL_Date[TIMEZONE]>
Full date and time of the email message based on the given TIMEZONE (e.g. Europe/Berlin).

<OTRS_FAQ_Author>
Author name of the knowledge base article.

<OTRS_FAQ_Category>
Category name of the knowledge base article.

<OTRS_FAQ_CategoryID>
Category ID of the knowledge base article.

<OTRS_FAQ_ItemID>
Item ID of the knowledge base article.

<OTRS_FAQ_Language>
Language of the knowledge base article.

<OTRS_FAQ_Number>
Knowledge base article number of the knowledge base article.

<OTRS_FAQ_State>
State (visibility) of the knowledge base article.

<OTRS_FAQ_Title>
Title of the knowledge base article.

<OTRS_MERGE_TO_TICKET>
Ticket number of the new ticket where the original ticket is merged.

<OTRS_NOTIFICATION_RECIPIENT_UserFirstname>
First name of the recipient user of the appointment.

<OTRS_NOTIFICATION_RECIPIENT_UserFullname>
Full name of the recipient user of the appointment.

<OTRS_NOTIFICATION_RECIPIENT_UserLastname>
Last name of the recipient user of the appointment.

<OTRS_NOTIFICATION_RECIPIENT_UserLogin>
Login name of the recipient user of the appointment.

<OTRS_NOTIFICATION_RECIPIENT_UserPassword>
Password of the recipient user of the appointment. Masked and replaced by *xxx*.

<OTRS_OWNER_ChangeTime>
Alias of `<OTRS_TICKET_OWNER_ChangeTime>`.

<OTRS_OWNER_CreateTime>
Alias of `<OTRS_TICKET_OWNER_CreateTime>`.

<OTRS_OWNER_UserEmail>
Alias of `<OTRS_TICKET_OWNER_UserEmail>`.

<OTRS_OWNER_UserFirstname>
Alias of `<OTRS_TICKET_OWNER_UserFirstname>`.

<OTRS_OWNER_UserFullname>
Alias of `<OTRS_TICKET_OWNER_UserFullname>`.

<OTRS_OWNER_UserID>
Alias of `<OTRS_TICKET_OWNER_UserID>`.

<OTRS_OWNER_UserLastname>
Alias of `<OTRS_TICKET_OWNER_UserLastname>`.

<OTRS_OWNER_UserLogin>
Alias of `<OTRS_TICKET_OWNER_UserLogin>`.

<OTRS_OWNER_UserPw>
Alias of `<OTRS_TICKET_OWNER_UserPw>`.

<OTRS_OWNER_ValidID>
Alias of `<OTRS_TICKET_OWNER_ValidID>`.

<OTRS_QUEUE>
Alias of `<OTRS_TICKET_Queue>`.

<OTRS_REDIRECT_TO>
Email address where the ticket is redirected to.

<OTRS_RESPONSIBLE_ChangeTime>
Alias of `<OTRS_TICKET_RESPONSIBLE_ChangeTime>`.

<OTRS_RESPONSIBLE_CreateTime>
Alias of `<OTRS_TICKET_RESPONSIBLE_CreateTime>`.

<OTRS_RESPONSIBLE_UserEmail>
Alias of `<OTRS_TICKET_RESPONSIBLE_UserEmail>`.

<OTRS_RESPONSIBLE_UserFirstname>
Alias of `<OTRS_TICKET_RESPONSIBLE_UserFirstname>`.

<OTRS_RESPONSIBLE_UserFullname>
Alias of `<OTRS_TICKET_RESPONSIBLE_UserFullname>`.

<OTRS_RESPONSIBLE_UserID>
Alias of `<OTRS_TICKET_RESPONSIBLE_UserID>`.

<OTRS_RESPONSIBLE_UserLastname>
Alias of `<OTRS_TICKET_RESPONSIBLE_UserLastname>`.

<OTRS_RESPONSIBLE_UserLogin>
Alias of `<OTRS_TICKET_RESPONSIBLE_UserLogin>`.

<OTRS_RESPONSIBLE_UserPw>
Alias of `<OTRS_TICKET_RESPONSIBLE_UserPw>`.

<OTRS_RESPONSIBLE_ValidID>
Alias of `<OTRS_TICKET_RESPONSIBLE_ValidID>`.

`<OTRS_TICKET_AccountedTime>`
Total time accounted in a ticket.

`<OTRS_TICKET_Age>`
Age of the ticket in absolute seconds.

`<OTRS_TICKET_ArchiveFlag>`
Archive status of the ticket. Returns `y` or `n`.

`<OTRS_TICKET_ChangeBy>`
ID of the agent who changed the ticket.

`<OTRS_TICKET_Changed>`
Date and time when the ticket was changed.

`<OTRS_TICKET_CreateBy>`
ID of the agent who created the ticket.

`<OTRS_TICKET_Created>`
Date and time when the ticket was created.

`<OTRS_TICKET_CustomerID>`
Customer ID of the ticket.

`<OTRS_TICKET_CustomerUserID>`
Customer user ID of the ticket.

`<OTRS_TICKET_DynamicField_X>`
Stored value of the dynamic field where X is the internal name of the dynamic field.

`<OTRS_TICKET_DynamicField_X_Value>`
Displayed value of the dynamic field where X is the internal name of the dynamic field.

`<OTRS_TICKET_EscalationDestinationDate>`
Date of escalation in date and time, e. g. *2021-08-02 18:00:00*.

`<OTRS_TICKET_EscalationDestinationIn>`
Escalation in human readable format, e. g. *1h 4m*.

`<OTRS_TICKET_EscalationDestinationTime>`
Date of escalation in UNIX time, e. g. *1627904516*.

`<OTRS_TICKET_EscalationResponseTime>`
Response time of escalation in UNIX time, e. g. *1627904516*.

`<OTRS_TICKET_EscalationSolutionTime>`
Solution time of escalation in UNIX time, e. g. *1627904516*.

`<OTRS_TICKET_EscalationTime>`
Seconds total till escalation, e. g. *3600*.

`<OTRS_TICKET_EscalationTimeWorkingTime>`
Seconds of working/service time till escalation, e. g. *1800*.

`<OTRS_TICKET_EscalationUpdateTime>`
Update time of escalation in UNIX time, e. g. *1627904516*.

`<OTRS_TICKET_FirstResponseTime>`
Seconds total till first response, e. g. *3600*.

`<OTRS_TICKET_FirstResponseTimeDestinationDate>`
Date of first response time in date and time, e. g. *2021-08-02 18:00:00*.

`<OTRS_TICKET_FirstResponseTimeDestinationTime>`
Date of first response time in UNIX time, e. g. *1627904516*.

`<OTRS_TICKET_FirstResponseTimeEscalation>`
If true, ticket is escalated.

`<OTRS_TICKET_FirstResponseTimeNotification>`
If true, notify - x% of escalation has reached.

`<OTRS_TICKET_FirstResponseTimeWorkingTime>`
Seconds of working/service time till first response, e. g. *1800*.

`<OTRS_TICKET_GroupID>`
Group ID of the queue of the ticket.

`<OTRS_TICKET_ID>`
ID of the ticket.

`<OTRS_TICKET_Lock>`
Lock status of the ticket.

`<OTRS_TICKET_LockID>`
Lock ID of the ticket.

`<OTRS_TICKET_Number>`
Number of the ticket.

`<OTRS_TICKET_Owner>`
Alias of `<OTRS_TICKET_OWNER_UserLogin>`.

`<OTRS_TICKET_OWNER_ChangeTime>`
Change time of the owner agent of the ticket.

`<OTRS_TICKET_OWNER_CreateTime>`
Creation time of the owner agent of the ticket.

`<OTRS_TICKET_OWNER_UserEmail>`
Email of the owner agent of the ticket.

`<OTRS_TICKET_OWNER_UserFirstname>`
First name of the owner agent of the ticket.

`<OTRS_TICKET_OWNER_UserFullname>`
Full name of the owner agent of the ticket.

`<OTRS_TICKET_OWNER_UserID>`
User ID of the owner agent of the ticket.

`<OTRS_TICKET_OWNER_UserLastname>`
Last name of the owner agent of the ticket.

`<OTRS_TICKET_OWNER_UserLogin>`
Login name of the owner agent of the ticket.

`<OTRS_TICKET_OWNER_UserPw>`
Password of the owner agent of the ticket. Masked and replaced by *xxx*.

`<OTRS_TICKET_OWNER_ValidID>`
Validation ID of the owner agent of the ticket.

`<OTRS_TICKET_OwnerID>`
ID of the owner agent of the ticket.

<OTRS_TICKET_Priority>
Priority of the ticket.

<OTRS_TICKET_PriorityID>
Priority ID of the ticket.

<OTRS_TICKET_Queue>
Queue of the ticket.

<OTRS_TICKET_QueueID>
Queue ID of the ticket.

<OTRS_TICKET_RealTillTimeNotUsed>
UNIX timestamp of pending time.

<OTRS_TICKET_Responsible>
Alias of `<OTRS_TICKET_RESPONSIBLE_UserLogin>`.

<OTRS_TICKET_RESPONSIBLE_ChangeTime>
Change time of the responsible agent of the ticket.

<OTRS_TICKET_RESPONSIBLE_CreateTime>
Creation time of the responsible agent of the ticket.

<OTRS_TICKET_RESPONSIBLE_UserEmail>
Email of the responsible agent of the ticket.

<OTRS_TICKET_RESPONSIBLE_UserFirstname>
First name of the responsible agent of the ticket.

<OTRS_TICKET_RESPONSIBLE_UserFullname>
Full name of the responsible agent of the ticket.

<OTRS_TICKET_RESPONSIBLE_UserID>
User ID of the responsible agent of the ticket.

<OTRS_TICKET_RESPONSIBLE_UserLastname>
Last name of the responsible agent of the ticket.

<OTRS_TICKET_RESPONSIBLE_UserLogin>
Login name of the responsible agent of the ticket.

<OTRS_TICKET_RESPONSIBLE_UserPw>
Password of the responsible agent of the ticket. Masked and replaced by *xxx*.

<OTRS_TICKET_RESPONSIBLE_ValidID>
Validation ID of the responsible agent of the ticket.

<OTRS_TICKET_ResponsibleID>
ID of the responsible agent of the ticket.

<OTRS_TICKET_Service>
Service of the ticket.

<OTRS_TICKET_ServiceID>
Service ID of the ticket.

<OTRS_TICKET_SLA>
Service level agreement of the ticket.

<OTRS_TICKET_SLAID>
Service level agreement ID of the ticket.

<OTRS_TICKET_SolutionTime>
Seconds total till solution, e. g. *3600*.

<OTRS_TICKET_SolutionTimeDestinationDate>
Date of solution time in date and time, e. g. *2021-08-02 18:00:00*.

<OTRS_TICKET_SolutionTimeDestinationTime>
Date of solution time in UNIX time, e. g. *1627904516*.

<OTRS_TICKET_SolutionTimeEscalation>
If true, ticket is escalated.

<OTRS_TICKET_SolutionTimeNotification>
If true, notify - x% of escalation has reached.

<OTRS_TICKET_SolutionTimeWorkingTime>
Seconds of working/service time till solution, e. g. *1800*.

<OTRS_TICKET_State>
State of the ticket.

<OTRS_TICKET_StateID>
State ID of the ticket.

<OTRS_TICKET_StateType>
State type of the ticket.

<OTRS_TICKET_TicketID>
Ticket ID of the ticket.

<OTRS_TICKET_TicketNumber>
Ticket number of the ticket.

<OTRS_TICKET_TimeUnit>
Alias of `<OTRS_TICKET_AccountedTime>`.

<OTRS_TICKET_Title>
Title of the ticket.

<OTRS_TICKET_Type>
Type of the ticket.

<OTRS_TICKET_TypeID>
Type ID of the ticket.

<OTRS_TICKET_UnlockTimeout>
Unlock timeout of the ticket.

<OTRS_TICKET_UntilTime>
Pending time in seconds.

<OTRS_TICKET_UpdateTime>
Seconds total till update, e. g. *3600*.

<OTRS_TICKET_UpdateTimeDestinationDate>
Date of update time in date and time, e. g. *2021-08-02 18:00:00*.

<OTRS_TICKET_UpdateTimeDestinationTime>
Date of update time in UNIX time, e. g. *1627904516*.

<OTRS_TICKET_UpdateTimeEscalation>
If true, ticket is escalated.

<OTRS_TICKET_UpdateTimeNotification>
If true, notify - x% of escalation has reached.

<OTRS_TICKET_UpdateTimeWorkingTime>
Seconds of working/service time till update, e. g. *1800*.

<OTRS_TOKEN>
Token for the lost password feature.

<OTRS_UserFirstname>
Alias of `<OTRS_NOTIFICATION_RECIPIENT_UserFirstname>`.

<OTRS_UserFullname>
Alias of `<OTRS_NOTIFICATION_RECIPIENT_UserFullname>`.

<OTRS_UserLastname>
Alias of `<OTRS_NOTIFICATION_RECIPIENT_UserLastname>`.

<OTRS_UserLogin>
Alias of `<OTRS_NOTIFICATION_RECIPIENT_UserLogin>`.

11.10.3 OTRS Screen Reference

The following reference lists the screens where the OTRS tags can be used.

Note: `<OTRS_AGENT>` and `<OTRS_CURRENT>` are synonym and they will be replaced by the data of the agent doing the action. For example in process management or generic agent where the agent can overwrite the `UserID` parameter, it do not has to be the current ticket owner.

Forward action
`<OTRS_TICKET_State>`

Merge Action
`<OTRS_MERGE_TO_TICKET>`

Ticket::Frontend::AutomaticMergeText
`<OTRS_TICKET_*>`

Generic notifications
`<OTRS_CONFIG_*>`

Appointment notifications

- `<OTRS_APPOINTMENT_*>`
- `<OTRS_CALENDAR_*>`
- `<OTRS_CONFIG_*>`
- `<OTRS_NOTIFICATION_*>`
- `<OTRS_USER*>`

Process management sequence flow actions and script task activities

- `<OTRS_AGENT_*>`
- `<OTRS_CONFIG_*>`
- `<OTRS_CUSTOMER_*>`
- `<OTRS_NOTIFICATION_*>`

- `<OTRS_TICKET_*>`

Ticket notifications

- `<OTRS_APPOINTMENT_*>`
- `<OTRS_CALENDAR_*>`
- `<OTRS_CONFIG_*>`
- `<OTRS_NOTIFICATION_*>`
- `<OTRS_USER*>`

Lost password

- `<OTRS_NEWPW>`
- `<OTRS_USER_*>`

Auto responses

- `<OTRS_AGENT_*>`
- `<OTRS_CONFIG_*>`
- `<OTRS_CUSTOMER_*>`
- `<OTRS_NOTIFICATION_*>`
- `<OTRS_TICKET_*>`

Salutation

- `<OTRS_CONFIG_*>`
- `<OTRS_CURRENT_*>`
- `<OTRS_CUSTOMER_DATA_*>`
- `<OTRS_OWNER_*>`
- `<OTRS_RESPONSIBLE_*>`
- `<OTRS_TICKET_*>`

Signature

- `<OTRS_CONFIG_*>`
- `<OTRS_CURRENT_*>`
- `<OTRS_CUSTOMER_DATA_*>`
- `<OTRS_OWNER_*>`
- `<OTRS_RESPONSIBLE_*>`
- `<OTRS_TICKET_*>`

Templates

- `<OTRS_AGENT_*>`
- `<OTRS_CONFIG_*>`
- `<OTRS_CURRENT_*>`
- `<OTRS_CUSTOMER_*>`
- `<OTRS_CUSTOMER_DATA_*>`

- `<OTRS_OWNER_*>`
- `<OTRS_RESPONSIBLE_*>`
- `<OTRS_TICKET_*>`

11.11 Include Custom Scripts

A modern web application has been built with security-first mindset in place. It contains several mechanisms to make sure that all code in the front end is provided directly by the OTRS framework.

In addition, a security standard called Content Security Policy headers is now leveraged by the built-in web server. It serves as a last line of defense and in case some unexpected code still makes it to the client, it will make sure that the user' s browser simply refuses to run it.

However, a use-case still exists in which even a modern web application might still need to run some additional code, provided it is vetted and included by the administrators. One example could be an external integrated chat solution, but could also be an inline code snippet used for external web analytics.

See also:

Including custom scripts can be done in the *System Configuration* with the following settings:

- `AgentFrontend::ExternalScripts`
- `AgentFrontend::InlineScripts`
- `ExternalFrontend::ExternalScripts`
- `ExternalFrontend::InlineScripts`
- `WebApp::Server::AdditionalOrigins`

11.11.1 Web Analytics Example

In this example, we outline the steps to include custom scripts used by a web analytics provider in the external interface application. They will be run each time the application is used in order to track and report the behavior of the user for later analysis.

We start with the provided code snippet below, from a third party provider.

```
<script async src="https://www.example.com/track/js?id=UA-XXXXXX-Y"></script>
<script>
   window.dataLayer = window.dataLayer || [];
   function track(){dataLayer.push(arguments);}
   track('js', new Date());

   track('config', 'UA-XXXXXX-Y');
</script>
```

First line of the snippet refers to an *external* resource which should be loaded by the `script` tag.

In the script block below, we have some *inline* code which is run immediately when the page is loaded.

Adding External Script Resources

First, we need to include the external script location in the appropriate configuration:

1. Go to the *System Configuration* screen.
2. Search for the setting `ExternalFrontend::ExternalScripts`.
3. Click on the setting in order to edit it.
4. In case a value is already present, click on the plus button. Otherwise, proceed below.
5. Enter the full location of the *external* resource in the text field. For example: `https://www.example.com/track/js?id=UA-XXXXXX-Y`.
6. Click on the check mark in order to save the modified setting.
7. Deploy the modified system configuration.

