OTRS 3.1 - Admin Manual

OTRS 3.1 - Admin Manual

Copyright © 2003-2012 OTRS AG

René Bakker, Stefan Bedorf, Michiel Beijen, Shawn Beasley, Hauke Böttcher, Jens Bothe, Udo Bretz, Martin Edenhofer, Carlos Javier García, Martin Gruner, Manuel Hecht, Christopher Kuhn, André Mindermann, Marc Nilius, Elva María Novoa, Henning Oschwald, Martha Elia Pascual, Thomas Raith, Carlos Fernando Rodríguez, Stefan Rother, Burchard Steinbild, Daniel Zamorano.

This work is copyrighted by OTRS AG.

You may copy it in whole or in part as long as the copies retain this copyright statement.

The source code of this document can be found at source.otrs.org [http://source.otrs.org/viewvc.cgi/doc-admin/].

UNIX is a registered trademark of X/Open Company Limited. Linux is a registered trademark of Linus Torvalds.

MS-DOS, Windows, Windows 95, Windows 98, Windows NT, Windows 2000, Windows XP, Windows 2003, Windows Vista and Windows 7 are registered trademarks of Microsoft Corporation. Other trademarks and registered trademarks are: SUSE and YaST of SUSE Linux GmbH, Red Hat and Fedora are registered trademarks of Red Hat, Inc. Mandrake is a registered trademark of MandrakeSoft, SA. Debian is a registered trademark of Software in the Public Interest, Inc. MySQL and the MySQL Logo are registered trademarks of Oracle Corporation and/or its affiliates.

All trade names are used without the guarantee for their free use and are possibly registered trade marks.

OTRS AG essentially follows the notations of the manufacturers. Other products mentioned in this manual may be trademarks of the respective manufacturer.

Table of Contents

List of Tables

List of Examples

Preface

This book is intended for use by OTRS administrators. It also serves as a good reference for OTRS newbies.

The following chapters describe the installation, configuration and administration of the OTRS software. The first third of the text describes key functionality of the software, while the remainder serves as a reference to the full set of configurable parameters.

This book continues to be a work in progress, given a moving target on new releases. We need your feedback in order to make this a high quality reference document, one that is usable, accurate and complete. Please write to us if you find content missing in this book, if things are not explained well enough or even if you see spelling mistakes, grammatical errors or typos. Any kind of feedback is highly appreciated and should be made via our bug tracking system on *http://bugs.otrs.org* [http://bugs.otrs.org]. Thanks in advance for your contributions!

Chapter 1. Trouble Ticket Systems - The Basics

This chapter offers a brief introduction to trouble ticket systems, along with explaining the core concept of a trouble ticket. A quick example demonstrates the advantages of using such a system.

What is a trouble ticket system, and why do you need one?

The following example describes what a trouble ticket system is, and how you might benefit from using such a system at your company.

Let's imagine that Max is a manufacturer of video recorders. Max receives many mails from customers needing help with the devices. Some days, he is unable to respond promptly or even acknowledge the mails. Some customers get impatient and write a second mail with the same question. All mails containing support requests are stored in a single inbox file. The requests are not sorted, and Max answers the mails using a regular email program.

Since Max cannot reply fast enough to all the messages, he is assisted by the developers Joe and John in this. Joe and John use the same mail system, accessing the same inbox file. They don't know that Max often gets two identical requests from a desperate customer. Sometimes they both end up responding separately to the same request, with the customer receiving two different answers. Further, Max is unaware of the details of their responses. He is also unaware of the details of customer problems and their resolution, such as which problems occur with high frequency, or how much time and money he has to spend on customer support.

At a meeting, a colleague tells Max about trouble ticket systems and how they can solve Max's problems with customer support. After looking for information on the Internet, Max decides to install the Open Ticket Request System (OTRS) on a computer that is accessible from the web by both his customers and his employees. Now, the customer requests are no longer sent to Max's private inbox but to the mail account that is used for OTRS. The ticket system is connected to this mailbox and saves all requests in its database. For every new request, the system generates an auto-answer and sends it to the customer so that the customer knows that his request has arrived and will be answered soon. OTRS generates an explicit reference, the ticket number, for every single request. Customers are now happy because they receive an acknowledgement to their requests and it is not necessary to send a second message with the same question. Max, John and Joe can now login into OTRS with a simple web browser and answer the requests. Since the system locks a ticket that is answered, no message is edited twice.

Let's imagine that Mr. Smith makes a request to Max's company, and his message is processed by OTRS. John gives a brief reply to his question. But Mr. Smith has a follow-up question, which he posts via a reply to John's mail. Since John is busy, Max now answers Mr. Smith's message. The history function of OTRS allows Max to see the full sequence of communications on this request, and he responds with a more detailed reply. Mr. Smith does not know that multiple service representatives were involved in resolving his request, and he is happy with the details that arrived in Max's last reply.

Of course, this is only a short preview of the possibilities and features of trouble ticket systems. But if your company has to attend to a high volume of customer requests through mails and phone calls, and if different service representatives need to respond at different times, a ticket system

can be of great help. It can help streamline work flow processes, add efficiencies and improve your overall productivity. A ticket system helps you to flexibly structure your Support or Help Desk environment. Communications between customers and service staff become more transparent. The net result is an increase in service effectiveness. And no doubt, satisifed customers will translate into better financial results for your company.

What is a trouble ticket?

A trouble 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.

Trouble ticket systems such as OTRS handle trouble tickets like normal email. 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 be stored too with every email. Also, 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.

Chapter 2. OTRS Help Desk

This chapter describes the features of OTRS Help Desk (OTRS). You will find information about the hardware and software requirements for OTRS. Additionally, this chapter tells you how to get commercial support for OTRS, should you require it, and how to contact the community.

Basics

OTRS Help Desk (OTRS) is a web application which is installed on a web server and can be used with a web browser.

OTRS is separated into several components. The basic component is the OTRS framework that contains all central functions for the application and the ticket system. Via the web interface of the OTRS framework, it is possible to install additional applications such as ITSM modules, integrations with Network Monitoring solutions, a knowledge base (FAQ), et cetera.

Features

OTRS has many features. The following list gives an overview of the features included in the central framework.

The features of OTRS

- Web interface:
  - Easy and initial handling with any modern web browser, even with mobile phones or other mobile computers.
  - A web interface to administer the system via the web is available.
  - A web interface to handle customer requests by employees/agents via the web is integrated.
  - A web interface for customers is available to write new tickets, check the state and answer old tickets and search through their own tickets.
  - The web interface can be customized with different themes; own themes can be integrated.
  - Support for many languages.
  - The appearance of output templates can be customized (dtl).
  - Mails from and into the system can contain multiple attachments.
- Mail interface:
  - Support for mail attachments (MIME support).
  - Automatic conversion of HTML into plain text messages (more security for dangerous content and enables faster searching).
  - Mail can be filtered with the X-OTRS headers of the system or via mail addresses, e.g. for spam messages.

- PGP support, creation and import of own keys, signing and encrypting outgoing mail, signed and encrypted messages can be displayed.
- Support for viewing and encrypting S/MIME messages, handling of S/MIME certificates.
- Auto answers for customers, configurable for every queue.
- Email notifications for agents about new tickets, follow-ups or unlocked tickets.
- Follow-ups by references or In-Reply-To header entries.

- Tickets:
  - Expanded queue view, fast overview of new requests in a queue.
  - Tickets can be locked.
  - Creation of own auto answer templates.
  - Creation of own auto responders, configurable for every queue.
  - Ticket history, overview of all events for a ticket (changes of ticket states, replies, notes, etc.).
  - Print view for tickets.
  - Adding own (internal or external) notes to a ticket (text and attachments).
  - Ticket zooming.
  - Access control lists for tickets can be defined.
  - Forwarding or bouncing tickets to other mail addresses.
  - Moving tickets between queues.
  - Changing/setting the priority of a ticket.
  - The working time for every ticket can be counted.
  - Up-coming tasks for a ticket can be defined (pending features).
  - Bulk actions on tickets are possible.
  - Automatic and timed actions on tickets are possible with the "GenericAgent".
  - Full text search on all tickets is possible.
- System:
  - OTRS runs on many operating systems (Linux, Solaris, AIX, FreeBSD, OpenBSD, Mac OS 10.x, Microsoft Windows).
  - ASP support (active service providing).
  - Linking several objects is possible, e.g. tickets and FAQ entries.
  - Integration of external back-ends for the customer data, e.g. via AD, eDirectory or OpenLDAP.

- Setting up an own ticket identifier, e.g. Call#, Ticket# or Request#.
- The integration of your own ticket counter is possible.
- Support of several database systems for the central OTRS back-end, e.g. MySQL, PostgreSQL, Oracle, MSSQL).
- Framework to create stats.
- utf-8 support for the front- and back-end.
- Authentication for customers via database, LDAP, HTTPAuth or Radius.
- Support of user accounts, user groups and roles.
- Support of different access levels for several systems components or queues.
- Integration of standard answer texts.
- Support of sub queues.
- Different salutations and signatures can be defined for every queue.
- Email notifications for admins.
- Information on updates via mail or the web interface.
- Escalation for tickets.
- Support for different time zones.
- Simple integration of own add-ons or applications with the OTRS API.
- Simple creation of own front-ends, e.g. for X11, console.

New features of OTRS 3.1

1 GENERIC INTERFACE - A Web Service Framework

- GI is a flexible framework to allow web service interconnections of OTRS with third party applications.
- OTRS can act in both ways - as a provider (server, requested from remote) or requester (client, requesting remotely).
- Simple web service connections can be created without programming by configuring the Generic Interface.
- Complex scenarios can be realized by plugging in custom OTRS extensions that add perl code to the GI infrastructure on different architectural layers.
- *Connectors* expose parts of OTRS to Generic Interface web services.For example, a ticket connector exposes the ticket create/update function, so that they can be used in a web service regardless which network transport is used.
- A scheduler daemon process supports asynchronous event handling. This is useful to asynchronously start web service requests from OTRS to another system, after the agents

request has been answered (e.g. when a ticket has been created). Otherwise, it might block the response, resulting in increased response times for the agent.

With the Generic Interface new web services can be configured easily by using existing OTRS modules, without additional code. They can be combined to create a new web service. When configuring a new web service connection, the administrator has to add:

- A new web service in the admin GUI
- The basic meta data (Transport type (SOAP), URL etc.) and
- Existing operations (part of a connector) and specify for each operation how the data must be mapped (inbound and outbound)

A Generic Interface Debugger will help the OTRS administrator to check how requests are coming in and how they are handled through the different layers.

1.1 Current Features

- Network transports: SOAP/HTTP. Others like REST and JSON are scheduled to be added in the future depending on customers demand.
- Configurable data mapping Graphical User Interface for key/value transformations with respect to incoming and outgoing data.
- Graphical debugger to check the configuration and flow of information of configured web services.
- A ticket connector allowing the use of OTRS as a web service for ticket handling.

1.2 Future Features

- Additional network transports (REST, JSON).
- The GI will replace the iPhoneHandle as the backend for mobile apps.
- Additional connectors will be added to provide more parts of OTRS for use with web services (e.g. to allow the creation, update or deletion of agents, users, services or CIs).

2 DYNAMIC FIELDS

The DynamicFields Feature replaces the existing ticket and article FreeText and FreeTime fields with a dynamic structure that will also allow to create custom forms in OTRS.

- An unlimited amount of fields can be configured using an own graphical user interface for administration.
- The fields can have different types that can be used for both, tickets and articles. Available by default are:
  - Text
  - Multiline text
  - Checkbox
  - Dropdown

  - Multi-select
  - Date
  - Date and time
- New custom field types (e.g. custom field type dropdown with an external data source) can be added with small effort as the fields are created in a modular, pluggable way.
- A future scenario is, that DynamicFields can be used for objects other than tickets or in custom modules. For example, a custom module adding objects to handle "orders" in OTRS could use the DynamicFields to attach properties/data to these orders.
- A database update script will transform historic FreeText fields and related configuration settings into the new structure.

3 TICKET MANAGEMENT IMPROVEMENTS

3.1 Ticket creation improved

- Multiple email addresses can now be specified as 'To:', 'CC:' or 'BCC:' when creating a new phone or email ticket.

3.2 Inbound phone call support

- Inbound phone calls can now be registered within an existing tickets (until now, only outbound calls were registered).

3.3 Ticket overview preview improved

- It is now possible to exclude articles of certain sender types (e.g. articles from internal agents) in the SysConfig from being displayed in the overview preview mode.
- A certain article type can be configured which will display articles of that type as expanded by default when the view is accessed.

3.4 Ticket move improved

- The screen shown after moving a ticket is now configurable. Options are the ticket zoom view (LastScreenView) or the ticket list (LastScreenOverview).

3.5 Bulk action improved

- With the new bulk action, outbound emails can now be sent from multiple tickets at the same time. As tickets can have different queues, and these queues each can have different templates, salutations and signatures, these are not used in the Bulk Action email.
- An additional bulk action allows configuring the ticket type for selected tickets.

3.6 Configurable Reject Sender Email Address

- The feature allows configuring an email address instead of the administrator address to reject the creation of new tickets by email. This feature can be used in all cases where customers are not allowed to create new tickets by email.

4 PROCESS AUTOMATION

4.1 Escalation events added

- OTRS will now create events for each of the available escalation types (response, update and resolution). This allows performing actions (such as notifications) before the escalation occurs, in the moment it occurs and in the moment that the escalation ends.

4.2 Notification mechanism improved

- A new generic agent notification module allows the OTRS administrator to define messages that will be shown in the agent web front-end when agents log into the system.

4.3 Time calculation improved

- All kind of times are from now on calculated by and based on the application server only solving the issues that were caused by variances between the clock times of application and data base servers.

4.4 GenericAgent improved

- The GenericAgent can now filter for tickets change time.
- In addition, the GenericAgent can set the ticket responsible for matched tickets.

5 USER INTERFACE, RICH TEXT EDITOR, CHARSET

5.1 User interface performance improved

- The speed for rendering and article display was improved, thanks to Stelios Gikas <stelios.gikas@noris.net>!

5.2 Rich Text Editor Update

- IOS5 support added.
- Block quotes can be left with the enter key.
- Update from CKEditor 3.4 to CKEditor 3.6, so improvements refer to the releases of CKEditor 3.5 [http://ckeditor.com/blog/CKEditor_3.5_released] and CKEditor 3.6 [http://ckeditor.com/blog/CKEditor_3.6_released].
- IE9 support improved.
- Resizable dialogs.

5.3 Unicode Support - Non-UTF-8 Internal Encodings Dropped

- UTF-8 is now the only allowed internal charset of OTRS.
- All language files are now formatted in UTF-8, which simplifies their handling and future improvements of the translation mechanism.

6 DATABASE DRIVER SUPPORT

6.1 PostgreSQL DRIVER compatibility improved

- PostgreSQL 9.1 support added.

- A new legacy driver is now available for PostgreSQL 8.1 or earlier versions.

6.2 MS SQL DRIVER compatibility improved

- The MS SQL driver now stores binary data in VARBINARY rather than deprecated type TEXT as well as NVARCHAR to store text strings rather than VARCHAR (for improved Unicode support).

7 MAIL INTEGRATION

7.1 Mail handling improved

- When connecting to IMAP mail accounts, it is now possible to handle emails from a specific email folder, other than the INBOX folder.
- OTRS can now also connect to IMAP servers using Transport Layer Security (TLS), useful for modern restricted environments.

Top new features of OTRS 3.0

Context

- User Centered redesign of the Graphical User Interface which results in a dramatic shift from a comprehensive but static to a more powerful and dynamic application using state-of-the art technologies like Ajax, xHTML and optimized CSS.

New Ticket and Article Indicator

- This new feature has been implemented on both ticket and article level. It allows an agent at a glance to check for any updates within a ticket or on the article level to check for new and unread articles. You benefit from increased transparency and decreased response times.

Optimized Fulltext Search

- The new search feature allows you to flexibly customize the way you browse the information base. Options the new search feature provides range from single search-string searches to complex multi-string boolean search operations including various operators. You benefit from fully customizable searches according to your needs.

New Ticket Zoom View

- The redesign based on Ajax technology allows agents to display complex and linked information structures in real-time while keeping the agents' current working environment. The agent will benefit from increased orientation and increased workflow efficiency.

Global Ticket Overviews

- Well known from OTRS 2.4 the global ticket overviews have been optimized to achieve increased inter- activity. Depending on the use case and preferences of your agents they can easily change the ticket overviews layout according to their special needs. Options are small, medium and large, each providing a different degree of information details.

Accessibility

- The redesign includes common accessibility standards WCAG and WAI-ARIA which also allows disabled users to better interact with OTRS Help Desk. The US Rehabilitation Acts Section 508 has been fulfilled.

New Customer Interface

- The customer web front-end can be integrated to your organizations intranet and is fully integrated into the redesigned help desk system.

Archive Feature

- OTRS 3.0 now offers a new archiving feature. With a separated archive you'll benefit from a reduced time spent for searches and increased display of results.

New features of OTRS 2.4

Licensing changed to AGPL Version 3

- Why AGPL instead of GPL? - AGPL and GPL are identical, with one exception: For software used in an SaaS environment Copyleft is effective in AGPL - which is not the case when using GPL. Keeping in mind the growing world of SaaS, ((otrs)) wants to ensure that future developments continue to return to the OTRS community. This is the reason for the switch to AGPL.

  Why v3 instead of v2? - GPL v2 is getting older and has, especially in the USA, various legal uncertainties. In the opinion of ((otrs)) GPL v3 is keeping the spirit of GPL v2, and at the same time has been tailored to new needs. ((otrs)) views GPLv3, more specifically AGPLv3, as being the best balanced Copyleft Open Source License available today, offering Protection for copyright owners and users and providing the best security under the law.

New Management Dashboard

- The need for a system-spanning, next to real-time, and personalized presentation of useful information led to an integrated Management Dashboard. It is possible to create plug-ins to display content from individual extensions alongside the standard content. Standard plug-ins are:
- Ticket volume (new & open) from the last 24h, 48h and 72h
- Calendar including an overview of upcoming events (escalations, auto-unlocks, etc.)
- System-wide overview of ticket distribution within the queues
- First Response Time/Solution Time of Queues
- Integration of RSS

New Standard Reports

- The new reports provided with OTRS 2.4 are:
- Created Tickets

- Closed Tickets
- SLA Analysis
- Required working time per customer / per queue
- Solution time analysis per customer / per queue
- Answer time analysis per customer / per queue

New Master/Slave Ticket Feature

- With the Master/Slave Ticket, it is possible to link multiple tickets of a similar nature, and handle them collectively. As soon as the problem is solved, only the master ticket must be closed. All other tickets will be closed automatically, and the solution text for the master ticket will be sent to all customers of slave tickets.

  A new link type 'Slave' will be available. All tickets with this Type of link will inherit the following actions from their Master ticket:

- Status change
- Email answers
- Change in FreeText fields
- Notes
- Pending time changes
- Priority changes
- Owner changes
- Responsibility changes

New Rich-Text/HTML E-Mail Support (WYSIWYG)

- With this feature, it is now possible to write e-mails, notes, and notifications in rich text format (HTML format). Using a WYSIWYG editor (What You See Is What You Get), it is possible to comfortably write using formatted text and even include in-line pictures.

New Out-Of-Office Feature

- With this new feature it is possible for all users to activate "out-of-office" to notify colleagues and OTRS of the period of their absence. The out-of-office feature is active for a time frame set by the user. Activation of this feature has the following effects:

  In the lists in which an agent can be selected as owner or responsible (i.e. Ticket creation or changing ownership), the period of absence and the time till return will be shown behind the user's name. This will help making the absence of the user more transparent.

  If an agent receives a follow-up during a period of absence, the ticket is automatically unlocked and a notification is sent to all agents in the queue. This allows immediate reaction to the customer follow-up by another service employee.

New Ticket Overviews and global Bulk Action

- Flexibility of presentation within the ticket overview is a must. Based on the "S/M/L" (Small/ Medium/Large) Ticket View every agent has the possibility to change the view for each type of overview (Queue View, Status View, etc) on-the-fly with a simple mouse click on the appropriate icon. This allows for the highest possible level of individualization and adjustment to any operational situation.

  Additionally, decentralization of the Bulk Action feature integrated the Bulk Action in all ticket overviews (Bulk Action allows processing of multiple tickets at a time).

Postmaster Filter recognizes Follow-Ups to internal forwarded messages

- Currently, e-mail replies to forwarded articles arrive in OTRS as email-external. The problem is that the answers to these forwarded articles can be seen by the customer in the web- interface. Although it is possible to classify e-mails of an entire domain as email-internal, this only shifts the problem. Also, such step makes it impossible to properly service customers in the domain, as the customer would not be able to track tickets in the customer web-interface any more. With this new feature, e-mail replies can be traced back, and email- internal or email-external will be set based upon the original Forward-Article type.

Configurable event based notifications

- Until now, a very inflexible notification could be sent to an agents and customers, for example Agent: New Ticket or Customer: Status Change. In order to make the notification system more flexible, a complete overhaul was performed on the messaging mechanism. The new system allows messaging to agents, customers, or a dedicated email address, based on the event taking place.

  With this, it is now possible to just inform the customer when the ticket has been closed. Or, for example, when a VIP customer creates a ticket, a message can be sent to a specific address. Events (i.e. TicketCreate, TicketStateUpdate, TicketPriorityUpdate, ArticleCreate), and all known message variables (i.e. <OTRS_TICKET_TicketNumber> <OTRS_TICKET_Priority>), are freely selectable for creating triggered messages via the web interface.

READ-ONLY Permissions and Notifications with watched Tickets

- In the current release of OTRS it is possible for a user to maintain a Watched Tickets List. This feature is dealing with tickets marked as "subscribed" by a user. It has the advantage that users no longer lose track of tickets marked as "sub- scribed", and are able to view them on an individual list. The "Read-Only" Feature - Up to now, tickets marked as "sub- scribed" were shown in a list, however, the agent could only actually view them if they were in a queue for which the agent had read permissions. With the "Read-Only" Feature, agents subscribed to a ticket always have read permissions on the ticket, even if the ticket is moved to a queue where the agent has no permissions. "Notify" Feature - Via a personalized setting, every agent can define whether or not to receive notifications about tickets, just as the owner and responsible of a ticket would receive. This allows for active tracking of watched tickets.

Secure SMTP

- OTRS can receive and send mails in multiple ways. All currently available methods for receiving emails have been implemented within OTRS 2.3 (POP3,POP3S,IMAP,IMAPS). Until now, there were two options for sending emails: using a local MTA (Sendmail, Postfix, etc.) or per SMTP. In OTRS 2.4.x, SMTPS (Secure SMTP) has been implemented in order to keep up to the growing security standards.

Hardware and software requirements

OTRS can be installed on many different operating systems. OTRS can run on linux and on other unix derivates (e.g. OpenBSD or FreeBSD). You can also deploy it on Microsoft Windows. OTRS does not have excessive hardware requirements. We recommend using a machine with at least a 2 GHz Xeon or comparable CPU, 2 GB RAM and a 160 GB hard drive for a small setup.

To run OTRS you'll also need to use a web server and a database server. Apart from that, on the OTRS machine, you should install perl and/or install some additional perl modules. The web server and Perl have to be installed on the same machine as OTRS. The database back-end can be installed locally or on another host.

For the web server we recommend using the Apache HTTP Server, because its module mod_perl improves greatly the performance of OTRS. Apart from that, OTRS should run on any web server that can execute Perl scripts.

You can deploy OTRS on different databases. You can choose between MySQL, PostgreSQL, Oracle, or Microsoft SQL Server. If you use MySQL you have the advantage that the database and some system settings can be configured during the installation, through a web front-end.

For Perl, we recommend using at least version 5.8.8. You need some additional modules which can be installed either with the Perl shell and CPAN or via the package manager of your operating system (rpm, yast, apt-get).

Software requirements

Perl support

- Perl 5.8.8 or higher

Web server support

- Apache2 + mod_perl2 or higher (recommended, mod_perl is really fast!)
- Webserver with CGI support (CGI is not recommended)
- Microsoft Internet Information Server (IIS) 6 or higher

Database support

- MySQL 4.1 or higher
- PostgreSQL 7.0 or higher (8.2 or higher recommended)
- Oracle 10g or higher
- Microsoft SQL Server 2005 or higher

The section in the manual about installation of Perl modules describes in more detail how you can set up those which are needed for OTRS.

If you install a binary package of OTRS, which was built for your operating system (rpm, Windows-Installer), either the package contains all Perl modules needed or the package manager of your system should take care of the dependencies of the Perl modules needed.

Web browser support

For the Agent interface of OTRS, you'll be OK if you use a modern browser with JavaScript support enabled. We support the following browsers:

- Internet Explorer 8.0 or higher
- Mozilla Firefox 3.6 or higher
- Google Chrome
- Opera 10 or higher
- Safari 4 or higher

We recommend to always use the latest version of your browser, because it has the best JavaScript and rendering performance. Dramatical performance varieties between the used browsers can occur with big data or big systems. We are happy to consult you on that matter.

For the OTRS Customer Interface, in addition to the browsers listed above, you can also use Internet Explorer versions 6 or 7, and we do not require JavaScript either.

Community

OTRS has a large user community. Users and developers discuss about OTRS and interchange information on related issues through the mailing-lists. You can use the mailing lists to discuss installation, configuration, usage, localization and development of OTRS. You can report software bugs in our bug tracking system.

The homepage of the OTRS community is: *http://www.otrs.com/open-source/*.

Commercial Support and Services for OTRS

Commercial support for OTRS is also available. You can find the available options on the website of OTRS Group, the company behind OTRS: *http://www.otrs.com/*.

OTRS Group provides subscription support services, customization, consulting and training for *OTRS Help Desk* [http://www.otrs.com/products/otrs-help-desk/] and *OTRS ITSM* [http://www.otrs.com/en/products/otrs-itsm/]. It also provides *Best Practice Product Editions* [http://www.otrs.com/solutions/]. With these Editions, OTRS Group helps organizations to design, deploy and optimize OTRS for each unique environment. Additionally, OTRS Group provides hosted versions including *OTRS OnDemand* [http://www.otrs.com/en/solutions/ondemand/] and *Managed OTRS* [http://www.otrs.com/en/solutions/managed-otrs/].

You can find more detailed information about OTRS Group on *http://www.otrs.com* and you can contact us via email on *sales at otrs.com* [mailto:sales@otrs.com].

Chapter 3. Installation

This chapter describes the installation and basic configuration of the central OTRS framework. It covers information on installing OTRS from source, or with a binary package such as an RPM or a Windows exectuable.

Topics covered here include configuration of the web and database servers, the interface between OTRS and the database, the installation of additional Perl modules, setting proper access rights for OTRS, setting up the cron jobs for OTRS, and some basic settings in the OTRS configuration files.

Follow the detailed steps in this chapter to install OTRS on your server. You can then use its web interface to login and administer the system.

The simple way - Installation of pre-built packages

You should use pre-built packages to install OTRS, since it is the simplest and most convenient method. You can find them in the download area at *http://www.otrs.org* [http://www.otrs.org] . The following sections describe the installation of OTRS with a pre-built or binary package on SUSE, Debian and Microsoft Windows systems. Only if you are unable to use the pre-built packages for some reason should you follow the manual process.

Installing the RPM on a SUSE Linux server

This section demonstrates the installation of a pre-built RPM package on a SUSE Linux distro. We have tested against all recent SLES and openSUSE versions. Before you start the installation, please have a look at *http://www.otrs.org/downloads* [http://www.otrs.org/downloads] and check if a newer OTRS RPM package is available. Always use the latest RPM package.

Install OTRS with yast (yast2) or via the command line and **rpm**. OTRS needs some Perl modules which are not installed on a SUSE system by default, and so we recommend using yast, since it addresses the package dependencies automatically.

If you decide to install OTRS via the command line and **rpm**, first you have to manually install the needed Perl modules. Assuming you saved the file `otrs.rpm` in the directory `/tmp`, you can execute the command specified in the following script to install OTRS.

```
linux:~ # rpm -i /tmp/otrs-xxx.rpm
otrs
 ################################################
Check OTRS user (/etc/passwd)... otrs exists.

Next steps:

[SuSEconfig]
 Execute 'SuSEconfig' to configure the web server.

[start Apache and MySQL]
 Execute 'rcapache restart' and 'rcmysql start' in case they don't
 run.
```

```
[install the OTRS database]
 Use a web browser and open this link:
 http://localhost/otrs/installer.pl

[OTRS services]
 Start OTRS 'rcotrs start-force' (rcotrs {start|stop|status|restart|
start-force|stop-force}).

Have fun!

 Your OTRS Team
 http://otrs.org/

linux:~ #
```

Script: Command to install OTRS.

After the installation of the OTRS RPM package, you have to run SuSEconfig, as shown in the following script.

```
linux:~ # SuSEconfig
Starting SuSEconfig, the SuSE Configuration Tool...
Running in full featured mode.
Reading /etc/sysconfig and updating the system...
Executing /sbin/conf.d/SuSEconfig.aaa_at_first...
Executing /sbin/conf.d/SuSEconfig.apache...
Including /opt/otrs/scripts/apache-httpd.include.conf
Executing /sbin/conf.d/SuSEconfig.bootsplash...
Executing /sbin/conf.d/SuSEconfig.doublecheck...
Executing /sbin/conf.d/SuSEconfig.guile...
Executing /sbin/conf.d/SuSEconfig.hostname...
Executing /sbin/conf.d/SuSEconfig.ispell...
Executing /sbin/conf.d/SuSEconfig.perl...
Executing /sbin/conf.d/SuSEconfig.permissions...
Executing /sbin/conf.d/SuSEconfig.postfix...
Setting up postfix local as MDA...
Setting SPAM protection to "off"...
Executing /sbin/conf.d/SuSEconfig.profiles...
Finished.
linux:~ #
```

Script: Running the SuSEconfig command.

The OTRS installation is done. Restart your web server to load the OTRS specific changes in its configuration, as shown in the script below.

```
linux:~ # rcapache restart
Shutting down httpd                                          done
Starting httpd [ PERL ]                                      done
linux:~ #
```

Script: Restarting the web server.

The next step is to setup the OTRS database, as described at section 3.2.4.

Installing OTRS on a CentOS system

On the OTRS Wiki you can find detailed instructions for setting up OTRS on a CentOS system. Please note that these instructions will also apply to Red Hat Linux systems since they use the same source: http://wiki.otrs.org/index.php?title=Installation_of_OTRS_3.0b1_on_CentOS_5.5 [http://wiki.otrs.org/index.php?title=Installation_of_OTRS_3.0b1_on_CentOS_5.5] .

Installing OTRS on a Debian system

On the OTRS Wiki you can find detailed instructions for setting up OTRS on a Debian system: http://wiki.otrs.org/index.php?title=Installation_on_Debian_5.04_lenny [http://wiki.otrs.org/index.php?title=Installation_on_Debian_5.04_lenny] .

Installing OTRS on a Ubuntu system

On the OTRS Wiki you can find detailed instructions for setting up OTRS on an Ubuntu system: http://wiki.otrs.org/index.php?title=Installation_on_Ubuntu_Lucid_Lynx_(10.4) [http://wiki.otrs.org/index.php?title=Installation_on_Ubuntu_Lucid_Lynx_(10.4)] .

Installing OTRS on Microsoft Windows systems

Installing OTRS on a Microsoft Windows system is very easy. Download the latest installer for Win32 from *http://www.otrs.org/downloads/* [http://www.otrs.org/downloads/] and save the file to your local file system. Then simply double-click on the file to execute the installer, and follow the few installation steps to setup the system. After that you will be able to login as OTRS administrator and configure the system according to your needs. To log in as OTRS administrator use the username 'root@localhost' and the default password 'root'.

Warning

Please change the password for the 'root@localhost' account as soon as possible.

Important

The Win32 installer for OTRS contains all needed components for OTRS, i.e. the Apache web server, the MySQL database server, Perl (with all needed modules) and Cron for Windows. For that reason you should only install OTRS on Windows systems that don't already have an installation of Apache or another web server, or MySQL.

Installation from source (Linux, Unix)

Preparing the installation from source

If you want to install OTRS from source, first download the source archive as .tar.gz, .tar.bz2, or .zip file from *http://www.otrs.org/downloads/* [http://www.otrs.org/downloads/]

Unpack the archive (for example, using **tar**) into the directory `/opt`, and rename the directory from otrs-3.1.x to otrs (see Script below).

```
linux:/opt# tar xf /tmp/otrs-3.1.tar.gz
linux:/opt# mv otrs-3.1 otrs
linux:/opt# ls
otrs
linux:/opt#
```

Script: First steps to install OTRS.

OTRS should NOT be run with root rights. You should add a new user for OTRS as the next step. The home directory of this new user should be `/opt/otrs`. If your web server is not running with the same user rights as the new 'otrs' user, which is the case on most systems, you have to add the new 'otrs' user to the group of the web server user (see Script below).

```
linux:/opt# useradd -r -d /opt/otrs/ -c 'OTRS user' otrs
linux:/opt# usermod -G nogroup otrs
linux:/opt#
```

Script: Adding a new user 'otrs', and adding it to a group.

Next, you have to copy some sample configuration files. The system will later use the copied files. The files are located in `/opt/otrs/Kernel` and `/opt/otrs/Kernel/Config` and have the suffix .dist (see Script below).

```
linux:/opt# cd otrs/Kernel/
linux:/opt/otrs/Kernel# cp Config.pm.dist Config.pm
linux:/opt/otrs/Kernel# cd Config
linux:/opt/otrs/Kernel/Config# cp GenericAgent.pm.dist GenericAgent.pm
```

Script: Copying some sample files.

The last step to prepare the installation of OTRS is to set the proper access rights for the files. You can use the script **otrs.SetPermissions.pl**, which is located in the `bin` directory, in the home directory of the 'otrs' user. You can execute the script with the following parameters:

`otrs.SetPermissions.pl` { *`Home directory of the OTRS user`* } { --otrs-user= *`OTRS user`* } { --web-user= *`Web server user`* } [--otrs-group= *`Group of the OTRS user`*] [--web-group= *`Group of the web server user`*]

If your web server is running with the same user rights as user 'otrs', the command to set the proper access rights is **otrs.SetPermissions.pl /opt/otrs --otrs-user=otrs --web-user=otrs**. On SUSE systems the web server is running with the user rights of 'wwwrun'. On Debian-based systems this is 'www-data'. You would use the command **otrs.SetPermissions.pl /opt/otrs --otrs-user=otrs --web-user=wwwrun --otrs-group=nogroup --web-group=www** to set the proper access rights.