Adding Inline Code Snippets

Next, we have to also include the inline part of the original snippet:

1. Go to the *System Configuration* screen.
2. Search for the setting `ExternalFrontend::InlineScripts`.
3. Click on the setting in order to edit it.
4. In case a value is already present, click on the plus button. Otherwise, proceed below.
5. Enter the full code snippet in the text area, minus any `script` tags. For example:

```
window.dataLayer = window.dataLayer || [];
function track(){dataLayer.push(arguments);}
track('js', new Date());

track('config', 'UA-XXXXXX-Y');
```

6. Click on the check mark in order to save the modified setting.
7. Deploy the modified system configuration.

Rebuilding the Application

In order to apply the changes, we also need to rebuild the application. Drop down to shell, and execute the following command:

```
bin/otrs.WebServer.pl --deploy-assets
```

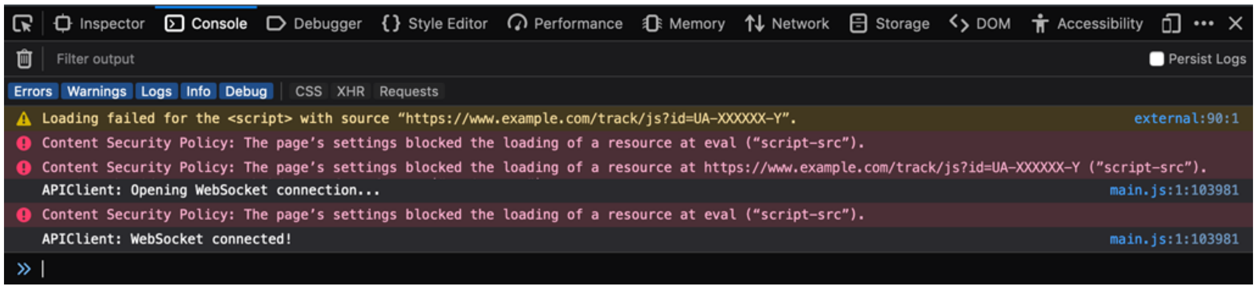
Whitelisting Additional Origins in the Security Headers

If you now try to access the external interface application, you will be able to verify that the scripts are included in the code. However, your browser will probably block access to all *inline* and *external* resources, therefore the code might fail with some errors.

This behavior is by design, since external resources can only be loaded if they are specifically whitelisted in the *Content Security Policy* headers.

To check for blocked code, please use suitable web browser inspection tools. In our example, we will be using Mozilla Firefox and its web console available via *Tools* → *Web Developer* → *Web Console* menu item, or via the *F12* shortcut key.

For the example code snippet, you might receive following errors in the console when the application is accessed:



```
Loading failed for the <script> with source “https://www.example.com/track/js?id=UA-XXXXXX-Y”.          external:90:1
Content Security Policy: The page’s settings blocked the loading of a resource at eval (“script-src”).
Content Security Policy: The page’s settings blocked the loading of a resource at https://www.example.com/track/js?id=UA-XXXXXX-Y (“script-src”).
APIClient: Opening WebSocket connection...                                                              main.js:1:103981
Content Security Policy: The page’s settings blocked the loading of a resource at eval (“script-src”).
APIClient: WebSocket connected!                                                                         main.js:1:103981
```

Fig. 14: Browser Console Errors and Warnings

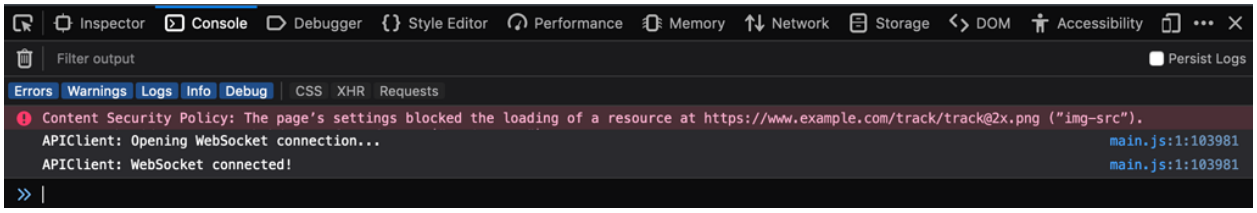
From the console errors we can see that the external script resource was prevented from being loaded (lines 1 and 3). In addition to that, two evaluation calls were also blocked (lines 2 and 5). All errors reference a *Content Security Policy* rule under the name of `script-src`, which signals script resources.

We need to add both the external resource and the evaluation calls to the additional origins list of the *Content Security Policy* headers:

1. Go to the *System Configuration* screen.
2. Search for the setting `WebApp::Server::AdditionalOrigins`.
3. Click on the setting in order to edit it.
4. In case a value for `script-src` is already present, click on the plus button next to it. Otherwise, proceed below.
5. Enter the domain part only of the blocked resource in the text field. For example: `https://www.example.com`. This allows the external resource to be loaded.
6. Click on the plus button next to the field, so another value is added.
7. Enter the following directive in the new field, including the quotes: `'unsafe-eval'`. This allows the evaluation calls to be executed.
8. Click on the check mark in order to save the modified setting.
9. Deploy the modified system configuration.

There is no need to rebuild the external interface application at this point, as the additional origins configuration should be immediately in effect.

If you reload the external interface application, you might get some additional errors. In our example, it might be the following:



```
Content Security Policy: The page's settings blocked the loading of a resource at https://www.example.com/track/track@2x.png ("img-src").
APIClient: Opening WebSocket connection...                                                        main.js:1:103981
APIClient: WebSocket connected!                                                                   main.js:1:103981
```

Fig. 15: Additional Browser Console Errors and Warnings

This error points that an additional resource that was also blocked, an image at a specific location (line 1). We can deduce this via the name of the *Content Security Policy* rule `img-src`, which references an image resource. In order to add it to the whitelist, try the following:

1. Go to the *System Configuration* screen.
2. Search for the setting `WebApp::Server::AdditionalOrigins`.
3. Click on the setting in order to edit it.
4. In case a value for `img-src` is already present, click on the plus button next to it. Otherwise, proceed below.
5. Enter the domain part only of the blocked resource in the text field. For example: `https://www.example.com`. This allows the external image resource to be loaded.
6. Click on the check mark in order to save the modified setting.
7. Deploy the modified system configuration.

Try again to reload the external interface application and check if there are more errors. If not, your scripts are now probably working as expected.

Unfortunately, it is not possible to predict what kind of resources your scripts might be requiring. But, no worries, you can whitelist most of them, just make sure to follow the trail of hints shown in the browser console log. Find a corresponding header rule in the configuration and update it accordingly.

Note: Some resources might only be requested by others, hence several iterations might be needed until everything is configured properly.

Warning: Whitelisting external resources opens potential security risks in your OTRS application! Only allow those resources that you are sure are not malicious and come from reputable sources. Keep in mind that if something is secure today, does not mean it will be tomorrow. Stay safe!

11.12 Allow Program Safe to Run

External programs to be run by OTRS and directories where OTRS can read from or write to are blocked by default due to security reasons. If you would like to use an external program in scripts, the `'PROGRAM' is not safe to run` message appears in the log file where `PROGRAM` is the name of program.

There is no graphical user interface to add additional program or directory to the allow list. A system administrator who has file system access has to add the allowed programs and directories to configuration file.

To add programs and directories to the allow list:

1. Open `$OTRS_HOME/Kernel/Config.pm`.
2. Add the following settings:

```
$Self->{'SystemConfiguration::ValueType::SystemCommand::BinaryWhiteList'}->{'999-
↪Custom'} = [
   'program_name',
];

$Self->{'SystemConfiguration::ValueType::SystemCommand::DirectoryWhiteList'}->{
↪'999-Custom'} = [
   '/path/to/directory',
];
```

The first setting lists the base names of allowed commands which can be run in a system command. The second setting lists the absolute paths of allowed directories where the system can read from or write to during a system command redirect.

In the example above the `999-Custom` is a unique identifier for expanding the allow lists.

Warning: Do not use the same unique identifier anywhere else in the system in the scope of this setting. Otherwise a previous allow list will be overwritten!

11.13 Custom Language File

To use this tutorial you need file system and command line access to the server where OTRS is running.

Note: This feature is only available to *On-Premise* customers. If you are a *Managed* customer, this feature is taken care of by the *Customer Solutions Team* in **OTRS**. Please contact us via support@otrs.com or in the OTRS Portal.

OTRS can be localized into other languages than English. The language files are stored in `.pm` files under the `$OTRS_HOME/Kernel/Language/` folder. There are three types of language files:

Framework language file

The files named based on a language code and optional dialect code separated by an underscore character (for example `de.pm` for German language file or `en_CA.pm` for the Canadian English language file) contain the translation of the core system. These files are part of the released package and they are being updated time to time with the translations made on OTRS Translation Portal.

Do not edit these files manually. They will be overwritten during the next version update.

Package language file

Packages like ITSM packages or features have own language files using the following naming convention: language code and optional dialect code followed by the package name separated by underscore characters (for example `de_OTRSServiceManagement.pm` for German language file or `en_CA_OTRSServiceManagement.pm` for the Canadian English language file of the `OTRSServiceManagement` package).

Do not edit these files manually. They will be overwritten during the next version update.

Custom language file

Custom language files are not part of the system. You have to create it manually and put it to the language folder. The name of the custom language file should be a language code and optional dialect code followed by the word `Custom` separated by underscore character (for example `de_Custom.pm` for German custom language file or `en_CA_Custom.pm` for the Canadian English custom language file). Since this file is not part of the release package, this will not be overwritten during version updates.

In a system where all kind of language files are present, the framework language file is loaded first then the package language files are loaded in alphabetical order and finally the custom language file is loaded. This ensures that the translations from the custom language file can override any previous translations.

There are two kind of usage of the custom language file:

1. To add translation to system resources created or changed after the installation.
2. To customize the existing translation.

During the configuration phase of the system the created or modified resources can contain new translatable strings like names, descriptions or any other attributes of the following resources:

- *Calendars*
- *FAQ Category*
- *System Configuration*
- *Dynamic Fields*
- *Process Management*
- *Priorities*
- *Services*
- *Service Level Agreements*
- *States*
- *Types*
- *Customers*
- *Customer Users*

It is recommended to always use English texts for the resources above and translate them using the custom language file even if the system is designed to be used in a specific language.

When the resources have been added to the system, you have to collect the translatable strings manually and you have to add them to the custom language file.

This example shows a system where a new drop-down dynamic field with translatable options and a new ticket state have been added. Additionally an existing translation has been changed.

The German custom language file should look like this:

```
# --
# Copyright (C) YEAR, https://your-company.com/
# --
# This software comes with ABSOLUTELY NO WARRANTY. For details, see
# the enclosed file COPYING for license information (GPL). If you
# did not receive this file, see https://www.gnu.org/licenses/gpl-3.0.txt.
# --

package Kernel::Language::de_Custom;
```

(continues on next page)

(continued from previous page)

```
use strict;
use warnings;
use utf8;

sub Data {
    my $Self = shift;

    # School dynamic field options
    $Self->{Translation}->{'middle school'} = 'Mittelschule';
    $Self->{Translation}->{'high school'} = 'Oberschule';
    $Self->{Translation}->{'University'} = 'Universität';

    # Ticket state
    $Self->{Translation}->{'closed with workaround'} = 'provisorisch geschlossen';

    # Override existing translation
    $Self->{Translation}->{'Internal News'} = 'Firmennachrichten';

    push @{ $Self->{JavaScriptStrings} // [] }, (
    );

    return;
}

1;
```

The first section contains copyright and license information. Since OTRS is licensed under GNU GPL version 3 it is recommended to apply the same license to the custom language file. Do not forget to change the year and the copyright holder in the second line.

The next section contains the package path which should be the relative path from the OTRS home folder and the name of the custom language file without the file extension. In the example above this is `Kernel::Language::de_Custom`. If you create the custom language file for another language you have to change the language prefix in the last segment of the path.

The most important section is the translation entries. Each entry contains the English string as key and the translation of the target language as value. It is recommended to group the entries and add a comment (a line starting with `#` character) which explains where the strings come from. This facilitates the maintenance of the custom language file.

It is possible to override the existing translation of the framework. In our example we override the German translation of `Internal News` with `Firmennachrichten` instead of the original `Interne Nachrichten`. To do this you have to search for the original string in the German language file (`de.pm`) then copy the original string to the custom language file and add different translation.

Any character can be in the strings but the apostrophe character has to be escaped as `\'` like `don\'t` because this character is used as enclosing character.

The language files for the new interface are now part of the built application (static JSON). When you add a custom language file to the file system, you need to rebuild the application for the change to be considered. To trigger the rebuild, restart the server with the `--deploy-assets` option:

```
otrs> /opt/otrs/bin/otrs.WebServer.pl --deploy-assets
```

During the build process, the language files will be refreshed and will take any `*_Custom.pm` into account.

11.14 Database Table Optimization

Note: This tutorial is only for on-Premise customers. The **Customer Solutions** team takes care of the database of managed customers.

Due to the nature of database tables they can be fragmented over the time. The fragmentation is caused by updating the values in the records. This is normal, but a fragmented table takes more storage and the queries can run slower.

This is a database operation topic and not related to OTRS strictly, but this tutorial can help to identify and solve the problem if the database is getting bigger and bigger.

To optimize the database tables:

1. Check the status of the tables using the SQL statement below.

```
SELECT
  TABLE_NAME,
  ROUND(DATA_LENGTH/1024/1024, 2) AS DATA_LENGTH,
  ROUND(INDEX_LENGTH/1024/1024, 2) AS INDEX_LENGTH,
  ROUND(DATA_FREE/1024/1024, 2) AS DATA_FREE,
  (data_free/(index_length+data_length)) AS FRAG_RATIO
FROM information_schema.tables
WHERE TABLE_SCHEMA = 'otrs'
AND DATA_FREE > 0
ORDER BY frag_ratio DESC;
```

2. Review and analyze the output which should look like this.

```
+---------------------------------+-------------+--------------+-----------+-------
↪-----+
| TABLE_NAME                      | DATA_LENGTH | INDEX_LENGTH | DATA_FREE | FRAG_
↪RATIO |
+---------------------------------+-------------+--------------+-----------+-------
↪-----+
| communication_log_object_entry |        0.02 |         0.03 |     10.00 |   213.
↪3333 |
| user_config                     |        1.19 |         0.05 |      4.00 |     3.
↪2405 |
| web_upload_cache                |        7.31 |         0.00 |     22.00 |     3.
↪0085 |
| article_data_mime               |        1.52 |         0.06 |      4.00 |     2.
↪5347 |
| sysconfig_deployment            |       10.02 |         0.02 |     16.00 |     1.
↪5950 |
| xml_storage                     |        1.50 |         1.92 |      4.00 |     1.
↪1689 |
| mail_queue                      |        5.52 |         0.05 |      4.00 |     0.
↪7191 |
| virtual_fs_db                   |        1.52 |         0.02 |      1.00 |     0.
↪6531 |
| sysconfig_default               |        6.41 |         0.31 |      4.00 |     0.
↪5953 |
| sysconfig_default_version       |       10.52 |         0.45 |      5.00 |     0.
↪4558 |
| package_repository              |       33.52 |         0.03 |     11.00 |     0.
```

(continues on next page)

(continued from previous page)

```
↪3279 |
+--------------------------------+-------------+--------------+-----------+-------
↪-----+
```

The output shows that the fragmentation ratio of the `communication_log_object_entry` table is 213 and the table contains 10 MB empty data.

3. If the fragmentation ratio is higher than 50% (0.5 in the `FRAG_RATIO` column) you should consider to optimize the table using the `OPTIMIZE` SQL statement.

```
OPTIMIZE TABLE otrs.communication_log_object_entry;
```

4. Verify if the table is not fragmented anymore. Use the same SQL statement as used in step 1.

```
+---------------------------+-------------+--------------+-----------+------------
↪+
| TABLE_NAME                | DATA_LENGTH | INDEX_LENGTH | DATA_FREE | FRAG_RATIO␣
↪|
+---------------------------+-------------+--------------+-----------+------------
↪+
| user_config               |        1.19 |         0.05 |      4.00 |     3.2405␣
↪|
| web_upload_cache          |        7.31 |         0.00 |     22.00 |     3.0085␣
↪|
| article_data_mime         |        1.52 |         0.06 |      4.00 |     2.5347␣
↪|
| sysconfig_deployment      |       10.02 |         0.02 |     16.00 |     1.5950␣
↪|
| xml_storage               |        1.50 |         1.92 |      4.00 |     1.1689␣
↪|
| mail_queue                |        5.52 |         0.05 |      4.00 |     0.7191␣
↪|
| virtual_fs_db             |        1.52 |         0.02 |      1.00 |     0.6531␣
↪|
| sysconfig_default         |        6.41 |         0.31 |      4.00 |     0.5953␣
↪|
| sysconfig_default_version |       10.52 |         0.45 |      5.00 |     0.4558␣
↪|
| package_repository        |       33.52 |         0.03 |     11.00 |     0.3279␣
↪|
+---------------------------+-------------+--------------+-----------+------------
↪+
```

5. Repeat step 3 for the other tables if needed. You can optimize more tables at the same time.

```
OPTIMIZE TABLE otrs.communication_log_object_entry, otrs.web_upload_cache, otrs.
↪package_repository;
```

See also:

Read the official manual of MySQL for detailed information.

11.15 Process Examples

This chapter assumes that you already know basically how to create and configure process items and how to place a process path on the drawing area. Please read the *Process Management* chapter for more information or participate in an OTRS administrator training.

This chapter contains two simple process examples from practice. You can use them yourself to apply your learned knowledge.

11.15.1 Application for Leave

This sample process describes a possible handling for a leave request. The agent creates a process ticket and enters the data. The ticket is then first sent to the head of department. After the approval of the head of department, the ticket is sent to the queue *HR Department*. If the department head does not give an approval, a ticket with the rejection is sent to the agent and the process ends.

The HR department receives the ticket with the approval of the head of department. Now it is checked if the agent still has enough free days available. If yes, a ticket is sent to the agent with the information *Confirmed* and the process ends. If there are not enough days available, a ticket with the information *Rejected* is sent to the agent and the process ends.

Enable the following *System Configuration* settings:

- `Ticket::Service`
- `Ticket::Type`

Create the following *Queues*:

- Administration
- HR Department

Create the following *Services*:

- Application for Leave

Create the following *Types*:

- Internal Request

Create the following *Dynamic Fields*:

Object	Type	Name	Label
Ticket	Text	`ProcessState`	Process State
Ticket	Date	`VacationStart`	Vacation Start
Ticket	Date	`VacationEnd`	Vacation End
Ticket	Text	`UsedVacationDays`	Days of Vacation Used
Ticket	Text	`OpenVacationDays`	Days of Vacation Open
Ticket	Text	`EmergencyTelephone`	Emergency Phone Number

Set the following *System Configuration* settings:

- `AgentFrontend::TicketDetailView::Widget::BusinessProcessInformation###DynamicField`
  - `ProcessState` → *1 - Enabled*
  - `VacationStart` → *1 - Enabled*
  - `VacationEnd` → *1 - Enabled*

  - `UsedVacationDays` → *1 - Enabled*
  - `OpenVacationDays` → *1 - Enabled*
  - `EmergencyTelephone` → *1 - Enabled*

Create a new process named *Application for Leave*.

Create the following user task activity dialogs with fields:

- *Recording the demand*
  - `CustomerID` → Display: *Show Field As Mandatory*
  - `DynamicField_ProcessState` → Default value: *approval*, Display: *Do not show Field*
  - `DynamicField_OpenVacationDays` → Display: *Show Field As Mandatory*
  - `DynamicField_UsedVacationDays` → Display: *Show Field As Mandatory*
  - `DynamicField_VacationStart` → Display: *Show Field As Mandatory*
  - `DynamicField_VacationEnd` → Display: *Show Field As Mandatory*
  - `Queue` → Default value: *Administration*, Display: *Do not show Field*
  - `Service` → Default value: *Application for Leave*, Display: *Do not show Field*
  - `State` → Default value: *open*, Display: *Do not show Field*
  - `Type` → Default value: *Internal Request*, Display: *Do not show Field*
- *Approved*
  - `DynamicField_ProcessState` → Default value: *approved*, Display: *Do not show Field*
- *Approval denied*
  - `Article` → Communication Channel: *OTRS*, Time Units: *Do not show Field*, Display: *Show Field*
  - `DynamicField_ProcessState` → Default value: *approval denied*, Display: *Do not show Field*
  - `State` → Default value: *closed successful*, Display: *Do not show Field*
- *Application for leave processed*
  - `Article` → Communication Channel: *OTRS*, Is visible to customer: no, Display: *Show Field As Mandatory*
  - `DynamicField_OpenVacationDays` → Display: *Show Field As Mandatory*
  - `DynamicField_UsedVacationDays` → Display: *Show Field As Mandatory*
  - `DynamicField_ProcessState` → Default value: *confirmation received*, Display: *Do not show Field*
  - `State` → Default value: *closed successful*, Display: *Do not show Field*
- *Not enough vacation days*
  - `DynamicFieldProcessState` → Default value: *not enough vacation days*, Display: *Do not show Field*
  - `State` → Default value: *closed unsuccessful*, Display: *Do not show Field*

Create the following sequence flows:

- *DynamicField_ProcessState = approval*

  - Name: `DynamicField_ProcessState`
  - Type: *String*
  - Value: *approval*
- *DynamicField_ProcessState = approval denied*
  - Name: `DynamicField_ProcessState`
  - Type: *String*
  - Value: *approval denied*
- *DynamicField_ProcessState = approved*
  - Name: `DynamicField_ProcessState`
  - Type: *String*
  - Value: *approved*
- *DynamicField_ProcessState = confirmation received*
  - Name: `DynamicField_ProcessState`
  - Type: *String*
  - Value: *confirmation received*
- *DynamicField_ProcessState = not enough vacation days*
  - Name: `DynamicField_ProcessState`
  - Type: *String*
  - Value: *not enough vacation days*
- *DynamicField_ProcessManagementActivityStatus = 0 (successful)*
  - Name: `DynamicField_ProcessManagementActivityStatus`
  - Type: *String*
  - Value: *0*

Create the following sequence flow actions:

- *Set ticket title* "*Application for leave: EmployeeName*"
  - Module: `TicketTitleSet`

  Click on the *Save* then the *Configure* button and add the following parameters:
  - Key: `Title`, value: *Application for leave: <OTRS_CUSTOMER_REALNAME>*.

Create the following user task activities:

- *Recording the application*

  Assign the user task activity dialog *Recording the demand*.
- *Approval*

  Assign the user task activity dialogs *Approval denied* and *Approved*.
- *Confirmation of HR department*

  Assign the user task activity dialogs *Application for leave processed* and *Not enough vacation days*.

- *Process complete*

Create the following script task activities:

- *Send ticket to queue “HR Department”*
    - Script: `TicketQueueSet`

  Click on the *Save* then the *Configure* button and add the following parameters:
    - Key: `Queue`, value: *HR Department*
    - Key: `UserID`, value: *1*
- *Send notification of confirmation to employee*
    - Script: `TicketArticleCreate`

  Click on the *Save* then the *Configure* button and add the following parameters:
    - Key: `Body`, value: *Good News! You can take your free time!*
    - Key: `CommunicationChannel`, value: *Internal*
    - Key: `ContentType`, value: *text/html;charset=UTF-8*
    - Key: `HistoryComment`, value: *Send note*
    - Key: `HistoryType`, value: *AddNote*
    - Key: `IsVisibleForCustomer`, value: *1*
    - Key: `SenderType`, value: *agent*
    - Key: `Subject`, value: *Your application for leave was successfully processed*
- *Send notification of denial to employee*
    - Script: `TicketArticleCreate`

  Click on the *Save* then the *Configure* button and add the following parameters:
    - Key: `Body`, value: *Your <OTRS_TICKET_Service> was denied. Please talk to your manager.*
    - Key: `CommunicationChannel`, value: *Internal*
    - Key: `ContentType`, value: *text/html;charset=UTF-8*
    - Key: `HistoryComment`, value: *Send note*
    - Key: `HistoryType`, value: *AddNote*
    - Key: `IsVisibleForCustomer`, value: *1*
    - Key: `SenderType`, value: *agent*
    - Key: `Subject`, value: *Approval denied*

Create the process path as follows:

Double click on the *DynamicField_ProcessState = approval* sequence flow and assign the *Set ticket title “Application for leave: EmployeeName”* sequence flow action.

Set the process state to *Active* and save it.

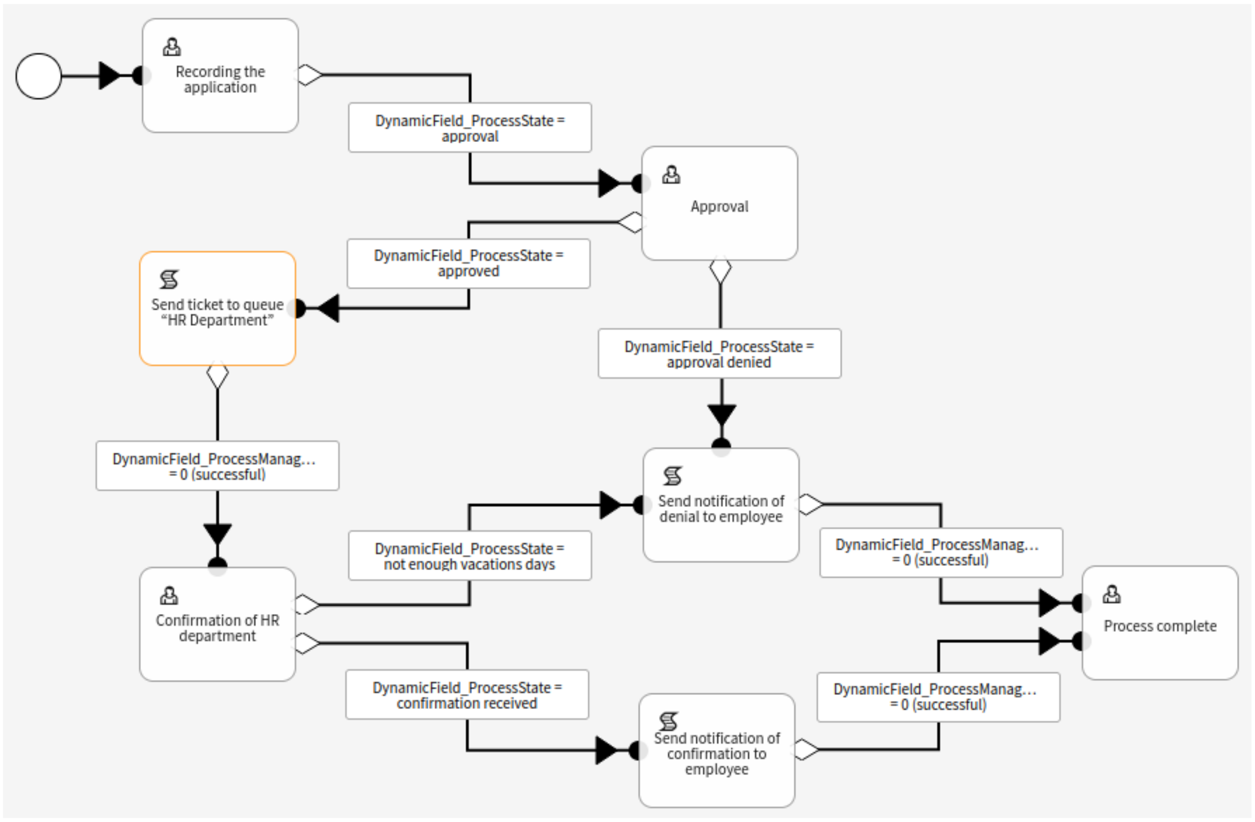

Recording the application
DynamicField_ProcessState = approval
Approval
DynamicField_ProcessState = approved
Send ticket to queue "HR Department"
DynamicField_ProcessState = approval denied
DynamicField_ProcessManag… = 0 (successful)
Send notification of denial to employee
DynamicField_ProcessState = not enough vacations days
DynamicField_ProcessManag… = 0 (successful)
Confirmation of HR department
Process complete
DynamicField_ProcessState = confirmation received
DynamicField_ProcessManag… = 0 (successful)
Send notification of confirmation to employee

Fig. 16: Application for Leave Process on Canvas

11.15.2 Defect Handling

This sample process describes a possible procedure for software defect handling. First, the possible software defect is recorded and classified via ticket. If it is a software defect, the ticket is sent to the queue *Development*. If it is not a software defect, the process ends at this point.

The ticket is now processed in development and runs through the *Troubleshooting* activity. If the defect has been fixed, the process ends.

If troubleshooting is unsuccessful, the ticket is returned to the *Development* queue for reprocessing.

Install the following packages:

- OTRSServiceManagement
- OTRSConfigurationManagement
- OTRSDynamicFieldCI

Enable the following *System Configuration* settings:

- `Ticket::Service`
- `Ticket::Type`

Create the following *Queues*:

- Technical Support
- Development

Create the following *Services*:

- Software

Create the following *States*:

- ready for release → State type: *open*

Create the following *Types*:

- Defect

Create the following *Dynamic Fields*:

Object	Type	Name	Label	Possible values
Ticket	Text	`ProcessState`	Process State	
Ticket	Configuration item	`AffectedSoftware`	Affected Software	
Ticket	Multiselect	`AffectedVersion`	Affected Version	
Ticket	Multiselect	`AffectedComponent`	Affected Component	
Ticket	Dropdown	`Severity`	Severity	• 1 - low • 2 - minor • 3 - medium • 4 - major • 5 - critical
Ticket	Dropdown	`Likelihood`	Likelihood	• very low • low • medium • high • very high
Ticket	Text Area	`Tasks`	Tasks	

Set the following *System Configuration* settings:

- *AgentFrontend::TicketDetailView::Widget::BusinessProcessInformation###DynamicField*
  - `ProcessState` → *1 - Enabled*
  - `AffectedSoftware` → *1 - Enabled*
  - `AffectedVersion` → *1 - Enabled*
  - `AffectedComponent` → *1 - Enabled*
  - `Severity` → *1 - Enabled*
  - `Likelihood` → *1 - Enabled*
  - `Tasks` → *1 - Enabled*

Create a new process named *Defect Handling*.

Create the following user task activity dialogs with fields:

- *Report new software defect*
  - `CustomerID` → Display: *Show Field As Mandatory*
  - `DynamicField_AffectedSoftware` → Display: *Show Field As Mandatory*
  - `DynamicField_AffectedComponent` → Display: *Show Field As Mandatory*
  - `DynamicField_AffectedVersion` → Display: *Show Field As Mandatory*
  - `DynamicField_Severity` → Display: *Show Field As Mandatory*
  - `DynamicField_Likelihood` → Display: *Show Field As Mandatory*
  - `Article` → Communication Channel: *OTRS*, Display: *Show Field As Mandatory*
  - `DynamicField_ProcessState` → Default value: *defect reported*, Display: *Do not show Field*

  - `Queue` → Default value: *Technical Support*, Display: *Do not show Field*
  - `Service` → Default value: *Software*, Display: *Do not show Field*
  - `State` → Default value: *open*, Display: *Do not show Field*
  - `Type` → Default value: *Defect*, Display: *Do not show Field*
- *Defect classification*
  - `DynamicField_Severity` → Display: *Show Field As Mandatory*
  - `DynamicField_Likelihood` → Display: *Show Field As Mandatory*
  - `Priority` → Display: *Show Field As Mandatory*
  - `Article` → Communication Channel: *OTRS*, Display: *Show Field*
  - `DynamicField_ProcessState` → Default value: *defect classified*, Display: *Do not show Field*
- *Planning of defect handling*
  - `DynamicField_Tasks` → Display: *Show Field As Mandatory*
  - `Owner` → Display: *Show Field As Mandatory*
  - `DynamicField_ProcessState` → Default value: *planning finished*, Display: *Do not show Field*
- *No Defect*
  - `DynamicField_ProcessState` → Default value: *no defect*, Display: *Do not show Field*
  - `State` → Default value: *closed unsuccessful*, Display: *Do not show Field*
- *Defect solved*
  - `Article` → Communication Channel: *OTRS*, Display: *Show Field As Mandatory*
  - `DynamicField_ProcessState` → Default value: *defect solved*, Display: *Do not show Field*
  - `State` → Default value: *ready for release*, Display: *Do not show Field*
- *Defect not solved*
  - `Article` → Communication Channel: *OTRS*, Display: *Show Field As Mandatory*
  - `DynamicField_ProcessState` → Default value: *defect not solved*, Display: *Do not show Field*

Create the following sequence flows:

- *DynamicField_ProcessState = defect reported*
  - Name: `DynamicField_ProcessManagementActivityStatus`
  - Type: *String*
  - Value: *0*
- *DynamicField_ProcessState = defect classified*
  - Name: `DynamicField_ProcessState`
  - Type: *String*
  - Value: *defect classified*
- *DynamicField_ProcessState = planning finished*
  - Name: `DynamicField_ProcessState`
  - Type: *String*

    - Value: *planning finished*

- *DynamicField_ProcessState* = *defect solved*
    - Name: `DynamicField_ProcessState`
    - Type: *String*
    - Value: *defect solved*
- *DynamicField_ProcessState* = *defect not solved*
    - Name: `DynamicField_ProcessState`
    - Type: *String*
    - Value: *defect not solved*
- *DynamicField_ProcessState* = *no defect*
    - Name: `DynamicField_ProcessState`
    - Type: *String*
    - Value: *no defect*
- *DynamicField_ProcessManagementActivityStatus* = *0 (successful)*
    - Name: `DynamicField_ProcessManagementActivityStatus`
    - Type: *String*
    - Value: *0*

Create the following sequence flow actions:

- *Set ticket title* “*Defect reported*”
    - Module: `TicketTitleSet`

    Click on the *Save* then the *Configure* button and add the following parameters:

    - Key: `Title`, value: *Defect reported*
- *Set linked CI in operational state*
    - Module: `ITSMConfigItemDataPush`

    Click on the *Save* then the *Configure* button and add the following parameters:

    - Main search parameters:
        * Class: *Software*
        * Incident State: *Operational*
        * Link Type: *Relevant to*
    - Additional configuration items matching conditions:
        * Key: `Limit`, value: *1*
    - Linked configuration items attributes to be updated:
        * Key: `IncidentStateID`, value: *3*

Create the following user task activities:

- *Report Software Defect*

    Assign the user task activity dialog *Report new software defect*.

- *Classification*

  Assign the user task activity dialogs *Defect classification* and *No Defect*.

- *Planning*

  Assign the user task activity dialog *Planning of defect handling*.

- *Defect Troubleshooting*

  Assign the user task activity dialogs *Defect solved* and *Defect not solved*.

- *Process complete*

Create the following script task activities:

- *Send ticket to queue "Development"*

  – Script: `TicketQueueSet`

  Click on the *Save* then the *Configure* button and add the following parameters:

  – Key: `Queue`, value: *Development*

  – Key: `UserID`, value: *1*

Create the process path as follows:

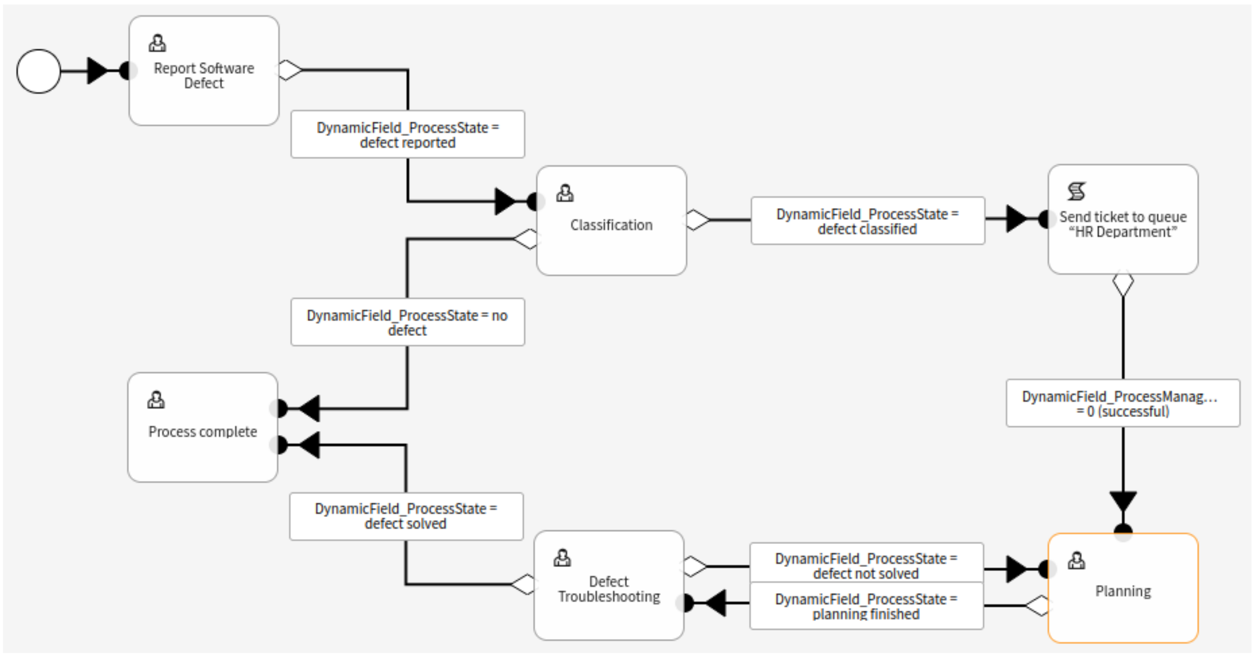

Report Software Defect
DynamicField_ProcessState = defect reported
Classification
DynamicField_ProcessState = defect classified
Send ticket to queue "HR Department"
DynamicField_ProcessState = no defect
DynamicField_ProcessManag… = 0 (successful)
Process complete
DynamicField_ProcessState = defect solved
Defect Troubleshooting
DynamicField_ProcessState = defect not solved
DynamicField_ProcessState = planning finished
Planning

Fig. 17: Defect Handling Process on Canvas

Double click on the *DynamicField_ProcessState = defect reported* sequence flow and assign the *Set ticket title "Defect reported"* sequence flow action.