Installation of Perl modules

OTRS needs some additional Perl modules, as described in Table 3-1. If you install OTRS from source, you will have to install these modules manually. This can be done either with the package manager of your Linux distribution (yast, apt-get) or, as described in this section, through the Perl shell and CPAN. If you're using ActiveState Perl, for instance on Windows, you could use PPM, the built-in Perl Package Manager. We recommend using your package manager if possible.

Table 3.1. Needed Perl modules for OTRS

Name	Description
DBI	Establishes a connection to the database back-end.
DBD::mysql	Contains special functions to connect to the MySQL database back-end (only required if MySQL is used).
DBD::pg	Contains special functions to connect to the PostgreSQL database back-end (only required if PostgreSQL is used).
Digest::MD5	Allows the use of the md5 algorithm.
CSS::Minifier	Minifies a CSS file and writes the output directly to another file.
Crypt::PasswdMD5	Provides interoperable MD5-based crypt functions.
MIME::Base64	Encodes / decodes Base64 strings, e.g. for mail attachments.
JavaScript:Minifier	Minifies a JavaScript file and writes the output directly to another file.
Net::DNS	Perl interface to the domain name system.
LWP::UserAgent	Processes HTTP requests.
Net::LDAP	Perl interface to a LDAP directory (only required if an LDAP back-end is used).
GD	Interface to the GD graphics library (only required if the OTRS stats module is used).
GD::Text, GD::Graph, GD::Graph::lines, GD::Text::Align	Some more text and graphic tools for the GD graphics library (only required if the OTRS stats module is used).
PDF::API2, Compress::Zlib	Needed to generate the PDF output for reports, search results and for the ticket print view.

You can verify which modules you need to install with **otrs.CheckModules.pl**. This script is located in the `bin` directory, in the home directory of the 'otrs' user (see Script below).

Please note that some modules are optional.

```
linux:~# cd /opt/otrs/bin/
linux:/opt/otrs/bin# ./otrs.CheckModules.pl
   o CGI............................ok (v3.49)
   o Crypt::PasswdMD5...............ok (v1.3)
   o CSS::Minifier..................ok (v0.01)
   o Date::Format...................ok (v2.24)
   o Date::Pcalc....................ok (v1.2)
   o DBI............................ok (v1.609)
   o DBD::mysql.....................ok (v4.013)
   o Digest::MD5....................ok (v2.36_01)
   o Encode::HanExtra...............ok (v0.23)
```

```
  o GD.............................ok (v2.44)
     o GD::Text....................ok (v0.86)
     o GD::Graph...................ok (v1.44)
     o GD::Graph::lines............ok (v1.15)
     o GD::Text::Align.............ok (v1.18)
  o IO::Scalar.....................ok (v2.110)
  o IO::Wrap.......................ok (v2.110)
  o JavaScript::Minifier...........ok (v1.05)
  o JSON...........................ok (v2.21)
     o JSON::PP....................ok (v2.27003)
     o JSON::XS....................Not installed! (Optional - Install
 it for faster AJAX/JavaScript handling.)
  o LWP::UserAgent.................ok (v5.829)
  o Mail::Internet.................ok (v2.06)
  o Mail::POP3Client...............ok (v2.18 )
     o IO::Socket::SSL.............ok (v1.31)
  o MIME::Base64...................ok (v3.07_01)
  o MIME::Tools....................ok (v5.428)
  o Net::DNS.......................ok (v0.65)
  o Net::POP3......................ok (v2.29)
  o Net::IMAP::Simple..............ok (v1.1916)
     o Net::IMAP::Simple::SSL......ok (v1.3)
  o Net::SMTP......................ok (v2.31)
     o Authen::SASL................ok (v2.15)
     o Net::SMTP::SSL..............ok (v1.01)
  o Net::LDAP......................ok (v0.4001)
  o PDF::API2......................ok (v0.73)
     o Compress::Zlib..............ok (v2.008)
  o SOAP::Lite.....................ok (v0.712)
  o Text::CSV......................ok (v1.18)
     o Text::CSV_PP................ok (v1.26)
     o Text::CSV_XS................Not installed! (Optional -
 Optional, install it for faster CSV handling.)
  o XML::Parser....................ok (v2.36)
linux:/opt/otrs/bin#
```

Script: Checking needed modules.

You should strive to install the missing modules from your Linux distribution's package management system. In that way, the packages will be automatically updated when new versions are available or when security issues are found. Please refer to your distribution's documentation on how to install additional packages. If the (correct version of) the module is not available from the package repositories, you can also install from CPAN, the Comprehensive Perl Archive Network.

To install one of the modules from above via CPAN, you have to execute the command **perl -e shell -MCPAN**. The Perl shell will be started in interactive mode and the CPAN module will be loaded. If CPAN is already configured, you can install the modules with the command **install** followed by the name of the module. CPAN takes care of the dependencies of a module to other Perl modules and will let you know if other modules are needed.

Execute also the commands **perl -cw bin/cgi-bin/index.pl perl -cw bin/cgi-bin/customer.pl** and **perl -cw bin/otrs.PostMaster.pl** after changing into the directory `/opt/otrs`. If the output of both commands is "syntax OK", your Perl is properly set up (see Script below).

```
linux:~# cd /opt/otrs
linux:/opt/otrs# perl -cw bin/cgi-bin/index.pl
cgi-bin/installer.pl syntax OK
linux:/opt/otrs# perl -cw bin/cgi-bin/customer.pl
cgi-bin/customer.pl syntax OK
linux:/opt/otrs# perl -cw bin/otrs.PostMaster.pl
bin/otrs.PostMaster.pl syntax OK
linux:/opt/otrs#
```

Script: Syntax check.

Configuring the Apache web server

This section describes the basic configuration of the Apache web server with mod_cgi for OTRS. The web server should be able to execute CGI scripts. OTRS won't work if the Perl scripts cannot be parsed. Check the configuration files of your web server, and search for the line that loads the CGI module. If you see something like the following, the CGI module should already be in use.

LoadModule cgi_module /usr/lib/apache2/modules/mod_cgi.so

To access the web interface of OTRS conveniently via a short address, Alias and ScriptAlias entries are needed. Most Apache installations have a `conf.d` directory included. On Linux systems you can find this directory very often under `/etc/apache` or `/etc/apache2`. Log in as root, change to the `conf.d` directory and copy the appropriate template in `/opt/otrs/scripts/apache2-httpd.include.conf` to a file called `otrs.conf` in the Apache configuration directory.

Restart your web server to load the new configuration settings. On most systems you can start/restart your web server with the command **/etc/init.d/apache2 restart** (see Script below).

```
linux:/etc/apache2/conf.d# /etc/init.d/apache2 restart
Forcing reload of web server: Apache2.
linux:/etc/apache2/conf.d#
```

Script: Restarting the web server.

Now your web server should be configured for OTRS.

If you choose to increase performance and you can install mod_perl, then you can leave mod_cgi off, and configure the Apache web server for use with mod_perl, in the following manner:

Please ensure that mod_perl is installed and loaded, in order to take advantage of this feature. Due to the nature of the start-up script, your server will not fail to start if mod_perl is not properly loaded or compiled in your apache web server, unless mod_cgi is also on. Technically speaking you can leave mod_cgi on as well, but you should not.

Search your /etc/apache* directory for mod_perl.so (see Script below) to see if the module is already loaded.

```
 #:/ grep -Rn mod_perl.so /etc/apache*
```

Script: Searching for mod_perl.

When you use the appropriate start script listed above and the module is loaded, the script (when commented in) /opt/otrs/scripts/apache2-perl-startup.pl can be used to load the perl modules into memory one time, saving on load times and increasing performance.

Configuring the database

The simple way - Using the web installer (works only with MySQL)

If you use MySQL as the database back-end, you can use the OTRS web installer: *http://localhost/otrs/installer.pl* [http://localhost/otrs/installer.pl] .

When the web installer starts, please follow the next steps to setup your system:

1. Check out the information about the OTRS offices and click on next to continue (see Figure below).

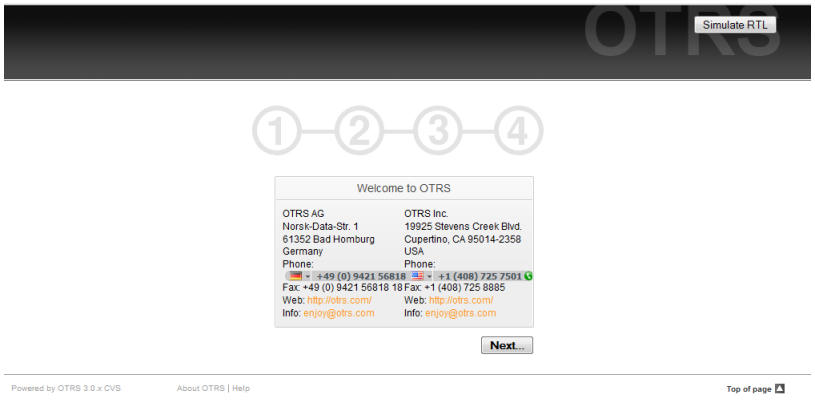

Figure: Welcome screen.

2. Read the GNU Affero General Public License (see Figure below) and accept it, by clicking the corresponding button at the bottom of the page.

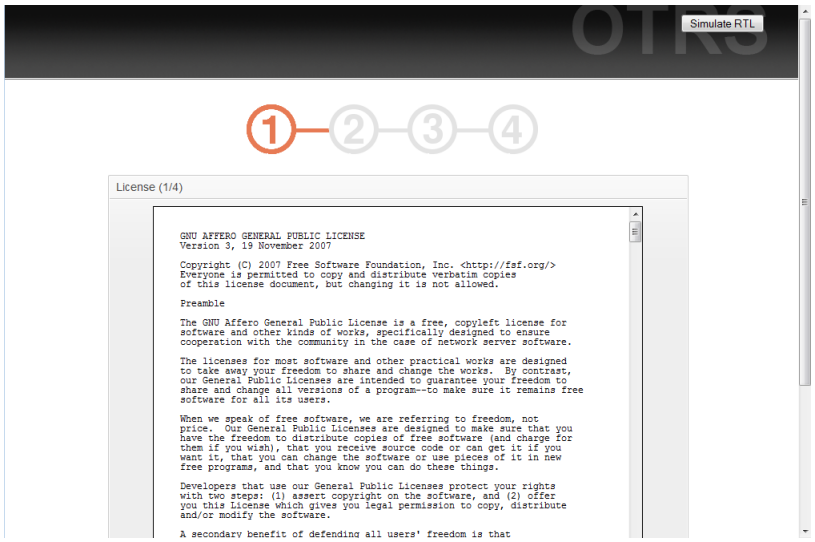

Figure: GNU Affero General Public License.

3. Provide the username and password of the administrator, the DNS name of the computer which hosts OTRS and the type of database system to be used. After that, check the settings (see Figure below).

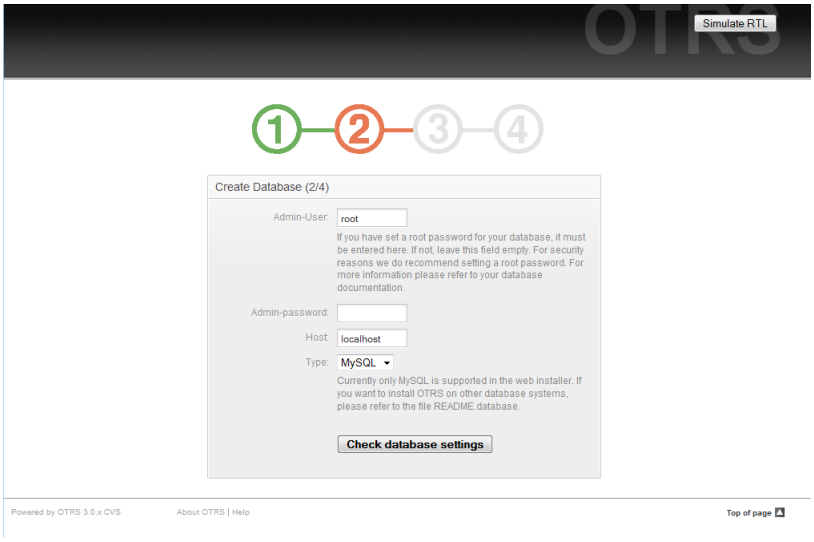

Figure: Database initial settings.

You will be notified if the check was successful. Press OK to continue (see Figure below).

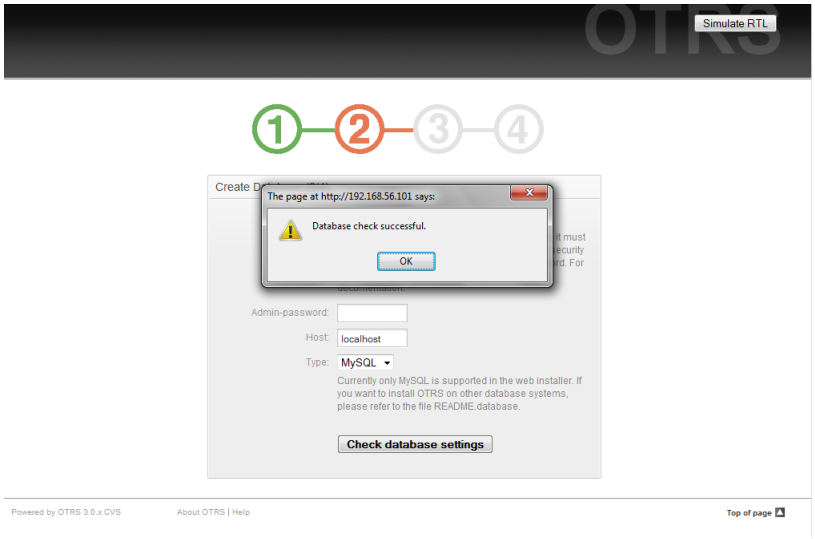

Figure: Notification for successful check.

4. Create a new database user, choose a name for the database and click on 'Next' (see Figure below).

Warning

It is never a good idea to use default passwords. Please change the default password for the OTRS database!

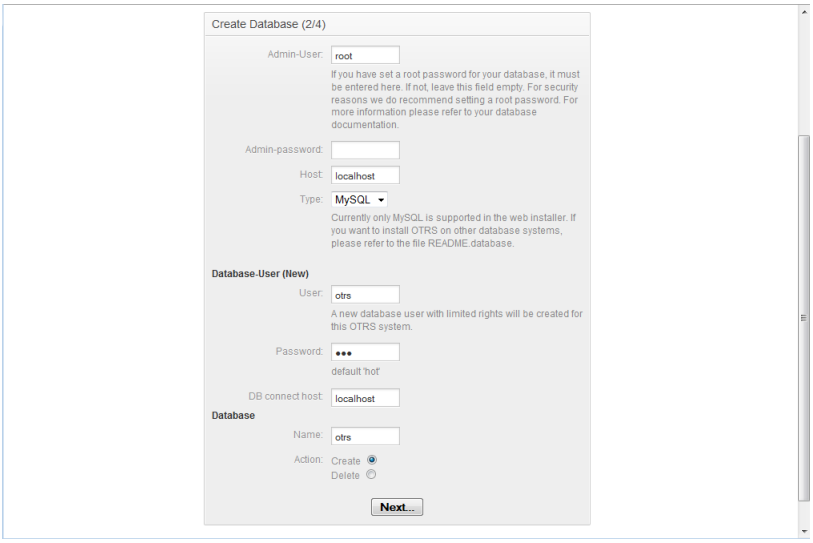

Figure: Database settings.

If the database and its user were successfully created, you will get a setup notification, as shown in Figure. Click 'Next' to go to the next screen.

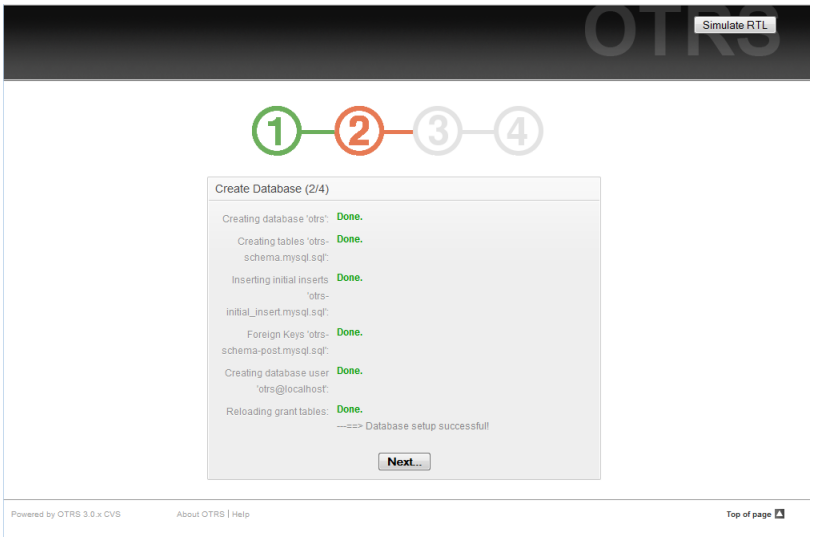

Figure: Notification indicating successful database setup.

5. Provide all the required system settings and click on 'Next' (see Figure below).

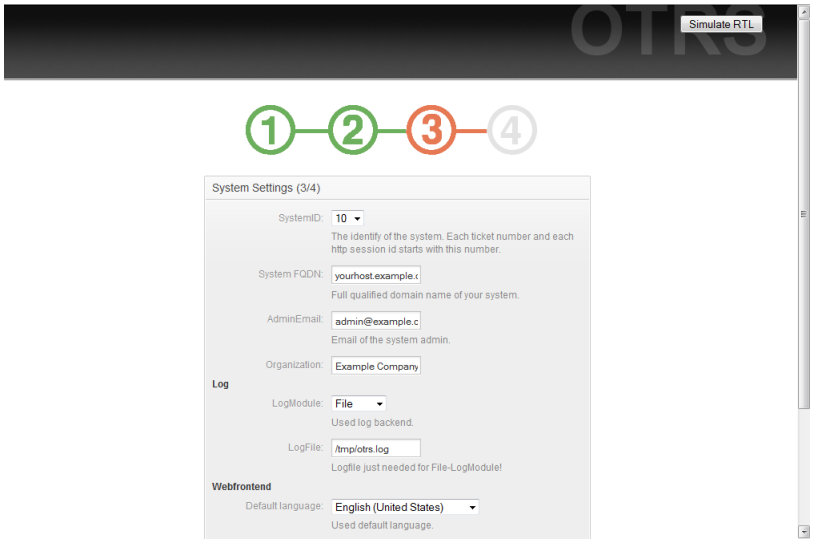

Figure: System settings.

6. If you want, you can provide the needed data to configure your inbound and outbound mail, or skip this step by pressing the right button at the bottom of the screen (see Figure below).

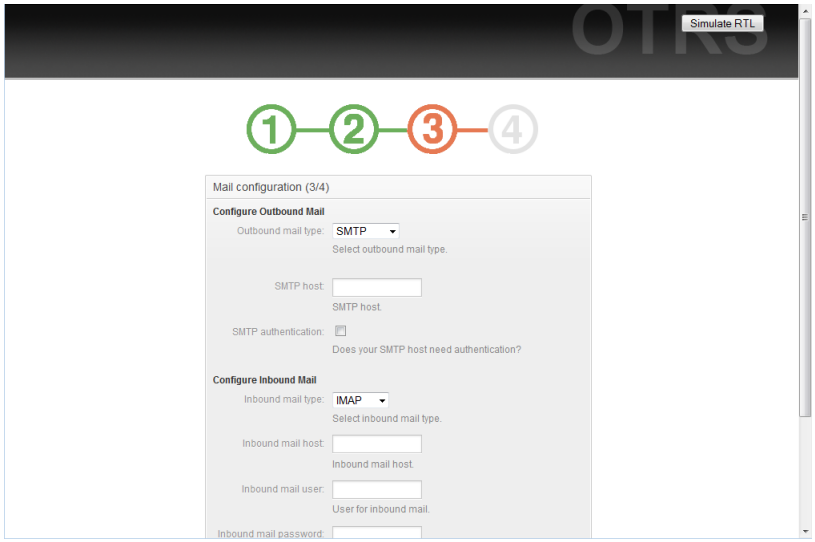

Figure: Mail configuration.

7. Restart the OTRS service now to use the new configuration settings as shown in the script below.

```
linux:~ # rcotrs restart-force
Shutting down OTRS
 Disable /opt/otrs/bin/otrs.PostMaster.pl ... done.
no crontab for otrs
 Shutting down cronjobs ... failed!
Shutting down OTRS (completely)
 Shutting down Apache ... done.
```

```
 Shutting down MySQL ... done.

 done
Starting OTRS (completely)
 Starting Apache ... done.
 Starting MySQL ... done.
Starting OTRS
 Checking Apache ... done.
 Checking MySQL ... done.
 Checking database connect... (It looks Ok!).
 Enable /opt/otrs/bin/otrs.PostMaster.pl ... done.
 Checking otrs spool dir...  done.
 Creating cronjobs (source /opt/otrs/var/cron/*) ... done.

  -->> http://linux.example.com/otrs/index.pl <<--

 done

 done
linux:~ #
```

Script: Restarting the OTRS service.

Congratulations! Now the installation of OTRS is finished and you should be able to work with the system (see Figure below). To log into the web interface of OTRS, use the address *http://localhost/otrs/index.pl* [http://localhost/otrs/index.pl] from your web browser. Log in as OTRS administrator, using the username 'root@localhost' and the password 'root'. After that you can configure the system for your needs.

Warning

Please change the password for the 'root@localhost' account as soon as possible.

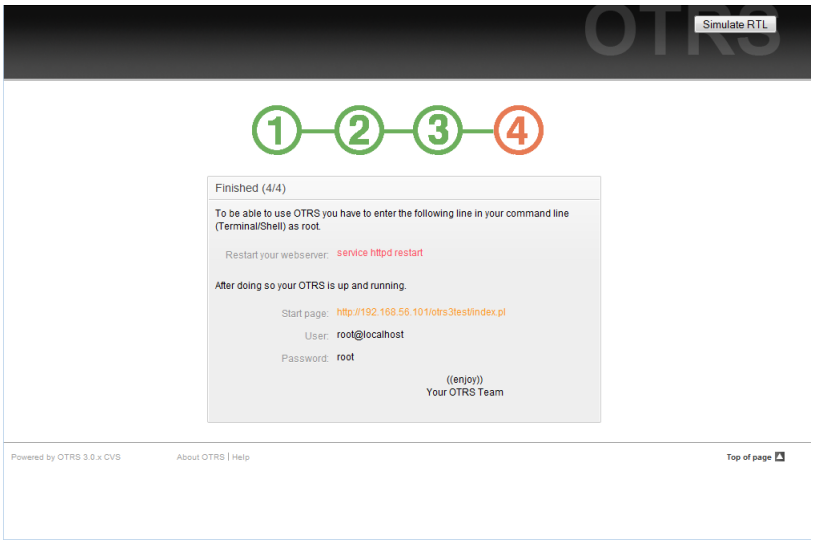

Figure: Final steps to install OTRS.

Installing the OTRS database manually

If you can't use the web installer to setup the OTRS database, you have to set it up manually. Scripts with the SQL statements to create and configure the database are located in `scripts/database`, in the home directory of the 'otrs' user (see Script below).

```
linux:~# cd /opt/otrs/scripts/database/
linux:/opt/otrs/scripts/database# ls
otrs-initial_insert.db2.sql         otrs-schema.mysql.sql
otrs-schema.oracle.sql
otrs-initial_insert.mssql.sql       otrs-schema-post.db2.sql
otrs-initial_insert.mysql.sql       otrs-schema.postgresql.sql
otrs-initial_insert.oracle.sql
otrs-initial_insert.postgresql.sql  otrs-schema-post.mssql.sql
otrs-initial_insert.xml             otrs-schema-post.mysql.sql
otrs-schema.db2.sql                 otrs-schema-post.oracle.sql
 otrs-schema-post.postgresql.sql
otrs-schema.mssql.sql               otrs-schema.xml
linux:/opt/otrs/scripts/database#
```

Script: Files needed to create and configure the database.

To setup the database for the different database back-ends, the .sql files must be processed in a specific order.

Create the OTRS database manually step by step

1. Creating the DB: Create the database that you want to use for OTRS, with your database client or your database interface.
2. Creating the tables: With the otrs-schema.DatabaseType.sql files (e.g. `otrs-schema.oracle.sql`, `otrs-schema.postgresql.sql`) you can create the tables in your OTRS database.
3. Inserting the initial system data: OTRS needs some initial system data to work properly (e.g. the different ticket states, ticket and notification types). Depending on the type of your database, use one of the files `otrs-initial_insert.mysql.sql`, `otrs-initial_insert.oracle.sql`, `otrs-initial_insert.postgresql.sql` or `otrs-initial_insert.mssql.sql` .
4. Creating references between tables: The last step is to create the references between the different tables in the OTRS database. Use the otrs-schema-post.DatabaseType.sql file to create these (e.g. `otrs-schema-oracle.post.sql`, `otrs-schema-post.postgresql.sql`).

After you have finished the database setup, you should check and set proper access rights for the OTRS database. It should be enough to grant access to one user. Depending on the database server you are using, setting up the access rights differs, but it should be possible either with your database client or your graphical database front-end.

If your database and the access rights are configured properly, you have to tell OTRS which database back-end you want to use and how the ticket system can connect to the database. Open the file `Kernel/Config.pm` located in the home directory of the 'otrs' user, and change the parameters shown in the script below according to your needs.

```
    # DatabaseHost
    # (The database host.)
    $Self->{'DatabaseHost'} = 'localhost';

    # Database
    # (The database name.)
    $Self->{Database} = 'otrs';

    # DatabaseUser
    # (The database user.)
    $Self->{DatabaseUser} = 'otrs';

    # DatabasePw
    # (The password of database user.)
    $Self->{DatabasePw} = 'some-pass';
```

Script: Parameters to be customized.

Setting up the cron jobs for OTRS

OTRS needs some cron jobs to work properly. The cron jobs should be run with the same user rights that were specified for the OTRS modules. That means that the cron jobs must be inserted into the crontab file of the 'otrs' user.

All scripts with the cron jobs are located in `var/cron`, in the home directory of the 'otrs' user (see Script below).

```
linux:~# cd /opt/otrs/var/cron
linux:/opt/otrs/var/cron# ls
aaa_base.dist                generic_agent.dist
 rebuild_ticket_index.dist
cache.dist                   pending_jobs.dist        session.dist
fetchmail.dist               postmaster.dist          unlock.dist
generic_agent-database.dist  postmaster_mailbox.dist
linux:/opt/otrs/var/cron#
```

Script: Files needed to create the cron jobs.

These scripts have a suffix of '.dist'. You should copy them to files with the suffix removed. If you use bash, you might want to use the command listed in Script below.

```
linux:/opt/otrs/var/cron# for foo in *.dist; do cp $foo `basename
 $foo .dist`; done
linux:/opt/otrs/var/cron# ls
aaa_base                generic_agent-database.dist
 rebuild_ticket_index
aaa_base.dist           generic_agent.dist
 rebuild_ticket_index.dist
cache                   pending_jobs                session
cache.dist              pending_jobs.dist           session.dist
fetchmail               postmaster                  unlock
```

```
fetchmail.dist          postmaster.dist              unlock.dist
generic_agent           postmaster_mailbox
generic_agent-database  postmaster_mailbox.dist
linux:/opt/otrs/var/cron#
```

Script: Copying and renaming all the files needed to create the cron jobs.

Table 3-2 describes the different cron jobs.

Table 3.2. Description of several cron job scripts.

Script	Function
aaa_base	Sets the basics for the crontab of the 'otrs' user.
cache	Removes expired cache entries from disk. Clears the loader cache for CSS and JavaScript files.
fetchmail	Used only if new mails will be fetched with fetchmail into the ticket system.
generic_agent	Executes the jobs of the GenericAgent that are not stored in the database but in own config files.
generic_agent-database	Executes the jobs of the GenericAgent that are stored in the database.
pending_jobs	Checks system for pending tickets, and closes them or sends reminders if needed.
postmaster	Checks the message queue of the ticket system, and delivers messages that are still in the queues.
postmaster_mailbox	Fetches the mails from the POP3 accounts that were specified in the admin area, in the section for "PostMaster Mail Accounts".
rebuild_ticket_index	Rebuilds the ticket index, which improves the speed of the QueueView.
session	Removes old and no longer needed session IDs.
unlock	Unlocks tickets in the system.

To setup all cron jobs, the script `bin/Cron.sh` located in the home directory of the 'otrs' user can be used. When this script is executed, it needs a parameter to specify whether you want to install, remove or reinstall the cron jobs. The following parameters can be used:

`Cron.sh` { *`start`* } { *`stop`* } { *`restart`* } [*`OTRS user`*]

Because the cron jobs need to be installed in the crontab file of the 'otrs' user, you need to be logged in as 'otrs'. If you are logged in as root, you can switch to 'otrs' with the command **su otrs**. Execute the commands specified in Script below to install the cron jobs.

Warning

Please note that other crontab entries of the 'otrs' user will be overwritten or removed by the `Cron.sh` script. Please change the `Cron.sh` script to retain other crontab entries as needed.

```
linux:/opt/otrs/var/cron# cd /opt/otrs/bin/
linux:/opt/otrs/bin# su otrs
linux:~/bin$ ./Cron.sh start
/opt/otrs/bin
Cron.sh - start/stop OTRS cronjobs
Copyright (C) 2001-2009 OTRS AG, http://otrs.org/
(using /opt/otrs) done
linux:~/bin$ exit
exit
linux:/opt/otrs/bin#
```

Script: Installing the cron jobs.

The command **crontab -l -u otrs**, which can be executed as root, shows you the crontab file of the 'otrs' user, and you can check if all entries are placed correctly (see Script below).

```
linux:/opt/otrs/bin# crontab -l -u otrs
# --
# cron/aaa_base - base crontab package
# Copyright (C) 2001-2012 OTRS AG, http://otrs.org/
# --
# This software comes with ABSOLUTELY NO WARRANTY.
# --
# Who gets the cron emails?
MAILTO="root@localhost"

# --
# cron/cache - delete expired cache
# Copyright (C) 2001-2012 OTRS AG, http://otrs.org/
# This software comes with ABSOLUTELY NO WARRANTY.
# --
# delete expired cache weekly (Sunday mornings)
20 0 * * 0  $HOME/bin/otrs.CacheDelete.pl --expired >> /dev/null
30 0 * * 0  $HOME/bin/otrs.LoaderCache.pl -o delete >> /dev/null

# --
# cron/fetchmail - fetchmail cron of the OTRS
# Copyright (C) 2001-2012 OTRS AG, http://otrs.org/
# --
# This software comes with ABSOLUTELY NO WARRANTY.
# --
# fetch every 5 minutes emails via fetchmail
#*/5 * * * * /usr/bin/fetchmail -a >> /dev/null

# --
# cron/generic_agent - otrs.GenericAgent.pl cron of the OTRS
# Copyright (C) 2001-2012 OTRS AG, http://otrs.org/
# --
# --
# This software comes with ABSOLUTELY NO WARRANTY.
# --
# start generic agent every 20 minutes
*/20 * * * * $HOME/bin/GenericAgent.pl >> /dev/null
```

```
# example to execute GenericAgent.pl on 23:00 with
# Kernel::Config::GenericAgentMove job file
#0 23 * * * $HOME/bin/otrs.GenericAgent.pl -c
 "Kernel::Config::GenericAgentMove" >> /dev/null
# --
# cron/generic_agent - GenericAgent.pl cron of the OTRS
# Copyright (C) 2001-2012 OTRS AG, http://otrs.org/
# --
# This software comes with ABSOLUTELY NO WARRANTY.
# --
# start generic agent every 10 minutes
*/10 * * * * $HOME/bin/otrs.GenericAgent.pl -c db >> /dev/null
# --
# cron/pending_jobs - pending_jobs cron of the OTRS
# Copyright (C) 2001-2012 OTRS AG, http://otrs.org/
# --
# This software comes with ABSOLUTELY NO WARRANTY.
# --
# check every 120 min the pending jobs
45 */2 * * * $HOME/bin/otrs.PendingJobs.pl >> /dev/null
# --
# cron/postmaster - postmaster cron of the OTRS
# Copyright (C) 2001-2012 OTRS AG, http://otrs.org/
# --
# This software comes with ABSOLUTELY NO WARRANTY.
# --
# check daily the spool directory of OTRS
#10 0 * * * test -e /etc/init.d/otrs & /etc/init.d/otrs cleanup
 >> /dev/null; test -e /etc/rc.d/init.d/otrs && /etc/rc.d/init.d/otrs
 cleanup >> /dev/null
10 0 * * * $HOME/bin/otrs.CleanUp.pl >> /dev/null
# --
# cron/postmaster_mailbox - postmaster_mailbox cron of the OTRS
# Copyright (C) 2001-2012 OTRS AG, http://otrs.org/
# --
# This software comes with ABSOLUTELY NO WARRANTY.
# --
# fetch emails every 10 minutes
*/10 * * * * $HOME/bin/otrs.PostMasterMailbox.pl >> /dev/null
# --
# cron/rebuild_ticket_index - rebuild ticket index for OTRS
# Copyright (C) 2001-2012 OTRS AG, http://otrs.org/
# --
# This software comes with ABSOLUTELY NO WARRANTY.
# --
# just every day
01 01 * * * $HOME/bin/otrs.RebuildTicketIndex.pl >> /dev/null

# --
# cron/session - delete old session ids of the OTRS
# Copyright (C) 2001-2012 OTRS AG, http://otrs.org/
# --
# This software comes with ABSOLUTELY NO WARRANTY.
# --
```

```
# delete every 120 minutes old/idle session ids
55 */2 * * * $HOME/bin/otrs.DeleteSessionIDs.pl --expired >> /dev/null

# --
# cron/unlock - unlock old locked ticket of the OTRS
# Copyright (C) 2001-2012 OTRS AG, http://otrs.org/
# --
# This software comes with ABSOLUTELY NO WARRANTY.
# --
# unlock every hour old locked tickets
35 * * * * $HOME/bin/otrs.UnlockTickets.pl --timeout >> /dev/null

linux:/opt/otrs/bin#
```

Script: Crontab file.