Double click on the *DynamicField_ProcessState = defect solved* sequence flow and assign the *Set linked CI in operational state* sequence flow action.

Set the process state to *Active* and save it.

11.16 Cookies and Local Storage

The applications need cookies and local storage for proper working. Both the cookie and the local storage are stored on the user' s computer. This storage does not include personal data, they are only needed for the operation of the application.

The only cookie stored when the user visits the external interface is `AuthenticationCustomer`. This cookie stores the session ID of the logged in customer user and has a lifetime of 16 hours.

Additionally, the following key-value pairs are stored in the web browser' s local storage:

```
Plugins/Store/PublicChat/External/MutationsSharerStorage = {
    "type": "Plugins/Store/PublicChat/External/publicUuid",
    "payload": {
        "publicUuid": "<uuid-string>"
    }
}
Plugins/Store/LoggerOptions/External = {
    "logLevel": "info",
    "logRecord": false
}
Plugins/Store/PublicChat/External/MutationsSharerNotification = notification-
↪<notification-id>
Apps/External/Plugins/Store = {
    "accessToken": null,
    "deviceUuid": null,
    "isLanguageSet": false,
    "language": "en",
    "passwordExpirySoonWarning": null
}
Apps/External/Plugins/Store/MutationsSharerStorage = {
    "type": "language",
    "payload": {
        "language": "en"
    }
}
Apps/External/Plugins/Store/PrivacyNotice = {
    "isNoticeVisible": true
}
Apps/External/Plugins/Store/MutationsSharerNotification = notification-<notification-
↪id>
Plugins/Store/PerlProfiler/External = {
    "perlProfiler": {}
}
Plugins/Store/PublicChat/External = {
    "publicUuid": "<uuid-string>"
}
```

The UUID values in the local storage are auto-generated when the user visits the external interface for the first time. The local storage has no expiration date and stores data used for the application itself. Additional notification entries might get spawned during the usage, and are used to sync the data state across multiple instances of the application.

11.17 Credit Card Filter

This feature offers options for the responsible handling of sensitive data, like credit card numbers, in your **OTRS** system. With the help of this function, it is possible to hide credit card numbers for new tickets and articles, as well as articles already stored in the system. Additionally, flexibly configurable warning messages for the credit card number can be automatically set and showed.

This mechanism for recognizing credit cards works for cards with 13, 15 or 16 digits. All articles of a ticket in the system will be scanned for valid credit card numbers before storing. Then, the majority of the digits will be encoded before storage in the system.

Benefits

- Improved data protection.
- Less security risks for credit card data.

Target Groups

- Customer service
- Finance
- Sales
- Order management
- Complaint management

11.17.1 Legacy Credit Card Masking

A console command exists to treat already stored credit card numbers in the system. This command will mask any unmasked valid credit card number in the article database table.

Note: This feature is only available to *On-Premise* customers. If you are a *Managed* customer, this feature is taken care of by the *Customer Solutions Team* in **OTRS**. Please contact us via support@otrs.com or in the OTRS Portal.

To mask the existing credit card numbers:

1. Make sure that setting `OTRSCreditCardFilter::ActiveMaskEnabled` is not enabled.
2. Create one or more tickets with valid credit card information.

```
Issuing Network,Card Number
JCB,3528988095245935
JCB,3112606824580636
JCB,3096030869937728
JCB,3112437499296450
JCB,3096010732100407
JCB,3528461498782367
JCB,3112892137191440
JCB,3088814635323630
```

3. Open the ticket detail view to make sure that the credit card numbers are not masked.
4. Execute the following command in the command line to mask the credit card numbers:

```
otrs> /opt/otrs/bin/otrs.Console.pl Maint::Ticket::MaskCreditCard --restart yes
```

5. Refresh the ticket detail view. The article the body will shown as:

```
Issuing Network,Card Number
JCB,352898xxxxxx5935
JCB,311260xxxxxx0636
JCB,309603xxxxxx7728
JCB,311243xxxxxx6450
JCB,309601xxxxxx0407
JCB,352846xxxxxx2367
JCB,311289xxxxxx1440
JCB,308881xxxxxx3630
```

This script starts masking the credit card numbers on last articles first, since they are the most common used, so the results can be seen faster.

See also:

For more information about the `Maint::Ticket::MaskCreditCard` parameters, execute the following command:

```
otrs> /opt/otrs/bin/otrs.Console.pl Maint::Ticket::MaskCreditCard --help
```

The architecture of this script is designed to avoid affecting the system performance by working in batches and waiting between each batch. The number of processed articles per batch and the wait time between batches can be fine tuned to match system performance.

The script is also designed to remember last processed article and start again from that, allowing to stop the process at a certain time and resume later. There is an override to force starting again from the beginning.

You could specify an end date so only articles until that date will be processed (e.g. if you started automatic masking at a certain date), also it is possible to specify the number of articles to process per run.

11.17.2 Mask Credit Card

This feature is a complete subsystem that allows the following:

- Automatically show a warning message next to a credit card number (not storable).
- Mask credit card numbers for new tickets and articles.
- Mask credit card numbers contained in articles already stored in the system.

The credit card detection mechanism requires credit card numbers of 13, 15 or 16 digits. These credit card numbers should be at least potentially valid numerically, which means they pass the Luhn algorithm test.

For the current version only a subgroup of all potentially valid credit card number are detected. This is the list of the credit cards numbers that are considered valid:

- Visa 16 digits starting with a 4.
- Visa 13 digits starting with a 4.
- MasterCard 16 digits starting with 51 to 55.
- Discover 16 digits starting with 6011, 6121-29 to 6229-25, 644 to 649 or 65.
- JCB 16 digits starting with 3088, 3096, 3112, 3158, 3337 or 3528 to 3589.
- JCB 15 digits starting with 1800, 2100 or 2131.

- American Express 15 digits starting with 34 or 37.

For successful detection, the digits of these credit card numbers are allowed without separation or with a single separator in groups of digits as 4-4-4-4, 4-4-4-3, 4-4-4-1 or 4-6-5 (the last combination for American Express only). Allowed separators are `-`, `+`, `/`, `.` or a combination thereof.

Valid credit card numbers that are a subset of a bigger number are not considerate as credit card numbers. This is to avoid false positives, e.g. a serial number that contains a (not intentionally) valid credit card number. Valid credit card numbers should be enclosed by at least one non-numeric character.

11.17.3 Active Credit Card Masking

When this feature is enabled, every article will be scanned for valid credit card numbers before it is saved on the database. In case of any findings in subject or body, all but the first six and the last four digits will be replaced by a configurable masking character.

See also:

The behavior can be changed with the following settings in the system configuration:

- `OTRSCreditCardFilter::ActiveMaskEnabled`
- `OTRSCreditCardFilter::MaskedCharacter`

For example *1234-5678-9012-3456* becomes *1234-56xx-xxxx-3456*.

Warning: This procedure is permanent and irreversible!

To use this feature:

1. Create a ticket with the following article body:

```
Issuing Network,Card Number
JCB 15 digit,180061388939823
JCB 15 digit,180079668437698
JCB 15 digit,180001434886883
JCB 15 digit,180044208063503
JCB 15 digit,180010497338476
JCB 15 digit,210004248524033
JCB 15 digit,210012319871803
JCB 15 digit,180094846333594
JCB 15 digit,210084424984649
JCB 15 digit,210012951351973
JCB 15 digit,210008094074787
JCB 15 digit,210081171733450
```

2. Open the ticket detail view to see the created ticket. The body of the article will shown as:

```
Issuing Network,Card Number
JCB 15 digit,180061xxxxx9823
JCB 15 digit,180079xxxxx7698
JCB 15 digit,180001xxxxx6883
JCB 15 digit,180044xxxxx3503
JCB 15 digit,180010xxxxx8476
JCB 15 digit,210004xxxxx4033
JCB 15 digit,210012xxxxx1803
JCB 15 digit,180094xxxxx3594
```

(continues on next page)

(continued from previous page)

```
JCB 15 digit,210084xxxxx4649
JCB 15 digit,210012xxxxx1973
JCB 15 digit,210008xxxxx4787
JCB 15 digit,210081xxxxx3450
```

3. You can also try with valid credit card numbers in the subject or fetch a mail with valid credit card information using a postmaster account.

11.17.4 Credit Card Warning Message

It is possible to display a warning message next to the credit card number.

To use this feature:

1. Make sure that setting `OTRSCreditCardFilter::ActiveMaskEnabled` is not enabled.
2. Make sure that setting `OTRSCreditCardFilter::WarningTextEnabled` is enabled.
3. Define your custom message in setting `OTRSCreditCardFilter::WarningText` in the system configuration.
4. Create a ticket with the following article body:

```
Issuing Network,Card Number
JCB,3528988095245935
JCB,3112606824580636
JCB,3096030869937728
JCB,3112437499296450
JCB,3096010732100407
JCB,3528461498782367
JCB,3112892137191440
JCB,3088814635323630
```

5. Open the ticket detail view. The warning message should appear next to the credit card number.

```
Issuing Network,Card Number
JCB,3528988095245935 Reminder: You should not store credit card numbers in this␣
↪product!
JCB,3112606824580636 Reminder: You should not store credit card numbers in this␣
↪product!
JCB,3096030869937728 Reminder: You should not store credit card numbers in this␣
↪product!
JCB,3112437499296450 Reminder: You should not store credit card numbers in this␣
↪product!
JCB,3096010732100407 Reminder: You should not store credit card numbers in this␣
↪product!
JCB,3528461498782367 Reminder: You should not store credit card numbers in this␣
↪product!
JCB,3112892137191440 Reminder: You should not store credit card numbers in this␣
↪product!
JCB,3088814635323630 Reminder: You should not store credit card numbers in this␣
↪product!
```

You can also try with valid credit card numbers in the subject or fetch a mail with valid credit card information using a postmaster account.

11.18 Performance Tuning

There is a list of performance enhancing techniques for your OTRS installation, including configuration, coding, memory use, and more.

11.18.1 Ticket Search Index

OTRS uses a special search index to perform full-text searches across fields in articles from different communication channels.

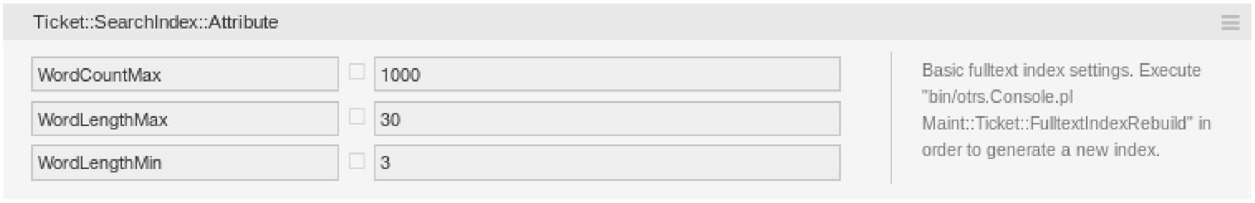
To create an initial index, use this command:

```
otrs> /opt/otrs/bin/otrs.Console.pl Maint::Ticket::FulltextIndex --rebuild
```

Note: Actual article indexing happens via an OTRS daemon job in the background. While articles which were just added in the system are marked for indexing immediately, it could happen their index is available within a few minutes.

There are some options available for fine tuning the search index:

`Ticket::SearchIndex::Attribute`
Basic full-text index settings.


Ticket::SearchIndex::Attribute

WordCountMax	1000
WordLengthMax	30
WordLengthMin	3

Basic fulltext index settings. Execute "bin/otrs.Console.pl Maint::Ticket::FulltextIndexRebuild" in order to generate a new index.

Fig. 18: `Ticket::SearchIndex::Attribute` Setting

Note: Run the following command in order to generate a new index:

```
otrs> /opt/otrs/bin/otrs.Console.pl Maint::Ticket::FulltextIndexRebuild
```

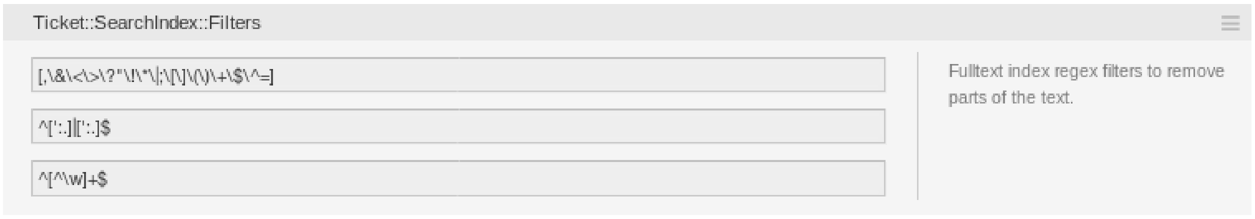
`WordCountMax`
Defines the maximum number of words which will be processed to build up the index. For example only the first 1000 words of an article body are stored in the article search index.

`WordLengthMin` and `WordLengthMax`
Used as word length boundaries. Only words with a length between these two values are stored in the article search index.

`Ticket::SearchIndex::Filters`
Full-text index regular expression filters to remove parts of the text.

There are three default filters defined:

- The first filter strips out special chars like: , & < > ? “! * | ; [ ] ( ) + $ ^ =
- The second filter strips out words which begin or ends with one of following chars: ‘: .


Ticket::SearchIndex::Filters

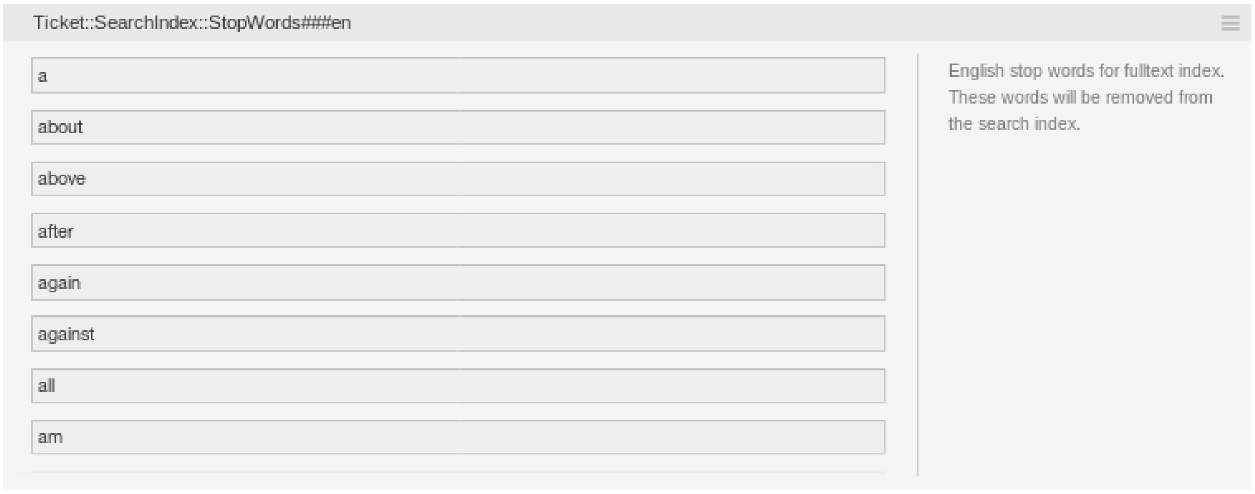
```
[,\&\<\>\?"\!\*\|;\[\]\(\)\+\$\^=]
^[':.]|[':.]$
^[^\w]+$
```

Fulltext index regex filters to remove parts of the text.

Fig. 19: `Ticket::SearchIndex::Filters` Setting

- The third filter strips out words which do not contain a word-character: a-z, A-Z, 0-9, _

`Ticket::SearchIndex::StopWords`
English stop words for full-text index. These words will be removed from the search index.


Ticket::SearchIndex::StopWords###en

```
a
about
above
after
again
against
all
am
```

English stop words for fulltext index. These words will be removed from the search index.

Fig. 20: `Ticket::SearchIndex::StopWords###en` Setting

There are so-called stop-words defined for some languages. These stop-words will be skipped while creating the search index.

See also:

If your language is not in the system configuration settings or you want to add more words, you can add them to this setting:

- `Ticket::SearchIndex::StopWords###Custom`

11.18.2 Document Search

OTRS uses Elasticsearch for its document search functionality. For a good introduction into the concepts, installation and usage of Elasticsearch, please follow the Quick start chapter in the official documentation.

Heap Size

Elasticsearch is written in Java and therefore runs in a Java Virtual Machine (JVM) on any cluster node. Such a JVM uses a part of the memory, called *heap*, which size can be configured in configuration file `jvm.options`.

The heap minimum and maximum configurations are by default set to a value of 1 GB and can be modified with the following options:

- `Xms1g`: minimum heap size.
- `Xmx1g`: maximum heap size.

If the `Xms` has a lower value than `Xmx`, the JVM will resize the used heap anytime the current limit is exceeded, until the value of `Xmx` is reached. Such a resizing causes the service to pause until it is finished, which may decrease the speed and reactivity of the search or indexing actions. Therefore it is highly recommended to set those configurations to an equal value.

Warning: If the maximum heap size is exceeded, the related cluster node stops working and might even shutdown the service.

The higher the heap maximum value is set, the more memory can be used by Elasticsearch, which also increases the possible pauses for garbage collection, done by the JVM. Therefore it is recommended to set a value for `Xmx`, that is not higher than 50% of the physical memory.

For more information and good rules of thumb about the heap size, please follow the Setting the heap size chapter in the official documentation.

Disk Allocation

During the run-time of the service, Elasticsearch inspects the available disk space and therefore decides whether to allocate new shards to the related cluster node or even relocate shards away from that particular node. Such behavior will be controlled by the current disk capacity and can be configured in configuration file `elasticsearch.yml`. Enclosed are some important configurations, that come with good default values, but might be important:

`cluster.routing.allocation.disk.watermark.low`
Default value of 85%. If this limit is exceeded, Elasticsearch will not allocate more shards to the related cluster node. The operation of that node is not influenced and data can still be indexed and searched.

`cluster.routing.allocation.disk.watermark.high`
Default value of 90%. If this limit is exceeded, Elasticsearch will try to relocate existing shards to other nodes (if possible), that have enough space available.

`cluster.routing.allocation.disk.watermark.flood_stage`
Default value of 95%. If this limit is exceeded, Elasticsearch will update the configuration of all indices to read-only index blocks `index.blocks.read_only_allow_delete`, that have at least one shard allocated to the related cluster node. Since then, it is not possible to index new data to such indices and restricted to searches and delete actions.

Note: If the flood stage was exceeded and certain indices are configured to read-only mode, such configuration *will not* automatically be changed by Elasticsearch. If the related disks contains enough free space again, due to manual actions, it is needed change the configuration back to normal mode manually.

For more information about disk watermarks and disk-based shard allocation, please follow the Disk-based Shard Allocation chapter in the official documentation.

11.18.3 Article Storage

There are three different back end modules for the article storage of phone, email and internal articles. The used article storage can be configured in the setting `Ticket::Article::Backend::MIMEBase::ArticleStorage`.

`Kernel::System::Ticket::Article::Backend::MIMEBase::ArticleStorageDB`
This default module will store attachments in the database. It also works with multiple front end servers, but requires much storage space in the database.

Note: Don't use this with large setups.

`Kernel::System::Ticket::Article::Backend::MIMEBase::ArticleStorageFS`
Use this module to store attachments on the local file system. It is fast, but if you have multiple front end servers, you must make sure the file system is shared between the servers. Place it on an NFS share or preferably a SAN or similar solution.

Note: Recommended for large setups.

`Kernel::System::Ticket::Article::Backend::MIMEBase::ArticleStorageAmazonS3`
Use this module to store attachments in any AWS S3 compatible object storage.

There is a default connection setup to the AWS S3 file storage in `Kernel/Config/Defaults.pm`. To activate the connection for your environment you have to add the following code snippet to `Kernel/Config.pm`:

```
$Self->{'Ticket::Article::Backend::MIMEBase::ArticleStorage'} =
↪'Kernel::System::Ticket::Article::Backend::MIMEBase::ArticleStorageAmazonS3';

$Self->{'Ticket::Article::Backend::MIMEBase::ArticleStorageAmazonS3'} = {
    'Active'            => 1,
    'Endpoint'          => 'http://127.0.0.1:9000',
    'Region'            => 'local',
    'AwsAccessKey'      => 'minioadmin',
    'AwsSecretKey'      => 'minioadmin',
    'Bucket'            => 'storage',
    'TestBucket'        => 'unit-test',
    'HealthCheck'       => '/minio/health/live',
    'MaxObjectSize'     => 1024 * 1024 * 20,
    'Reconnect'         => 2,
    'BackoffOnFailure'  => 1,
    'BackoffRetryCount' => 150,
};
```

You can switch from one back end to the other on the fly. You can switch the back end in the system configuration, and then run this command line utility to put the articles from the database onto the file system or the other way around:

```
otrs> /opt/otrs/bin/otrs.Console.pl Admin::Article::StorageSwitch --target␣
↪ArticleStorageFS
```

You can use the `--target` option to specify the target back end.

Note: The entire process can take considerable time to run, depending on the number of articles you have and the available CPU power and/or network capacity.

To ensure that you have access to all articles during the migration process you can use the `Ticket::Article::Backend::MIMEBase::ArticleStorageBackendCheckOrder` system configuration setting. This setting specifies additional article storage back ends to be checked for articles and their attachments. It is recommended to add here the old article storage back end.

11.18.4 Tuning the Web Server

The built-in web server of OTRS can handle small and medium setups out of the box. When OTRS serves many users simultaneously, it may be necessary to tweak the web server configuration to increase the number of worker processes, for example.

The web server configuration file is located in `Kernel/WebApp.conf`, and all settings there are documented. The `worker` setting can be increased to deploy more processes for serving HTTP requests on capable servers.

11.18.5 Caching

OTRS caches a lot of temporary data in `/opt/otrs/var/tmp`. Please make sure that this uses a high performance file system and storage. If you have enough RAM, you can also try to put this directory on a ramdisk like this:

```
otrs> /opt/otrs/bin/otrs.Console.pl Maint::Session::DeleteAll
otrs> /opt/otrs/bin/otrs.Console.pl Maint::Cache::Delete
root> mount -o size=16G -t tmpfs none /opt/otrs/var/tmp
```

Note: Add persistent mount point in `/etc/fstab`.

Warning: This will be a non-permanent storage that will be lost on server reboot. All your sessions (if you store them in the file system) and your cache data will be lost.

11.18.6 Clustering

For very high loads, it can be required to operate OTRS on a cluster of multiple front end servers. This is a complex task with many pitfalls. Therefore, OTRS Group provides support for clusters in its managed OTRS environment exclusively.

11.18.7 Limits of Objects

There is no technical limitation of how many objects can be used in the system but using high number of objects may affect the system performance. The proposed limits apply only to objects that are set as *valid*. The objects set as *invalid* or *invalid-temporarily* are not used by the system.

To keep the system fast and responsive the following limits for *valid* objects should not be exceeded:

- Mail accounts: 10
- Postmaster filters: 50
- ACLs: 80
- Dynamic fields: 300
- Dynamic field dropdown or multiselect values per field: 100
- Services: 500
- SLAs: 50
- Queues: 200
- Configuration item classes: 20
- Configuration item objects: 20,000
- Processes: 50
- Generic agents: 30 (frequency max once per hour per generic agent)
- Ticket states: 20
- Ticket types: 10
- Appointment calendars: 50
- Articles per ticket: 500

11.19 Cache Memcached Fast

This feature allows the use of a memory back end for the caches used in **OTRS**. It makes possible to allocate the entire cache through the Round Robin method on several memcached servers and provides smooth deployment of **OTRS** in a high-availability environment.

The number of memcached servers that can be used is unlimited. After proper setup of the memcached servers, you can use **OTRS** to configure which of them should be used by OTRS. We strongly recommend that configuration is carried out only by our **OTRS** consultants.

The *MemcachedFast* OTRS cache back end uses the CPAN module Cache::Memcached::Fast, which is a Perl client for memcached, a memory cache daemon.

Memcached is a free and open source, high-performance, distributed memory object caching system. It is often used to speed up big dynamic database-driven websites by caching data and objects in RAM to reduce the number of times an external data source (such as a database or API) must be read.

11.19.1 Basics

Note: You need to change the `Cache::Module` setting to use `MemcachedFast`, if you have not done this yet.

It is required to have at least 1 memcache server. It is recommended to have more (smaller) servers than a single big one. High availability clusters will definitely need more than 1 server. Only 1 server means a single point of failure.

It is recommended to use the latest versions of the required CPAN modules `Storable` and `Cache::Memcached::Fast`.

Note: Memcached does not handle authentication and replication.

Warning: Do not use patches that will install replication, like `repcached`. It is not supported.

11.19.2 Memory Settings

By default, memcached generally only sets aside a small amount of RAM for the cache. This may differ depending on the operating system or Linux distribution, but it varies between 64 MB and 512 MB. The more memory you can give to memcached, the better is. At least 4 GB is required for OTRS. Large deployments may want to give even more.

A rough guideline from our experience for a front end server without database is to divide the memory equally between the OS, Apache and memcached (1/3).

To increase the amount of memory available to memcached, modify the value passed along with `-m`. This value is in megabytes. For example, to specify 2 GB of cache space, pass `-m 2048`. Your setup most likely has a configuration file where you can set this. On Linux, memcached stores its settings in `/etc/memcached.conf`.

Note: Restart the memcache daemon, when changing the configuration.

11.19.3 Security

Memcached does not handle authentication. The user is responsible for data security. So make sure to secure your network (e.g. subnetting, separated network or firewalling).

But also make sure that each OTRS instance can communicate with each memcached server.

11.19.4 Monitoring

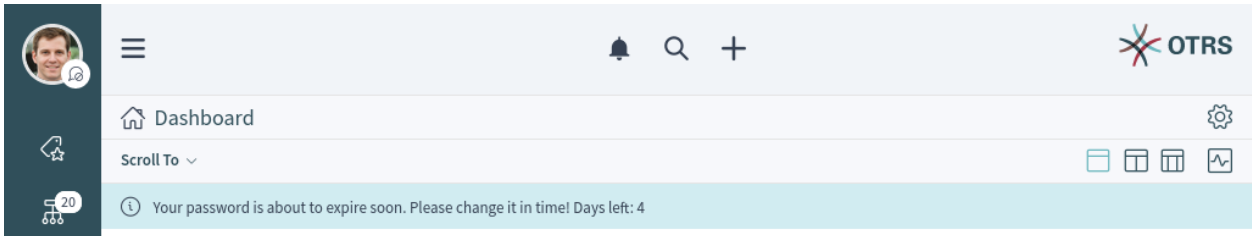
It is recommended to use monitoring tools, like phpmemcacheadmin. There are also extensions for Nagios.

11.20 LDAP Password Notifications

This feature displays the directory service password policy password expiration message that was previously displayed to users only after they logged on to corporate workstations.

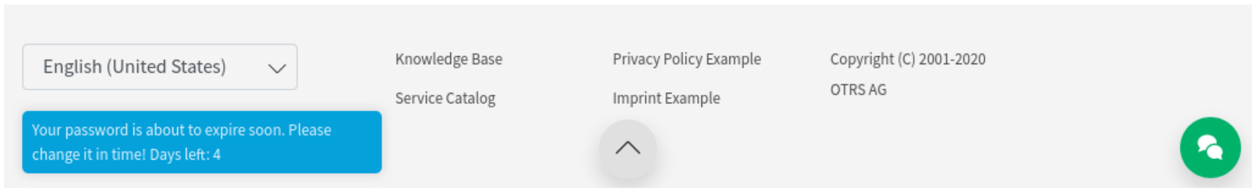
Because of the messages, users can keep their passwords up to date so they can avoid having to update them after log in. This is especially important for users, either agent or a customer user, who need to respond to a ticket in a very short time. In this case, the user can always respond quickly to messages because they do not have to spend valuable minutes updating their expired password.

Notifications can be displayed in several phases, which can be individually configured in the system settings. In the first phase, OTRS informs users via a warning message a few days before the password expires, telling them how many days are left until it needs to get changed.



Your password is about to expire soon. Please change it in time! Days left: 4

Fig. 21: Notification on Agent Interface



Your password is about to expire soon. Please change it in time! Days left: 4

Fig. 22: Notification on External Interface

If companies provide a separate website for updating passwords, it is possible to configure notifications with a link to that website so that users can update their expiring passwords for the notification immediately.

This feature is not enabled by default.

To enable the feature:

1. Go to the *System Configuration* screen.
2. Search for the setting `AuthLDAP::PasswordNotification::Enabled`.
3. Enable the feature.

See also:

Relevant system configurations:

- `AgentFrontend::NotifyDaysBeforeExpiry`
- `AgentFrontend::PasswordChangeLink`

- `AgentFrontend::NotifyModule###9001-AgentLDAPPassword-Check`
- `ExternalFrontend::NotifyDaysBeforeExpiry`
- `ExternalFrontend::PasswordChangeLink`
- `ExternalFrontend::NotifyModule###9001-CustomerLDAPPassword-Check`

11.21 Out of Office

This feature *ignores* out-of-office messages. As a rule, after processing a ticket, a final message is sent to a customer stating that the request has been completed. Then, the ticket is closed. If the message recipient has an out-of-office message turned on, this gets sent back to **OTRS** as a reply. This causes the closed ticket to be automatically re-opened. An employee must then manually close the ticket again. This can be very time-consuming, especially if you have frequent email traffic.

With the feature, the system stops responding to out-of-office messages, which means tickets are not re-opened when these messages are received. You can flexibly configure which out-of-office messages should be ignored.

This feature allows to create articles for a ticket with the `X-OTRS-OutOfOffice` header without changing the status of the ticket. It also prevents the locking of the ticket if mentioned header is present.

To configure this feature:

1. Go to the *System Configuration* screen.
2. Search for the setting `OTRSOutOfOffice-Header` and enable it.
3. Search for the setting `PostmasterX-Header` and add the `X-OTRS-OutOfOffice` header to the end of the list.
4. Deploy the settings.
5. Go to the *Postmaster Filters* screen and add a new postmaster filter.
6. Set email header `X-OTRS-OutOfOffice` with value *1* of an incoming email based on whatever regular expression or text you want to use (e.g. Subject: *[Out of Office]*).
7. Click on the *Save* button.

11.22 Delete Attachments

This feature enables the automatic deletion of ticket attachments that were closed in a definable time span. Furthermore, file types can be defined as additional filter criteria.

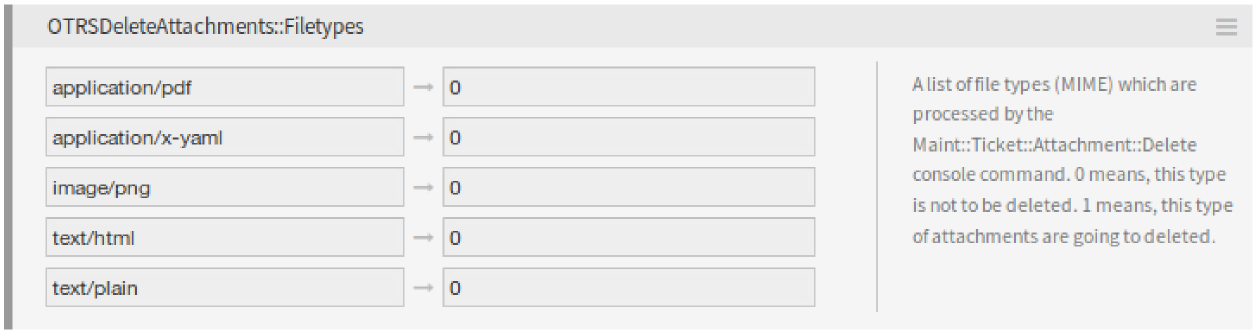
Note: This feature is only available to *On-Premise* customers. If you are a *Managed* customer, this feature is taken care of by the *Customer Solutions Team* in **OTRS**. Please contact us via support@otrs.com or in the OTRS Portal.

11.22.1 Enable Attachment Deletion

This feature is not enabled by default, it must be configured by an administrator first.

To use this feature:

1. Go to the *System Configuration* screen.
2. Search for the setting `OTRSDeleteAttachments::Filetypes` and add some file MIME types as key-value pairs. The key is the MIME type, the value is the deletion flag.
   - 0: This type of attachments are not to be deleted.
   - 1: This type of attachments are going to deleted.



OTRSDeleteAttachments::Filetypes

application/pdf	0
application/x-yaml	0
image/png	0
text/html	0
text/plain	0

A list of file types (MIME) which are processed by the Maint::Ticket::Attachment::Delete console command. 0 means, this type is not to be deleted. 1 means, this type of attachments are going to deleted.

Fig. 23: File Types Setting

11.22.2 Automatic File Type Population

There is a console command `Maint::Ticket::Attachment::AddMIMETypes` to populate the system configuration setting automatically.

```
otrs> /opt/otrs/bin/otrs.Console.pl Maint::Ticket::Attachment::AddMIMETypes
Searching attachment types received the last 90 days...
|       Added undefined MIME type: application/x-yaml
|       Added undefined MIME type: image/png
|       Added undefined MIME type: text/html
|       Added undefined MIME type: application/pdf
|       Added undefined MIME type: text/plain
Updating SysConfig...
Done.
```

The console command collects MIME types of attachments, that are in closed tickets and received within the last 90 days. Use the `--all` option to iterate over all closed tickets.

See also:

Run the command with the `--help` option to see the possibilities.

```
otrs> bin/otrs.Console.pl Maint::Ticket::Attachment::AddMIMETypes --help
```

11.22.3 Delete Attachments From Tickets

The attachment deletion is done by console command `Maint::Ticket::Attachment::Delete`. Check the following settings to customize the job:

`OTRSDeleteAttachments::Queues`
A list of queues (specified by their names) which are processed by the `Maint::Ticket::Attachment::Delete` console command.

> **Warning:** If no queues are defined, all queues will be affected.

`OTRSDeleteAttachments::TimeSpan`
The starting time (in days) from the current date from which the tickets should be analyzed for deletion.

`OTRSDeleteAttachments::TimePeriod`
The end time (in days) from the current date up to which the tickets should be analyzed for deletion.

Example:

```
TimeSpan: 10
TimePeriod: 40
                           Eligible  tickets
              |=========================================|
                                                        |=============|
                            TimePeriod (40)              TimeSpan (10)
|-------------|-------------|-------------|-------------|-------------|
50 days ago   40 days ago   30 days ago   20 days ago   10 days ago   Today
                                                        |
                                                       \ /
                                                 Ticket close time
```

Means that all tickets between 10 days ago from the current date and 40 days ago from the current date are analyzed for deletion.

The value of `TimePeriod` must be always higher than the value of `TimeSpan`.

See also:

Run the command with the `--help` option to see the possibilities.

```
otrs> bin/otrs.Console.pl Maint::Ticket::Attachment::Delete --help
```

11.22.4 Delete Attachments Automatically

Updating the file types and deleting the attachments can be run in the background by the OTRS daemon. Enable the following settings to run the jobs in a regular basis:

`Daemon::SchedulerCronTaskManager::Task###OTRSDeleteAttachments`
Delete attachments from tickets.

`Daemon::SchedulerCronTaskManager::Task###OTRSDeleteAttachments_AddMIMETypes`
Populate system configuration MIME types for attachment deletion.

Note: Do not forget to restart the OTRS daemon after the system configuration is changed.

11.23 Automatic Time Accounting

This feature enables automatic time accounting in ticket create screens, ticket detail view and any screens which have *Accounted time* field. When an agent opens these screens, a timer starts counting the seconds and displays the elapsed time in human readable way in the breadcrumb bar. When the ticket is created or the note is added to an existing ticket, the value of the counter is added automatically to the *Accounted time* field which is read only now.

The agent can pause and resume the timer any time. The timer remembers its value. If an agent opens the ticket detail view, spends some time to read the articles but do not add new note, the timer remembers the spent time. Next time when the agent opens the same ticket detail view again, the timer continues counting from the elapsed time.

Times are saved into the database per ticket and agent pairs so accounted times are preserved even on closing session.

There are two configurable thresholds when the timer changes its color from green to yellow or to red.

Additionally when this feature is enabled, display of *Accounted time* in ticket detail view is in *HH:MM:SS* format. All values saved in the ticket for accounted time is in seconds.

To activate this feature:

1. Go to the *System Configuration* screen.
2. Search for the setting `AgentFrontend::Ticket::AccountedTime` and make sure it is enabled.
3. Search for the setting `AgentFrontend::AutoTimeAccountingActive` and enable it.
4. Deploy the settings.
5. Go to a ticket create screen or open a ticket detail view and see the running timer.