Upgrading the OTRS Framework

These instructions are for people upgrading OTRS from version *3.0* to *3.1*, and apply both for RPM and source code (tarball) upgrades.

If you are running a lower version of OTRS you have to follow the upgrade path to 3.0 first (1.1->1.2->1.3->2.0->2.1->2.2->2.3->2.4->3.0->3.1 ...)!

Please note that if you upgrade from OTRS 2.2 or earlier, you have to take an extra step; please read http://bugs.otrs.org/show_bug.cgi?id=6798.

If you need to do a "patch level upgrade", which is an upgrade for instance from OTRS version 3.1.1 to 3.1.3, you should skip steps 8, 10 and 12-19.

Please note that for upgrades from 3.1.beta1 or 3.1.beta2, an additional step 20 is needed!

If you are using Microsoft SQL Server as the DBMS for OTRS, please refer to the manual, chapter "Upgrading Microsoft SQL Server Data Types" for instructions how to upgrade the data types used by OTRS (http://doc.otrs.org/3.1/en/html/upgrading-mssql-datatypes.html).

1. Stop all relevant services.

   e. g. (depends on used services):

   ```
   shell> /etc/init.d/cron stop
   shell> /etc/init.d/postfix stop
   shell> /etc/init.d/apache stop
   ```

2. Backup everything below $OTRS_HOME (default: OTRS_HOME=/opt/otrs):

   - `Kernel/Config.pm`
   - `Kernel/Config/GenericAgent.pm`
   - `Kernel/Config/Files/ZZZAuto.pm`
   - `var/*`

3. Backup the database.

4. Make sure that you have backed up everything ;-)
5. Setup new system (optional)

   If possible, try this install on a separate machine for testing first.
6. Install the new release (tar or RPM).

   - With the tarball:

     ```
     shell> cd /opt
     shell> tar -xzf otrs-x.x.x.tar.gz
     shell> ln -s otrs-x.x.x otrs
     ```

     Restore old configuration files.

     - `Kernel/Config.pm`
     - `Kernel/Config/GenericAgent.pm`
     - `Kernel/Config/Files/ZZZAuto.pm`

   - With the RPM:

     ```
     shell> rpm -Uvh otrs-x.x.x.-01.rpm
     ```

     In this case the RPM update automatically restores the old configuration files.
7. Own themes

   Note: The OTRS themes between 3.0 and 3.1 are NOT compatible, so don't use your old themes!

   Themes are located under $OTRS_HOME/Kernel/Output/HTML/*/*.dtl (default: OTRS_HOME=/opt/otrs).
8. Set file permissions.

   If the tarball is used, execute:

   ```
   shell> cd /opt/otrs/
   shell> bin/otrs.SetPermissions.pl
   ```

   with the permissions needed for your system setup.
9. Apply the database changes (part 1/2):

   ```
   shell> cd /opt/otrs/

   # MySQL:
   ```

```
shell> cat scripts/DBUpdate-to-3.1.mysql.sql | mysql -p -f -u root
 otrs
# PostgreSQL 8.2+:
shell> cat scripts/DBUpdate-to-3.1.postgresql.sql | psql otrs
# PostgreSQL, older versions:
shell> cat scripts/DBUpdate-to-3.1.postgresql_before_8_2.sql | psql
 otrs
```

NOTE: If you use PostgreSQL 8.1 or earlier, you need to activate the new legacy driver for these older versions. Do this by adding a new line to your `Kernel/Config.pm` like this:

```
$Self->{DatabasePostgresqlBefore82} = 1;
```

Run the migration script (as user 'otrs', NOT as root):

You must execute the migration script to migrate some data from the old database structure to the new one. Please run:

```
shell> scripts/DBUpdate-to-3.1.pl
```

Apply the database changes (part 2/2):

```
# MySQL:
shell> cat scripts/DBUpdate-to-3.1-post.mysql.sql | mysql -p -f -u
 root otrs
# PostgreSQL 8.2+:
shell> cat scripts/DBUpdate-to-3.1-post.postgresql.sql | psql otrs
# PostgreSQL, older versions:
shell> cat scripts/DBUpdate-to-3.1-post.postgresql_before_8_2.sql |
 psql otrs
```

10. Refresh the configuration and delete caches. Please run:

```
shell> bin/otrs.RebuildConfig.pl
shell> bin/otrs.DeleteCache.pl
```

11. Update your web server configuration

Note: this applies only if you use the Apache web server together with mod_perl2, and do not use the configuration file directly from the OTRS installation directory (e. g. with a symlink from the Apache configuration directory).

Please add a new setting to the Apache configuration file for OTRS:

```
# set mod_perl2 option for generic interface
<Location /otrs/nph-genericinterface.pl>
```

```
  PerlOptions -ParseHeaders
</Location>
```

Please see the file /opt/otrs/scripts/apache2-httpd.include.conf for an example of where this new option needs to be added (inside the <IfModule mod_perl.c> block).

In this file, you will also note a new section on caching:

```
<IfModule mod_headers.c>
    <Directory "/opt/otrs/var/httpd/htdocs/skins/*/*/css-cache">
        <FilesMatch "\.(css|CSS)$">
            Header set Cache-Control "max-age=2592000 must-
revalidate"
        </FilesMatch>
    </Directory>

    <Directory "/opt/otrs/var/httpd/htdocs/js/js-cache">
        <FilesMatch "\.(js|JS)$">
            Header set Cache-Control "max-age=2592000 must-
revalidate"
        </FilesMatch>
    </Directory>
</IfModule>
```

Please activate this in your local installation too, and make sure that mod_headers is installed and active.

12. Restart your services.

    e. g. (depends on used services):

```
shell> /etc/init.d/cron start
shell> /etc/init.d/postfix start
shell> /etc/init.d/apache start
```

    Now you can log into your system.

13. Check installed packages

    In the package manager, check if all packages are still marked as correctly installed or if any require reinstallation or even a package upgrade.

14. Check for encoding issues

    OTRS 3.1 only allows UTF-8 as internal charset. Non-UTF-8 installations of OTRS must switch to UTF-8.

15. Escalation events

    If you want to use the new escalation events in your system, you need to activate the corresponding GenericAgent job in Kernel/Config/GenericAcent.pm. Please look into Kernel/Config/GenericAgent.pm.dist for an example of how to do this.

16. Ticket event handlers

    The Event name TicketFreeTextUpdate_$Counter was renamed to TicketDynamicFieldUpdate_$FieldName. If you have any custom event handlers for these events, please adapt them.

17. DynamicField user preferences module

    If you had one or more active custom settings for "PreferencesGroups###Freetext", you need to adapt them to work with the new DynamicFields engine. The PrefKey setting must be changed to "UserDynamicField_DynamicField", where the part after the _ is the name of the dynamic field. Existing values would need to be renamed in the database as well.

18. Custom free field default value event handler

    If you used the event handler Ticket::EventModulePost###TicketFreeFieldDefault (not active by default), you'll need to migrate its configuration to the new setting Ticket::EventModulePost###TicketDynamicFieldDefault.

    The configuration of this is slightly different; where you had to specify a Counter indicating the TicketFreeText number previously, now you need to specify the name of the DynamicField (for migrated fields, this will be DynamicField_TicketFreeKey$Counter and DynamicField_TicketFreeText$Counter. You need two separate entries now if you want to set both the key and the text field.

19. FreeText/Time based ACLs

    If you have any ACLs defined which involve freetext or freetime fields, you need to adjust these ACL definitions.

    Please have a look at http://doc.otrs.org/3.1/en/html/acl.html. There you can find a list of all possible ACL settings. In general, you need to add the prefix "DynamicField_" to existing free field definitions, and you can add a new "DynamicField" section to the "Properties" list for situations when a ticket does not exist yet.

20. Database Upgrade During Beta Phase

    This step is ONLY needed if you upgrade from 3.1.beta1 or 3.1.beta2! Please apply the required database changes as follows:

```
MySQL:
 shell> cat scripts/DBUpdate-3.1.beta.mysql.sql | mysql -p -f -u
 root otrs

PostgreSQL 8.2+:
 shell> cat scripts/DBUpdate-3.1.beta.postgresql.sql | psql otrs

PostgreSQL, older versions:
 shell> cat scripts/DBUpdate-3.1.beta.postgresql_before_8_2.sql |
 psql otrs
```

21. Well done!

Upgrading Windows Installer

There is currently no in-place upgrade tool available for OTRS installations that were done with the Windows Installer. The upgrade process basically consists of backing up the database and the filesystem, uninstalling OTRS, installing the new version, restoring the database and running the upgrade procedure if needed.

Upgrading is described in FAQ# 4200351 [http://faq.otrs.org/otrs/public.pl?Action=PublicFAQ;ItemID=351], and there is also an informative YouTube video [http://www.youtube.com/watch?v=sf0R-reMTWc] available.

Upgrading Microsoft SQL Server Data Types

Starting OTRS version *3.1*, OTRS uses the *NVARCHAR* data type rather than *VARCHAR* or *TEXT*, to store textual data. This is because the *NVARCHAR* type has full support for Unicode, whereas the old data types store data in UCS-2 format, which is a sub-set of Unicode. Also, the *TEXT* data type is deprecated since *SQL Server 2005*. Due to this, starting with OTRS version 3.1, the minimal SQL Server version required for operation with OTRS is now *Microsoft SQL Server 2005*.

Because dropping and re-creating these indexes is a time-consuming operation, especially on large databases, please plan enough time for performing the upgrade. We would recommend that you perform the upgrade on a copy of the database prior to doing the actual conversion to test the upgrade procedure and to time how much time will be needed on your specific environment.

Please make sure that, before you start, there is enough space available on the database server. Make sure the free space on your database server is at least 2.5x the current size of the database.

Important

This upgrade procedure will upgrade all fields of the mentioned data types to the new types. This procedure first removes any indexes and constraints in which these fields are referenced, upgrades the fields, and then adds the indexes and constraints back. It will do so on all tables found in the SQL Server database that OTRS uses. If you would have stored non-OTRS tables in the OTRS database, and these tables contain columns of the data types VARCHAR or TEXT, these will also be updated.

1. Open a Command Line on the OTRS server.
2. Change directory to the OTRS root directory. If you're using the default OTRS installer this would be C:\Program Files\OTRS\OTRS.
3. Run the following command:

```
shell> perl scripts/DUpdate-to-3.1.mssql-datatypes.pl
```

4. This will generate three scripts in the specified directory scripts\database\update. Run these scripts on the SQL Server database, via SQL Server Management Studio or isql.

Chapter 4. First steps

This chapter's goal is to give a quick overview of OTRS and the structure of its web interface. The terms agents, customers and administrators are introduced. We also login as the OTRS admininstrator and take a closer look at the user preferences available for every account.

Agent web interface

The agent web interface allows agents to answer customer requests, create new tickets for customers or other agents, write tickets about telephone calls with customers, write FAQ entries, edit customer data, etc.

Supposing your OTRS host is reachable via the URL *http://www.example.com* [http://www.example.com/] , then the OTRS login screen can be reached by using the address *http://www.example.com/otrs/index.pl* [http://www.example.com/otrs/index.pl] in a web browser (see Figure below).

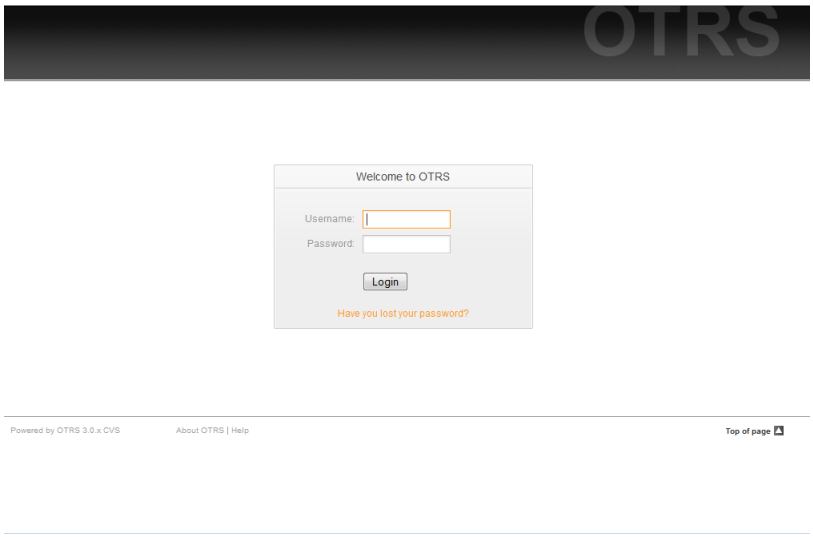

Figure: Login screen of the agent interface.

Customer web interface

Customers have a separate web interface in OTRS, through which they can create new accounts, change their account settings, create and edit tickets, get an overview on tickets that they created, etc.

Continuing with the above example, the customer login screen can be reached by using the URL *http://www.example.com/otrs/customer.pl* [http://www.example.com/otrs/customer.pl] with a web browser (see Figure below).

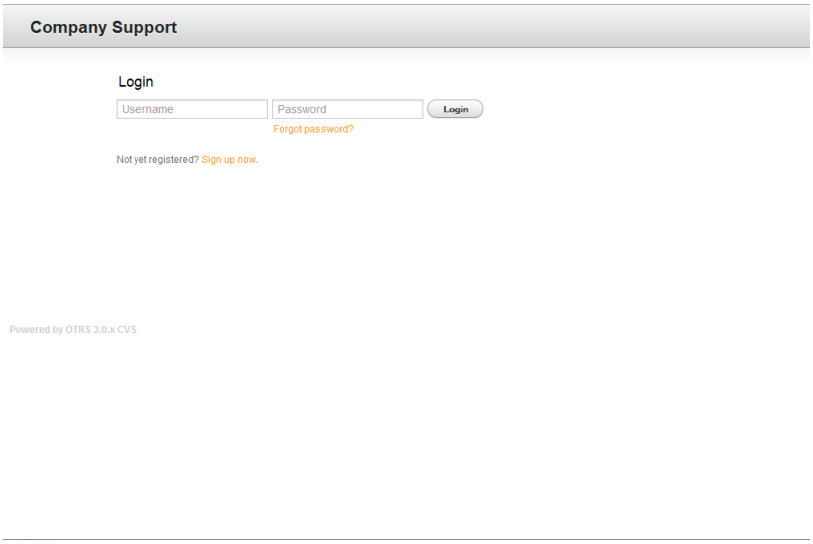

Figure: Login screen of the customer interface.

Public web interface

In addition to the web interfaces for agents and customers, OTRS also has a public web interface that is available through the FAQ-Module. This module needs to be installed separately. It provides public access to the FAQ system, and lets visitors search through FAQ entries without any special authorization.

In our example, the public web interface can be reached via either of the URLs below: *http://www.example.com/otrs/faq.pl* [http://www.example.com/otrs/faq.pl] , *http://www.example.com/otrs/public.pl* [http://www.example.com/otrs/public.pl]

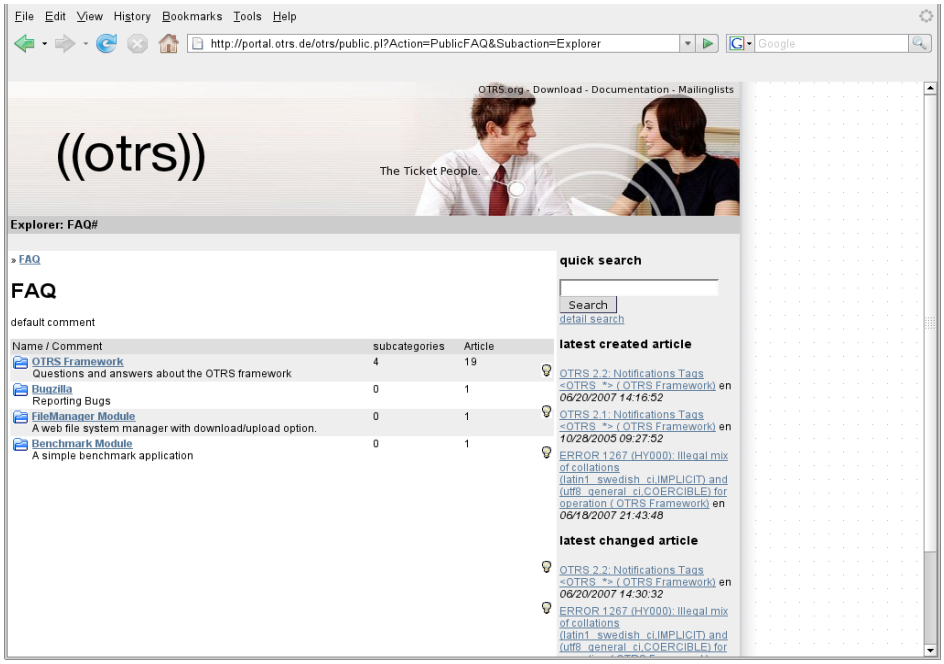

Figure: Public web interface.

First login

Access the login screen as described in the section Agent web interface . Enter a user name and a password. Since the system has just been freshly installed and no users have yet been created, login as OTRS administrator first, using 'root@localhost' for username and 'root' for password.

Warning

This account data is valid on every newly installed OTRS system. You should change the password for the OTRS administrator as soon as possible! This can be done via the preferences screen for the OTRS administrator account.

If you don't want to login as OTRS administrator, just enter the user name and password for your normal agent account.

In case you have forgotten your password, you can request the system for a new password. Simply press the link below the Login button, enter the mail address that is registered for your OTRS account into the input field, and press the Submit button (see Figure).

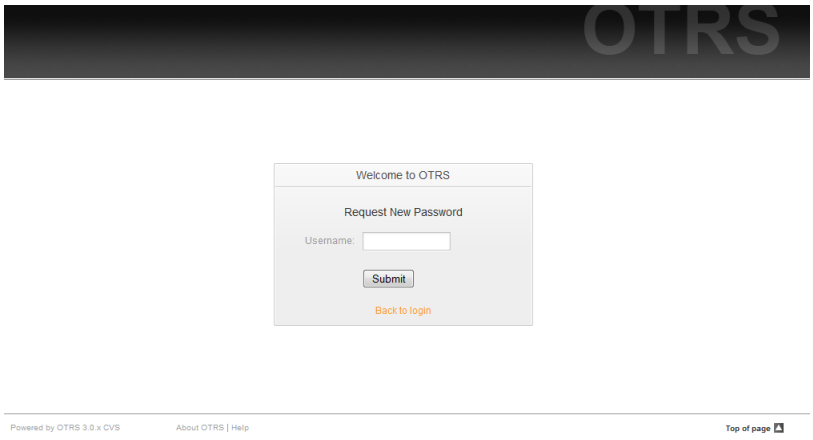

Figure: Request new password.

The web interface - an overview

On successfully logging into the system, you are presented with the Dashboard page (see Figure below). The Dashboard is completely customizable. It shows your locked tickets, allows direct access through menus to the queue, status and escalation views, and also holds options for creation of new phone and e-mail tickets. It also presents a quick summary of the tickets which are pending, escalated, new and open.

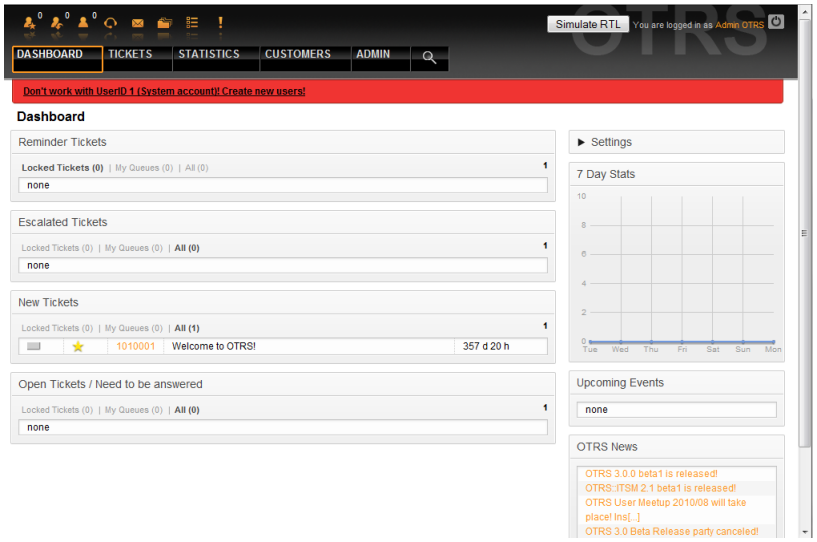

Figure: Dashboard of the agent interface.

To improve clarity, the general web interface is separated into different areas. The top row of each page shows some general information such the current username, the logout button, icons listing the number of locked tickets with direct access to them, links to create a new phone/e-mail ticket, etc. There are also icons to go to the queue, status and escalation views.

Below the icons row is the navigation bar. It shows a menu that enables you to navigate to different areas or modules of the system, lettng you execute some global actions. Clicking on the Dashboard button takes you to the dashboard which is the default start page after login. If you click on the Tickets button, you will get a submenu with options to change the ticket's view, create a new ticket (phone/e-mail) or search for a specific ticket. The Statistics button presents a menu that allows choosing from an overview of the registered statistics, creating a new one or importing an existing one. The Customers button leads you to the Customer Managment screen. By clicking the Admin button, you can access all the administrator modules, allowing you to create new agents, queues, etc. There is also a Search button to make ticket searches.

If any associated applications are also installed, e.g. the File Manager or the Web Mailer, buttons to reach these applications are also displayed.

The red bar below the navigation bar shows different system messages. If you are logged in as OTRS administrator, you get a message warning you not to work using this system account.

Below the title of the section you are currently in, there are several subsections each in a separate box. These boxes can be relocated within the same column by clicking on and dragging the box header, and dropping them elsewhere.

In the left column, you can see information on some tickets clasified as - reminder, escalated, new and open. In each of the categories, you are also able to see all tickets you are allowed to access, how many tickets you have locked and how many are located in "My Queues". "My Queues" are queues that you identify in your user configuration account preferences as those you have a special interest in tracking.

In the right column is the Settings button. Click on it to expand the section and see the various settings, as shown in Figure. You can then check or uncheck the individual settings options, and save your changes. This section is fixed, so you can not drag and drop it.

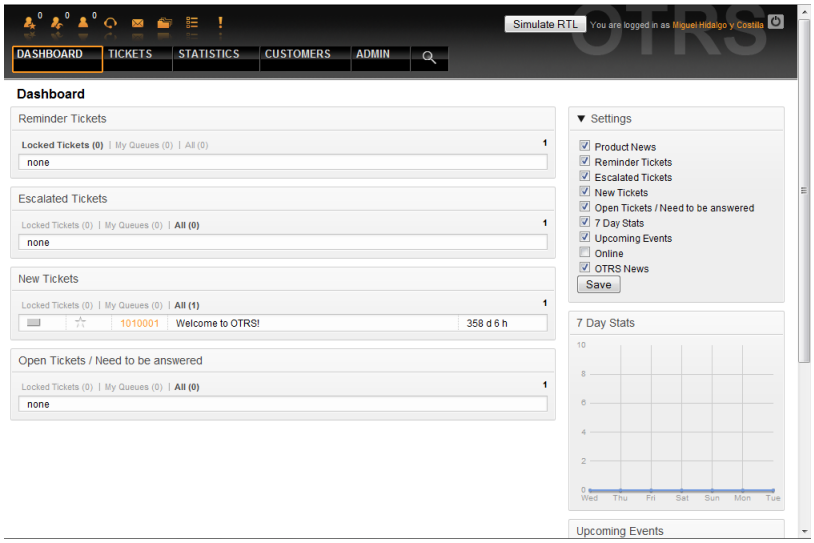

Figure: Dashboard Settings.

Below the settings area, you can see a section with a graph of ticket activity over the past 7 days. Further below is a section showing Upcoming Events and OTRS News.

Finally at the bottom of the page, the site footer is displayed (see Figure below). It contains links to directly access the OTRS official website, or go to the Top of the page.

Figure: Footer.

What is a queue?

On many mail systems, it is common for all messages to flow into an Inbox file, where they remain stored. New messages are appended at the end of the Inbox file. The mail client program used to read and write mails reads this Inbox file and presents the content to the user.

A queue in OTRS is somewhat comparable to an Inbox file, since it too can store many messages. A queue also has features beyond those of an Inbox mail file. As an OTRS agent or user, one needs to remember which queue a ticket is stored in. Agents can open and edit tickets in a queue, and also move tickets from one queue to another. But why would they move tickets?

To explain it more practically, remember the example of Max's company described in an example of a ticket system. Max installed OTRS in order to allow his team to better manage support for company customers buying video recorders.

One queue holding all requests is enough for this situation. However, after some time Max decides to also sell DVD recorders. Now, the customers have questions not only about the video recorder, but also the new product. More and more emails get into the one queue of Max's OTRS and it's hard to have a clear picture of what's happening.

Max decides to restructure his support system, and adds two new queues. So now three queues are being used. Fresh new mails arriving at the ticket system are stored into the old queue titled "raw". Of the two new queues, one titled "video recorder" is for video recorder requests, while the other one titled "dvd recorder" is for dvd recorder requests.

Max asks Sandra to watch the "raw" queue and sort (dispatch) the mails either into "video recorder" or "dvd recorder" queue, depending on the customer request. John only has access to the "video recorder" queue, while Joe can only answer tickets in the "dvd recorder" queue. Max is able to edit tickets in all queues.

OTRS supports access management for users, groups and roles, and it is easy to setup queues that are accessible only to some user accounts. Max could also use another way to get his requests into the different queues, with filter rules. Else, if two different mail addresses are used, Sandra only has to dispatch those emails into the two other queues, that can't be dispatched automatically.

Sorting your incoming messages into different queues helps you to keep the support system structured and tidy. Because your agents are arranged into different groups with different access rights on queues, the system can be optimized even further. Queues can be used to define work flow processes or to create the structure of a company. Max could implement, for example, another queue called "sales", which could contain the sub queues "requests", "offers", "orders", "billing", etc. Such a queue structure could help Max to optimize his order transactions.

Improved system structures, such as through the proper design of queues, can lead to significant time and cost savings. Queues can help to optimize the processes in your company.

User preferences

OTRS users such as customers, agents and the OTRS administrator can configure their account preferences as per their needs. Agent can access the configuration screen by clicking on their login name at the top right corner of the web interface (see Figure below), and customers must click on the "Preferences" link (see Figure below).

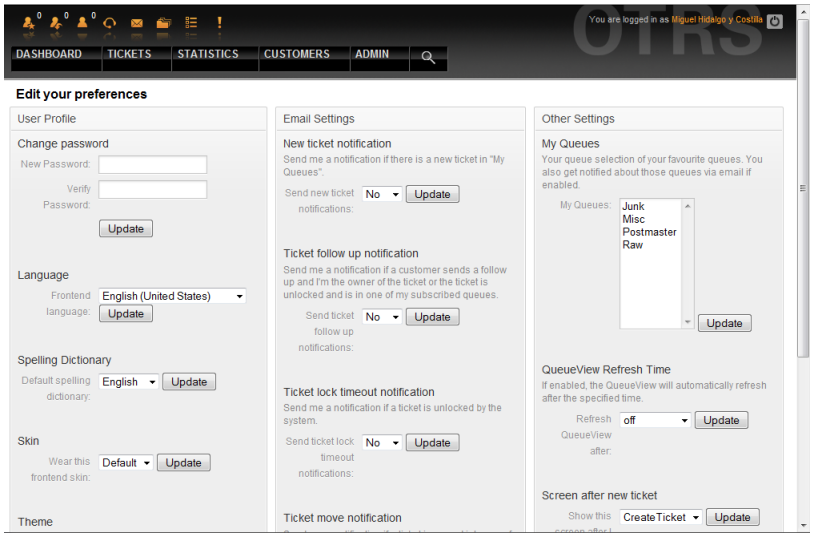

Figure: Agent's personal preferences.

An agent can configure 3 different categories of preferences: user profile, email settings and other settings. The default posibilities are:

User Profile

- Change the current password.

- Adjust the interface language.
- Switch the frontend skin.
- Shift the frontend theme.
- Activate and configure the out-of-office time.

Email Settings

- Select events that trigger email notifications to the agent.

Other Settings

- Select the queues you want to monitor in "My Queues".
- Set the refresh period for the queue view.
- Set the screen to be displayed after a ticket is created.

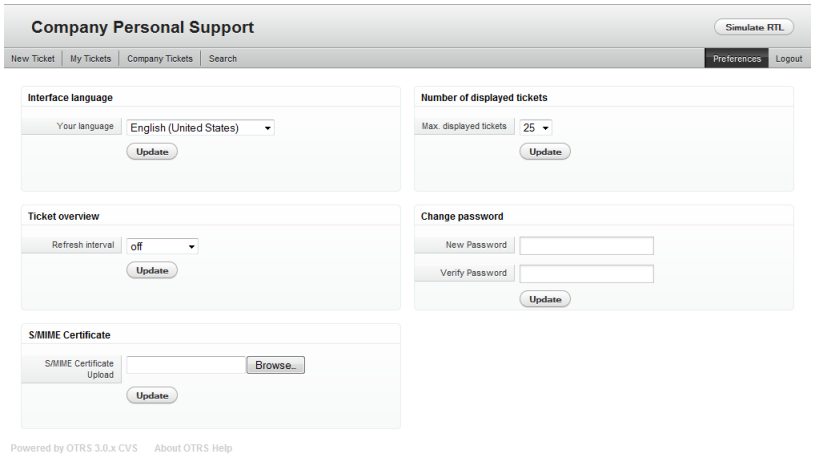

Figure: Customer's personal preferences.

A customer can select the web interface language, set the refresh interval for the ticket overview, and choose the maximum amount of shown tickets. It is also possible to set a new password.

Chapter 5. The ADMIN area of OTRS

Basics

OTRS administrators use the Admin page on the OTRS web interface to configure the system - adding agents, customers and queues, ticket and mail settings, installing additional packages such as FAQ and ITSM, and much more.

Agents who are members of the *admin* group can access the Admin area by clicking the *Admin* link in the navigation bar (see Figure below). The rest of the agents won't see this link.

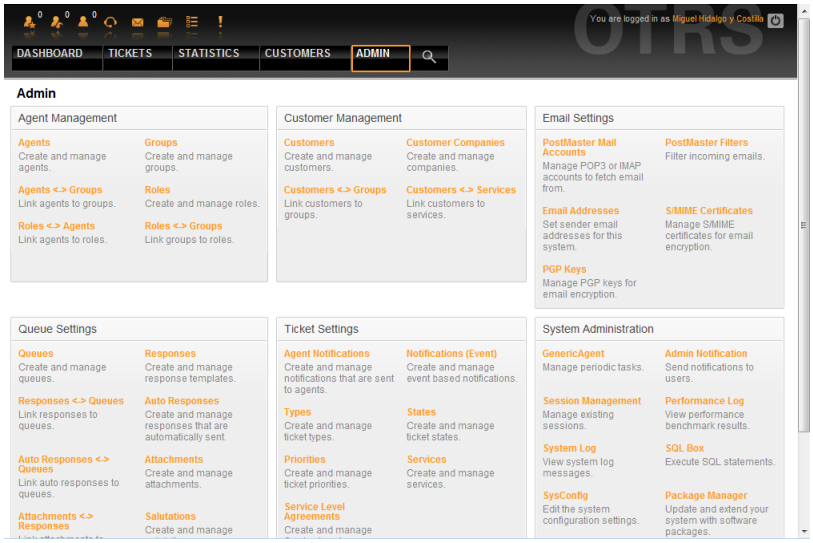

Figure: OTRS Admin screen.

Agents, Groups and Roles

Agents

By clicking the link *Agents*, you get access to the agent management screen of OTRS (see Figure below). Administrators can add, change or deactivate agent accounts. Administrators can also manage agent preferences, for instance the language and notification settings for their interface.

Note

An OTRS agent account may be deactivated but not deleted. Deactivation is done by setting the Valid flag to *invalid* or *invalid-temporarily*.

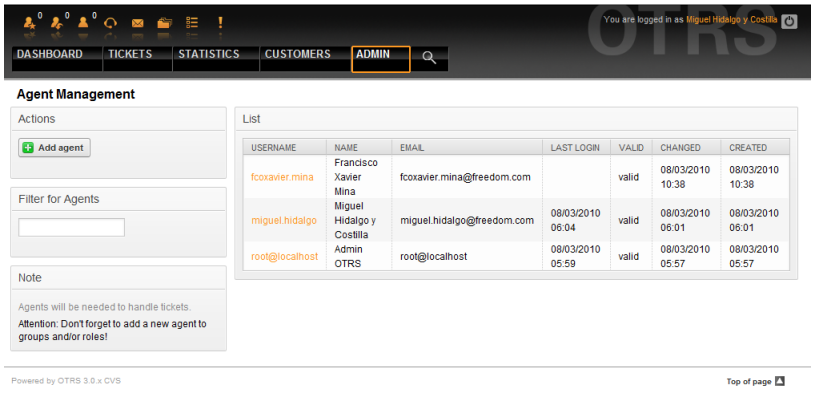

Figure: Agent management.

To register an agent, click on the "Add agent" button, type all the needed data and press the Submit button at the bottom of the screen, as shown in Figure.

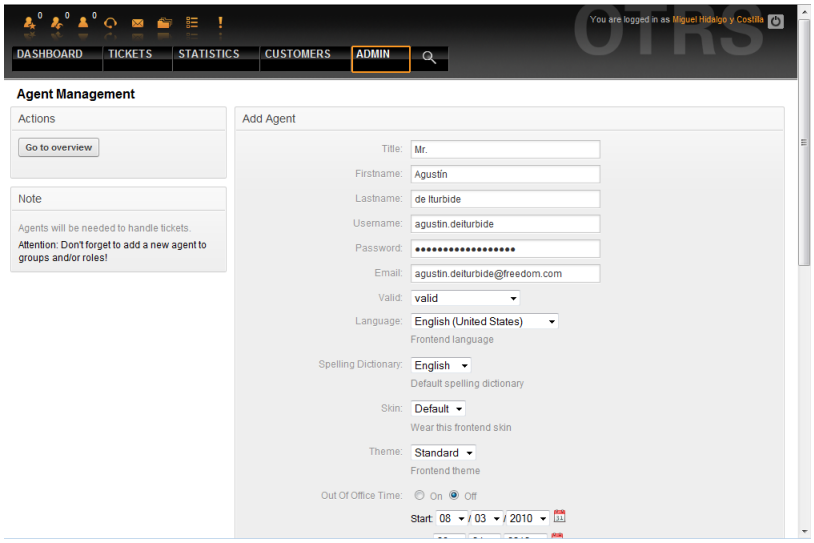

Figure: Adding a new agent.