See also:

Relevant system configurations:

- `AgentFrontend::AutoTimeAccountingActive`
- `AgentFrontend::AutoTimeAccounting::ThresholdYellow`
- `AgentFrontend::AutoTimeAccounting::ThresholdRed`
- `AgentFrontend::AutoTimeAccounting::WhiteListGroups`
- `Daemon::SchedulerCronTaskManager::Task###TimeAccountingAutoTrackCleanup`

There is a console command `Maint::Ticket::TimeAccounting::AutoTrackCleanup` to delete old time accounting auto-track entries. This command is run periodically by the OTRS Daemon. The command can be run manually, too.

For more information review the setting `Daemon::SchedulerCronTaskManager::Task###TimeAccountingAutoTra` or execute the command with the `--help` option.

```
otrs> bin/otrs.Console.pl Maint::Ticket::TimeAccounting::AutoTrackCleanup --help
```

11.24 Ticket Queue Selection

This feature makes it possible to automatically move a ticket to preferred queues on the basis of ticket and customer data when it is created. Some similar selection can be also done by using several email addresses or moving tickets manually.

Administrators can define in the system configuration on which data the queue selection should be executed. For example, it would be possible to define certain customer names, which causes their tickets to move into a special queue where the appropriate customer consultant can work on it immediately.

It would also be possible to define keywords, like *problem* or *computer*, for the ticket title so that those tickets are automatically routed to the first-level-support queue.

The feature can be configured with the setting `OTRSTicketQueueSelection::Configuration`. Key is the queue name and value is a semicolon-separated list of ticket and/or customer user settings and their expected value as regular expression.

This example shows how to configure a set of rules to move a ticket from any selected queue to a defined destination queue based on a ticket attribute, on ticket creation time.

To configure rules for newly created tickets with title `Test` to be moved to queue *Junk*:

1. Go to the *System Configuration* screen.
2. Search for the setting `OTRSTicketQueueSelection::Configuration` and open it for editing.
3. Enter *Junk* for key and `Ticket::Title='^Test$'` for content.
4. Create a ticket with title *Test* and check that the ticket is moved according to your configuration.

As seen in this example there is only one rule to that the ticket must match to be moved to the queue *Junk*. If more rules are needed, they can be added to the same content field with a `;` character as a separator, like:

```
Ticket::Title='^Test$';CustomerUser::UserFirstname='^John$';
↪CustomerUser::UserLastname='^Doe$'
```

Each rule is taken by the system as an AND condition, so for these new rules the ticket will need to match the title condition **and** the customer user condition in order to be moved to the *Junk* queue.

The rules can be based on ticket attributes `Ticket::<Attribute>` or on customer user attributes `CustomerUser::<Attribute>`.

11.25 Link Object Connector

OTRS can accept requests from external systems to interact with the link object. The functionality needs to be configured in the *Web Services* module of the administrator interface. There are example configurations for both SOAP and REST methods.

To import the web service configuration:

1. Go to the *Web Services* screen.
2. Click on the *Add Web Service* button in the left sidebar.
3. Click on the *Import Web Service* button in the left sidebar.
4. Select the file `development/webservices/GenericLinkConnectorSOAP.yml` or `development/webservices/GenericLinkConnectorREST.yml` within the OTRS home folder.

5. Click on the *Import* button.

These files contain the complete information to setup the web service. The web service can be customized after the import process, if needed. The result of this process will be the same as the manual procedure explained below, but with some useful comments are added to the operations. The imported web services are ready and able to handle all operations.

It is possible to setup the configuration manually, if you would not like to use the builtin configuration examples.

Note: The following steps are needed only, when the web services were not imported as explained above.

The following steps are required to provide a full working configuration:

1. Go to the *Web Services* screen.
2. Click on the *Add Web Service* button in the left sidebar.
3. Fill the form with the following data:
   - Name: LinkObjectConnector
   - Debug threshold: Debug
   - Network transport: HTTP::SOAP
4. Click on the *Save* button to enable the *Add Operation* selection and the *Configure* button for network transport.
5. Click on the network transport *Configure* button to show the transport options.
6. Use the following parameters:
   - Namespace: https://otrs.org/LinkObjectConnector/
   - Maximum message length: 100000
7. Click on the *Save* button. Wait until the page refreshes and then click on the *Go back to web service* button to continue.
8. The *Add Operation* selection provides seven operations for link:

   ```
   Link::LinkAdd
   Link::LinkDelete
   Link::LinkDeleteAll
   Link::LinkList
   Link::PossibleLinkList
   Link::PossibleObjectsLst
   Link::PossibleTypesList
   ```

9. Add the `Link::LinkAdd` operation and set the name `LinkAdd` in the next screen. It does not need any further configuration.
10. Click on the *Save* button. Wait until the page refreshes and then click on the *Go back to web service* button to continue.
11. Repeat step 9 and step 10 for the other six `Link::` operations.
12. After adding all operations, click on the *Save and Finish* button from the *Edit Web Service: LinkObjectConnector* screen.

Now you are on the *Web Service Management* overview screen and the new web service is ready and able to handle all operations.

Note: When the new web service is fully tested, the debug threshold could be changed to *Notice*. This will reduce the amount of data that will be kept in the database and only notice and error messages will be logged.

11.25.1 Usage

The link object feature provides functionality to link items (e.g. tickets, changes) from objects (e.g. `Ticket`, `ITSMChangeManagement`) in order to provide a logical connection.

To make easier to read and understand these requests here are some SOAP request templates showing the parameters for each operation.

`Link::LinkAdd` Request Template

```
<RootElement>
    <UserLogin>?</UserLogin>
    <Password>?</Password>
    <SessionID>?</SessionID>
    <SourceObject>?</SourceObject>
    <SourceKey>?</SourceKey>
    <TargetObject>?</TargetObject>
    <TargetKey>?</TargetKey>
    <Type>?</Type>
    <State>?</State>
</RootElement>
```

`UserLogin` and `Password`
Used for authentication. This is optional, but either a user login with a password or a session ID have to be provided.

`SessionID`
Used for authentication. This is optional, but either a user login with a password or a session ID have to be provided.

`SourceObject`
Object where the link originates.

`SourceKey`
ID of the originating object.

`TargetObject`
Object where the link points to.

`TargetKey`
ID of the target object.

`Type`
Type of the link.

`State`
State of the link, usually `Valid`.

`Link::LinkDelete` Request Template

```
<RootElement>
    <UserLogin>?</UserLogin>
    <Password>?</Password>
    <SessionID>?</SessionID>
    <Object1>?</Object1>
    <Key1>?</Key1>
    <Object2>?</Object2>
    <Key2>?</Key2>
    <Type>?</Type>
</RootElement>
```

`UserLogin` and `Password`
Used for authentication. This is optional, but either a user login with a password or a session ID have to be provided.

`SessionID`
Used for authentication. This is optional, but either a user login with a password or a session ID have to be provided.

`Object1`
Object where the link originates.

`Key1`
ID of the originating object.

`Object2`
Object where the link points to.

`Key2`
ID of the target object.

`Type`
Type of the link.

`Link::LinkDeleteAll` Request Template

```
<RootElement>
    <UserLogin>?</UserLogin>
    <Password>?</Password>
    <SessionID>?</SessionID>
    <Object>?</Object>
    <Key>?</Key>
</RootElement>
```

`UserLogin` and `Password`
Used for authentication. This is optional, but either a user login with a password or a session ID have to be provided.

`SessionID`
Used for authentication. This is optional, but either a user login with a password or a session ID have to be provided.

`Object`
Object where the links originate.

Key
ID of the originating object.

`Link::LinkList` Request Template

```
<RootElement>
    <UserLogin>?</UserLogin>
    <Password>?</Password>
    <SessionID>?</SessionID>
    <Object>?</Object>
    <Key>?</Key>
    <Object2>?</Object2>
    <State>?</State>
    <Type>?</Type>
    <Direction>?</Direction>
</RootElement>
```

`UserLogin` and `Password`
Used for authentication. This is optional, but either a user login with a password or a session ID have to be provided.

`SessionID`
Used for authentication. This is optional, but either a user login with a password or a session ID have to be provided.

`Object`
Object where the links originate.

`Key`
ID of the originating object.

`Object2`
Optional restriction of the object where the links point to.

`State`
State of the links, usually `Valid`.

`Type`
Optional restriction of the link type.

`Direction`
Optional restriction of the link direction (`Source` or `Target`).

`Link::PossibleLinkList` Request Template

```
<RootElement>
    <UserLogin>?</UserLogin>
    <Password>?</Password>
    <SessionID>?</SessionID>
</RootElement>
```

`UserLogin` and `Password`
Used for authentication. This is optional, but either a user login with a password or a session ID have to be provided.

SessionID
: Used for authentication. This is optional, but either a user login with a password or a session ID have to be provided.

Link::PossibleObjectsList Request Template

```
<RootElement>
    <UserLogin>?</UserLogin>
    <Password>?</Password>
    <SessionID>?</SessionID>
    <Object>?</Object>
</RootElement>
```

UserLogin and Password
: Used for authentication. This is optional, but either a user login with a password or a session ID have to be provided.

SessionID
: Used for authentication. This is optional, but either a user login with a password or a session ID have to be provided.

Object
: Object where the links originate.

Link::PossibleTypesList Request Template

```
<RootElement>
    <UserLogin>?</UserLogin>
    <Password>?</Password>
    <SessionID>?</SessionID>
    <Object1>?</Object1>
    <Object2>?</Object2>
</RootElement>
```

UserLogin and Password
: Used for authentication. This is optional, but either a user login with a password or a session ID have to be provided.

SessionID
: Used for authentication. This is optional, but either a user login with a password or a session ID have to be provided.

Object1
: Object where the link originates.

Object2
: Object where the link points to.

11.25.2 Examples

The following examples use all objects and types that are available in OTRS using the default configuration. If you have changed the link default configurations, the examples might have to be modified.

Create New Link

This example will show how to create a link, using the `Link::LinkAdd` generic interface operation.

1. Send the following SOAP request to the OTRS system.

```
<RootElement>
    <UserLogin>johnsmith</UserLogin>
    <Password>Secret123</Password>
    <SourceObject>Ticket</SourceObject>
    <SourceKey>1</SourceKey>
    <TargetObject>Ticket</TargetObject>
    <TargetKey>2</TargetKey>
    <Type>Normal</Type>
    <State>Valid</State>
</RootElement>
```

2. The server response should be something similar to this one.

```
<namesp1:RootElementResponse>
    <Success>1</Success>
</namesp1:RootElementResponse>
```

Remove Link

This example will show how to remove the previously created link, using the `Link::LinkDelete` generic interface operation.

1. Send the following SOAP request to the OTRS system.

```
<RootElement>
    <UserLogin>johnsmith</UserLogin>
    <Password>Secret123</Password>
    <Object1>Ticket</Object1>
    <Key1>1</Key1>
    <Object2>Ticket</Object2>
    <Key2>2</Key2>
    <Type>Normal</Type>
</RootElement>
```

2. The server response should be something similar to this one.

```
<namesp1:RootElementResponse>
    <Success>1</Success>
</namesp1:RootElementResponse>
```

Remove All Links

This example will show how to alternatively remove the previously created link (and all other links of this ticket), using the `Link::LinkDeleteAll` generic interface operation.

1. Send the following SOAP request to the OTRS system.

```
<RootElement>
    <UserLogin>johnsmith</UserLogin>
    <Password>Secret123</Password>
    <Object>Ticket</Object>
    <Key>1</Key>
</RootElement>
```

2. The server response should be something similar to this one.

```
<namesp1:RootElementResponse>
    <Success>1</Success>
</namesp1:RootElementResponse>
```

Show All Links

This example will show all links of an object, using the `Link::LinkList` generic interface operation.

1. Send the following SOAP request to the OTRS system.

```
<RootElement>
    <UserLogin>johnsmith</UserLogin>
    <Password>Secret123</Password>
    <Object>Ticket</Object>
    <Key>1</Key>
    <State>Valid</State>
</RootElement>
```

2. The server response should be something similar to this one.

```
<namesp1:RootElementResponse>
    <RootElement>
        <Direction>Source</Direction>
        <Key>2</Key>
        <Object>Ticket</Object>
        <Type>Normal</Type>
    </RootElement>
    <RootElement>
        <Direction>Source</Direction>
        <Key>3</Key>
        <Object>Ticket</Object>
        <Type>ParentChild</Type>
    </RootElement>
    <RootElement>
        <Direction>Target</Direction>
        <Key>4</Key>
        <Object>Ticket</Object>
        <Type>ParentChild</Type>
    </RootElement>
</namesp1:RootElementResponse>
```

Show All Possible Links

This example will show all possible link types between objects, using the `Link::PossibleLinkList` generic interface operation.

1. Send the following SOAP request to the OTRS system.

```
<RootElement>
    <UserLogin>johnsmith</UserLogin>
    <Password>Secret123</Password>
</RootElement>
```

2. The server response should be something similar to this one.

```
<namesp1:RootElementResponse>
    <RootElement>
        <Object1>Ticket</Object1>
        <Object2>Ticket</Object2>
        <Type>Normal</Type>
    </RootElement>
    <RootElement>
        <Object1>Ticket</Object1>
        <Object2>Ticket</Object2>
        <Type>ParentChild</Type>
    </RootElement>
</namesp1:RootElementResponse>
```

Show All Possible Objects

This example will show all objects that can be linked to a given object, using the `Link::PossibleObjectsList` generic interface operation.

1. Send the following SOAP request to the OTRS system.

```
<RootElement>
    <UserLogin>johnsmith</UserLogin>
    <Password>Secret123</Password>
    <Object>Ticket</Object>
</RootElement>
```

2. The server response should be something similar to this one.

```
<namesp1:RootElementResponse>
    <PossibleObject>Ticket</PossibleObject>
</namesp1:RootElementResponse>
```

Show All Possible Types

This example will show all link types between two given objects, using the `Link::PossibleTypesList` generic interface operation.

1. Send the following SOAP request to the OTRS system.

```
<RootElement>
    <UserLogin>johnsmith</UserLogin>
    <Password>Secret123</Password>
    <Object1>Ticket</Object1>
    <Object2>Ticket</Object2>
</RootElement>
```

2. The server response should be something similar to this one.

```
<namesp1:RootElementResponse>
    <PossibleType>Normal</PossibleType>
    <PossibleType>ParentChild</PossibleType>
</namesp1:RootElementResponse>
```

11.26 Advanced Editor

This feature allows OTRS administrators and OTRS developers with experience using Template Toolkit to write Template Toolkit code snippets in OTRS templates. This can facilitate, for example, the customization of response templates with multiple languages because you can customize the same template for many languages by using conditions without adding completely new response templates. This saves resources within **OTRS** and gives you a cleaner overview of your response templates.

This feature extends the standard editor by enabling logical if-else conditions as well as loops and much more.

To insert code snippets:

1. Go to the *System Configuration* screen.
2. Search for the setting `AdvancedEditor` and enable it.
3. Open *Templates*, *Signatures* or *Salutations* modules in the administrator interface.
4. Add or modify a resource in these screens.
5. Find the new icon in the Rich Text editor toolbar (the last one).

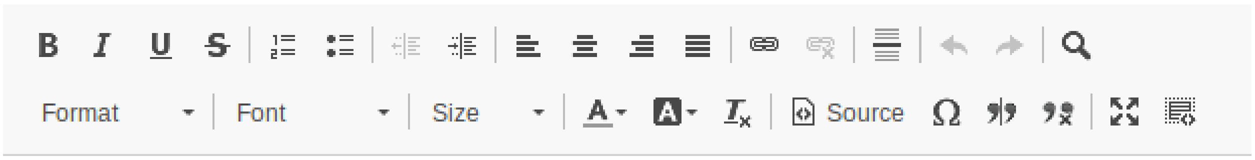
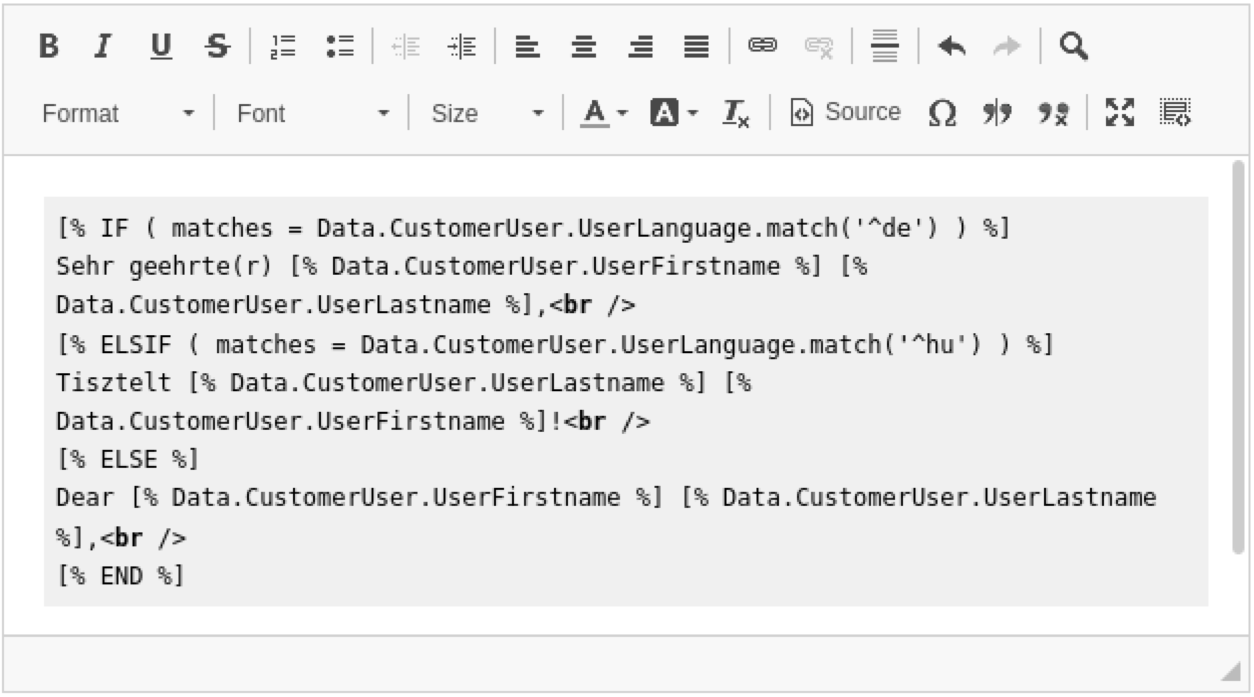


Fig. 24: Insert Code Snippet Icon

6. Click on the new icon to open a new window.
7. Select *TemplateToolkit* from the *Language* drop-down.
8. Write the following lines into the text box.

```
[% IF ( matches = Data.CustomerUser.UserLanguage.match('^de') ) %]
Sehr geehrte(r) [% Data.CustomerUser.UserFirstname %] [% Data.CustomerUser.
↪UserLastname %],<br />
[% ELSIF ( matches = Data.CustomerUser.UserLanguage.match('^hu') ) %]
Tisztelt [% Data.CustomerUser.UserLastname %] [% Data.CustomerUser.UserFirstname
↪%]!<br />
[% ELSE %]
Dear [% Data.CustomerUser.UserFirstname %] [% Data.CustomerUser.UserLastname %],
↪<br />
[% END %]
```

9. Click on the *OK* button to save the code snippet.



```
[% IF ( matches = Data.CustomerUser.UserLanguage.match('^de') ) %]
Sehr geehrte(r) [% Data.CustomerUser.UserFirstname %] [%
Data.CustomerUser.UserLastname %],<br />
[% ELSIF ( matches = Data.CustomerUser.UserLanguage.match('^hu') ) %]
Tisztelt [% Data.CustomerUser.UserLastname %] [%
Data.CustomerUser.UserFirstname %]!<br />
[% ELSE %]
Dear [% Data.CustomerUser.UserFirstname %] [% Data.CustomerUser.UserLastname
%],<br />
[% END %]
```

Fig. 25: Rich Text Editor Code Snippet

10. Click on the *Save* or *Save and finish* button.
11. Add the template, signature or salutation to the queue you want to use it.
12. Create a new ticket, choose a customer user (for this example we will call him John Doe), then choose the queue you have configured your template for.