After the new agent account has been created, you should make the agent a member of one or more groups or roles. Information about groups and roles is available in the Groups and Roles sections of this chapter.

Groups

Every agent's account should belong to at least one group or role. In a brand new installation, there are three pre-defined groups available, as shown in Table 5-1.

Table 5.1. Default groups available on a fresh OTRS installation

Group	Description
admin	Allowed to perform administrative tasks in the system.
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

In a brand new OTRS installation, the group *users* is initially empty. The agent 'root@localhost' belongs by default to the admin and stats groups.

You can access the group management page (see Figure below) by clicking the *Groups* link in the admin area.

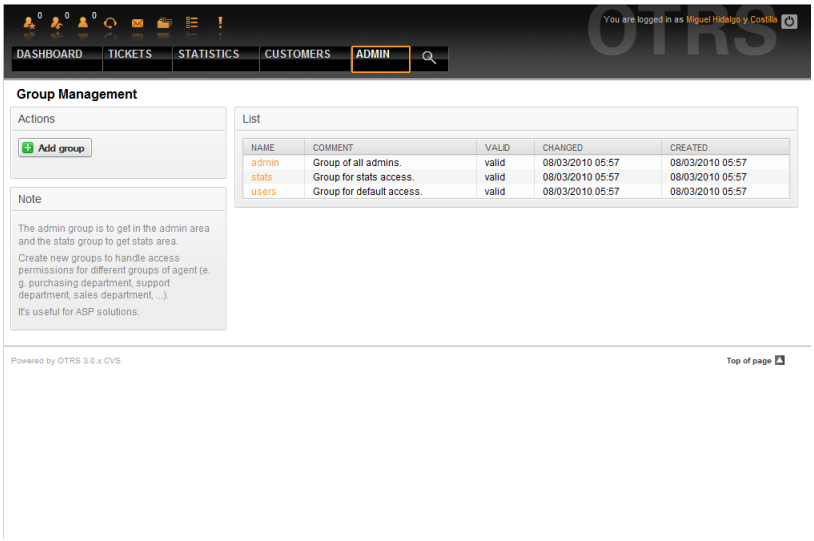

Figure: Group management.

Note

As with agents, an OTRS group may be deactivated but not deleted. Deactivation is done by setting the Valid flag to *invalid* or *invalid-temporarily*.

To add an agent to a group, or to change the agents who belong to a group, you can use the link *Agents <-> Groups* from the Admin page (see Figure below).

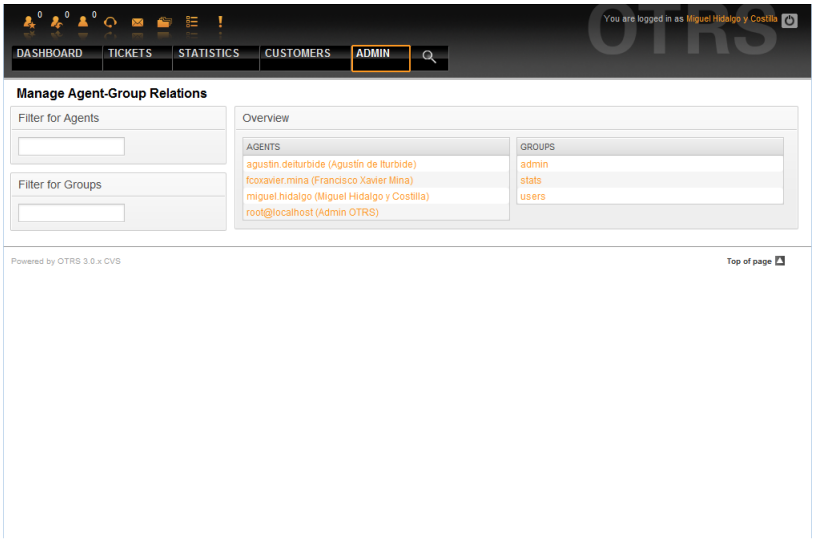

Figure: Group management.

An overview of all groups and agents in the system is displayed. You can also use the filters to find a specific entity. If you want to change the groups that an agent is member of, just click on the agent's name (see Figure below). To change the agents associated with a group, just click on the group you want to edit (see Figure below).

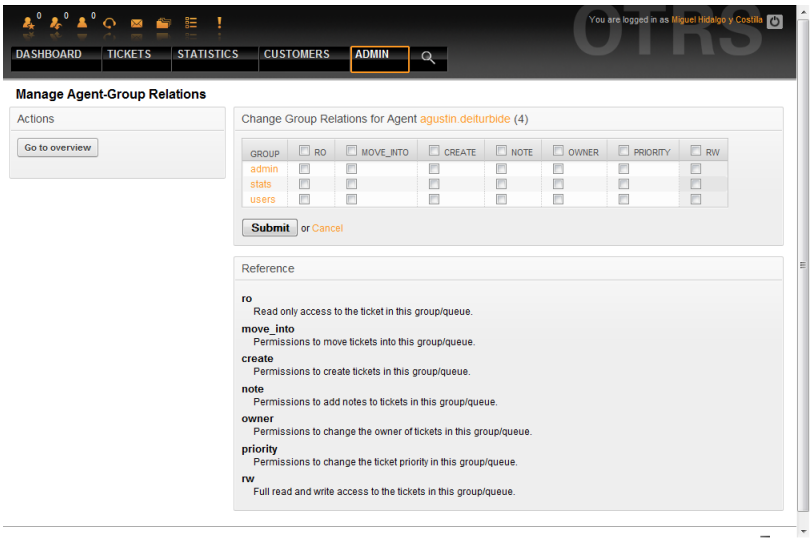

Figure: Change the groups an agent belongs to.

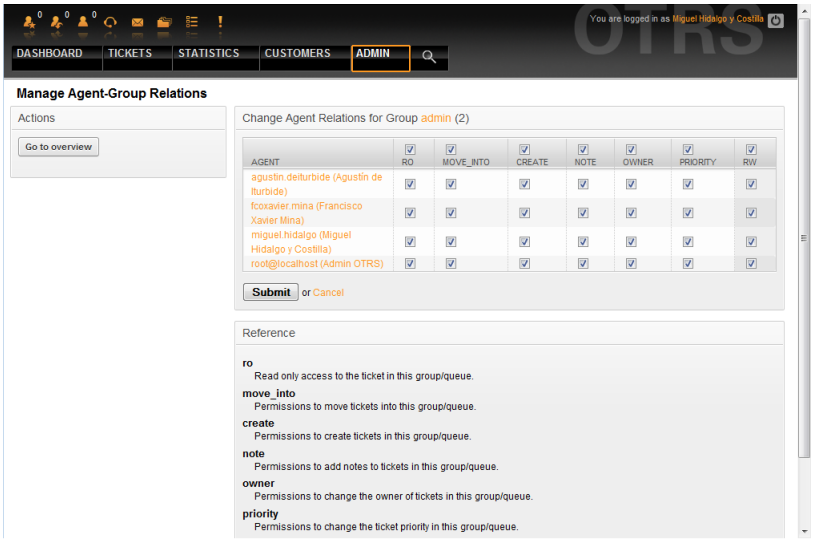

Figure: Change the agents that belong to a specific group.

Each group has a set of rights associated with it, and each member agent may have some combination of these rights for themselves. A list of the permissions / rights is shown in Table 5-2.

Table 5.2. Rights associated with OTRS Groups

Right	Description
ro	Read only access to the tickets, entries and queues of this group.
move into	Right to move tickets or entries between queues or areas that belong to this group.
create	Right to create tickets or entries in the queues or areas of this group.
owner	Right to update the owner of tickets or entries in queues or areas that belong to this group.
priority	Right to change the priority of tickets or entries in queues or areas that belong to this group.
rw	Full read and write access on tickets or entries in the queues or areas that belong to this group.

Note

By default, the QueueView only lists tickets in queues that an agent has *rw* access to, i.e., the tickets the agent needs to work on. If you want to change this behaviour, you can set Ticket::Frontend::AgentTicketQueue###ViewAllPossibleTickets to *Yes*.

Roles

Roles are a powerful feature to manage the access rights of many agents in a very simple and quick manner. They are particularly applicable on large, complex support systems with a lot of agents, groups and queues. An example below explains when they may be used.

Suppose that you have a system with 100 agents, 90 of them with access to a single queue called "support" where all support requests are handled. The "support" queue contains some

sub queues. The other 10 agents have permission to access all queues of the system. These 10 agents dispatch tickets, watch the raw queue and move spam messages into the "junk" queue.

The company now opens a new department that sells some products. Order request and acceptance, order confirmation, bills, etc. must be processed, and some of the company's agents shall do this via OTRS. The different agents have to get access to the new queues that must be created.

Because it would take a long time to change the access rights for the different agents manually, roles that define the different access levels can be created. The agents can then be added to one or more roles, with their rights automatically changed. If a new agent account is created, it is also possible to add this account to one or more roles.

Note

Roles are really useful when maintaining larger OTRS installations. You should take care in their use though. Mixing Agent to Group with Agent to Role mappings can make for a complex access control scheme, difficult to understand and maintain. If you wish to use only roles and disable the Agents <-> Groups option in the Admin area, you can do so by modifying the Frontend::Module###AdminUserGroup in the SysConfig. Be aware that this won't remove already existing Agents to Group assignments!

You can access the role management section (see Figure below) by clicking the *Roles* link on the Admin page.

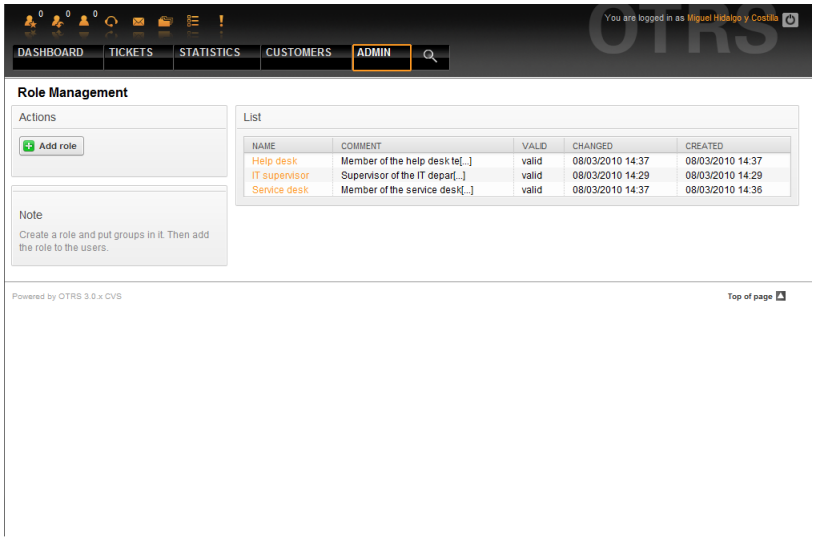

Figure: Role management.

Note

As with agent and groups, roles once created can be deactivated but not deleted. To deactivate, set the Valid option to *invalid* or *invalid-temporarily*.

An overview of all roles in the system is displayed. To edit a role's settings, click on the role's name. In a fresh new OTRS installation, there are no roles defined by default. To register one, click on the "Add role" button, provide the needed data and submit it (see Figure below).

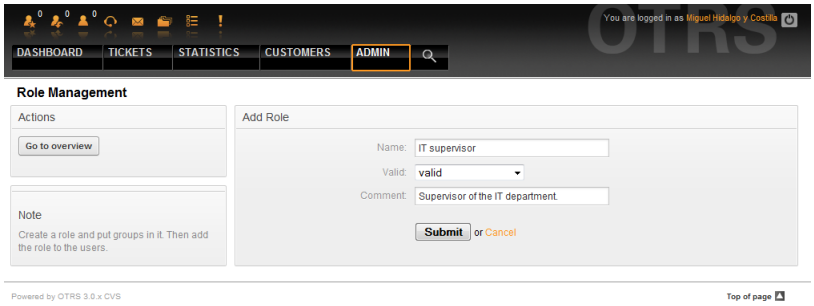

Figure: Adding a new role.

To get an overview of all roles and agents in the system, click on the link Roles <-> Agents on the Admin page. You can also use filters to find a specific element. If you want to change the roles associated with an agent, just click on the agent's name (see Figure below). To change the agents associated with a role, click on the role you want to edit (see Figure below).

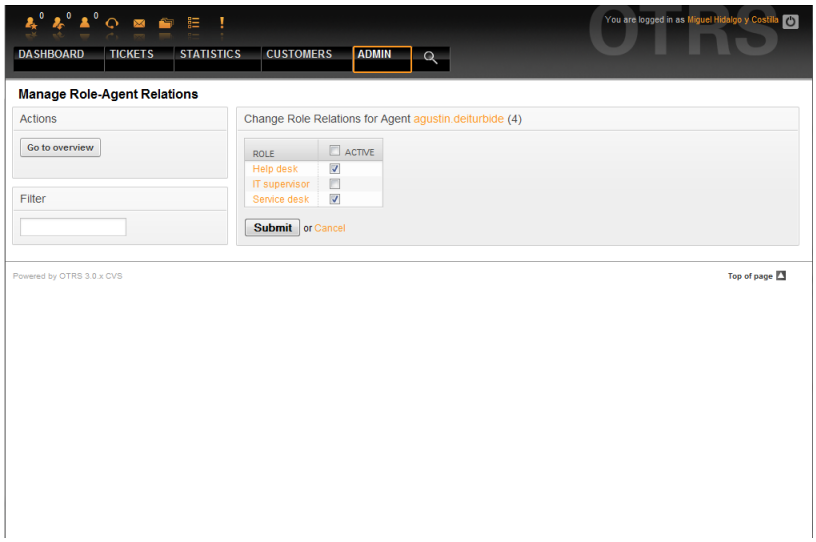

Figure: Change the Roles associated with an Agent.

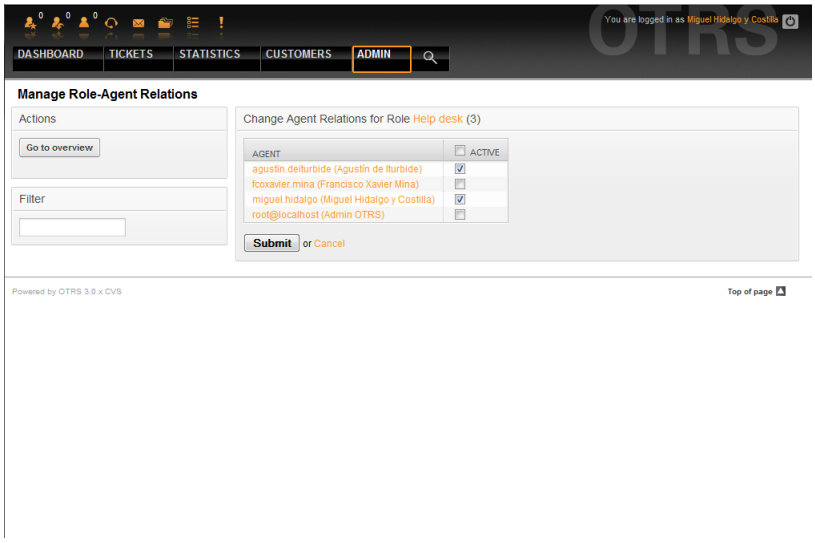

Figure: Change the Agents associated with a specific Role.

To get an overview of all roles and groups in the system, click on the link Roles <-> Groups on the Admin page. You will see a similar screen as the one shown in the Figure. You can also use filters to find a specific entity.

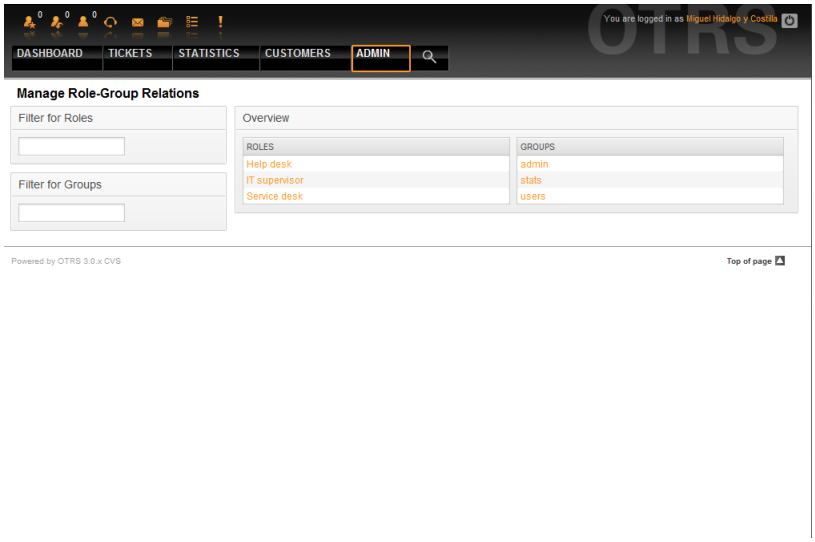

Figure: Manage Roles-Groups relations.

To define the different access rights for a role, click on the name of a role or a group (see below the Figures 5.13 and 5.14, respectively).

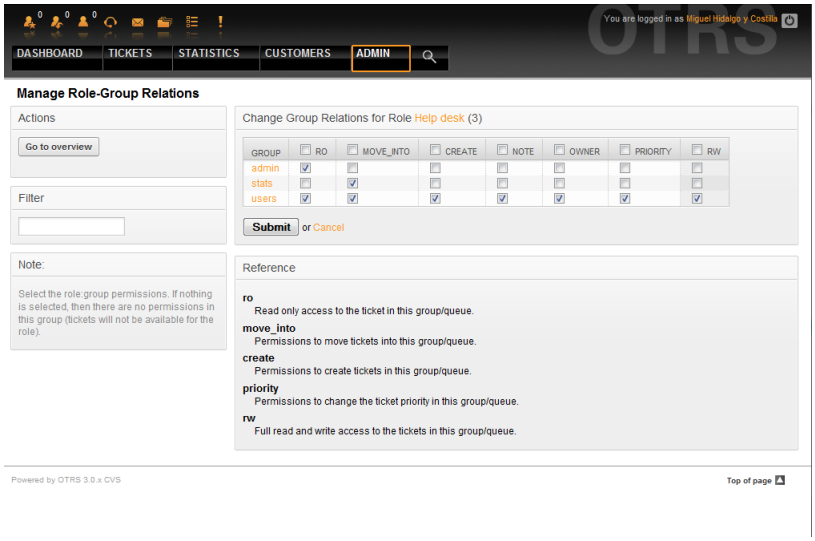

Figure: Change Group relations for a Role.

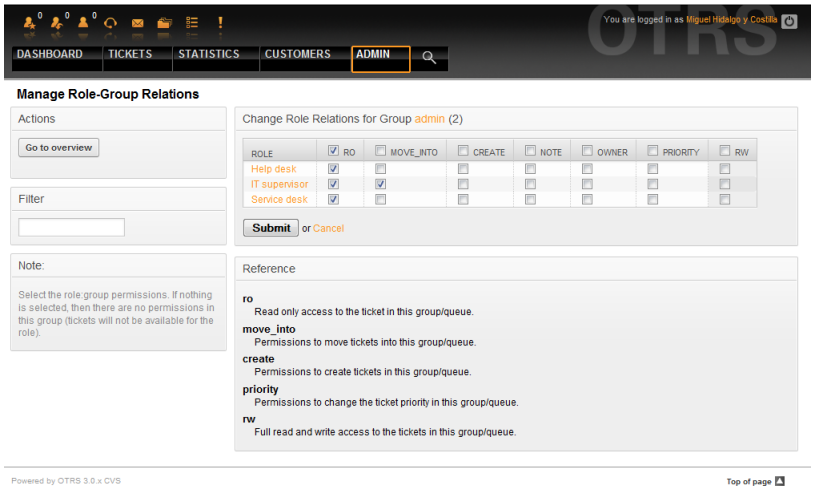

Figure: Change Role relations for a Group.

Customers and Customer Groups

Customers

OTRS supports different types of users. Using the link "Customers" (via the navigation bar, or the Admin page), you can manage the accounts of your customers (see Figure below), who can log into the system via the Customers interface (customer.pl). Through this interface, your customers can create tickets and access them as they are updated. It is important to know that a customer is needed for the ticket history in the system.

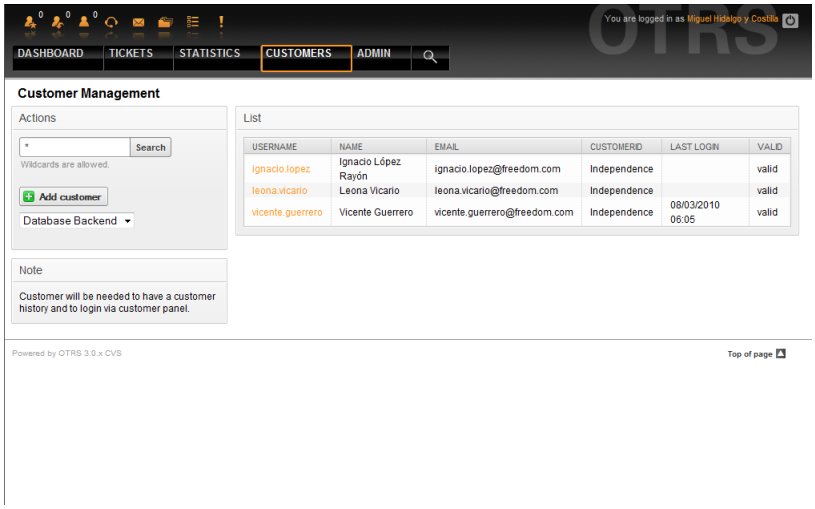

Figure: Customer management.

You can search for a registered customer, or edit their settings by clicking on their name. You also have the possibility to change the customer back-end, for further information please refer to the chapter about external back-ends.

To create a new customer account, click on the "Add customer" button (see Figure below). Some of the fields are mandatory, i.e., they have to contain values, so if you leave one of those empty, it will be highlighted in red.

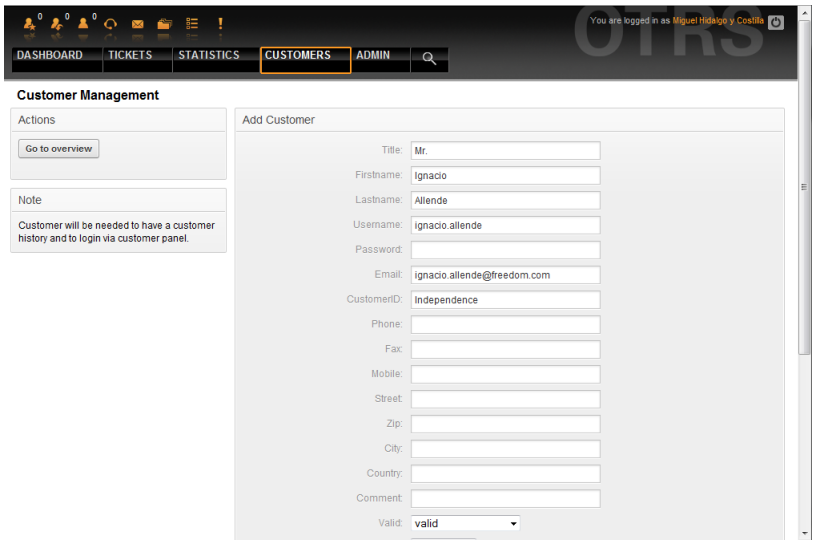

Figure: Adding a customer.

Customers can access the system by providing their username and password. The CustomerID is needed by the system to identify the user and associated tickets. Since the email address is a unique value, it can be used as the ID.

Note

As with agents, groups and roles, customers can not be deleted from the system, only deactivated by setting the Valid option to *invalid* or *invalid-temporarily*.

Customer Groups

Customer users can also be added to a group, which can be useful if you want to add customers of the same company with access to one or a few queues. First create the group to which your customers will belong, via the Group management module. Then add the queues and select the new group for the queues.

The next step is to activate the customer group support. This can be done with the configuration parameter CustomerGroupSupport, from the Admin SysConfig option. Using the parameter CustomerGroupAlwaysGroups, you can specify the default groups for a newly added customer, so that every new account will be automatically added to these groups.

Through the link "Customers <-> Groups" you can manage which customer shall belong to the different groups (see Figure below).

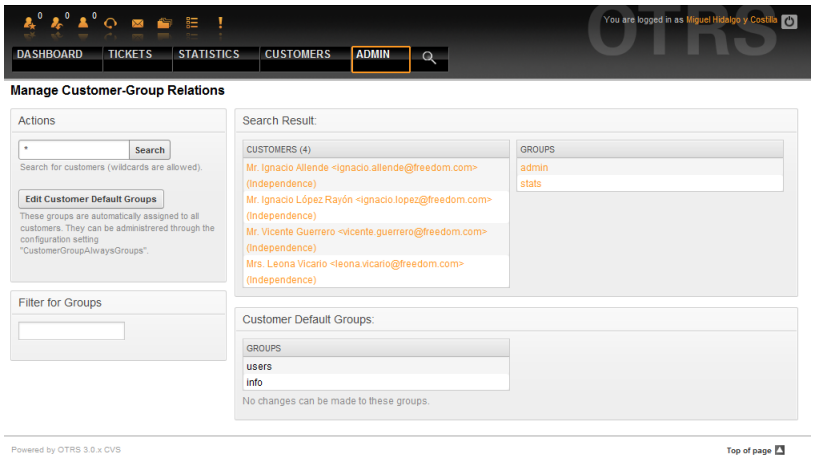

Figure: Customer-Group relations management.

To define the different groups a customer should be part of and vice versa, click on the corresponding customer username or group (see below the Figures 5.16 and 5.17, respectively).

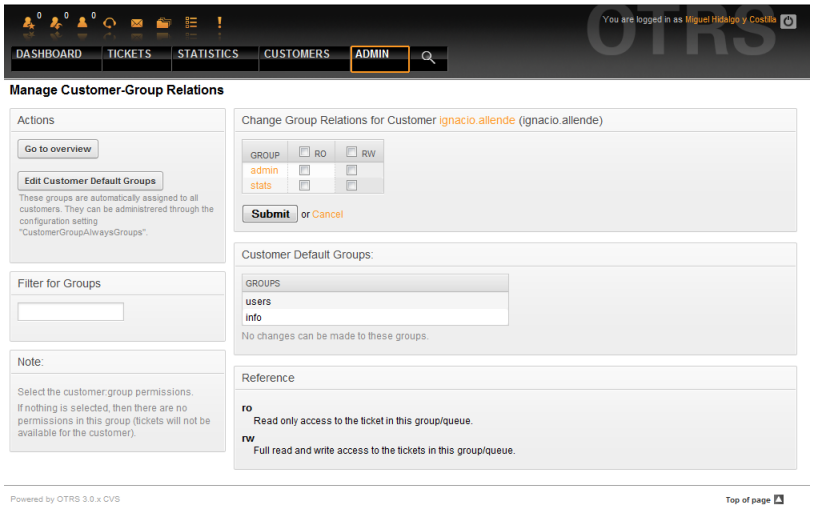

Figure: Change Group relations for a Customer.

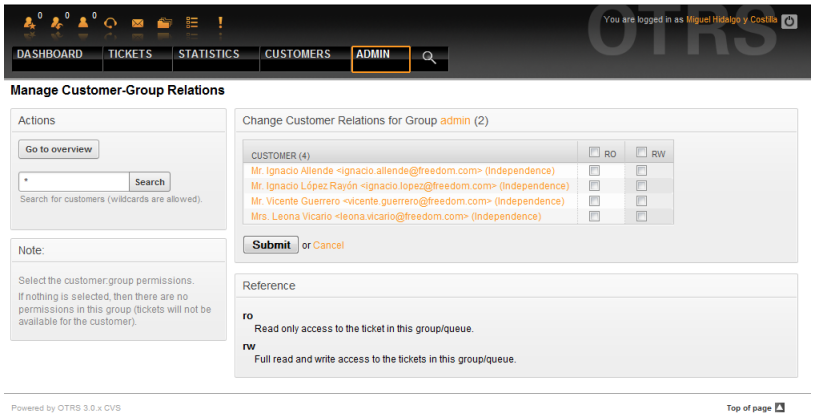

Figure: Change Customer relations for a Group.

Queues

Clicking on the link "Queues" of the Admin page, you can manage the queues of your system (see Figure below). In a new 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.

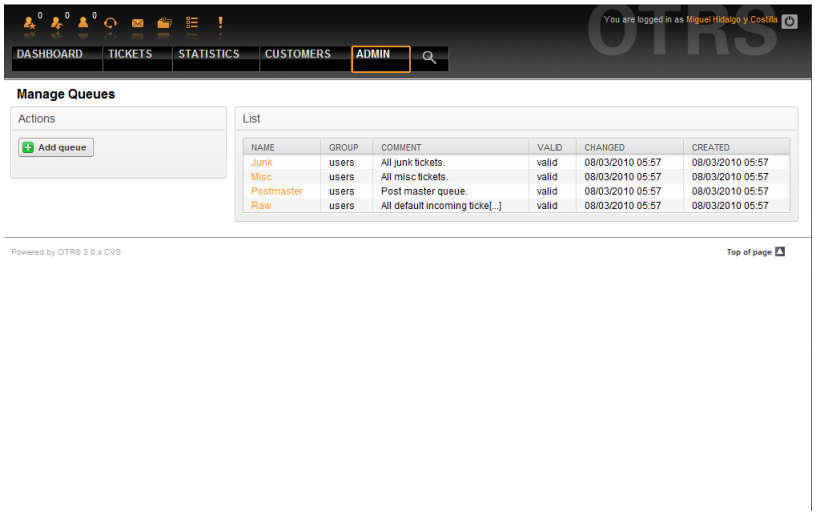

Figure: Queue management.

Here you can add queues (see Figure below) and modify them. You can specify the group that should use the queue. You can also set the queue as a sub-queue of an existing queue.

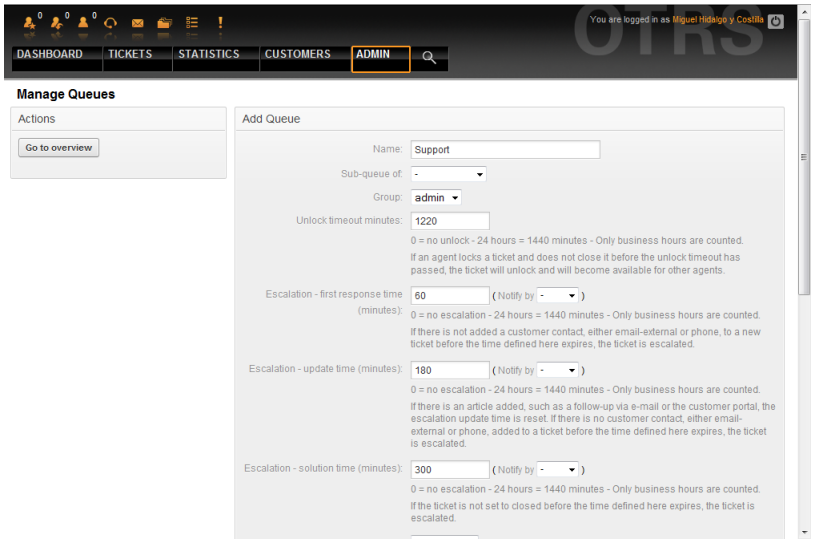

Figure: Adding a new queue.

You can define an unlock timeout for a queue - if an agent locks a ticket and does not close it before the unlock timeout has passed, the ticket will be automatically unlocked and made available for other agents to work on.

There are three escalation time settings that can be associated at queue level:

Escalation - First Response Time

- After creation of the ticket, if the time defined here expires without any communication to the customer, either by email or phone, the ticket is escalated.

Escalation - Update Time

- If there is any customer followup via e-mail or the customer portal and recorded in the ticket, the escalation update time is reset. If there is no customer contact before the time defined here expires, the ticket is escalated.

Escalation - Solution Time

- If the ticket is not closed before the time defined here expires, the ticket is escalated.

With 'Ticket lock after a follow-up', you can define if a ticket should be set to 'locked' to the old owner if a ticket that has been closed and later is re-opened. This ensures that a follow up for a ticket is processed by the agent that has previously handled that ticket.

The parameter for the system address specifies the email address that will be used for the outgoing tickets of this queue. There is also possibility to associate a queue with a salutation and a signature, for the email answers. For more detailed information, please refer to the sections email addresses, salutations and signatures.

Note

As with agents, groups and customers, queues cannot be deleted, only deactivated, by setting the Valid option to *invalid* or *invalid-temporarily*.

Salutations, signatures, attachments and responses

Salutations

A salutation is a text module for a response. Salutations can be linked to one or more queues, as described in the section about queues. A salutation is used only if a ticket from a queue the salutation is linked to, is answered. To manage the different salutations of your system, use the "Salutations" link of the admin area (see Figure below).

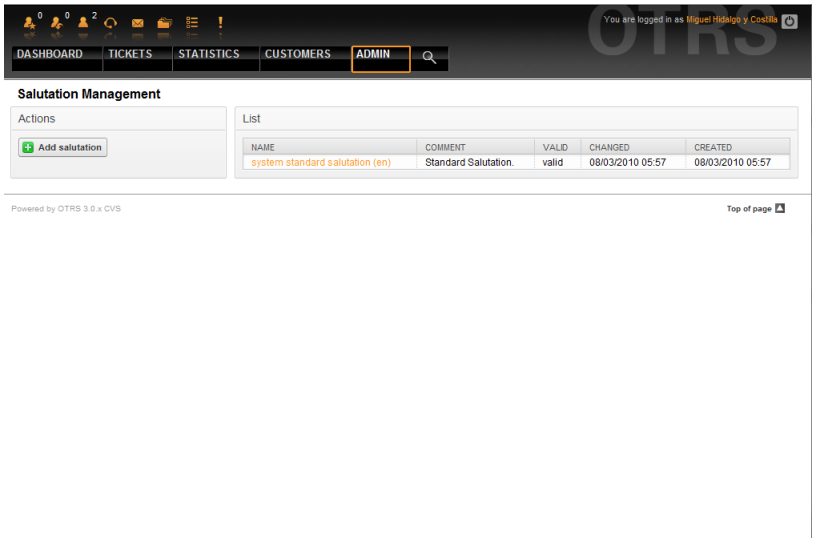

Figure: Salutation management.

After a default installation there is already one salutation available, "system standard salutation (en)".

To create a new salutation, press the button "Add salutation", provide the needed data and submit it (see Figure below).

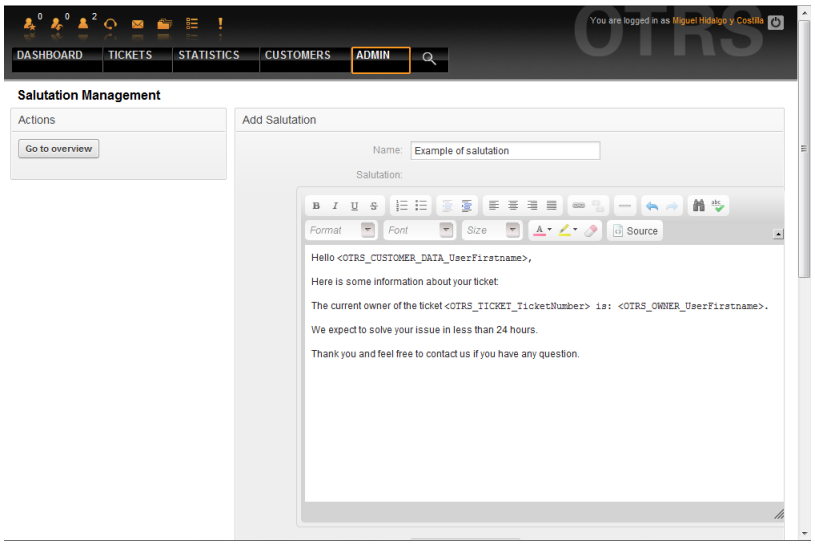

Figure: Adding a new salutation.

It is possible to use variables in salutations. When you respond to a ticket, the variable names will be replaced by their values.

The different variables you can use in responses are listed in the lower part of the salutation screen. If you use, for example, the variable <OTRS_LAST_NAME> the last name of the ticket's sender will be included in your reply.

Note

As with other OTRS entities, salutations cannot be deleted, only deactivated by setting the Valid option to *invalid* or *invalid-temporarily*.

Signatures

Another text module for a response is the signature. Signatures can be linked to a queue, as described in the section about the queues. Only if a signature is linked to a queue will it be included into the response text. Through the "Signatures" link of the Admin page, you can manage the signatures in your system (see Figure below).

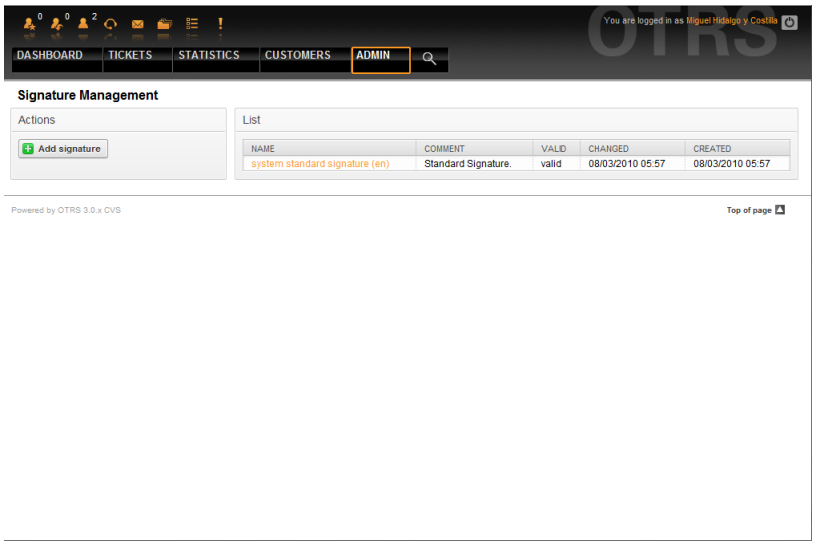

Figure: Signatures management.

After a fresh installation of OTRS, there is one predefined signature stored in your system, "system standard signature (en)".

To create a new signature, press the button "Add signature", provide the needed data and submit it (see Figure below).

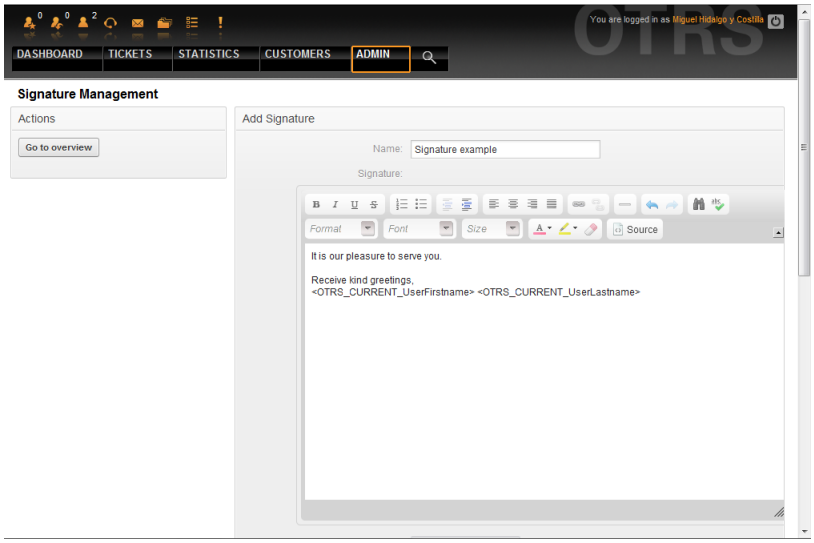

Figure: Adding a new signature.

Like salutations, signatures can also contain dynamical content, such as the first and last name of the agent who answers the ticket. Here too, variables can be used to replace the content of the signature text for every ticket. See the lower part of the signatures screen for the variables which can be used. If you include the variable <OTRS_LAST_NAME> in a signature for example, the last name of the agent who answers the ticket will replace the variable.

Note

As with salutations, signatures too cannot be deleted, only deactivated by setting the Valid option to *invalid* or *invalid-temporarily*.

Attachments

You can also optionally add one or more attachments for a response. If the response is selected, the attachments will be attached to the message composition window. If necessary, the agent can remove the attachment from an individual response before sending it to the customer.

Through the "Attachment" link of the Admin page, you can load the attachments into the database of the system (see Figure below).

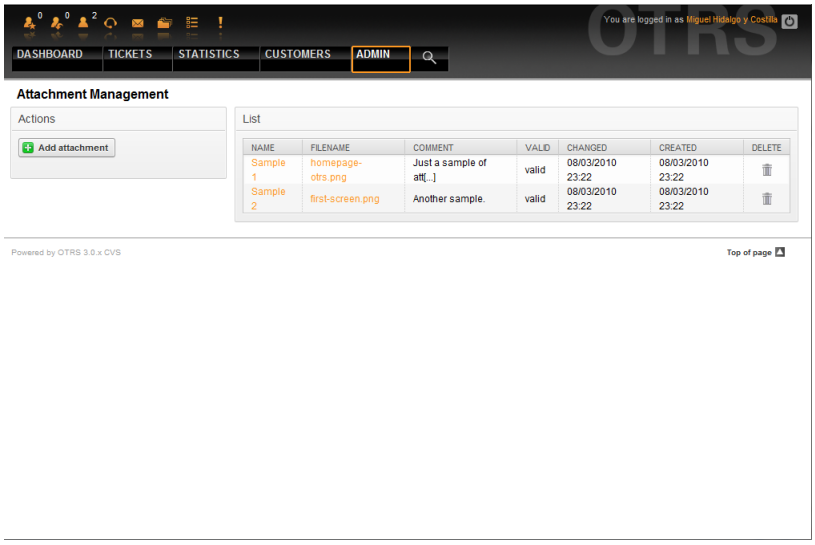

Figure: Attachments management.

To create a new attachment, press the button "Add attachment", provide the needed data and submit it (see Figure below).

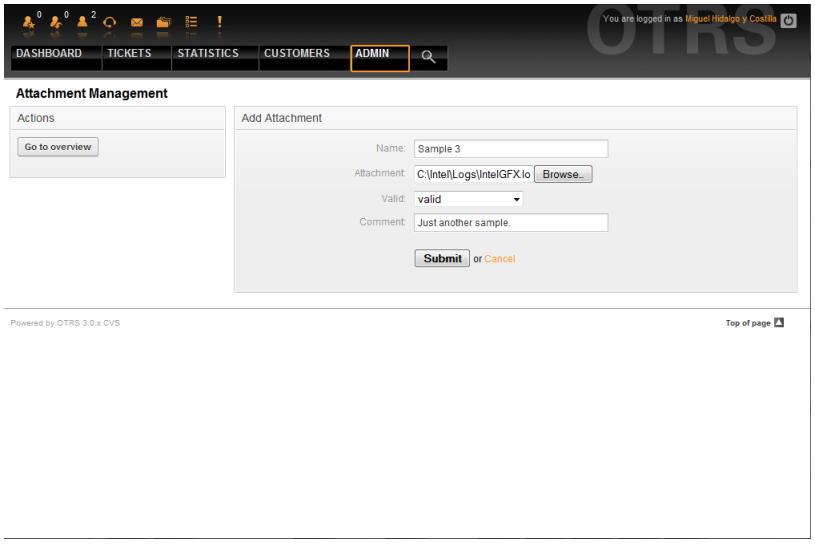

Figure: Adding a new attachment.

If an attachment is stored it can be linked to one or more responses. Click on the "Attachment <-> Responses" link of the Admin page (see Figure below).

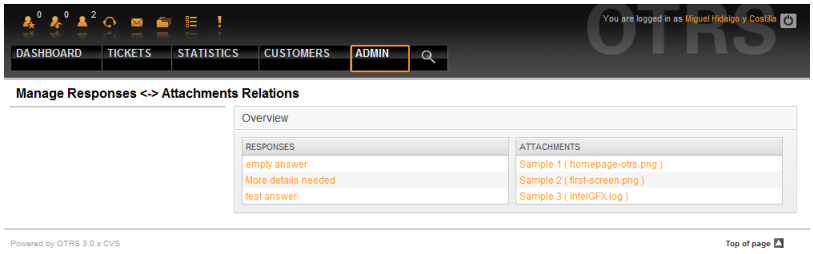

Figure: Linking Attachments to Responses.

To associate different attachments with a specific response and vice versa, click on the corresponding response name or attachment (see below the Figures 5.27 and 5.28, respectively).

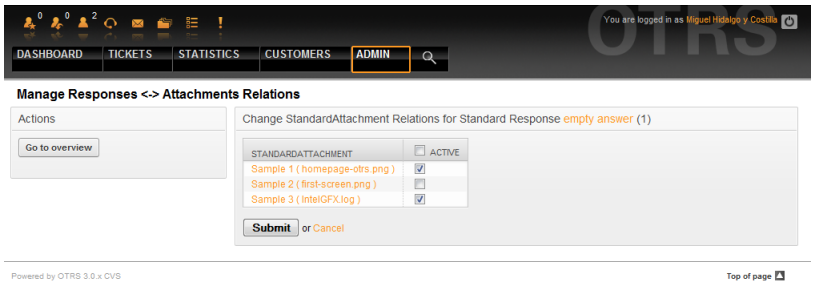

Figure: Change Attachment relations for a Response.

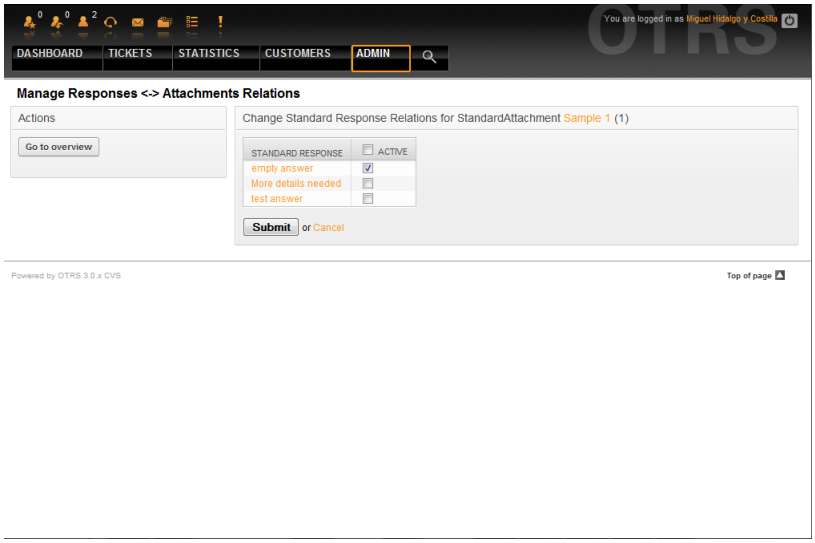

Figure: Change Response relations for an Attachment.

Responses

To speed up the answering of tickets and to standardize the look of answers, you can define responses in OTRS. A response can be linked to one or more queues and vice versa. In order to be able to use a response quickly, the different responses are displayed below every ticket in the QueueView or in "My Queues".

On a fresh OTRS installation, the "empty answer" response is defined for every queue. Clicking the "Responses" link on the Admin page brings you to the Responses management page (see Figure below).

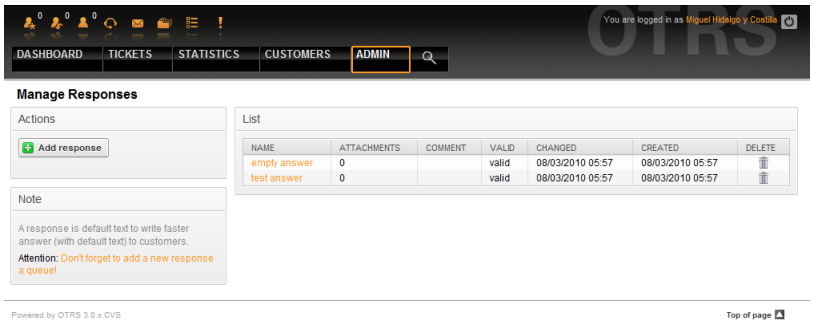

Figure: Responses management.

To create a new response, click on the "Add response" button, provide the required data and submit it (see Figure below).

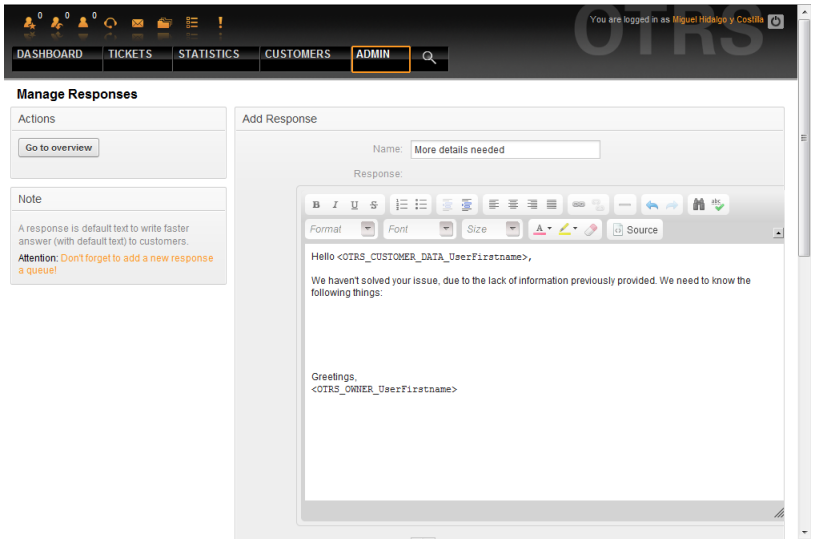

Figure: Adding a response.

To add/remove responses to one or more queues, click on the "Responses <-> Queues" link on the Admin page (see Figure below). You can also use filters to get information on a specific entity.

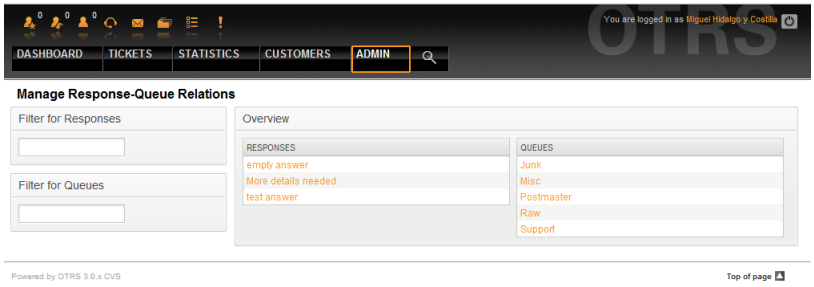

Figure: Response-Queue relations management.

To define the different responses that will be available for a queue and vice versa, click on the corresponding response or queue (see below the Figures 5.32 and 5.33, respectively).

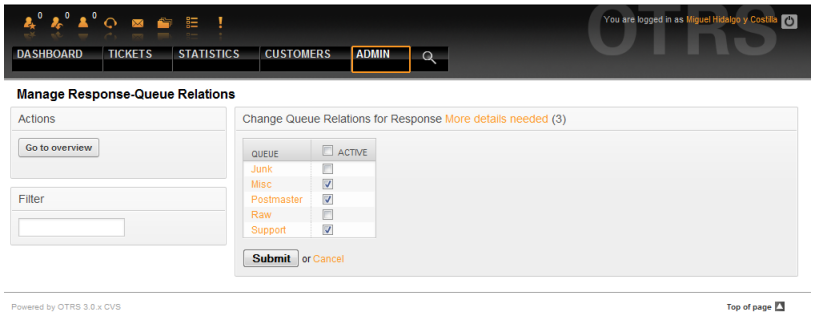

Figure: Change Queue relations for a Response.

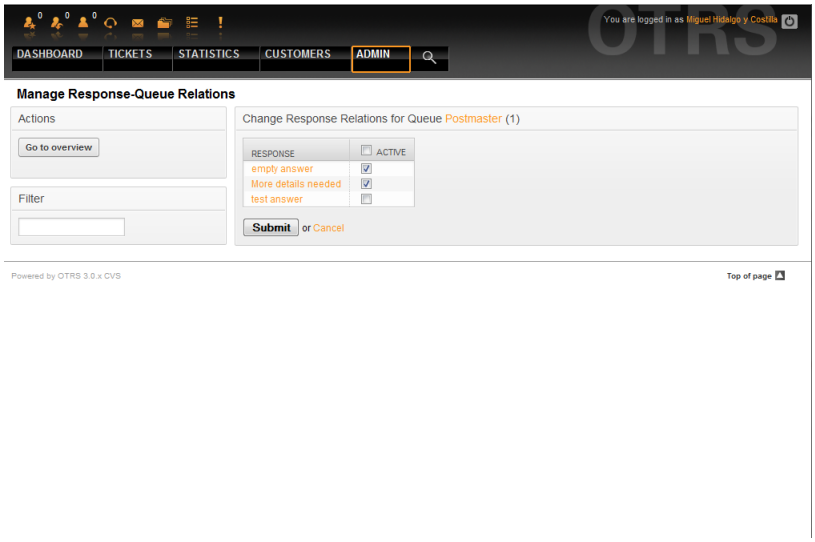

Figure: Change Response relations for a Queue.

The structure of a response is intuitive. It includes the salutation associated with the queue, then the text of the response, then the quoted ticket text, and finally the signature associated with the queue.

Auto responses

OTRS allows you to send automatic responses to customers on the occurence of certain events, such as the creation of a ticket in certain queue, reception of a follow-up message on a ticket, closure or rejection of a ticket, etc. To manage such responses, click the link "Auto responses" on the Admin page (see Figure below).

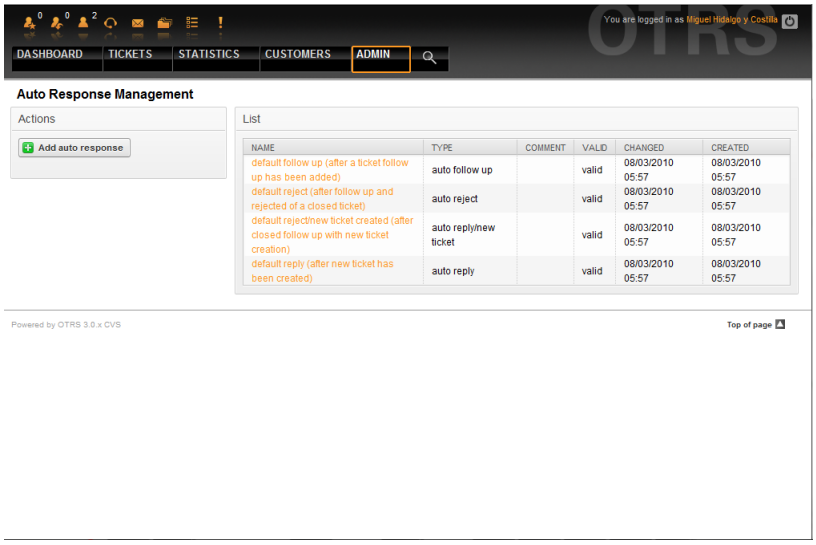

Figure: Auto Response management.

To create an automatic response, click on the button "Add auto response", provide the needed data and submit it (see Figure below).

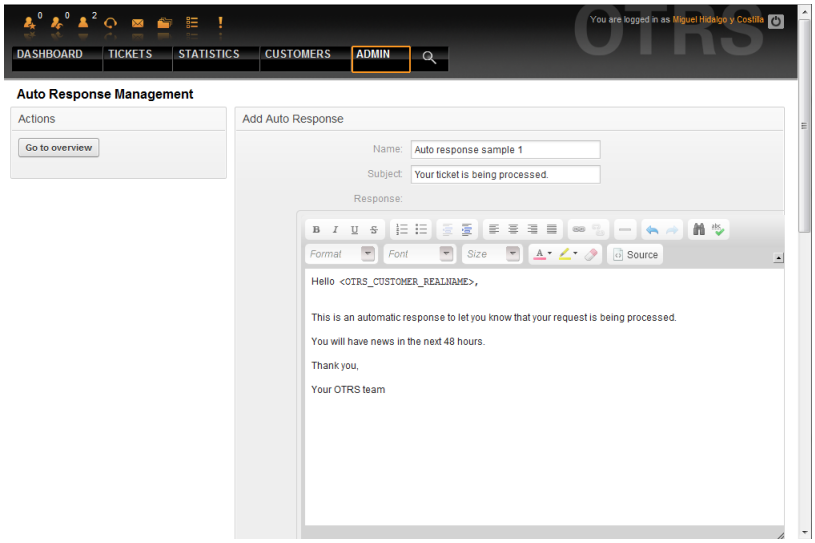

Figure: Adding an Auto Response.

The subject and text of auto responses can be generated by variables, just as in signatures and salutations. If you insert, for example, the variable <OTRS_CUSTOMER_EMAIL[5]> into the body of the auto answer, the first 5 lines of the customer mail text will be inserted into the auto answer. You will find more details about the valid variables that can be used at the bottom of the screen shown in the Figure.

For every automatic answer, you can specify the event that should trigger it. The system events that are available after a default installation are described in the Table 5-3.

Table 5.3. Events for Auto answers

Name	Description
auto reply	Creation of a ticket in a certain queue.
auto reply/new ticket	Reopening of an already closed ticket, e.g. if a customer replies to such ticket.
auto follow up	Reception of a follow-up for a ticket.
auto reject	Automatic rejection of a ticket, done by the system.
auto remove	Deletion of a ticket, done by the system.

Note

As with other OTRS entities, Auto responses too cannot be deleted, only deactivated, by setting the Valid option to *invalid* or *invalid-temporarily*.

To add an auto response to a queue, use the "Auto Response <-> Queues" link on the Admin page (see Figure below). All system events are listed for every queue, and an auto answer with the same event can be selected or removed via a listbox.

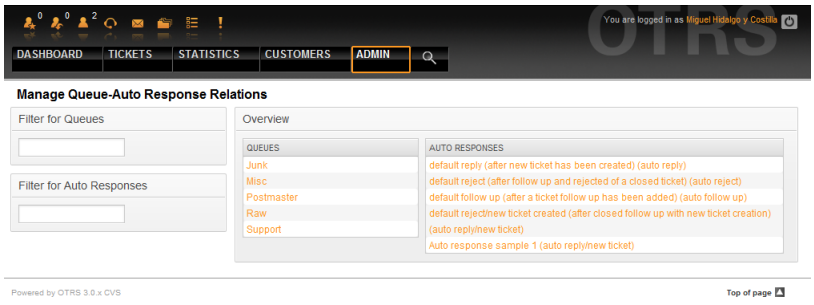

Figure: Queue-Auto Response relations management.

To define the different auto responses that will be available for a queue, click on the corresponding queue name (see Figure below). It is also possible to edit an existing auto response - to do so, click on the reponse and edit in the same manner as editing a new auto response.

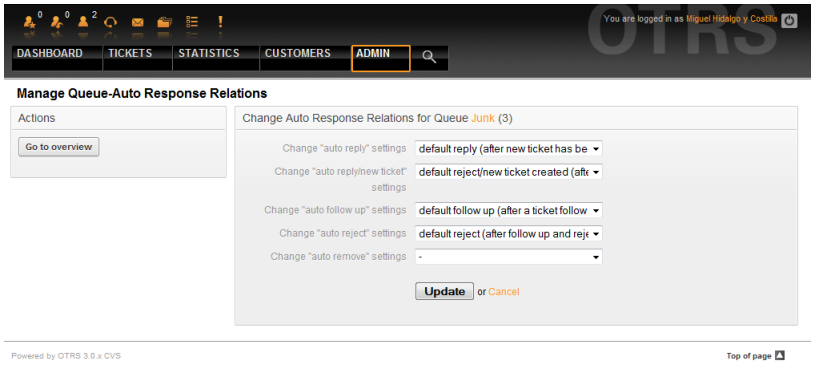

Figure: Change Auto Response relations for a Queue.

Email addresses

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 the "Email Addresses" link from the Admin page to manage all email addresses of the system (see Figure below).

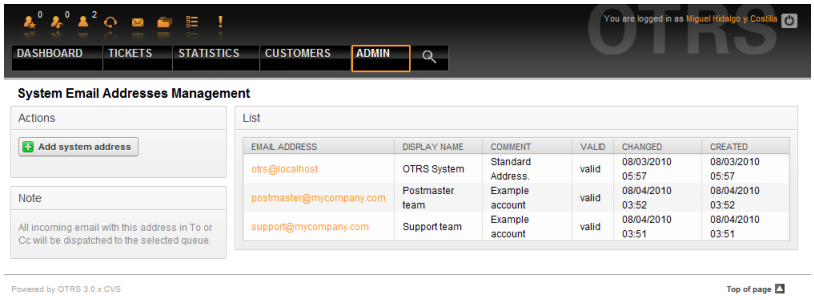

Figure: System Email Addresses management.

If you create a new mail address (see Figure below) you can select the queue or sub queue to be linked with it. This link enables the system to sort incoming messages via the address in the To: field of the mail into the right queue.

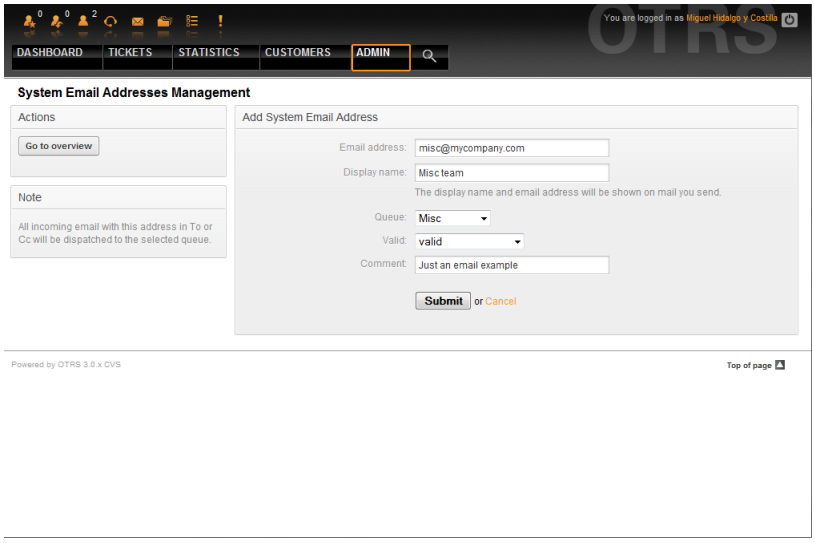

Figure: Adding a system Email Address.

Note

As with other OTRS entities, email addresses cannot be deleted, only deactivated by setting the Valid option to *invalid* or *invalid-temporarily*.

Notifications

OTRS allows notifications to be sent to agents and customers, on the occurence of certain events. Agents can set the system events for their own notifications via the preferences link.

Through the "Agent Notifications" link on the Admin page, you can manage the notifications of your system (see Figure below). You can use filters to list only certain notifications.

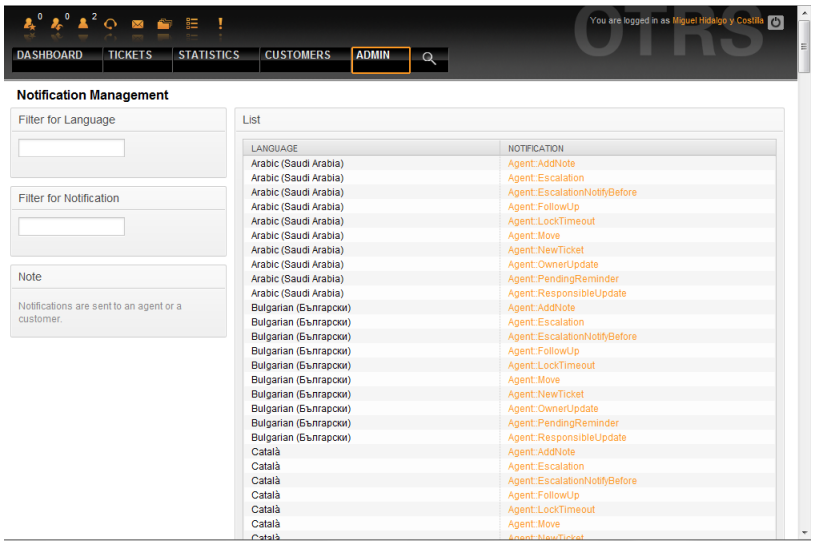

Figure: Notification management.

You can customize the subject and the text of the notifications. Click on the notification you want to change from the list, and its content will get loaded for editing (see Figure). Please note that there is a notification with the same name for each of the available languages.

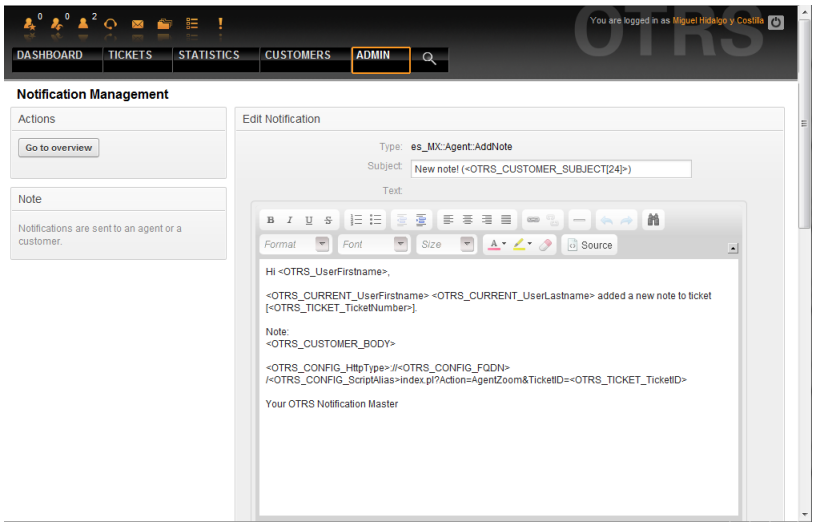

Figure: Customizing a Notification.

Just as with signatures and salutations, it is possible to dynamically create the content of a notification, by using special variables. You can find a list of variables at the bottom of the screen shown in the Figure.

It is also possible to create notifications based on events. You can specify in detail when and to whom you want such a notification to be sent. You can choose from a wide variety of parameters, such as: recipient group(s), agent(s), role(s), email address(es), type of event triggering the notification, ticket-type, state, priority, queue, lock, service, SLA, etc.

In order to see a list of all event based notifications, click on the link "Notifications (Event)" on the Admin page (see Figure).

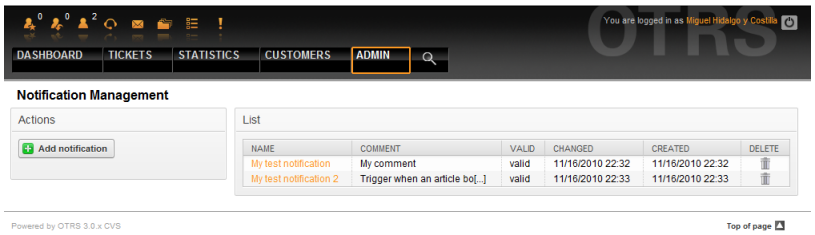

Figure: Event based Notification management.

As shown in Figure, you can create a new notification by clicking on the Add button (see Figure).

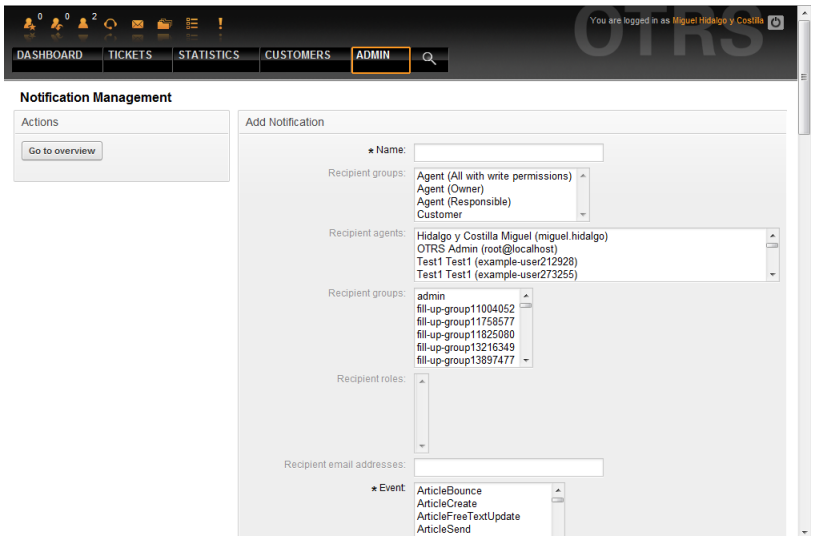

Figure: Registering an Event based Notification management.