    If the customer user language is German, it should print *Sehr geehrte(r) John Doe,*. If it is set to Hungarian, you should see *Tisztelt Doe John!*. If your customer user has no language set or it is set to English, you will see *Dear John Doe,*.

As you can see, conditions in templates can be very useful, but be careful to write correct Template Toolkit syntax. If you make mistakes there, you will get an error during choosing a template.

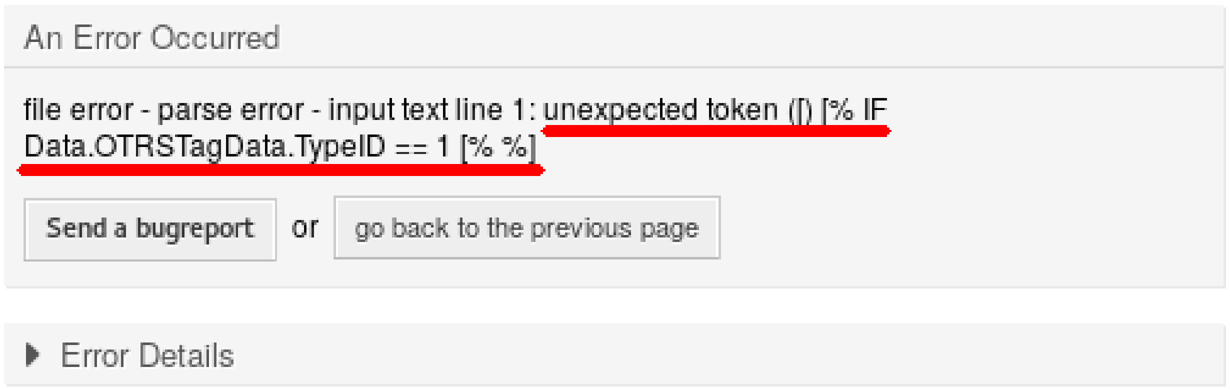
See also:

For detailed instructions on how to write Template Toolkit code, refer to the Template Toolkit User Manual.

11.26.1 Debugging Templates

Writing Template Toolkit code is not too difficult compared to other languages. Nevertheless errors can occur as we are all just humans.

If you have mistakes in your Template Toolkit syntax, saving a template will bring up an error screen that tell you about what went wrong and where to find the error.



An Error Occurred

file error - parse error - input text line 1: unexpected token ([) [% IF Data.OTRSTagData.TypeID == 1 [% %]

Fig. 26: Error Reason in Error Screen

In this example the **OTRS** system informed us, that there was an unexpected token `[` in the line `[% IF Data.OTRSTagData.TypeID == 1 [% %]`. A second opening bracket was entered inside the same Template Toolkit tag, which for sure is forbidden.

To fix the error:

1. Go back to the previous page.
2. Edit the template and save it.
3. Repeat these steps until no more errors are found.

11.27 System Monitoring

OTRS implements a basic interface to system monitoring suites.

Note: A network monitoring system, such as Nagios, Icinga2, HP OpenView or similar, capable of sending out events via email is required to use this feature.

11.27.1 Control Flow for Nagios

For Nagios, it works by receiving email messages sent by a network monitoring suite. New tickets are created in case of component failures. Once a ticket has been opened messages regarding the effected component are attached to this ticket. When the component recovers, the ticket state can be changed or the ticket can be closed. If an open ticket for a given host and service combination exists, all mails concerning this particular combination will be attached to the ticket until it' s closed.

The following control flow illustrates how mails are handled by this module and in which cases they trigger which action. Pretty much all checks are configurable using the regular expressions given by the parameters listed below.

- Mail matches `FromAddress`?
  - NO → Continue with regular mail processing
  - YES → Does a ticket with matching host and service combination already exist in **OTRS**?
    - NO → Does `State:` match `NewTicketRegExp`?
      - NO → Stop processing this mail (silent drop)
      - YES → Create new ticket, record host and service, attach mail
    - YES → Attach mail to ticket. Does `State:` match `CloseTicketRegExp`?
      - NO → Continue with regular mail processing
      - YES → Change ticket type as configured in `CloseActionState`

Besides of a few additional sanity checks this is how the system monitoring module treats incoming mails. By changing the regular expressions it should be possible to adopt it to different monitoring systems.

11.27.2 Icinga2 Acknowledge

For Icinga2, it works by specifying a host and a service in ticket dynamic fields. This combination of host and service is used after a ticket lock is set to generate a HTTP request which is send to the configured Icinga2 host. In Icinga2, this request is used to create or to confirm new incidents.

A new ticket is created with values in the specified dynamic fields which are needed as combination of host and service for the communication to the Icinga2 host. After this newly created ticket is locked to an agent, a HTTP request is send to the configured Icinga2 host. In the Icinga2 host a new acknowledge is created or an existing one is confirmed.

11.27.3 System Configuration for System Monitoring

There are two useful functions, which can be used to automatize the workflow based on the messages from the system monitoring suites. However, both functions are turned off by default, and they can be enabled in the system configuration.

Note: The *OTRS Configuration Management* feature is required to use these functions.

Link Incident Ticket with Configuration Item

An already opened incident ticket can be linked with the affected configuration item. This is only possible when a subsequent system monitoring email arrives.

The setting `SystemMonitoring::LinkTicketWithCI` needs to be enabled to use this feature.

This feature has no agent interface, but in case the incident ticket linking with configuration item and the incident state setting are enabled, the list of configuration items and the list of services can be changed based on the incident tickets.

For more information, read the *Services* and the *Configuration Items* chapters in the OTRS::ITSM manual.

Set Incident State

It is possible to set the incident state of a configuration item automatically when a system monitoring email arrives.

The setting `SystemMonitoring::SetIncidentState` needs to be enabled to use this feature.

See also:

External programs to be run by OTRS are blocked by default due to security reasons. You have to add the program to the allow list as described in the *Allow Program Safe to Run* chapter.

11.28 Hide/Show Dynamic Fields

A dynamic field enables the extension of the information stored in a ticket and the individual configuration of your **OTRS**. However, dynamic fields can also cause confusion when they appear in the wrong context.

With this feature you can build flexible and dynamic forms by hiding or showing fields based on ticket information, permissions and more. This is controlled by ticket *Access Control Lists (ACL)*.

A typical example of an internal IT department process is the ordering of hardware or software in **OTRS**. Dynamic fields help structure this order process through drop-down lists. For example, after choosing the hardware to be ordered, the feature ensures that only appropriate choices will appear in the next step, e.g. screen, keyboard and printer. The dynamic fields defined for software, like word processing, image processing or spreadsheet software, will not be displayed.

The hide/show dynamic fields function can be used in agent and external interface. The following screens can be configured.

Agent interface:

- New Email Ticket
- New Phone Ticket
- New Process Ticket (incl. Process Activity Dialogs)
- New SMS Ticket
- Close Ticket
- Change Customer
- Send Email Outbound
- Change Free Fields

- Link Objects
- Merge Ticket
- Move Ticket
- Add Note
- Change Owner
- Set Pending Time
- Add Phone Call Inbound
- Add Phone Call Outbound
- Change Priority
- Change Responsible
- Send SMS Outbound
- Forward via Email
- Redirect via Email
- Reply via Email
- Reply to All via Email
- Reply via Note
- Reply via SMS

External interface:

- Create New Ticket
- Create New Process Ticket (incl. Process Activity Dialogs)
- Ticket Reply (within Ticket Detail View)

Note: This feature works with ticket dynamic fields only. Other dynamic field object types are not supported.

The following example shows how to use this feature when dynamic fields should be displayed based on specific rules.

Goals:

- If brand *VW* is selected, all dynamic fields should be hidden and just *VW Model* is displayed.
- If VW model *Up* is selected, all dynamic fields should be displayed except for the fields *Peugeot Model* and *Peugeot Production Facility*.

Create the following dynamic fields:

Object	Type	Name	Label	Possible values
Ticket	Dropdown	`Brand`	Brand	• `VW` → VW • `Peugeot` → Peugeot
Ticket	Dropdown	`VWModel`	VW Model	• `Up` → Up • `Polo` → Polo • `Golf` → Golf • `T5` → T5
Ticket	Dropdown	`VWProductionFacility`	VW Production Facility	• `Barcelona` → Barcelona • `Berlin` → Berlin • `Bratislava` → Bratislava
Ticket	Dropdown	`PeugeotModel`	Peugeot Model	• `207` → 207 • `307` → 307
Ticket	Dropdown	`PeugeotProductionFacility`	Peugeot Production Facility	• `Poissy` → Poissy • `Madrid` → Madrid • `Trnava` → Trnava
Ticket	Dropdown	`Fuel`	Fuel	• `Gasoline` → Gasoline • `Diesel` → Diesel • `Gas` → Gas
Ticket	Multiselect	`Accessories`	Accessories	• `CDRadio` → CD Radio • `GPS` → GPS • `ProximitySensors` → Proximity Sensors • `RearCamera` → Rear Camera • `ClimateControl` → Climate Control
Ticket	Textarea	`Remarks`	Remarks	
Ticket	Date	`RegistrationDate`	Registration Date	
Ticket	Date	`InvoiceDate`	Invoice Date	

Note: All *Dropdown* and *Multiselect* dynamic fields should have the option *Add empty value* set to *Yes* in their configuration.

Add the dynamic fields to the *New Phone Ticket* screen via setting `Forms###AgentFrontend::TicketCreate::Phone::CreateProperties`:

```
- Label: Dynamic Fields
  Collapsible: 1
  Fields:
  - Name: DynamicField_Brand
  - Name: DynamicField_VWModel
  - Name: DynamicField_VWProductionFacility
  - Name: DynamicField_PeugeotModel
  - Name: DynamicField_PeugeotProductionFacility
  - Name: DynamicField_Fuel
  - Name: DynamicField_Accessories
  - Name: DynamicField_Remarks
  - Name: DynamicField_RegistrationDate
  - Name: DynamicField_InvoiceDate
```

Import this ACL:

```
---
- ChangeBy: root@localhost
  ChangeTime: 2019-07-22 11:44:25
  Comment: ''
  ConfigChange:
    PossibleNot:
      Form:
      - PeugeotModel
      - PeugeotProductionFacility
      - Accessories
      - Fuel
      - Remarks
      - RegistrationDate
      - InvoiceDate
  ConfigMatch:
    Properties:
      Ticket:
        DynamicField_Brand:
        - VW
  CreateBy: root@localhost
  CreateTime: 2019-07-22 11:40:43
  Description: ''
  ID: 1
  Name: ACL-VW
  StopAfterMatch: 0
  ValidID: 1
```

Detailed explanation:

```
DynamicField_Brand:
- VW
```

The condition for this ACL rule. If brand *VW* is selected, the rule will come into action. The array contains the used possible values. These are keys found in your database inside the `dynamic_field` table in column `config`. In this example it is a dynamic field of type *Dropdown*.

```
PossibleNot:
  Form:
  - PeugeotModel
  - PeugeotProductionFacility
  - Accessories
  - Fuel
  - Remarks
  - RegistrationDate
  - InvoiceDate
```

This section lists the dynamic fields that should not be visible. In this example the dynamic fields *VW Model* and *VW Production Facility* are visible. All other dynamic fields will be hidden.

Import this second ACL:

```
---
- ChangeBy: root@localhost
  ChangeTime: 2019-07-22 12:06:24
  Comment: ''
  ConfigChange:
    Possible:
      Ticket:
        DynamicField_Accessories:
        - CDRadio
        - ClimateControl
        DynamicField_Fuel:
        - Gasoline
        DynamicField_VWProductionFacility:
        - Bratislava
    PossibleAdd:
      Form:
      - Accessories
      - Fuel
      - Remarks
      - RegistrationDate
      - InvoiceDate
    PossibleNot:
      Form:
      - PeugeotModel
      - PeugeotProductionFacility
  ConfigMatch:
    Properties:
      Ticket:
        DynamicField_Brand:
        - VW
        DynamicField_VWModel:
        - Up
  CreateBy: root@localhost
  CreateTime: 2019-07-22 11:47:02
  Description: ''
  ID: 2
  Name: ACL-VW-Up
  StopAfterMatch: 0
  ValidID: 1
```

Detailed explanation:

```
DynamicField_Brand:
- VW
DynamicField_VWModel:
- Up
```

In this example two conditions should be met. Brand has to be *VW* and VW model has to be *Up* for this rule to come into action. It will be triggered only if an agent selects brand *VW* **and** VW model *Up*.

```
PossibleAdd:
  Form:
  - Accessories
  - Fuel
  - Remarks
  - RegistrationDate
  - InvoiceDate
```

Here the dynamic fields *VW Model* and *VW Production Facility* were already visible and they remain, but *Accessories*, *Fuel*, *Remarks*, *Registration Date* and *Invoice Date* has to be re-added to the fields that are visible. This is done in the *PossibleAdd* section as the first ACL sets this fields as not shown and both ACLs works together. If this was done in the *Possible* section for example, the result will be that only these fields explicitly will be shown and *VW Model* and *VW Production Facility* will be hidden as they are not longer part of the (new) *Possible* section.

```
PossibleNot:
  Form:
  - PeugeotModel
  - PeugeotProductionFacility
```

Just *Peugeot Model* and *Peugeot Production Facility* are invisible (in our example it does not make much sense to configure a Peugeot model if the user has a VW Up).

In addition to the visibility of dynamic fields there is the possibility to show just some of the possible values of a dynamic field. Combined into ACL rules like in here, makes it easier to handle big multiselects.

```
Possible:
  Ticket:
    DynamicField_Accessories:
    - CDRadio
    - ClimateControl
    DynamicField_Fuel:
    - Gasoline
    DynamicField_VWProductionFacility:
    - Bratislava
```

In our example a VW Up can have just CD Radio and Climate Control as extra accessories, just Gasoline as fuel and can be produced just in Bratislava.

If we would have an ACL rule for Peugeot 207 for example, there may be other extras, fuel options and production locations selectable.

Note: If you are showing dynamic fields using the *Possible* option based on a `DynamicField_NameX` value, is normally desirable to include the dynamic field that triggers the ACL to be part of the fields to be displayed in the *Possible* or *PossibleAdd* sections (if apply). Otherwise if *Possible* or *PossibleAdd* contains other fields and not the trigger, the latest will not be shown after the value is selected.

Note: The mandatory status of the fields can not be changed using this method.

11.29 Ticket Forms

For service organizations that use the **OTRS** helpdesk software in different departments or that have to respond to many different types of customer requests, this feature is a definite must-have. With it you can display different ticket masks and forms in the agent and external interfaces, depending on which dynamic fields are relevant to the customer request. Moreover, the ticket creation process is faster as headlines and message body fields are pre-filled with the necessary information. You can define required fields for forms as well as display pre-filled fields, such as ticket types and services, in the external interface based on the customer group settings. You can also pre-fill ticket masks by adding attachments, using Rich Text format, or using the OTRS smart tags `<OTRS_CONFIG_*>` and `<OTRS_CURRENT_*>` in the template body.

With this feature the following screens can be configured:

- Agent interface
  - New phone ticket
  - New email ticket
- External interface
  - New ticket

In addition, the *Hide/Show Dynamic Fields* feature enables you to define which dynamic fields you want to display in your ticket forms and which ones you want to hide. This gives you more freedom to configure all ticket screens to your specific needs.

The *Dynamic Ticket Templates* feature uses the extension to the ACLs mechanism to create ticket templates based on the selected ticket type and service. These templates include a predefined ticket body, title, attachments and full dynamic fields visibility configuration.

For this example we will use the same fields as in the example of *Hide/Show Dynamic Fields*, but is necessary to comment or remove the ACLs.

Note: While the dynamic ticket templates can work in conjunction with other ACL rules, it is recommended to test the templates without other affecting ACLs, and after the templates work as needed, incorporate other ACLs one by one.

To show the full functionalities of the dynamic ticket templates we can simply remove all dynamic fields from the *New Phone Ticket* screen by resetting the system configuration setting `Forms###AgentFrontend::TicketCreate::Phone::CreateProperties` to the default value, or disable the specific dynamic fields in the YAML configuration of this form:

```yaml
- Label: Dynamic Fields
  Collapsible: 1
  Fields:
  - Name: DynamicField_Brand
    Inactive: 1
  - Name: DynamicField_VWModel
    Inactive: 1
  - Name: DynamicField_VWProductionFacility
```

(continues on next page)

(continued from previous page)

```
    Inactive: 1
  - Name: DynamicField_PeugeotModel
    Inactive: 1
  - Name: DynamicField_PeugeotProductionFacility
    Inactive: 1
  - Name: DynamicField_Fuel
    Inactive: 1
  - Name: DynamicField_Accessories
    Inactive: 1
  - Name: DynamicField_Remarks
    Inactive: 1
  - Name: DynamicField_RegistrationDate
    Inactive: 1
  - Name: DynamicField_InvoiceDate
    Inactive: 1
```

To define a dynamic ticket template it is necessary to specify a ticket type and a service. If these features are not activated by default, please follow the next steps before start using this feature.