Please note that the content of the event based notifications can also be dynamically created by using the special variables listed at the bottom of the screen shown in the Figure.

SMIME

OTRS can process incoming S/MIME encoded messages and sign outgoing mails. Before this feature can be used, you need to activate it and change some config parameters in the SysConfig.

The "S/MIME Certificates" link on the Admin page allows you to manage your S/MIME certificates (see Figure below). You can add or remove certificates, and also search through the SMIME data.

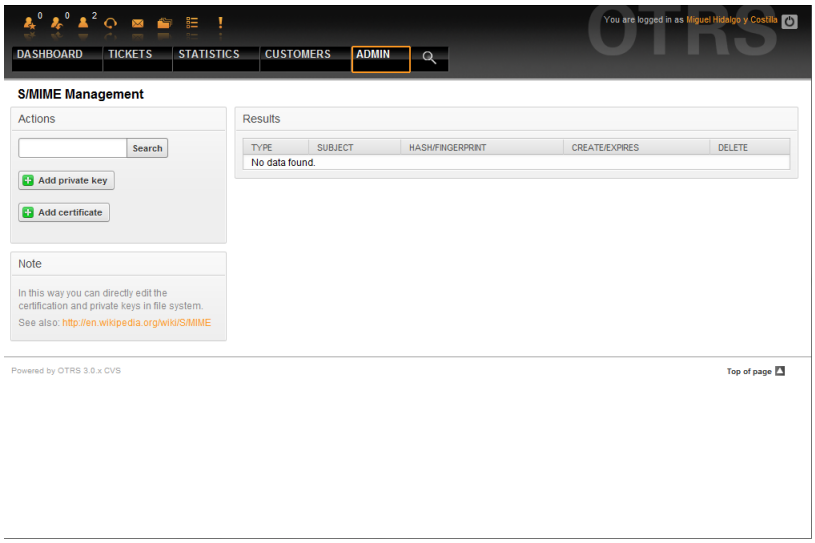

Figure: S/MIME management.

PGP

OTRS handles PGP keys, which allows you to encrypt/decrypt messages and to sign outgoing messages. Before this feature can be used, you need to activate it and change some config parameters in the SysConfig.

Through the "PGP Keys" link on the Admin page, it is possible to manage the key ring of the user who shall be used for PGP with OTRS (see Figure below), e.g. the local OTRS user or the web server user. It is possible to add and remove keys and signatures, and you can search through all data in your key ring.

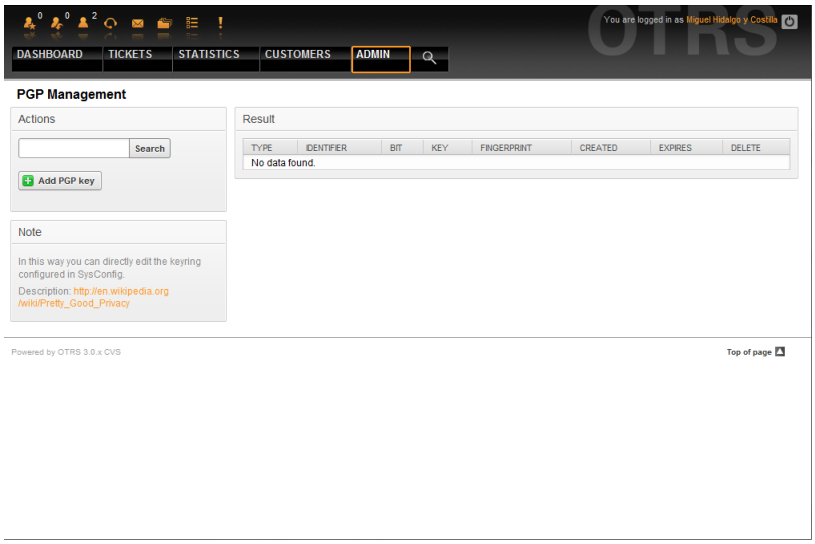

Figure: PGP management.

States

Through the "States" link on the Admin page, you can manage the different ticket states you want to use in the system (see Figure below).

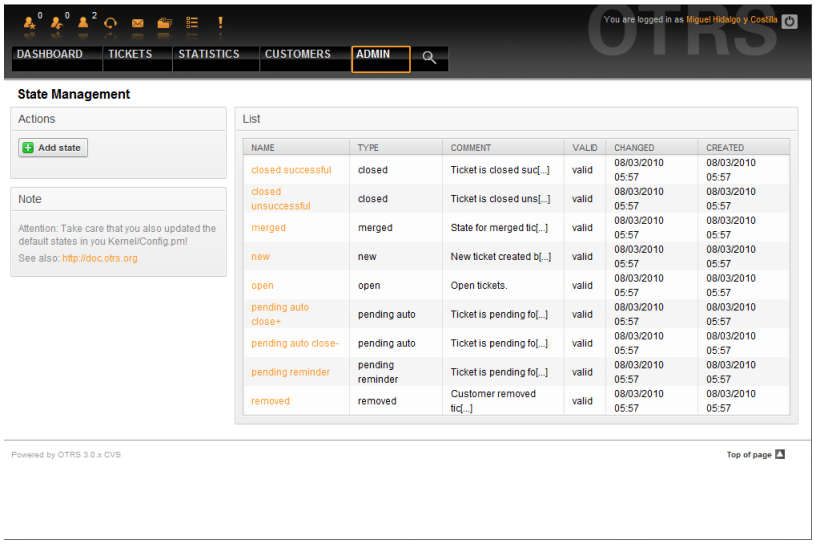

Figure: State management.

After a default setup, there are some states defined:

- closed successful
- closed unsuccessful
- merged
- new
- open
- pending auto close+
- pending auto close-
- pending reminder
- removed

Every state is linked to a type, which needs to be specified if a new state is created. By default the state types are:

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

SysConfig

The SysConfig link leads to the section where many OTRS configuration options are maintained.

The SysConfig link on the Admin page loads the graphical interface for system configuration (see Figure below). You can upload your own configuration files for the system, as well as backup all your current settings into a file. Almost all configuration parameters of the OTRS framework and installed applications can be viewed and changed through this interface. Since all configuration parameters are sorted into groups and sub groups, it is possible to navigate quickly through the multitude of the parameters. It is also possible to perform a full-text search through all the configuration parameters.

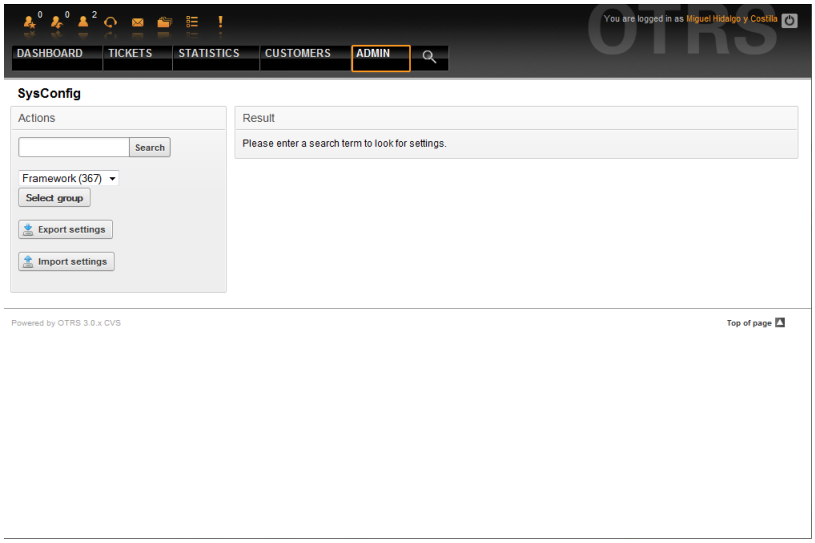

Figure: The graphical interface for system configuration (SysConfig).

The graphical interface for system configuration is described in more detail in the chapter "Configuring the system through the web interface".

Using mail accounts

There are several possibilities to transport new emails into the ticket system. One of them is the otrs.PostMaster.pl module that pipes the mails directly into the system. Another possibility is the use of mail accounts which can be administrated through the web interface. The "PostMaster Mail Accounts" link on the Admin page loads the management console for the mail accounts (see Figure below). OTRS supports the mail protocols: POP3, POP3S, IMAP and IMAPS.

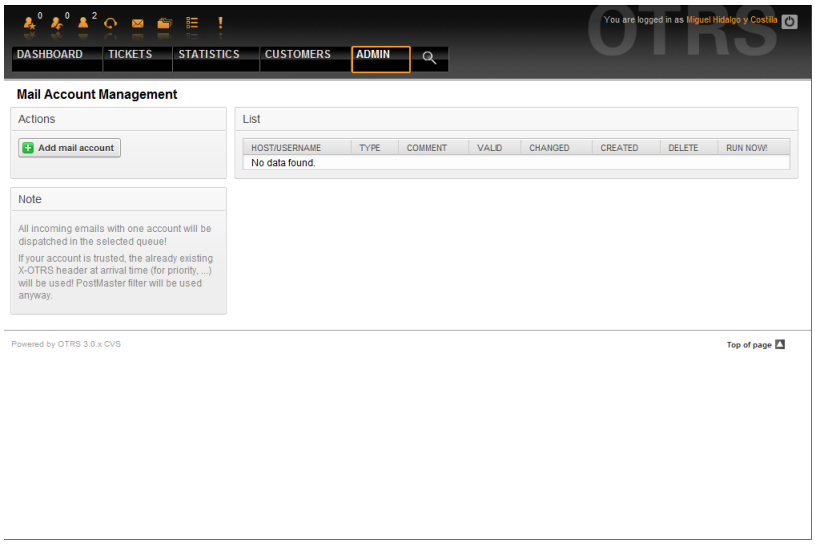

Figure: Mail account management.

See the section about the PostMaster mail accounts for more details.

Filtering incoming messages

OTRS has the capability to filter incoming messages, as reflected by incoming messages being sorted automatically into queues, or spam mails being moved into a specific queue. It is irrelevant whether `PostMaster.pl` or mail accounts are used to get messages into the ticket system. Filter rules can be created through the link "PostMaster Filter" on the Admin page (see Figure below).

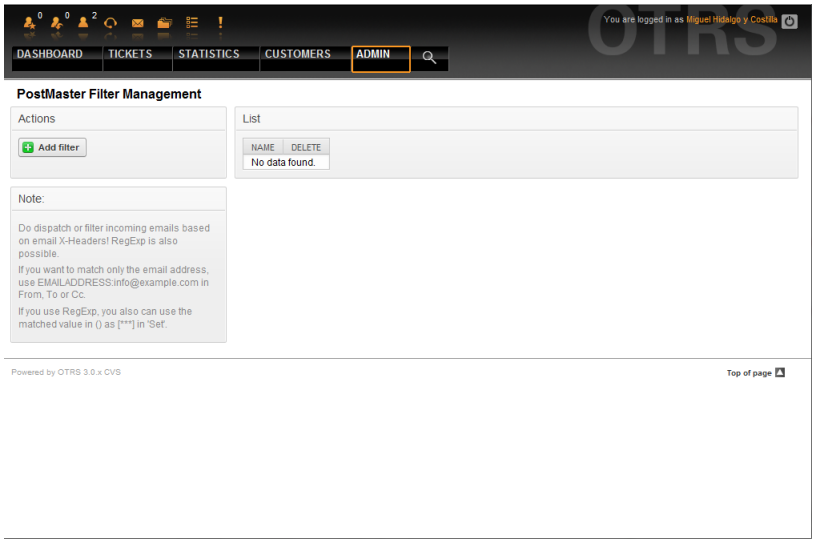

Figure: PostMaster filter management.

A filter consists of one or more criteria that must match for the defined actions to be executed on the email. Filter criteria may be defined for the headers or the body of an email, e.g. search for specific header entries or strings in the body, even regular expressions are allowed. All actions for a filter rule are triggered by X-OTRS headers, which are inserted if the filter criteria match. The ticket system evaluates the inserted X-OTRS headers and executes the specific actions. X-OTRS headers can be used to sort an incoming message into a specific queue, change the priority of the message or ignore the message and not deliver it to the system. The Table 5-4 lists the different X-OTRS headers and their meaning.

Note: You also can use X-OTRS-FollowUp-* headers for follow up emails.

Table 5.4. Function of the different X-OTRS-headers

Name	Possible values	Description
X-OTRS-Priority:	1 very low, 2 low, 3 normal, 4 high, 5 very high	Sets the priority of a ticket.
X-OTRS-Queue:	Name of a queue in the system.	Sets the queue where the ticket shall be sorted. If set in X-OTRS header, all other filter rules that try to sort a ticket into a specific queue are ignored.
X-OTRS-Lock:	lock, unlock	Sets the lock state of a ticket.
X-OTRS-Ignore:	Yes or True	If this X-OTRS header is set to "Yes", the incoming message

Name	Possible values	Description
		will completely be ignored and never delivered to the system.
X-OTRS-State:	new, open, closed successful, closed unsuccessful, ...	Sets the next state of the ticket.
X-OTRS-State-PendingTime:	e. g. 2010-11-20 00:00:00	Sets the pending time of a ticket (you also should sent a pending state via X-OTRS-State).
X-OTRS-Type:	default (depends on your setup)	Sets the type of a ticket (if Ticket::Type support is active).
X-OTRS-Service:	(depends on your setup)	Sets the service of a ticket (if Ticket::Service support is active).
X-OTRS-SLA:	(depends on your setup)	Sets the SLA of a ticket (if Ticket::Service support is active).
X-OTRS-CustomerUser:	CustomerUser	Sets the customer user for the ticket.
X-OTRS-CustomerNo:	CustomerNo	Sets the customer ID for this ticket.
X-OTRS-SenderType:	agent, system, customer	Sets the type of the ticket sender.
X-OTRS-ArticleType:	email-external, email-internal, email-notification-ext, email-notification-int, phone, fax, sms, webrequest, note-internal, note-external, note-report	Sets the article type for the incoming ticket.
X-OTRS-DynamicField-<DynamicFieldName>:	Depends on Dynamic Field configuration (Text: Notebook, Date: 2010-11-20 00:00:00, Integer: 1)	Saves an additional info value for the ticket on <DynamicFieldName> Dynamic Field.
X-OTRS-Loop:	True	If this X-OTRS header is set, no auto answer is delivered to the sender of the message (mail loop protection).

A name must be specified for every filter rule. Filter criteria can be specified in the section "Filter Condition". Choose via the listboxes for "Header 1", "Header 2" and so on for the parts of the messages where you would like to search, and specify on the right side the values you wish to filter on. In the section "Set Email Headers", you can choose the actions that are triggered if the filter rules match. You can select for "Header 1", "Header 2" and so on to select the X-OTRS-Header and set the associated values (see Figure below).

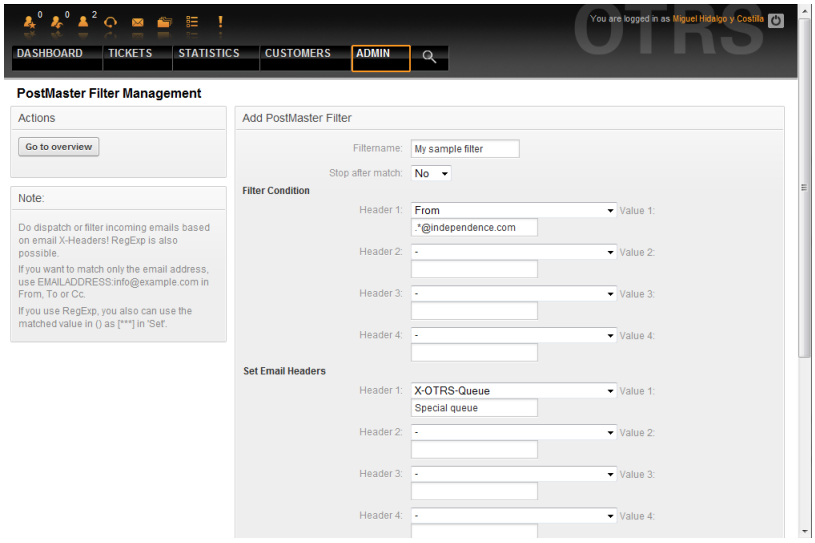

Figure: Add a PostMaster filter.

Example 5.1. Sort spam mails into a specific queue

A useful filter rule could be to let OTRS automatically move mails marked for spam with a spam detection tool such as SpamAssassin, into the "Junk" queue. SpamAssassin adds the "X-Spam-Flag" header to every checked mail. When the mail is marked as spam, the Header is set to "Yes". So the filter criteria would be "X-Spam-Flag: Yes". To create a filter rule with this criteria you can insert the name as, for example, "spam-mails". In the section for "Filter Condition", choose "X-Spam-Flag:" for "Header 1" from the listbox. Insert "Yes" as value for this header. Now the filter criteria is specified. To make sure that all spam mails get directed into the "Junk" queue, choose in the section for "Set Email Headers", the "X-OTRS-Queue:" entry for "Header 1". Specify "Junk" as value for this header. Finally add the new filter rule to activate it for new messages into the system.

There are additional modules, that can be used to filter incoming messages more specifically. These modules might be useful on larger, more complex systems.

Executing automated jobs with the GenericAgent

The GenericAgent is a tool to execute tasks automatically. In its absence such tasks would need to be done by a human person, a real agent. The GenericAgent, for example, can close or move tickets, send notifications on escalated tickets, etc.

Click the link "GenericAgent" on the Admin page (see Figure below). A table with currently automated jobs is displayed which can be edited to switch to executing jobs manually, or removing them.

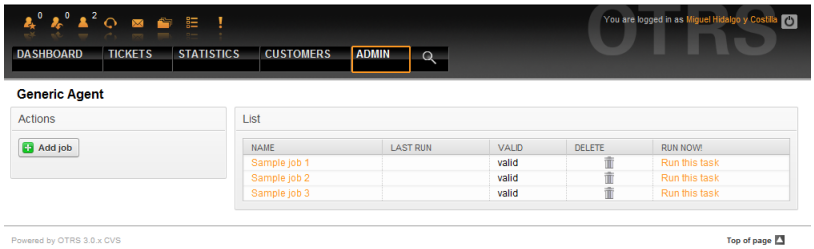

Figure: Job list for the GenericAgent.

Click the "Add job" button to create a new job. You first need to supply a name for the job, as also the times when the job should be executed. Different criteria to select the tickets to work on and the new properties of those tickets can also be set (see Figure below).

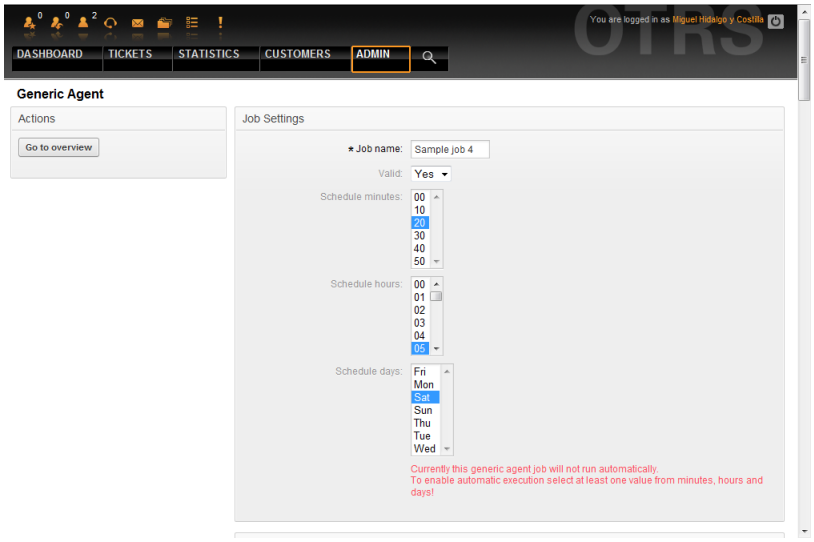

Figure: Creating a job for the GenericAgent.

On completing the job creation, all affected tickets by the job are listed. This list helps you verify that the job has the expected behavior. No changes are made to these tickets yet. The job will actually be activated only when it is saved into the job list.

Admin email

OTRS administrators can send messages to specific users or groups. The "Admin Notification" link on the Admin page opens the screen where the agents and groups that should be notified can be selected (see Figure below).

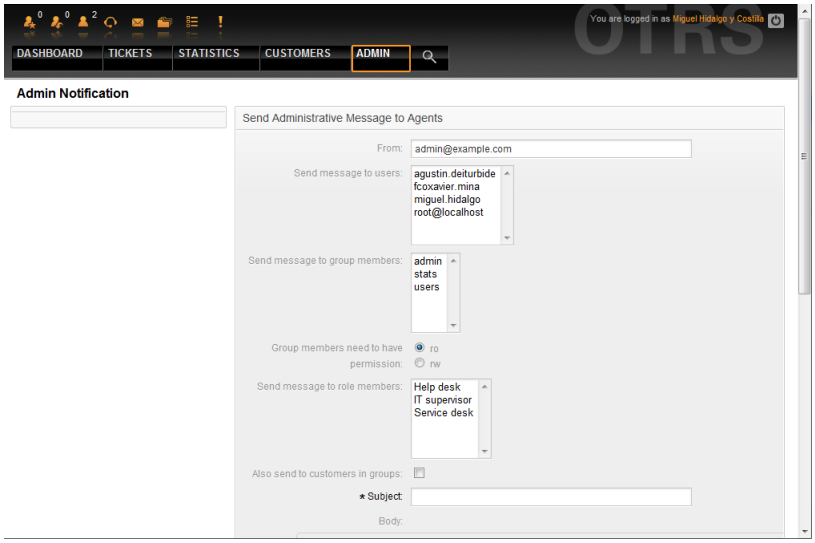

Figure: Admin notification.

It is possible to specify the sender, subject and body text of the notification. You can also select the agents, groups and roles who should receive the message.

Session management

You can see all logged in users and their session details by clicking the "Session Management" link in the admin area (see Figure below).

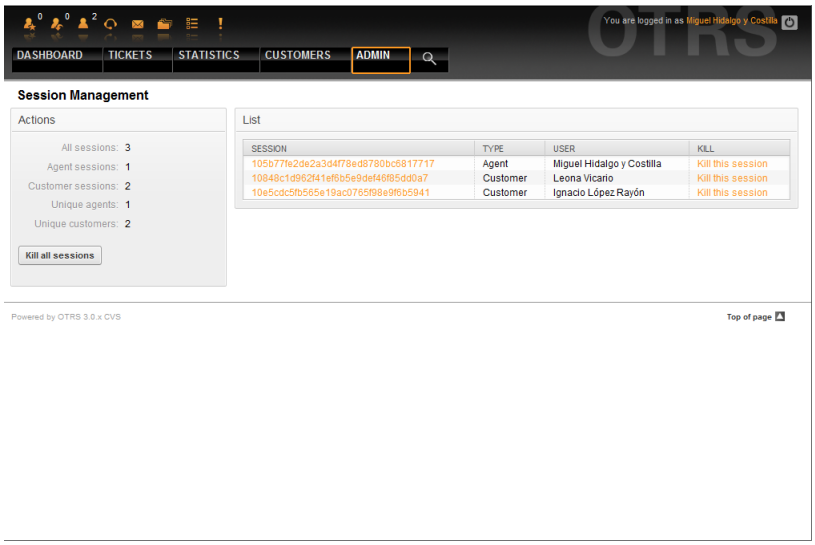

Figure: Session management.

Some statistics about all active sessions are displayed, e.g. how many agents and customer users are logged in, number of active sessions. Any individual session can be removed by clicking on the *Kill this session* link on the right-hand side of the list. You also have the option to *Kill all sessions*, which can be useful if you'd like to bring the system down. Detailed information for every session is available, too (see Figure below).

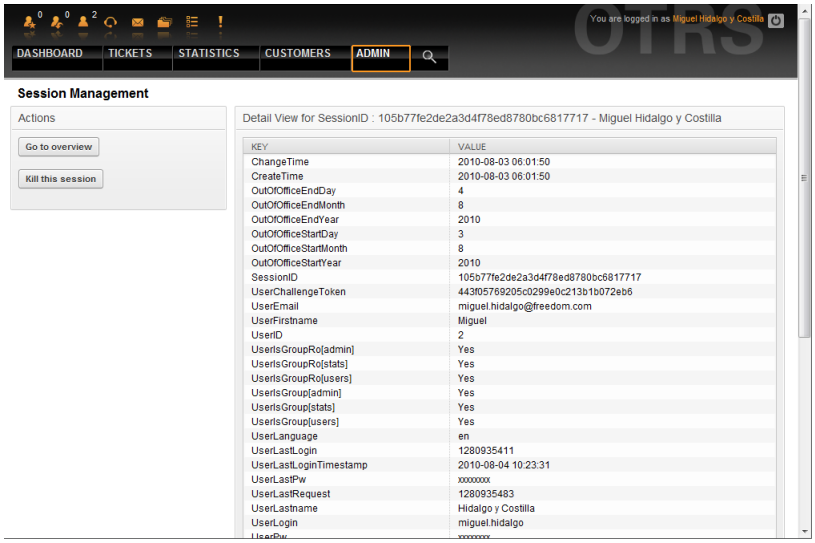

Figure: Session details.

System Log

The "System Log" link on the Admin page shows the log entries of the system, reverse chronologically sorted with most recent first (see Figure below).

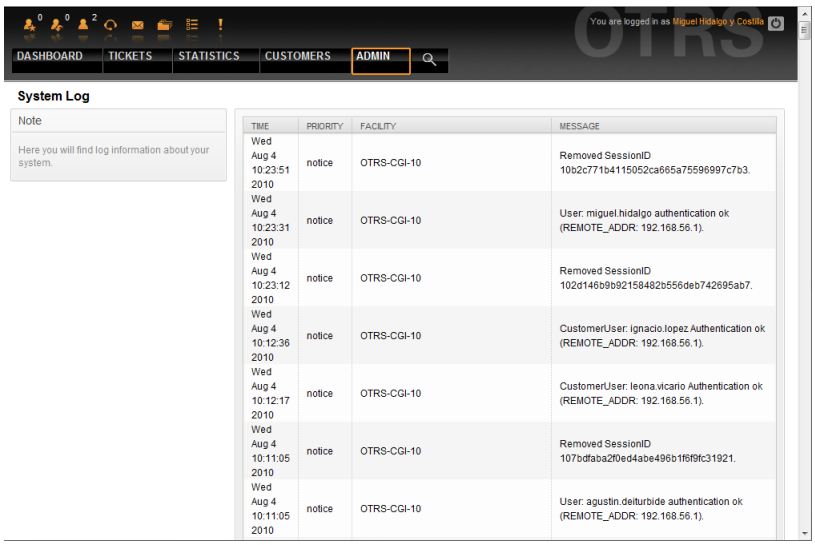

Figure: System Log.

Each line in the log contains a time stamp, the log priority, the system component and the log entry itself.

Note

System logs are available via the web interface only on Linux / Unix systems. On Windows systems, you can see the logs using a text editor and opening the file `[install_dir]otrs\var\log\otrs.log`.

SQL queries via the SQL box

The "SQL Box" link on the Admin page opens a screen that lets you query the content of the tables in the OTRS database (see Figure below). It is not possible to change the content of the tables, only select queries are allowed.

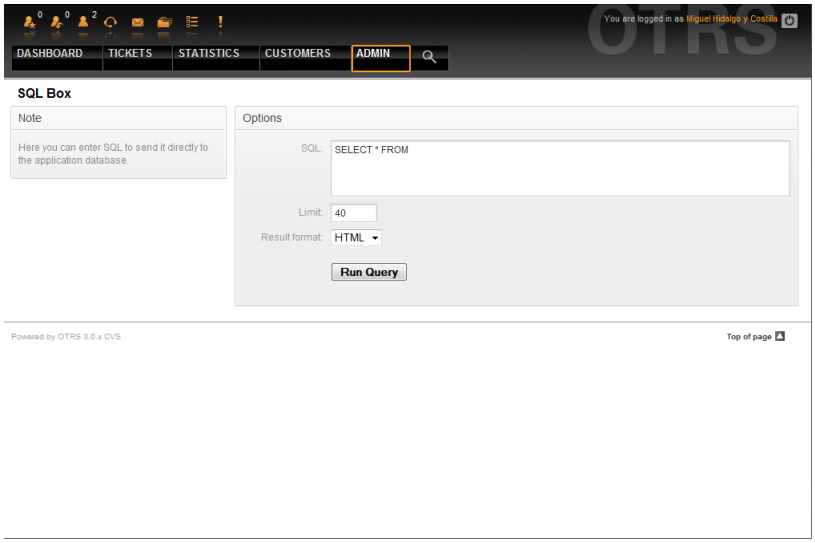

Figure: SQL Box.

Package Manager

Using the "Package Manager" link on the Admin page, you can install and manage packages that extend the functionality of OTRS (see Figure below). See the Additional applications section for a discussion on the extensions that are available from the OTRS repositories.

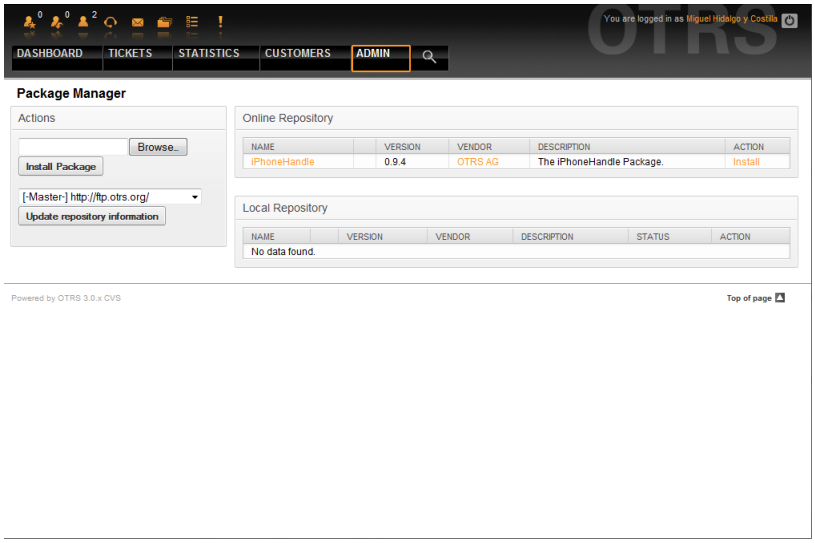

Figure: Package Manager.

The Package Manager shows the OTRS addon packages you currently have installed on your server, together with their version numbers.

You can install packages from a remote host by selecting the repository in the *Online Repository* section, and clicking the *Update repository information* button. The available packages are displayed in the corresponding table. The right side of the screen shows the available packages. To install a package, click on *Install*. After installation, the package is displayed in the *Local Repository* section.

To upgrade an installed package, the list of available packages in the online repository will show *Upgrade* in the Action column for any package that has a higher version than the one locally installed. Just click Upgrade and it will install the upgrade on your system.

In some cases, such as when your OTRS system is not connected to the Internet, you can also install packages you have downloaded to a local disk. Click the *Browse* button on the Actions side bar, and select the .opm file on your disk. Click *Open* and then *Install Package*. After installation the package is displayed in the *Local Repository* section. You can use the same steps for updating a package that is already installed.

In special cases, you might want to configure the Package Manager, e.g., to use a proxy or to use a local repository. Just take a look at the available options in SysConfig under Framework:Core::Package.

Web Services

The Web Services link leads to the graphical interface where web services (for the OTRS Generic Interface) are created and maintained (see Figure below).

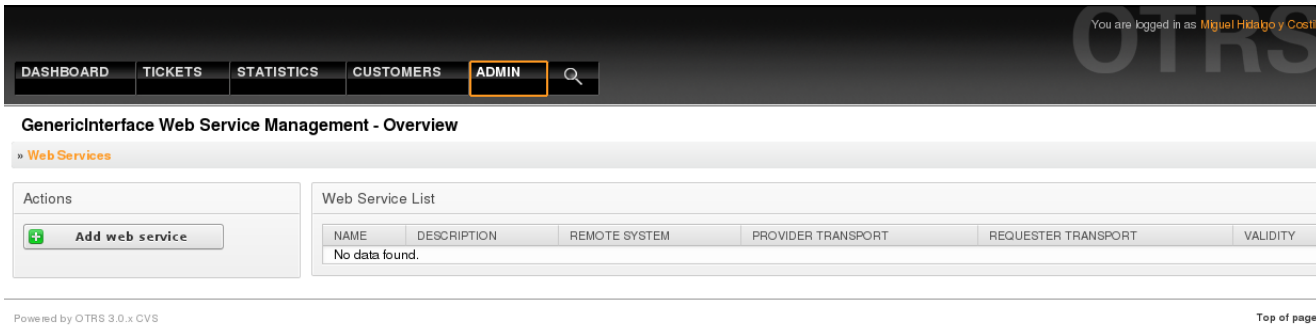

Figure: The graphical interface for web services.

The graphical interface for web services configuration is described in more detail in the section "Web Service Graphical Interface".

Dynamic Fields

Dynamic Fields is the place where you setup an manage custom fields for tickets and articles (see figure below).

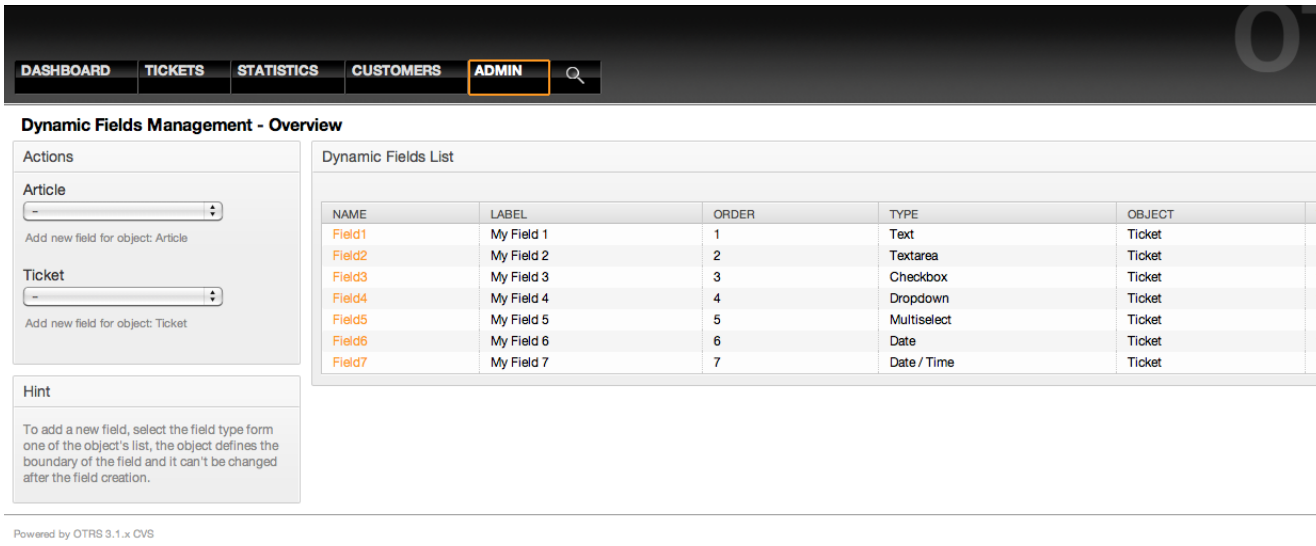

Figure: The dynamic fields overview screen with some dynamic fields.

The dynamic fields configuration is described in more detail in the section "Dynamic Fields Configuration".

Each dynamic field type has its own configuration settings and therefore its own configuration screen.

Note

In the OTRS framework, dynamic fields can only be linked to tickets and articles by default, but they can be extended to other objects as well.

Chapter 6. System Configuration

OTRS config files

All OTRS configuration files are stored in the directory `Kernel` and in its subdirectories. There is no need to manually change any other file than `Kernel/Config.pm`, because the rest of the files will be changed when the system gets upgraded. Just copy the configuration parameters from the other files into `Kernel/Config.pm` and change them as per your needs. This file will never be touched during the upgrade process, so your manual settings are safe.

The file `Kernel/Config/Defaults.pm` contains the parameters of the central OTRS framework. It defines all basic system settings such as the mail configuration, database connection, default charset and standard language. The file `Kernel/Config/Files/Ticket.pm` contains all configuration parameters for the trouble ticket system.

In the directory `Kernel/Config/Files` there are some other files that are parsed when the OTRS login page is accessed. If additional applications like the FAQ or the File Manager are installed, the configuration files for those can also be found in the mentioned path.

If the OTRS web interface is accessed, all .xml files in the `Kernel/Config/Files` directory are parsed in alphabetical order, and the settings for the central framework and additional applications will be loaded. Afterwards, the settings in the two files `Kernel/Config/Files/ZZZAAuto.pm` and `Kernel/Config/Files/ZZZAuto.pm` will be evaluated. Both files are used by the graphical interface for system configuration and should never be changed manually. Lastly, the file `Kernel/Config.pm` that contains your individual settings and manually changed configuration parameters, will be parsed. Reading the configuration files in this order makes sure that your specific configuration settings are used by the system.

Configuring the system through the web interface

Since OTRS 2.0, nearly all configuration parameters of the central framework or additional installed applications, can be changed easily with the graphical interface for system configuration. Log in as OTRS administrator and follow the SysConfig link on the Admin page to execute the new configuration tool (see Figure below).

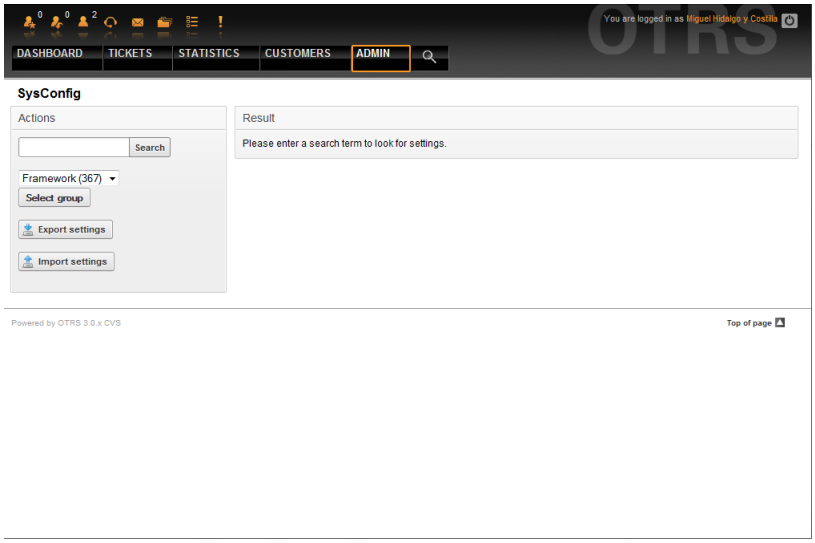

Figure: The graphical interface for system configuration.

OTRS currently has over 600 configuration parameters, and there are different ways to quickly access a specific one. With the full text search, all configuration parameters can be scanned for one or more keywords. The full text search not only searches through the names of the configuration parameters, but also through the descriptions of the parameters. This allows an element to be found easily even if its name is unknown.

Furthermore, all configuration parameters are sorted in main groups and sub groups. The main group represents the application that the configuration parameter belongs to, e.g. "Framework" for the central OTRS framework, "Ticket" for the ticket system, "FAQ" for the FAQ system, and so on. The sub groups can be accessed if the application is selected from the groups listbox and the "Select group" button is pressed.

Every configuration parameter can be turned on or off via a checkbox. If the parameter is turned off, the system will ignore this parameter or use a default. It is possible to switch a changed configuration parameter back to the system default using the Reset link. The Update button submits all changes to system configuration parameters.

If you want to save all the changes you made to your system's configuration, for example to setup a new installation quickly, you can use the "Export settings" button, which will create a .pm file. To restore your own settings, just press the "Import settings" and select the .pm created before.

Note

For security reasons, the configuration parameters for the database connection cannot be changed in the SysConfig section. They have to be set manually in `Kernel/Config.pm`.

Chapter 7. Sending/Receiving emails

Sending emails

Via Sendmail (default)

OTRS can send out emails via Sendmail [http://www.sendmail.org/], Postfix [http://www.postfix.org/], Qmail [http://www.qmail.org] or Exim [http://www.exim.org]). The default configuration is to use Sendmail and should work out-of-the-box.

You can configure the sendmail settings via the graphical configuration frontend (Framework::Core::Sendmail)

Via SMTP server or smarthost

OTRS can send emails via SMTP (Simple Mail Transfer Protocol / RFC 821 [http://www.ietf.org/rfc/rfc821.txt]) or Secure SMTP. You will want to use this on non-UNIX platforms (e.g. Windows).

The SMTP server settings can be configured via the SysConfig (Framework::Core::Sendmail). If you don't see SMTPS available as an option, the required Perl modules are missing. In that case, please refer to "Installation of Perl modules required for OTRS" for instructions.

Receiving emails

Mail accounts configured via the OTRS GUI

OTRS is able to receive emails from POP3, POP3S, IMAP, and IMAPS mail accounts.

Configure your mail accounts via the PostMaster Mail Accounts link on the Admin page.

If a new mail account is to be created (see Figure below), then it's mail server name, login name and password must be specified. Also, you need to select the mail server type, which can be POP3, POP3S, IMAP or IMAPS. If you don't see your server type available as an option, the required Perl modules are missing on your system. In that case, please refer to "Installation of Perl modules required for OTRS" for instructions.

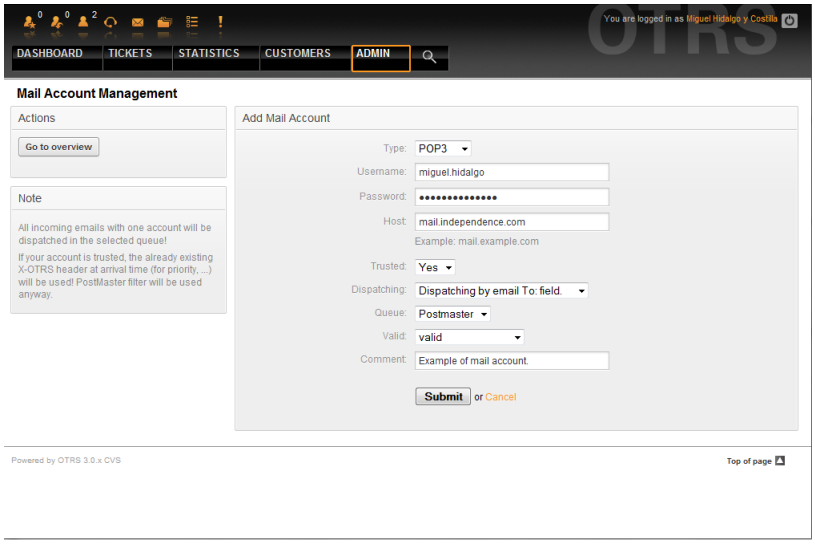

Figure: Adding a mail account.

If you select Yes for the value of the Trusted option, 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 the Trusted option to Yes only for known senders. X-OTRS-Headers are used by the filter module in OTRS. The X-OTRS headers are explained in this table in more detail. Any postmaster filter rules created are executed, irrespective of the Trusted option's setting.

The distribution of incoming messages can be controlled if they need to be sorted by queue or by the content of the "To:" field. For the Dispatching field, if "Dispatching by selected queue" is selected, all incoming messages will be sorted into the specified queue. The address where the mail was sent to is disregarded in this case. If "Dispatching by email To: field" is selected, 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 E-mail address management section of the Admin page. 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 PostmasterDefaultQueue after a default installation.

All data for the mail accounts are saved in the OTRS database. The `otrs.PostMasterMailbox.pl` script, which is located in the `bin` directory of your OTRS installation, uses the settings in the database and fetches the mail. You can execute `./bin/otrs.PostMasterMailbox.pl` manually to check if all your mail settings are working properly.

On a normal installation, the mail will be fetched every 10 minutes by the `postmaster_mailbox` cron job. For further information about modifying cron jobs, please refer to the "Setting up the cron jobs for OTRS" section.

Note

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.

Via command line program and procmail (otrs.PostMaster.pl)

If you cannot use mail accounts to get the email into OTRS, the command line program `bin/otrs.PostMaster.pl` might be a way around the problem. It takes the mails via STDIN and pipes them directly into OTRS. That means email will be available in your OTRS system if the MDA (mail delivery agent, e.g. procmail) executes this program.

To test `bin/otrs.PostMaster.pl` without an MDA, execute the command of the following script.

```
linux:/opt/otrs# cd bin
linux:/opt/otrs/bin# cat ../doc/sample_mails/test-email-1.box | ./
otrs.PostMaster.pl
linux:/opt/otrs/bin#
```

Script: Testing PostMaster without the MDA.

If the email is shown in the QueueView, then your setup is working.

Procmail is a very common e-mail filter in Linux environments. It is installed on most systems. If not, have a look at the *procmail homepage* [http://www.procmail.org/].

To configure procmail for OTRS (based upon a procmail configured MTA such as sendmail, postfix, exim or qmail), use the `~otrs/.procmailrc.dist` file and copy it to `.procmailrc` and add the lines of the script below.

```
SYS_HOME=$HOME
PATH=/bin:/usr/bin:/usr/local/bin
# --
# Pipe all email into the PostMaster process.
# --
:0 :
| $SYS_HOME/bin/otrs.PostMaster.pl
```

Script: Configuring procmail for OTRS.

All email sent to the local OTRS user will be piped into `bin/otrs.PostMaster.pl` and then shown in your QueueView.

Fetching emails via POP3 or IMAP and fetchmail for otrs.PostMaster.pl

In order to get email from your mail server, via a POP3 or IMAP mailbox, to the OTRS machine/local OTRS account and to procmail, use fetchmail [http://fetchmail.berlios.de/].

Note

A working SMTP configuration on the OTRS machine is required.

You can use the `.fetchmailrc.dist` in the home directory of OTRS and copy it to `.fetchmailrc`. Modfiy/change it for your needs (see the Example 7-1 below).

Example 7.1. .fetchmailrc

```
#poll (mailserver) protocol POP3 user (user) password (password) is
 (localuser)
poll mail.example.com protocol POP3 user joe password mama is otrs
```

Don't forget to set the .fetchmailrc to 710 ("chmod 710 .fetchmailrc")!

With the `.fetchmailrc` from the Example 7-1 above, all email will be forwarded to the local OTRS account, if the command **fetchmail -a** is executed. Set up a cronjob with this command if you want to fetch the mails regularly.

Filtering/dispatching by OTRS/PostMaster modules (for more complex dispatching)

If you use the bin/otrs.PostMaster.pl or bin/otrs.PostMasterMailbox.pl method, you can insert or modify X-OTRS header entries with the PostMaster filter modules. With the X-OTRS headers, the ticket system can execute some actions on incoming mails, sort them into a specific queue, change the priority or change the customer ID, for example. More information about the X-OTRS headers are available in the section about adding mail accounts from the OTRS Admin page.

There are some default filter modules:

Note

The job name (e.g. $Self->{'PostMaster::PreFilterModule'}->{'JobName'}) needs to be unique!

Kernel::System::PostMaster::Filter::Match is a default module to match on some email header (e.g. From, To, Subject, ...). It can set new email headers (e.g. X-OTRS-Ignore: yes or X-OTRS-Queue: spam) if a filter rule matches. The jobs of the Example 7-2 can be inserted in `Kernel/Config.pm`

Example 7.2. Example jobs for the filter module Kernel::System::PostMaster::Filter::Match

```
# Job Name: 1-Match
# (block/ignore all spam email with From: noreply@)
$Self->{'PostMaster::PreFilterModule'}->{'1-Match'} = {
    Module => 'Kernel::System::PostMaster::Filter::Match',
    Match => {
        From => 'noreply@',
    },
    Set => {
        'X-OTRS-Ignore' => 'yes',
    },
};
# Job Name: 2-Match
# (sort emails with From: sales@example.com and Subject: **ORDER**
# into queue 'Order')
$Self->{'PostMaster::PreFilterModule'}->{'2-Match'} = {
```

```
        Module => 'Kernel::System::PostMaster::Filter::Match',
        Match => {
            To => 'sales@example.com',
            Subject => '**ORDER**',
        },
        Set => {
            'X-OTRS-Queue' => 'Order',
        },
    };
```

Kernel::System::PostMaster::Filter::CMD is a default module to pipe the email into an external command. The output is given to STDOUT and if the result is true, then set new email header (e.g. X-OTRS-Ignore: yes or X-OTRS-Queue: spam). The Example 7-3 can be used in `Kernel/Config.pm`

Example 7.3. Example job for the filter module Kernel::System::PostMaster::Filter::CMD

```
    # Job Name: 5-SpamAssassin
    # (SpamAssassin example setup, ignore spam emails)
    $Self->{'PostMaster::PreFilterModule'}->{'5-SpamAssassin'} = {
        Module => 'Kernel::System::PostMaster::Filter::CMD',
        CMD => '/usr/bin/spamassassin | grep -i "X-Spam-Status: yes"',
        Set => {
            'X-OTRS-Ignore' => 'yes',
        },
    };
```

Of course it's also possible to develop your own PostMaster filter modules.

Chapter 8. Time related functions

Setting up business hours, holidays and time zones

Some functions in OTRS, like escalations and automatic unlocking of tickets, depend on a proper configuration of business hours, time zones and holidays. You can define these via the SysConfig interface, in Framework > Core::Time. You can also specify different sets of business hours, holidays and time zones as separate 'Calendars' in Framework > Core::Time::Calendar1 through Framework > Core::Time::Calendar9. Calendars can be defined by queue settings, or on SLA levels. This means that, for example, you can specify a calendar with 5 x 8 business hours for your 'standard' SLA, but create a separate calendar with 7 x 24 support for your 'gold' SLA; as well as set a calendar for your 'Support-USA' queue with a different time window than your 'Support-Japan' queue. OTRS can handle up to 99 different calendars.

Business Hours

Set up the working hours for your system in SysConfig Framework > Core::Time::TimeWorkingHours, or for your specific calendar in the calendar's configuration. OTRS can handle a granularity of one hour. Checking the marks in the boxes 8, 9, 10 ... 18 corresponds with business hours of 8 AM - 6 PM.

Only during business hours can tickets escalate, notifications for escalated and pending tickets be sent, and tickets be unlocked.

Fixed date holidays

Holidays that are on a fixed date every year, such as New Year's Day or the Fourth of July, can be specified in TimeVacationDays, or in the corresponding section for the calendars 1-9.

Tickets will not escalate nor get unlocked on dates defined in TimeVacationDays.

Note

By default, OTRS ships with the *German* holidays installed.

TimeVacationDaysOneTime

Holidays such as Easter that do not have a yearly fixed date but instead vary each year, can be specified in TimeVacationDaysOneTime.

Tickets will not escalate and will not be unlocked on dates defined in TimeVacationDaysOneTime.

Note

OTRS does not ship with any One-Time holidays pre-installed. This means that you need to add holidays, such as Easter or Thanksgiving, to the system when configuring OTRS.

Automated Unlocking

Locked tickets can be automatically unlocked by the system. This feature might be useful if, for example, an agent has locked tickets that need to be processed, but he can't work on them for some reason, say because he is out of the office on an emergency. The automated unlock feature unlocks tickets after a given time to ensure that no locked tickets will be forgotten, thereby allowing other agents to process them.

The amount of time before a ticket is unlocked can be specified in the queue settings for every queue. The module `bin/otrs.UnlockTickets.pl`, which is executed periodically as a cron job, performs the automated unlocking of tickets.

Notifications on unlocked tickets are sent out only to those agents that have the queue with the unlocked tickets set in "My queues", and that have activated the notification on unlocked tickets in their personal preferences.

Tickets will be unlocked if all of the following conditions are met:

- There is an *unlock timeout* defined for the queue the ticket is in.
- The ticket is set to *locked*.
- The ticket state is *open*.

The unlock timer will be reset if an agent adds a new external article to the ticket. It can be of any of the following types: *email-external*, *phone*, *fax*, *sms*, or *note-external*.

Also, if the last article in the ticket is created by an agent, and a customer adds another one, either via web or email response, the unlock timer will be reset.

The last event that will reset the unlock timer is when the ticket is assigned to another agent.

Chapter 9. Ticket Responsibility & Ticket Watching

From OTRS 2.1 on, it is possible to assign a person as being responsible for a ticket, additionally to its owner. Moreover, all activities connected with the ticket can be watched by someone other than the ticket owner. These two functionalities are implemented with the TicketResponsible and TicketWatcher features, and facilitate the assignment of tasks and working within hierarchical team structures.

Ticket Responsibility

The ticket responsiblility feature facilitates the complete processing of a ticket by an agent other than the ticket owner. Thus an agent who has locked a ticket can pass it on to another agent, who is not the ticket owner, in order for the second to respond to a customer request. After the request has been dealt with, the first agent can withdraw the ticket responsibility from the second agent.

With the configuration parameter Ticket::Responsible, the ticket responsibility feature can be activated. This will cause 3 new links to appear in the ticket activities menu of a zoomed ticket in the agent interface.

Ticket responsibility can be assigned by calling up the ticket content and clicking on the "Responsible" link in the ticket activities menu of a zoomed ticket in the agent interface (see the Figure below).

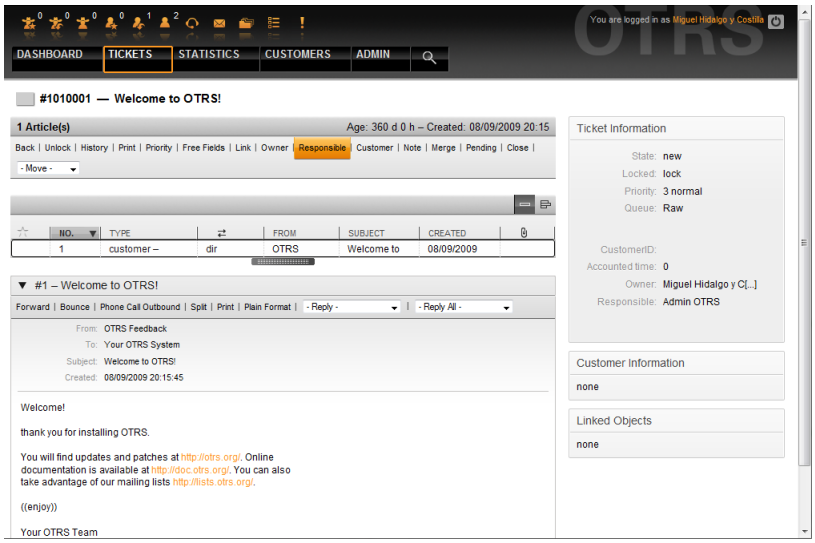

Figure: Changing the Responsibility of a ticket in its zoomed view.

After clicking on "Responsible", a pop-up dialog to change the responsibility of that ticket will open (see Figure below). This dialog can also be used to send a message to the new responsible agent.

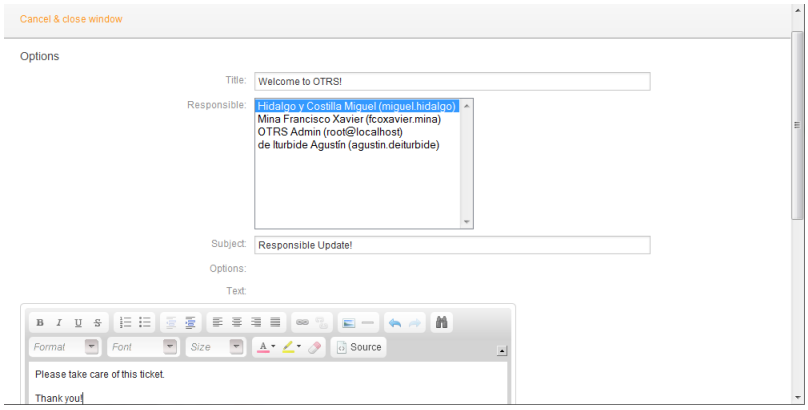

Figure: Pop-up dialog to change a ticket's responsibility.

The list of all tickets for which an agent is responsible, can be accessed through the Responsible view of the OTRS agent interface, as soon as the ticket responsibility feature gets activated.

Ticket watching

From OTRS 2.1 on, select agents such as supervisors can watch certain tickets within the system without processing them, by using the TicketWatcher feature.

The TicketWatcher feature can be activated with the configuration parameter Ticket::Watcher which adds new links to your actions toolbar. Using Ticket::WatcherGroup, one or more user groups with permission to watch tickets can also be defined.

In order to watch a ticket, go to its zoomed view and click on the "Subscribe" link in the ticket activities menu (see Figure below).

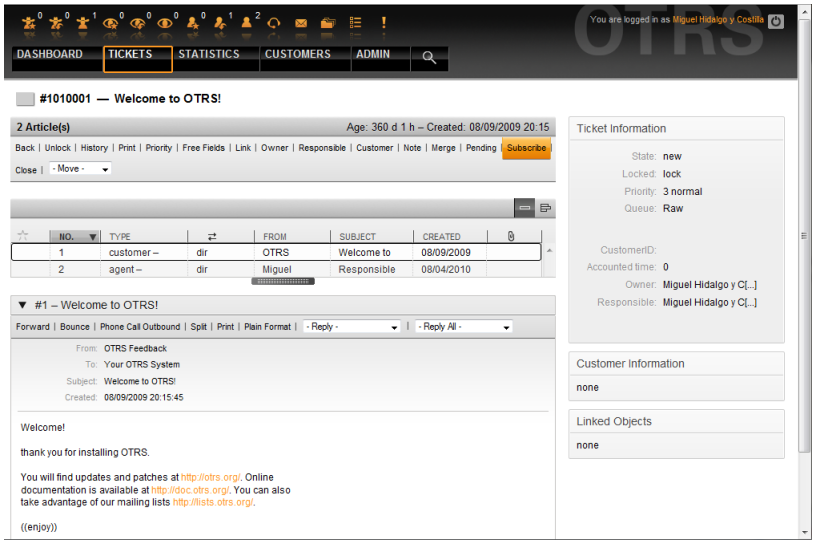

Figure: Subscribing to watching a ticket in its zoomed view.

If you no longer want to watch a specific ticket, go to its zoomed view and click on the "Unsubscribe" link in the ticket activities menu (see Figure below).

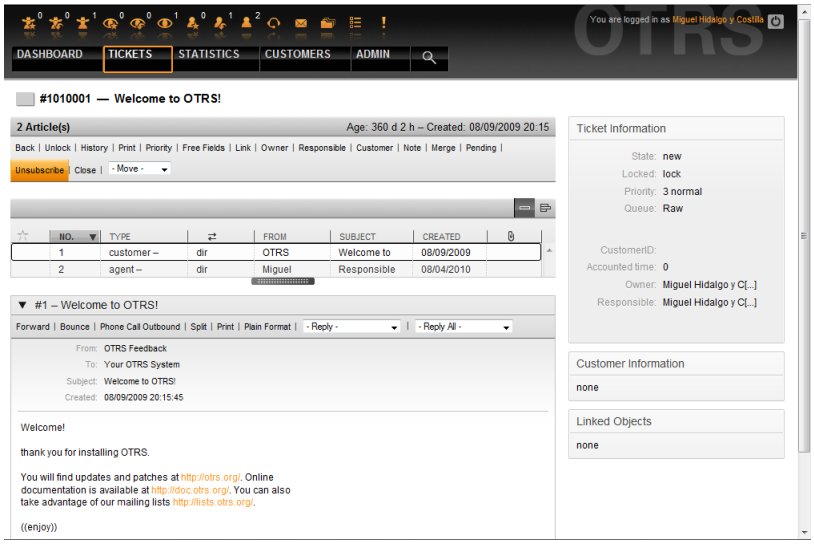

Figure: Unsubscribing from watching a ticket in its zoomed view.

The list of all watched tickets can be accessed through the Watched view of the OTRS agent interface (see Figure below), as soon as the ticket watcher feature gets activated.

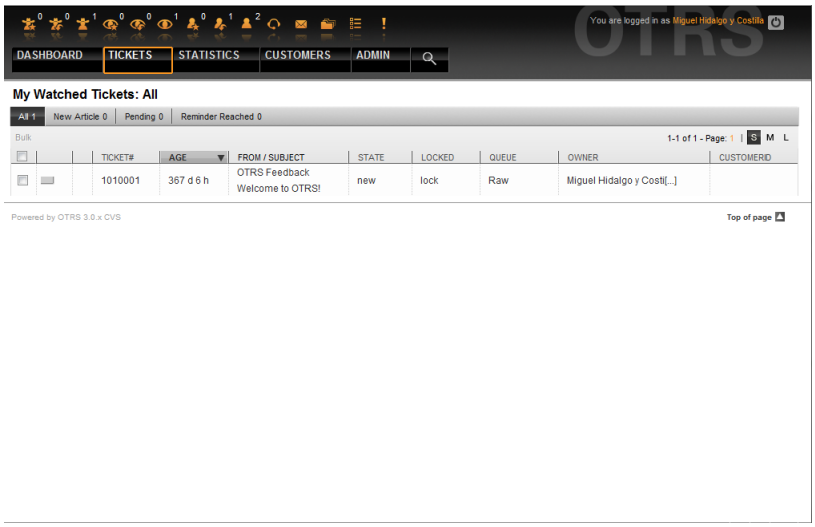

Figure: Watched tickets view.

Chapter 10. Customizing the PDF output

This section handles the configurable options for PDF output in OTRS.

If you use the Print action from anywhere within the OTRS interface, this generates a formatted PDF file. You can deactivate this by modifying the configuration parameter PDF to create HTML output instead.

You can adjust the look of the files generated by OTRS by creating your own logo and adding it to PDF::LogoFile. You can use PDF::PageSize to define the standard page size of the generated pdf file (DIN-A4 or Letter), and also PDF::MaxPage to specify the maximum number of pages for a pdf file, which is useful if a user generates a huge output file by mistake.

The Perl CPAN modules PDF::API2 and Compress::Zlib must be installed for the generation of pdf files. In many distributions they are available as packages and can be easily installed, using the respective package manager. In case this is not possible, they have to be installed with CPAN. For further information about installing Perl modules, please refer to the "Installation of Perl modules" section.

Chapter 11. Using external backends

Customer data

OTRS works with many customer data attributes such as username, email address, phone number, etc. These attributes are displayed in both the Agent and the Customer frontends, and also used for the authentication of customers.

Customer data used or displayed within OTRS is highly customizable. The following information is however always needed for customer authentication:

- User login
- Email address
- Customer ID

Use configuration parameters of the following script in your `Kernel/Config.pm` file, if you want to display customer information in your agent interface.

```
# Ticket::Frontend::CustomerInfo*
# (show customer info on Compose (Phone and Email), Zoom and
# Queue view)
$Self->{'Ticket::Frontend::CustomerInfoCompose'} = 1;
$Self->{'Ticket::Frontend::CustomerInfoZoom'} = 1;
$Self->{'Ticket::Frontend::CustomerInfoQueue'} = 0;
```

Script: Kernel/Config.pm configuration parameters.

Customer user backend

You can use two types of customer backends, DB and LDAP. If you already have another customer backend (e.g. SAP), it is of course possible to write a module that uses it.

Database (Default)

Example 11-1 shows the configuration of a DB customer backend, which uses customer data stored in the OTRS database.

Example 11.1. Configuring a DB customer backend

```
# CustomerUser (customer database backend and settings)
$Self->{CustomerUser} = {
    Name => 'Database Datasource',
    Module => 'Kernel::System::CustomerUser::DB',
    Params => {
        # if you want to use an external database, add the required
 settings
#            DSN => 'DBI:odbc:yourdsn',
#            DSN =>
 'DBI:mysql:database=customerdb;host=customerdbhost',
```

```
#            User => '',
#            Password => '',
            Table => 'customer_user',
            # if your frontend is unicode and the charset of your
            # customer database server is iso-8859-1, use these
 options.
#           SourceCharset => 'iso-8859-1',
#           DestCharset => 'utf-8',

            # CaseSensitive will control if the SQL statements need
 LOWER()
            #   function calls to work case insensitively. Setting
 this to
            #   1 will improve performance dramatically on large
 databases.
            CaseSensitive => 0,
        },
# customer unique id
CustomerKey => 'login',

# customer #
CustomerID => 'customer_id',
CustomerValid => 'valid_id',
    CustomerUserListFields => ['first_name', 'last_name', 'email'],
    CustomerUserSearchFields => ['login', 'last_name', 'customer_id'],
    CustomerUserSearchPrefix => '',
    CustomerUserSearchSuffix => '*',
    CustomerUserSearchListLimit => 250,
    CustomerUserPostMasterSearchFields => ['email'],
    CustomerUserNameFields => ['title','first_name','last_name'],
    CustomerUserEmailUniqCheck => 1,
#    # show not own tickets in customer panel, CompanyTickets
#    CustomerUserExcludePrimaryCustomerID => 0,
#    # generate auto logins
#    AutoLoginCreation => 0,
#    AutoLoginCreationPrefix => 'auto',
#    # admin can change customer preferences
#    AdminSetPreferences => 1,
#    # cache time to live in sec. - cache any database queries
#    CacheTTL => 0,
#    # just a read only source
#    ReadOnly => 1,
    Map => [
        # note: Login, Email and CustomerID needed!
        # var, frontend, storage, shown (1=always,2=lite), required,
 storage-type, http-link, readonly, http-link-target
        [ 'UserTitle',      'Title',      'title',      1, 0, 'var',
 '', 0 ],
        [ 'UserFirstname',  'Firstname',  'first_name', 1, 1, 'var',
 '', 0 ],
        [ 'UserLastname',   'Lastname',   'last_name',  1, 1, 'var',
 '', 0 ],
        [ 'UserLogin',      'Username',   'login',      1, 1, 'var',
 '', 0 ],
```

```
        [ 'UserPassword',   'Password',   'pw',         0, 0, 'var',
 '', 0 ],
        [ 'UserEmail',      'Email',      'email',      1, 1, 'var',
 '', 0 ],

#        [ 'UserEmail',      'Email', 'email',           1, 1, 'var',
 '$Env{"CGIHandle"}?Action=AgentTicketCompose&ResponseID=1&TicketID=
$Data{"TicketID"}&ArticleID=$Data{"ArticleID"}', 0 ],
        [ 'UserCustomerID', 'CustomerID', 'customer_id', 0, 1, 'var',
 '', 0 ],

#        [ 'UserCustomerIDs', 'CustomerIDs', 'customer_ids', 1, 0,
 'var', '', 0 ],
        [ 'UserPhone',        'Phone',       'phone',        1, 0,
 'var', '', 0 ],
        [ 'UserFax',          'Fax',         'fax',          1, 0,
 'var', '', 0 ],
        [ 'UserMobile',       'Mobile',      'mobile',       1, 0,
 'var', '', 0 ],
        [ 'UserStreet',       'Street',      'street',       1, 0,
 'var', '', 0 ],
        [ 'UserZip',          'Zip',         'zip',          1, 0,
 'var', '', 0 ],
        [ 'UserCity',         'City',        'city',         1, 0,
 'var', '', 0 ],
        [ 'UserCountry',      'Country',     'country',      1, 0,
 'var', '', 0 ],
        [ 'UserComment',      'Comment',     'comments',     1, 0,
 'var', '', 0 ],
        [ 'ValidID',          'Valid',       'valid_id',     0, 1,
 'int', '', 0 ],
    ],
    # default selections
    Selections => {
        UserTitle => {
            'Mr.' => 'Mr.',
            'Mrs.' => 'Mrs.',
        },
    },
};
```

If you want to customize the customer data, change the column headers or add new ones to the customer_user table in the OTRS database. As an example, the script below shows how to add a new field for room number.

```
linux:~# mysql -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 116 to server version: 5.0.18-Debian_7-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use otrs;
```

```
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> ALTER TABLE customer_user ADD room VARCHAR (250);
Query OK, 1 rows affected (0.01 sec)
Records: 1  Duplicates: 0  Warnings: 0

mysql> quit
Bye
linux:~#
```

Script: Adding a room field to the customer_user table.

Now add the new column to the MAP array in `Kernel/Config.pm`, as shown in the following script.

```
    # var, frontend, storage, shown (1=always,2=lite), required,
 storage-type, http-link, readonly
    [...]
    [ 'UserRoom',      'Room',      'room',       0, 1, 'var', '',
 0 ],
```

Script: Adding a room field to the Kernel/Config.pm file.

It is also possible to edit all this customer information via the Customers link in the Agent interface.