To activate ticket type and service features:

1. Go to the *System Configuration* screen.
2. Search for the setting `Ticket::Service` and enable it.
3. Search for the setting `Ticket::Type` and enable it.

To activate ticket type and service features for the external interface:

1. Go to the *System Configuration* screen.
2. Search for the setting `ExternalFrontend::TicketCreate###Service` and enable it.
3. Search for the setting `ExternalFrontend::TicketCreate###TicketType` and enable it.

Create the following services:

- *Peugeot Service*
- *VW Service*

All services should be available for the customer user who will use this feature, or marked as default services.

The goals of this example are the follows:

- If the selected service is *VW Service* and the selected dynamic ticket template is *VW Template 1*, the ticket form should look like:

Subject	VW Service Request 1
Body	– Oil Change – Oil Filter Change – Air Filter Change – Fluids Check
Dynamic Fields	– `VWModel`: visible and required – `VWProductionFacility`: visible

- If the selected service is *VW Service* and the selected dynamic ticket template is *VW Template 2*, the ticket form should look like:

Subject	VW Service Request 2
Body	The selected accessory reports the following issues:
Dynamic Fields	– `VWModel`: visible – `Accessories`: visible and required – `Remarks`: visible – `RegistationDate`: visible – `InvoiceDate`: visible

- If the selected service is *Peugeot Service*, the ticket form should look like:

Subject	Peugeot Service Request
Body	The car computer reports the following issues:
Dynamic Fields	– `PeugeotModel`: visible – `PeugeotProductionFacility`: visible – `Fuel`: visible

To create a dynamic ticket template:

1. Open the *Dynamic Ticket Templates* module of the *Ticket Settings* group in the administrator interface.
2. Click on the *Add template* button in the left sidebar.
3. Fill in the required fields.
4. Click on the *Save* button.

If you have added other dynamic fields by your own, the template form will show more fields than the ones explained in this examples. There is nothing to worry about, just leave all other dynamic fields as *Hide* while you create the new templates.

There are special dynamic fields from other packages that are not displayed in the template form. These special fields have a determined behavior and they should be always hidden, always shown or their visibility depends on other configurations, so they can not be part of the template definition. Also any dynamic field marked as *Internal* will not be displayed in the template form.

See also:

A quick way to know if a dynamic field is *Internal* is to look at dynamic fields in the overview table of the *Dynamic Fields Management* screen. *Internal* dynamic fields can not be deleted, and they does not have a icon in the *Delete* column.

Create the *VW Template 1* dynamic ticket template with the following data:

Field name	Value
Name	`VWTemplate1`
Comment	VW Template 1
Valid	valid
Frontend	Agent and External Interface
Type	Unclassified
Service	VW Service
Subject	VW Service Request 1
Body	• Oil Change • Oil Filter Change • Air Filter Change • Fluids Check
Attachments	
`Brand`	Hide
`VWModel`	Show as mandatory
`VWProductionFacility`	Show
`PeugeotModel`	Hide
`PeugeotProductionFacility`	Hide
`Fuel`	Hide
`Accessories`	Hide
`Remarks`	Hide
`RegistationDate`	Hide
`InvoiceDate`	Hide

Create the *VW Template 2* dynamic ticket template with the following data:

Field name	Value
Name	`VWTemplate2`
Comment	VW Template 2
Valid	valid
Frontend	Agent and External Interface
Type	Unclassified
Service	VW Service
Subject	VW Service Request 2
Body	The selected accessory reports the following issues:
Attachments	
`Brand`	Hide
`VWModel`	Show
`VWProductionFacility`	Hide
`PeugeotModel`	Hide
`PeugeotProductionFacility`	Hide
`Fuel`	Hide
`Accessories`	Show as mandatory
`Remarks`	Show
`RegistationDate`	Show
`InvoiceDate`	Show

Create the *Peugeot Template* dynamic ticket template with the following data:

Field name	Value
Name	`PeugeotTemplate`
Comment	Peugeot Template
Valid	valid
Frontend	Agent and External Interface
Type	Unclassified
Service	Peugeot Service
Subject	Peugeot Service Request
Body	The car computer reports the following issues:
Attachments	
`Brand`	Hide
`VWModel`	Hide
`VWProductionFacility`	Hide
`PeugeotModel`	Show
`PeugeotProductionFacility`	Show
`Fuel`	Show
`Accessories`	Hide
`Remarks`	Show
`RegistationDate`	Show
`InvoiceDate`	Show

Once there are one or more templates activated in the system, you can use them.

Note: ACL restrictions will be ignored for the superuser account (UserID 1).

To activate a dynamic ticket template:

1. Click on the *Take Phone Call* menu item of the *Action* menu in the header bar.

   There is a new field called *Dynamic Ticket Template*. By default it is empty and it will be filled automatically when you select the appropriate combination of ticket type and service.

2. Fill the form with the following data:

   - Type: Unclassified
   - Customer user: Customer 1
   - Queue: Misc
   - Service: VW Service

3. The *Dynamic Ticket Template* field will be automatically populated with *VW Template 1* and *VW Template 2*.
4. Select each template and match the resulting *New Phone Ticket* form with the expected results.
5. Now change the selected service to *Peugeot Service*. As there is only one template defined for this combination of ticket type and service, the template will be automatically selected for you.
6. Compare the resulting *New Phone Ticket* form with the expected results.

This complete example can be tested also *New Email Ticket* or external *New Ticket* screen without changing anything.

The template definition is independent from the screen and once a template is defined, it can be used on any ticket creation screen, but please review the dynamic field configuration on each screen for the default fields configuration when a template is not selected.

11.29.1 Dynamic Ticket Templates Customer Groups

For this example we will use the *Peugeot Template* defined in the example above.

The goal to have a pre-filled form to create a new ticket in external interface.

To configure a customer group:

1. Go to the *System Configuration* screen.
2. Search for the setting `CustomerGroupSupport` and enable it.
3. Search for the setting `Ticket::DynamicTicketTemplate::CustomerGroup` and enable it.
4. Go to the *Groups* screen.
5. Create the group *Peugeot Customer*.
6. Go to the *Customers   Groups* screen.
7. Assign *Customer 1* customer to the *Peugeot Customer* group.
8. Go to the *Dynamic Ticket Templates   Groups* screen.
9. Assign the *Peugeot Customer* group to the *Peugeot Template* template.
10. Login in the external interface as customer user of *Customer 1* and create a new ticket.
11. Compare the pre-filled new ticket form with the expected results. Also notice that the ticket type is pre-selected as default and service is also pre-selected as *Peugeot Service*.

You can also set the optional settings, but for this example we will not use these optional settings, so you can leave it as default.

See also:

If no template is assigned to a group and the notification is enabled, such notification can be customized by adding different recipients or updating the notification body. To do this, edit the *Missing DTT assignment for CustomerGroup* notification in the *Ticket Notifications* screen of the administrator interface.

Missing DTT assignment for CustomerGroup
    If no template is assigned to a group and the notification is enabled, such notification can be customized by adding different recipients or updating the notification body.

To configure the optional settings for the notification:

1. Go to the *System Configuration* screen.
2. Search for the setting `Ticket::DynamicTicketTemplate::CustomerGroup::Notify` and enable it to send a notification if a customer does not use a template to create a ticket.
3. Add agent login names to `Ticket::DynamicTicketTemplate::CustomerGroup::NotifyAgents` setting. The agent list in this setting will receive the notification when a user creates a ticket without using a template.

11.30 Service Management

The service desk (which, according to ITIL, is not a process but a function) is usually the ticket system's main field of application. All user messages and notifications from system monitoring and internal IT organization converge here. The ITIL service management process, closely interwoven with the service desk, describes which work steps, information, escalations and/or interfaces are relevant in connection with the processing of incidents or service requests.

This feature contains objects and basic functionalities needed for common features and processes of ITIL implementation. It contains the general catalog, which is the basic for ITSM relevant configurations in the service management. Additionally, adds statistics to the system for ensuring that all service level agreements are appropriate and satisfy the agreements, as well as to monitor and report on service levels.

The incident and problem management processes within OTRS::ITSM are based on ITIL recommendations and ITIL terminology. At the same time, user comfort was a main consideration, and terms known from OTRS have been retained as much as possible.

Some features of the service management are not enabled by default. This chapter guides you how to activate or setup these features.

11.30.1 Activate Incident and Problem Management

The incident and problem management feature is turned off by default.

To activate the incident and problem management feature:

1. Go to the *System Configuration* screen.
2. Search for the setting `IncidentProblemManagement::Active` and enable it.
3. Activate the actions in the following settings:

```
AgentFrontend::Ticket::Action###IncidentProblemManagementAdditionalFields
AgentFrontend::Ticket::Action###IncidentProblemManagementDecision
```

4. Enable the dynamic field property cards in the widget type configuration `AgentFrontend::TicketDetailView::WidgetType###Properties` by setting the value of the key `IsVisible` to `1` (available) or `2` (available and visible by default).

   The following dynamic field property cards have been added to the widget type configuration:

```
DynamicField_ITSMCriticality
DynamicField_ITSMImpact
DynamicField_ITSMReviewRequired
DynamicField_ITSMDecisionResult
DynamicField_ITSMRepairStartTime
DynamicField_ITSMRecoveryStartTime
DynamicField_ITSMDecisionDate
DynamicField_ITSMDueDate
```

5. Enable the dynamic fields in the form configurations by setting the key `Inactive` to the value `0`:

```
Forms###AgentFrontend::TicketCreate::Email::CreateProperties
- DynamicField_ITSMImpact
- DynamicField_ITSMDueDate

Forms###AgentFrontend::TicketCreate::Phone::CreateProperties
```

(continues on next page)

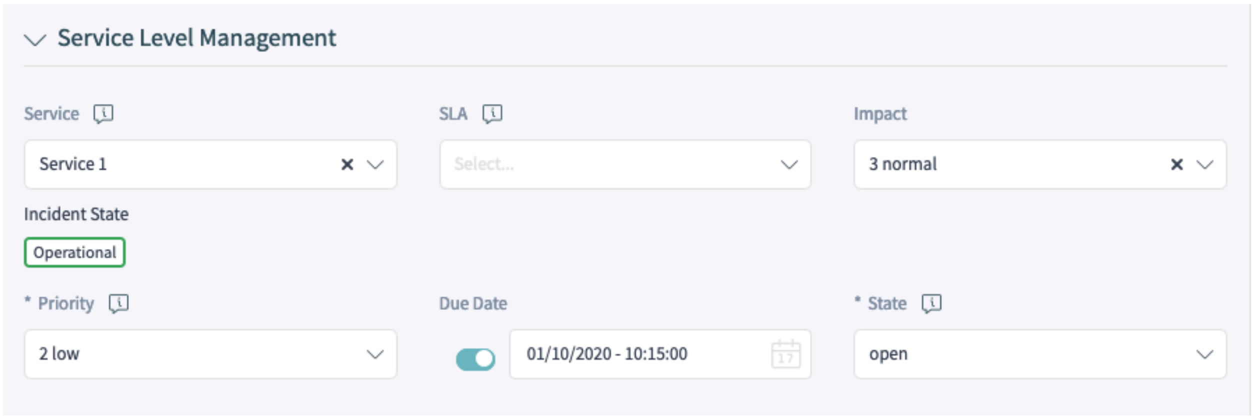
(continued from previous page)

```
- DynamicField_ITSMImpact
- DynamicField_ITSMDueDate

Forms###AgentFrontend::TicketCreate::SMS::CreateProperties
- DynamicField_ITSMImpact
- DynamicField_ITSMDueDate

Forms###AgentFrontend::Ticket::Action::Priority
- DynamicField_ITSMImpact

Forms###AgentFrontend::Ticket::Action::Close
- DynamicField_ITSMReviewRequired

Forms###AgentFrontend::TicketArticle::Action::Reply
- DynamicField_ITSMReviewRequired

Forms###AgentFrontend::TicketArticle::Action::ReplyAll
- DynamicField_ITSMReviewRequired
```

6. Deploy the modified configuration.

11.30.2 Hiding Service Incident State in Forms

This section describes how to hide the incident state field in a form if selecting a service. In default state, the incident state is shown in a form after selecting a service:



Service Level Management

Service: Service 1	SLA: Select...	Impact: 3 normal
Incident State: Operational		
* Priority: 2 low	Due Date: 01/10/2020 - 10:15:00	* State: open

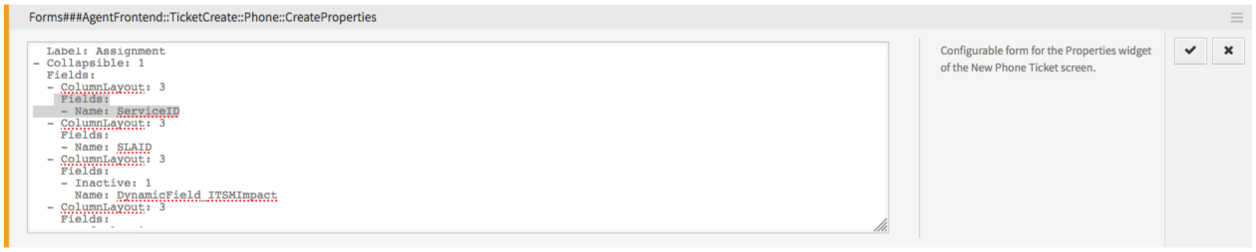
Fig. 27: Service Incident State in the New Phone Ticket Form

In order to hide the service incident state in a form, you need to edit the YAML configuration of the relevant form and add the following part:

```yaml
- Name: ServiceID
  Config:
    HideIncidentState: 1
```

The following example shows how to hide the service incident state for the *New Phone Ticket* form:

1. Search for the setting `Forms###AgentFrontend::TicketCreate::Phone::CreateProperties`.
2. Search for the `ServiceID` field in the YAML configuration:


Forms###AgentFrontend::TicketCreate::Phone::CreateProperties

```
  Label: Assignment
- Collapsible: 1
  Fields:
  - ColumnLayout: 3
    Fields:
    - Name: ServiceID
  - ColumnLayout: 3
    Fields:
    - Name: SLAID
  - ColumnLayout: 3
    Fields:
    - Inactive: 1
      Name: DynamicField_ITSMImpact
  - ColumnLayout: 3
    Fields:
```

Configurable form for the Properties widget of the New Phone Ticket screen.

Fig. 28: YAML Configuration of the Form

3. Add the `Config` key with the `HideIncidentState` sub-key set to `1`:

```
- Name: ServiceID
  Config:
    HideIncidentState: 1
```

4. Deploy the modified settings.

After successful deployment the service incident state will be hidden in the *New Phone Ticket* form:

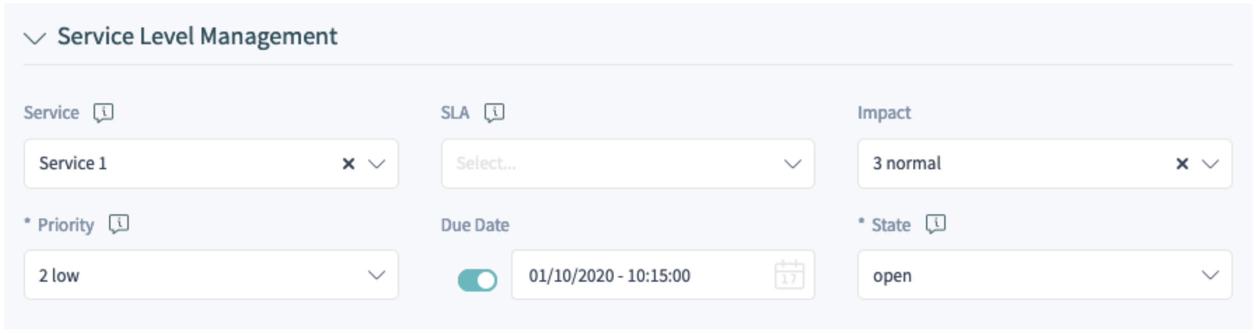


Fig. 29: New Phone Ticket Form without the Service Incident State

CHAPTER

TWELVE

GLOSSARY

Activity
Part of the process management.

Agent
An agent is a user which services customer users. The person working for your organization.

Back end
A configuration file, module, script or other functional item within OTRS which is not reachable via the browser.

Calendar
Defines working hours, time-zone, and holidays to be applied when using *escalation times*.

Customer
Physical organizations to which customer users are assigned. These provide the function to group users together which belong to a single organization.

Customer User
Physical persons, and their contact data, which belong to a customer. These are the people raising tickets.

Dynamic Field
A field used in tickets and articles to extend the data which can be saved in a ticket or article. They are saved from values in the database or accessed from outside resources.

Escalation Times
The defined amount of working time before a ticket breaches an agreement.

Front end
The graphical interface to OTRS as viewed in a browser.

Group
This is a resource within OTRS. An *agent* or a *customer* can be assigned permissions to this resource as needed. Additionally, they can be used for access control, processes and web services.

Invoker
An invoker is a special Perl module which allows OTRS to provide information to a remote system via REST or HTTP. Invokers must be developed in Perl by a back end developer.

Mapping
A mapping allows OTRS to provide inbound and outbound translation of data allowing us to offer a specific data construction to rebuild incoming structures to meet our needs.

Operation
Similar to a web hook, a provider operation offers a specific sub-set of OTRS Perl API functionality to external systems.

Process
: Part of the process management.

Process Path
: Part of the process management.

Resource
: A resource is any object which has effective permissions. This can be, but is not limited to, a ticket, dashboard widget, statistic, or module.

Screen
: A graphical interface to OTRS viewed in a browser (see also *Front end*).

Service
: A means of delivering added value to customers by making it easier for customers or helping them to achieve their desired results without them having to personally bear the responsibility for specific costs and risks.

Service Level Agreement
: A service level agreement (SLA) is a contract between a service provider (either internal or external) and the end user that defines the level of service expected from the service provider. SLAs are output-based in that their purpose is specifically to define what the customer will receive.

Service Request
: (see also *Ticket*).

Sequence Flow
: Part of the process management.

Sequence Flow Action
: Part of the process management.

SLA
: (see also *Service Level Agreement*).

Ticket
: A ticket is a collection of all communications with a customer during the course of a service request. A ticket contains article, which are the communication received from or sent to customers, agents, external systems, etc. Tickets belong to a customer user, are assigned to agents and reside in queues.

Transport
: A transport is the method chosen for communication. OTRS support REST and SOAP.

User Task Activity Dialog
: Part of the process management.

Working Time
: A definition of minutes passed during the defined working hours of any *calendar*.

INDEX