Customer with multiple IDs (Company tickets)

It is possible to assign more than one customer ID to a customer. This can be useful if a customer must access tickets of other customers, e.g. a supervisor wants to watch the tickets of his assistants. If a customer can access the tickets of another customer, the company ticket feature of OTRS is used. Company tickets can be accessed via the "Company Tickets" link in the customer panel.

To use company tickets, a new column with the IDs that should be accessible for a customer, has to be added to the customer_user table in the OTRS database (see Script below).

```
linux:~# mysql -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 124 to server version: 5.0.18-Debian_7-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use otrs;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> ALTER TABLE customer_user ADD customer_ids VARCHAR (250);
Query OK, 1 rows affected (0.02 sec)
Records: 1  Duplicates: 0  Warnings: 0
```

```
mysql> quit
Bye
linux:~#
```

Script: Adding a customer_ids field to the customer_user table.

Now the new column has to be added to the MAP array in `Kernel/Config.pm`, as shown in the script below.

```
    # var, frontend, storage, shown (1=always,2=lite), required,
 storage-type, http-link, readonly
    [...]
    [ 'UserCustomerIDs', 'CustomerIDs', 'customer_ids', 1, 0, 'var',
 '', 0 ],
```

Script: Adding a UserCustomerIDs field to the Kernel/Config.pm file.

Now, the new column for the multiple customer IDs can be edited via the Agent interface, in the section for the customer management.

To ensure that one customer can access the tickets of other customers, add the IDs of these other users into the new field for the multiple customer IDs. Each ID has to be separated by a semicolon (see Example 11-2 below).

Example 11.2. Using company tickets with a DB backend

The customers A, B and C exist in your system, and A wants to have access to the tickets of B and C via the customer panel. B and C should have no access to the tickets of other users.

To realize this setup, change the customer_user table and the mapping in `Kernel/Config.pm` as described above. Then load the settings for customer A via the Customers link in the Agent interface or via the Admin page. If the settings are displayed, add into the field for CustomerIDs the values "B;C;".

LDAP

If you have a LDAP directory with your customer data, you can use it as the customer backend with OTRS, as shown in Example 11-3.

Example 11.3. Configuring an LDAP customer backend

```
# CustomerUser
# (customer ldap backend and settings)
$Self->{CustomerUser} = {
    Name => 'LDAP Data Source',
    Module => 'Kernel::System::CustomerUser::LDAP',
    Params => {
        # 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 both your frontend and your LDAP are unicode, use
 this:
            SourceCharset => 'utf-8',
            DestCharset   => 'utf-8',
            # if your frontend is unicode and the charset of your
            # ldap server is iso-8859-1, use these options.
            # SourceCharset => 'iso-8859-1',
            # DestCharset => 'utf-8',
            # 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'],
    # show not own tickets in customer panel, CompanyTickets
    CustomerUserExcludePrimaryCustomerID => 0,
    # add an ldap filter for valid users (expert setting)
#    CustomerUserValidFilter => '(!(description=locked))',
    # administrator can't change customer preferences
    AdminSetPreferences => 0,
#    # cache time to live in sec. - cache any database queries
#    CacheTTL => 0,
    Map => [
        # note: Login, Email and CustomerID are mandatory!
        # var, frontend, storage, shown (1=always,2=lite), required,
 storage-type, http-link, readonly
        [ 'UserTitle',      'Title',      'title',           1, 0,
 'var', '', 0 ],
```

```
        [ 'UserFirstname',  'Firstname',  'givenname',       1, 1,
 'var', '', 0 ],
        [ 'UserLastname',   'Lastname',   'sn',              1, 1,
 'var', '', 0 ],
        [ 'UserLogin',      'Username',   'uid',             1, 1,
 'var', '', 0 ],
        [ 'UserEmail',      'Email',      'mail',            1, 1,
 'var', '', 0 ],
        [ 'UserCustomerID', 'CustomerID', 'mail',            0, 1,
 'var', '', 0 ],
#        [ 'UserCustomerIDs', 'CustomerIDs', 'second_customer_ids', 1,
 0, 'var', '', 0 ],
        [ 'UserPhone',      'Phone',      'telephonenumber', 1, 0,
 'var', '', 0 ],
        [ 'UserAddress',    'Address',    'postaladdress',   1, 0,
 'var', '', 0 ],
        [ 'UserComment',    'Comment',    'description',     1, 0,
 'var', '', 0 ],
    ],
};
```

If additional customer attributes are stored in your LDAP directory, such as a manager's name, a mobile phone number, or a department, and if you want to display this information in OTRS, just expand the MAP array in `Kernel/Config.pm` with the entries for these attributes, as shown in the following script.

```
    # var, frontend, storage, shown (1=always,2=lite), required,
 storage-type, http-link, readonly
    [...]
    [ 'UserPhone',      'Phone',      'telephonenumber', 1, 0, 'var',
 '', 0 ],
```

Script: Adding new fields to the Kernel/Config.pm file.

Customer with multiple IDs (Company tickets)

It is possible to assign more than one Customer ID to a customer, when using an LDAP backend. To use company tickets, a new field has to be added to the LDAP directory that contains the IDs accessible by the customer.

If the new field in the LDAP directory has been created, the new entry has to be added to the MAP array in `Kernel/Config.pm`, as shown in the script below.

```
    # var, frontend, storage, shown (1=always,2=lite), required,
 storage-type, http-link, readonly
    [...]
    [ 'UserCustomerIDs', 'CustomerIDs', 'customer_ids', 1, 0, 'var',
 '', 0 ],
```

Script: Maping new fields to the Kernel/Config.pm file.

The field for the multiple customer IDs has to be edited directly in the LDAP directory. OTRS can only read from LDAP, not write to it.

To ensure access by a customer to the tickets of other customers, add the customer IDs of the customers whose tickets should be accessed to the new field in your LDAP directory. Each ID has to be separated by a semicolon (see Example 11-4 below).

Example 11.4. Using Company tickets with an LDAP backend

The customers A, B and C exist in your system and A wants to have access to the tickets of B and C via the customer panel. B and C should have no access to tickets of other users.

To realize this setup, change the LDAP directory and the mapping in `Kernel/Config.pm` as described above. Then add into the field for CustomerIDs the values "B;C;" for customer A in your LDAP directory.

Use more than one customer backend with OTRS

If you want to utilize more than one customer data source used with OTRS (e.g. an LDAP and a database backend), the CustomerUser config parameter should be expanded with a number, e.g. "CustomerUser1", "CustomerUser2" (see Example 11-5 below).

Example 11.5. Using more than one customer backend with OTRS

The following configuration example shows usage of both an LDAP and a database customer backend with OTRS.

```
# 1. Customer user backend: DB
# (customer database backend and settings)
$Self->{CustomerUser1} = {
    Name => 'Customer Database',
    Module => 'Kernel::System::CustomerUser::DB',
    Params => {
        # if you want to use an external database, add the
        # required settings
#        DSN => 'DBI:odbc:yourdsn',
#        DSN => 'DBI:mysql:database=customerdb;host=customerdbhost',
#        User => '',
#        Password => '',
        Table => 'customer_user',
    },
    # customer unique id
    CustomerKey = 'login',
    # customer #
    CustomerID = 'customer_id',
    CustomerValid = 'valid_id',
    CustomerUserListFields => ['first_name', 'last_name', 'email'],
    CustomerUserSearchFields => ['login', 'last_name', 'customer_id'],
    CustomerUserSearchPrefix => '',
    CustomerUserSearchSuffix => '*',
    CustomerUserSearchListLimit => 250,
    CustomerUserPostMasterSearchFields => ['email'],
    CustomerUserNameFields => ['title','first_name','last_name'],
    CustomerUserEmailUniqCheck => 1,
#    # show not own tickets in customer panel, CompanyTickets
#    CustomerUserExcludePrimaryCustomerID => 0,
```

```
#    # generate auto logins
#    AutoLoginCreation => 0,
#    AutoLoginCreationPrefix => 'auto',
#    # admin can change customer preferences
#    AdminSetPreferences => 1,
#    # cache time to live in sec. - cache any database queries
#    CacheTTL => 0,
#    # just a read only source
#    ReadOnly => 1,
    Map => [

        # note: Login, Email and CustomerID needed!
        # var, frontend, storage, shown (1=always,2=lite), required,
 storage-type, http-link, readonly, http-link-target
        [ 'UserTitle',      'Title',      'title',        1, 0, 'var',
 '', 0 ],
        [ 'UserFirstname',  'Firstname',  'first_name',   1, 1, 'var',
 '', 0 ],
        [ 'UserLastname',   'Lastname',   'last_name',    1, 1, 'var',
 '', 0 ],
        [ 'UserLogin',      'Username',   'login',        1, 1, 'var',
 '', 0 ],
        [ 'UserPassword',   'Password',   'pw',           0, 0, 'var',
 '', 0 ],
        [ 'UserEmail',      'Email',      'email',        1, 1, 'var',
 '', 0 ],
        [ 'UserCustomerID', 'CustomerID', 'customer_id',  0, 1, 'var',
 '', 0 ],
        [ 'UserPhone',      'Phone',      'phone',        1, 0, 'var',
 '', 0 ],
        [ 'UserFax',        'Fax',        'fax',          1, 0, 'var',
 '', 0 ],
        [ 'UserMobile',     'Mobile',     'mobile',       1, 0, 'var',
 '', 0 ],
        [ 'UserStreet',     'Street',     'street',       1, 0, 'var',
 '', 0 ],
        [ 'UserZip',        'Zip',        'zip',          1, 0, 'var',
 '', 0 ],
        [ 'UserCity',       'City',       'city',         1, 0, 'var',
 '', 0 ],
        [ 'UserCountry',    'Country',    'country',      1, 0, 'var',
 '', 0 ],
        [ 'UserComment',    'Comment',    'comments',     1, 0, 'var',
 '', 0 ],
        [ 'ValidID',        'Valid',      'valid_id',     0, 1, 'int',
 '', 0 ],
    ],
    # default selections
    Selections => {
        UserTitle => {
            'Mr.' => 'Mr.',
            'Mrs.' => 'Mrs.',
        },
    },
```

```
};

# 2. Customer user backend: LDAP
# (customer ldap backend and settings)
$Self->{CustomerUser2} = {
    Name => 'LDAP Datasource',
    Module => 'Kernel::System::CustomerUser::LDAP',
    Params => {
        # 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 both your frontend and your LDAP are unicode, use this:
#        SourceCharset => 'utf-8',
#        DestCharset   => 'utf-8',
        # if your frontend is e. g. iso-8859-1 and the character set
 of your
        # ldap server is utf-8, use these options:
#        SourceCharset => 'utf-8',
#        DestCharset => 'iso-8859-1',

        # 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'],
    # show not own tickets in customer panel, CompanyTickets
```

```
    CustomerUserExcludePrimaryCustomerID => 0,
    # add a ldap filter for valid users (expert setting)
#    CustomerUserValidFilter => '(!(description=locked))',
    # admin can't change customer preferences
    AdminSetPreferences => 0,
    Map => [
        # note: Login, Email and CustomerID needed!
        # var, frontend, storage, shown (1=always,2=lite), required,
 storage-type, http-link, readonly
        [ 'UserTitle',      'Title',      'title',           1, 0,
 'var', '', 0 ],
        [ 'UserFirstname',  'Firstname',  'givenname',       1, 1,
 'var', '', 0 ],
        [ 'UserLastname',   'Lastname',   'sn',              1, 1,
 'var', '', 0 ],
        [ 'UserLogin',      'Username',   'uid',             1, 1,
 'var', '', 0 ],
        [ 'UserEmail',      'Email',      'mail',            1, 1,
 'var', '', 0 ],
        [ 'UserCustomerID', 'CustomerID', 'mail',            0, 1,
 'var', '', 0 ],
#        [ 'UserCustomerIDs', 'CustomerIDs', 'second_customer_ids', 1,
 0, 'var', '', 0 ],
        [ 'UserPhone',      'Phone',      'telephonenumber', 1, 0,
 'var', '', 0 ],
        [ 'UserAddress',    'Address',    'postaladdress',   1, 0,
 'var', '', 0 ],
        [ 'UserComment',    'Comment',    'description',     1, 0,
 'var', '', 0 ],
    ],
};
```

It is possible to integrate up to 10 different customer backends. Use the customer management interface in OTRS to view or edit (assuming write access is enabled) all customer data.

Backends to authenticate Agents and Customers

OTRS offers the option to authenticate agents and customers against different backends.

Authentication backends for Agents

DB (Default)

The backend to authenticate agents which is used by default is the OTRS database. Agents can be added and edited via the agent management interface in the Admin page (see Example 11-6 below).

Example 11.6. Authenticate agents against a DB backend

```
    $Self->{'AuthModule'} = 'Kernel::System::Auth::DB';
```

LDAP

If an LDAP directory has all your agent data stored, you can use the LDAP module to authenticate your users in OTRS (see Example 11-7 below). This module has only read access to the LDAP tree, which means that you cannot edit your user data via the agent management interface.

Example 11.7. Authenticate agents against an LDAP backend

```
# This is an example configuration for an LDAP auth. backend.
# (Make sure 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';

# 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)'
$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';

# 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,
};
```

The configuration parameters shown in the script below can be used to synchronize the user data from your LDAP directory into your local OTRS database. This reduces the number of requests to your LDAP server and speeds up the authentication with OTRS. The data synchronization is done when the agent authenticates the first time. Although the data can be syncronized into

the local OTRS database, the LDAP directory is the last instance for the authentication, so an inactive user in the LDAP tree can't authenticate to OTRS, even when the account data is already stored in the OTRS database. The agent data in the LDAP directory can't be edited via the web interface of OTRS, so the data has to be managed directly in the LDAP tree.

```
# 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'} = 'AuthSyncBackend';

# agent data sync against ldap
$Self->{'AuthSyncModule'} = 'Kernel::System::Auth::Sync::LDAP';
$Self->{'AuthSyncModule::LDAP::Host'} = 'ldap://ldap.example.com/';
$Self->{'AuthSyncModule::LDAP::BaseDN'} = 'dc=otrs, dc=org';
$Self->{'AuthSyncModule::LDAP::UID'} = 'uid';
$Self->{'AuthSyncModule::LDAP::SearchUserDN'} = 'uid=sys, ou=user,
 dc=otrs, dc=org';
$Self->{'AuthSyncModule::LDAP::SearchUserPw'} = 'some_pass';
$Self->{'AuthSyncModule::LDAP::UserSyncMap'} = {
    # DB -> LDAP
    UserFirstname => 'givenName',
    UserLastname  => 'sn',
    UserEmail     => 'mail',
};
[...]

# AuthSyncModule::LDAP::UserSyncInitialGroups
# (sync following group with rw permission after initial create of
 first agent
# login)
$Self->{'AuthSyncModule::LDAP::UserSyncInitialGroups'} = [
    'users',
];
```

Script: Synchronizing the user data from the LDAP directory into the OTRS database.

HTTPBasicAuth for Agents

If you want to implement a "single sign on" solution for all your agents, you can use HTTP basic authentication (for all your systems) and the HTTPBasicAuth module for OTRS (see Example 11-8 below).

Example 11.8. Authenticate Agents using HTTPBasic

```
# 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';

# Note:
#
# If you use this module, you should use as fallback
# the following configuration settings if the user is not authorized
```

```
# 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';
```

Radius

The configuration parameters shown in Example 11-9 can be used to authenticate agents against a Radius server.

Example 11.9. Authenticate Agents against a Radius backend

```
# 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';
```

Authentication backends for Customers

Database (Default)

The default user authentication backend for customers in OTRS is the OTRS database. With this backend, all customer data can be edited via the web interface of OTRS (see Example 11-10 below).

Example 11.10. Customer user authentication against a DB backend

```
# This is the auth. module againt the otrs db
$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";
```

LDAP

If you have an LDAP directory with all your customer data, you can use the LDAP module to authenticate your customers to OTRS (see Example 11-11 below). Because this module has only read-access to the LDAP backend, it is not possible to edit the customer data via the OTRS web interface.

Example 11.11. Customer user authentication against an LDAP backend

```
# This is an example configuration for an LDAP auth. backend.
# (make sure 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 does 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,
};
```

HTTPBasicAuth for customers

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). See Example 11-12 below.

Example 11.12. Customer user authentication with HTTPBasic

```
# 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';

# Note:
```

```
# If you use this module, you should use the following
# config settings as fallback, 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';
```

Radius

The settings shown in Example 11-13 can be used to authenticate your customers against a Radius server.

Example 11.13. Customer user authentication against a Radius backend

```
# This is a example configuration to auth. customer against a radius
 server
$Self->{'Customer::AuthModule'} = 'Kernel::System::Auth::Radius';
$Self->{'Customer::AuthModule::Radius::Host'} = 'radiushost';
$Self->{'Customer::AuthModule::Radius::Password'} = 'radiussecret';
```

Customize the customer self-registration

It is possible to customize the self-registration for new customers, accessible via the customer.pl panel. New optional or required fields, like room number, address or state can be added.

The following example shows how you can specify a required field in the customer database, in this case to store the room number of a customer.

Customizing the web interface

To display the new field for the room number in the customer.pl web interface, the .dtl file responsible for the layout in this interface has to be modified. Edit the `Kernel/Output/HTML/Standard/CustomerLogin.dtl` file, adding the new field around line 80 (see Script below).

```
[...]
<div class="NewLine">
    <label for="Room">$Text{"Room{CustomerUser}"}</label>
    <input title="$Text{"Room Number"}" name="Room" type="text"
 id="UserRoom" maxlength="50" />
</div>
[...]
```

Script: Displaying a new field in the web interface.

Customer mapping

In the next step, the customer mapping has to be expanded with the new entry for the room number. To ensure that the changes are not lost after an update, put the "CustomerUser" settings from the `Kernel/Config/Defaults.pm` into the `Kernel/Config.pm`. Now change the MAP array and add the new room number field, as shown in the script below.

```
# CustomerUser
# (customer database backend and settings)
$Self->{CustomerUser} = {
    Name => 'Database Backend',
    Module => 'Kernel::System::CustomerUser::DB',
    Params => {
        # if you want to use an external database, add the
        # required settings
#        DSN => 'DBI:odbc:yourdsn',
#        DSN => 'DBI:mysql:database=customerdb;host=customerdbhost',
#        User => '',
#        Password => '',
        Table => 'customer_user',
    },
    # customer unique id
    CustomerKey => 'login',
    # customer #
    CustomerID => 'customer_id',
    CustomerValid => 'valid_id',
    CustomerUserListFields => ['first_name', 'last_name', 'email'],
#    CustomerUserListFields => ['login', 'first_name', 'last_name',
 'customer_id', 'email'],
    CustomerUserSearchFields => ['login', 'last_name', 'customer_id'],
    CustomerUserSearchPrefix => '',
    CustomerUserSearchSuffix => '*',
    CustomerUserSearchListLimit => 250,
    CustomerUserPostMasterSearchFields => ['email'],
    CustomerUserNameFields => ['title', 'first_name', 'last_name'],
    CustomerUserEmailUniqCheck => 1,
#    # show not own tickets in customer panel, CompanyTickets
#    CustomerUserExcludePrimaryCustomerID => 0,
#    # generate auto logins
#    AutoLoginCreation => 0,
#    AutoLoginCreationPrefix => 'auto',
#    # admin can change customer preferences
#    AdminSetPreferences => 1,
#    # cache time to live in sec. - cache database queries
#    CacheTTL => 0,
#    # just a read only source
#    ReadOnly => 1,
    Map => [

        # note: Login, Email and CustomerID needed!
        # var, frontend, storage, shown (1=always,2=lite), required,
 storage-type, http-link, readonly, http-link-target
        [ 'UserTitle',      'Title',      'title',       1, 0, 'var',
 '', 0 ],
        [ 'UserFirstname',  'Firstname',  'first_name',  1, 1, 'var',
 '', 0 ],
        [ 'UserLastname',   'Lastname',   'last_name',   1, 1, 'var',
 '', 0 ],
        [ 'UserLogin',      'Username',   'login',       1, 1, 'var',
 '', 0 ],
```

```
        [ 'UserPassword',   'Password',   'pw',          0, 0, 'var',
 '', 0 ],
        [ 'UserEmail',      'Email',      'email',       1, 1, 'var',
 '', 0 ],
        [ 'UserCustomerID', 'CustomerID', 'customer_id', 0, 1, 'var',
 '', 0 ],
        [ 'UserPhone',      'Phone',      'phone',       1, 0, 'var',
 '', 0 ],
        [ 'UserFax',        'Fax',        'fax',         1, 0, 'var',
 '', 0 ],
        [ 'UserMobile',     'Mobile',     'mobile',      1, 0, 'var',
 '', 0 ],
        [ 'UserRoom',       'Room',       'room',        1, 0, 'var',
 '', 0 ],
        [ 'UserStreet',     'Street',     'street',      1, 0, 'var',
 '', 0 ],
        [ 'UserZip',        'Zip',        'zip',         1, 0, 'var',
 '', 0 ],
        [ 'UserCity',       'City',       'city',        1, 0, 'var',
 '', 0 ],
        [ 'UserCountry',    'Country',    'country',     1, 0, 'var',
 '', 0 ],
        [ 'UserComment',    'Comment',    'comments',    1, 0, 'var',
 '', 0 ],
        [ 'ValidID',        'Valid',      'valid_id',    0, 1, 'int',
 '', 0 ],
    ],
    # default selections
    Selections => {
        UserTitle => {
            'Mr.' => 'Mr.',
            'Mrs.' => 'Mrs.',
        },
    },
};
```

Script: Changing the map array.

Customize the customer_user table in the OTRS DB

The last step is to add the new room number column to the customer_user table in the OTRS database (see Script below). In this column, the entries for the room numbers will be stored.

```
linux:~# mysql -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6 to server version: 5.0.18-Debian_7-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use otrs;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
```

```
Database changed
mysql> ALTER TABLE customer_user ADD room VARCHAR (200);
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> quit
Bye
linux:~#
```

Script: Adding a new column to the customer_user table.

Now the new field for the room should be displayed in the customer.pl panel. New customers should have to insert their room number if they register a new account. If you use apache and use mod_perl for OTRS, you should restart the web server to activate the changes.

Chapter 12. States

Predefined states

OTRS allows you to change predefined ticket states and their types, or even add new ones. Two attributes are important for a state: the state name and the state type.

The default states of OTRS are: 'closed successful', 'closed unsuccessful', 'merged', 'new', 'open', 'pending auto close+', 'pending auto close-', 'pending reminder' and 'removed'.

New

Tickets are usually in this state when created from incoming e-mails.

Open

This is the default state for tickets assigned to queues and agents.

Pending reminder

After the pending time has expired, the ticket owner will receive a reminder email concerning the ticket. If the ticket is not locked, the reminder will be sent to all agents in the queue. Reminder tickets will only be sent out during business hours, and are repeatedly sent every 24 hours until the ticket state is changed by the agent. Time spent by the ticket in this status will still add towards the escalation time calculation.

Pending auto close-

Tickets in this status will be set to "Closed Unsuccessful" if the pending time has expired. Time spent by the ticket in this status will still add towards the escalation time calculation.

Pending auto close+

Tickets in this status will be set to "Closed Successful" if the pending time has expired. Time spent by the ticket in this status will still add towards the escalation time calculation.

Merged

This is the state for tickets that have been merged with other tickets.

Closed Successful

This is the end state for tickets that have been successfully resolved. Depending on your configuration, you may or may not be able to reopen closed tickets.

Closed Unsuccessful

This is the end state for tickets that have NOT been successfully resolved. Depending on your configuration, you may or may not be able to reopen closed tickets.

Customizing states

Every state has a name (state-name) and a type (state-type). Click on the States link on the Admin page and press the button "Add state" to create a new state. You can freely choose the name of a new state. The state types can not be changed via the web interface. The database has to be directly modified if you want to add new types or change existing names. The default state types should typically not be modified as this can yield unpredictable results. For instance, escalation calculations and the unlock feature are based on specific state types.

The name of an already existing state can be changed, or new states added through this screen. If the state "new" has been changed via the web interface, this change also has to be configured via the config file `Kernel/Config.pm` or via the SysConfig interface. The settings specified in the script below have to be modified to ensure that OTRS works with the changed state for "new".

```
[...]
# PostmasterDefaultState
# (The default state of new tickets.) [default: new]
$Self->{PostmasterDefaultState} = 'new';

# CustomerDefaultState
# (default state of new customer tickets)
$Self->{CustomerDefaultState} = 'new';
[...]
```

Script: Modifying the Kernel/Config.pm settings.

If a new state type should be added, the ticket_state_type table in the OTRS database needs to be modified with a database client program, as shown in the script below.

```
linux:~# mysql -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 23 to server version: 5.0.16-Debian_1-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use otrs;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> insert into ticket_state_type (name,comments) values
 ('own','Own
state type');
Query OK, 1 row affected (0.00 sec)

mysql> quit
Bye
linux:~#
```

Script: Modifying the OTRS database.

Now it is possible to use the new state type you just created. After a state has been linked with this new state type, the OTRS configuration also has to be changed to ensure that the new state is usable. Just modify the following options via SysConfig:

Ticket -> Frontend::Agent::Ticket::ViewPhoneNew > AgentTicketPhone###StateDefault - to define the default next state for new phone tickets.

Ticket -> Frontend::Agent::Ticket::ViewPhoneNew > AgentTicketPhone###StateType - to define the available next states for new phone tickets.

Ticket -> Frontend::Agent::Ticket::ViewEmailNew > AgentTicketEmail###StateDefault - to define the default next state for new email tickets.

Ticket -> Frontend::Agent::Ticket::ViewEmailNew > AgentTicketEmail###StateType - to define the available next states for new email tickets.

Ticket -> Frontend::Agent::Ticket::ViewPhoneOutbound > AgentTicketPhoneOutbound###State - to define the default next state for new phone articles.

Ticket -> Frontend::Agent::Ticket::ViewPhoneOutbound > AgentTicketPhoneOutbound###StateType - to define the available next states for new phone articles.

Ticket:Frontend::Agent::Ticket::ViewMove:Ticket::DefaultNextMoveStateType - to define the default next state after moving a ticket.

Ticket -> Frontend::Agent::Ticket::ViewBounce > StateDefault - to define the default next state after bouncing a ticket.

Ticket -> Frontend::Agent::Ticket::ViewBounce > StateType - to define the available next states in the bounce screen.

Ticket -> Frontend::Agent::Ticket::ViewBulk > StateDefault - to define the default next state in a bulk action.

Ticket -> Frontend::Agent::Ticket::ViewBulk > StateType - to define the available next states in the bulk action screen.

Ticket -> Frontend::Agent::Ticket::ViewClose > StateDefault - to define the default next state after closing a ticket.

Ticket -> Frontend::Agent::Ticket::ViewClose > StateType - to define the available next states in the close screen.

Ticket -> Frontend::Agent::Ticket::ViewCompose > StateDefault - to define the default next state in the Compose (reply) screen.

Ticket -> Frontend::Agent::Ticket::ViewCompose > StateType - to define the available next states in the Compose (reply) screen.

Ticket -> Frontend::Agent::Ticket::ViewForward > StateDefault - to define the default next state after forwarding a ticket.

Ticket -> Frontend::Agent::Ticket::ViewForward > StateType - to define the available next states in the Forward screen.

Ticket -> Frontend::Agent::Ticket::ViewForward > StateDefault - to define the default next state of a ticket in the free text screen.

Ticket -> Frontend::Agent::Ticket::ViewForward > StateType - to define the available next states in the free text screen.

Ticket -> Core::PostMaster > PostmasterDefaultState - to define the state of tickets created from emails.

Ticket -> Core::PostMaster > PostmasterFollowUpState - to define the state of tickets after a follow-up has been received.

Ticket -> Core::PostMaster > PostmasterFollowUpStateClosed - to define the state of tickets after a follow-up has been received on an already closed ticket.

Ticket -> Core::Ticket > ViewableStateType - to define the state types that are displayed at various places in the system, for example in the Queueview.

Ticket -> Core::Ticket > UnlockStateType - to define the state types for unlocked tickets.

Ticket -> Core::Ticket > PendingReminderStateType - to define the state type for reminder tickets.

Ticket -> Core::Ticket > PendingAutoStateType - to define the state type for Pending Auto tickets.

Ticket -> Core::Ticket > StateAfterPending - to define the state a ticket is set to after the Pending Auto timer of the configured state has expired.

Chapter 13. Modifying ticket priorities

OTRS comes with five default priority levels that can be modified via the "Priorities" link on the Admin page. 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. Also, OTRS orders tickets by internal database IDs in the QueueView.

Note

As with other OTRS entities, priorities may not be deleted, only deactivated by setting the Valid option to *invalid* or *invalid-temporarily*.

Important

If a new priority was added or if an existing one was changed, you might also want to modify some values in SysConfig:

- Ticket:Core::Postmaster::PostmasterDefaultPriority - defines the default priority for all incoming emails.
- Ticket:Frontend::Agent:Ticket::ViewPhoneNew:Priority - defines the default priority in the New Phone Ticket screen for agents.
- Ticket:Frontend::Agent:Ticket::ViewEmailNew:Priority - defines the default priority in the New Email Ticket screen for agents.
- Ticket:Frontend::Customer:Ticket::ViewNew:PriorityDefault - defines the default priority in the New Ticket screen in the Customer frontend.

Chapter 14. Creating your own themes

You can create your own themes so as to use the layout you like in the OTRS web frontend. To create own themes, you should customize the output templates to your needs.

More information on the syntax and structure of output templates can be found in the Developer Manual at *http://doc.otrs.org*, especialy in the chapter on *templates* [http://doc.otrs.org/developer/3.1/en/html/hacking.html#TemplatingMechanism].

As an example, perform the following steps to create a new theme called "Company":

1. Create a directory called `Kernel/Output/HTML/Company` and copy all files that you like to change, from `Kernel/Output/HTML/Standard` into the new folder.

   ### Important

   Only copy over the files you actually change. OTRS will automatically get the missing files from the Standard theme. This will make upgrading at a later stage much easier.

2. Customize the files in the directory `Kernel/Output/HTML/Company`, and change the layout to your needs.

3. To activate the new theme, add them in SysConfig under Frontend::Themes.

Now the new theme should be useable. You can select it via your personal preferences page.

Warning

Do not change the theme files shipped with OTRS, since these changes will be lost after an update. Create your own themes only by performing the steps described above.

Chapter 15. Localization of the OTRS frontend

OTRS offers multi-language support for its web interface.

Procedures for localization for the OTRS framework, steps to be followed to create a new language translation, as well as procedures for translation customizations, can be found in the "Language Translations" [http://doc.otrs.org/developer/3.1/en/html/contributing.html#translate] chapter from the developer manual on http://doc.otrs.org.

Chapter 16. PGP

OTRS has the capability to sign or encrypt outgoing messages with PGP. Further, encrypted incoming messages can be decrypted. Encryption and decryption are done with the GPL tool GnuPG. To setup GnuPG for OTRS, the following steps have to be performed:

1. Install GnuPG, via the package manager of your operating system.

2. Configure GnuPG for use with OTRS. The necessary directories for GnuPG and a private key have to be created. The command shown in the script below has to be executed as user 'otrs' from a shell.

```
   linux:~# su otrs
   linux:/root$ cd
   linux:~$ pwd
   /opt/otrs
   linux:~$ gpg --gen-key
   gpg (GnuPG) 1.4.2; Copyright (C) 2005 Free Software Foundation,
Inc.
   This program comes with ABSOLUTELY NO WARRANTY.
   This is free software, and you are welcome to redistribute it
   under certain conditions. See the file COPYING for details.

   gpg: directory `/opt/otrs/.gnupg' created
   gpg: new configuration file `/opt/otrs/.gnupg/gpg.conf' created
   gpg: WARNING: options in `/opt/otrs/.gnupg/gpg.conf' are not yet
active during t
   his run
   gpg: keyring `/opt/otrs/.gnupg/secring.gpg' created
   gpg: keyring `/opt/otrs/.gnupg/pubring.gpg' created
   Please select what kind of key you want:
      (1) DSA and Elgamal (default)
      (2) DSA (sign only)
      (5) RSA (sign only)
   Your selection? 1
   DSA keypair will have 1024 bits.
   ELG-E keys may be between 1024 and 4096 bits long.
   What keysize do you want? (2048)
   Requested keysize is 2048 bits
   Please specify how long the key should be valid.
        0 = key does not expire
     <n>  = key expires in n days
     <n>w = key expires in n weeks
     <n>m = key expires in n months
     <n>y = key expires in n years
   Key is valid for? (0)
   Key does not expire at all
   Is this correct? (y/N) y

   You need a user ID to identify your key; the software constructs
the user ID
   from the Real Name, Comment and Email Address in this form:
```

```
        "Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"

   Real name: Ticket System
   Email address: support@example.com
   Comment: Private PGP Key for the ticket system with address
support@example.com
   You selected this USER-ID:
   "Ticket System (Private PGP Key for the ticket system with
address support@examp
   le.com) <support@example.com>"

   Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? O
   You need a Passphrase to protect your secret key.

   Passphrase: secret
   Repeat passphrase: secret

   We need to generate a lot of random bytes. It is a good idea to
perform
   some other action (type on the keyboard, move the mouse, utilize
the
   disks) during the prime generation; this gives the random number
   generator a better chance to gain enough entropy.
   ++++++++++.++++++++++++++++++++++....+++++.+++++...++++++++++
+++++++++++++++.
   ++++++++++++++++++++++++.+++++.+++++.++++++++++++++++++++++++>
++++++++++>+++++
   .......>+++++<+++++.................................+++++

   Not enough random bytes available.  Please do some other work to
give
   the OS a chance to collect more entropy! (Need 280 more bytes)

   +++++++++.+++++..+++++++++..+++++....++++++++++++++++++.++++
++++++++++.++++
   ++++++++++++++++++++++++.++++++++++.+++++++++++++++.+++++++++
+.+++++++++++++++
   ..+++++>.+++++....>++++
+.....................................................
   ..........................................................>++++
+<+++++.........
   ............+++++^^^
   gpg: /opt/otrs/.gnupg/trustdb.gpg: trustdb created
   gpg: key 7245A970 marked as ultimately trusted
   public and secret key created and signed.

   gpg: checking the trustdb
   gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
   gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m,
0f, 1u
   pub   1024D/7245A970 2006-02-03
         Key fingerprint = 2ED5 BC36 D2B6 B055 7EE1  5833 1D7B F967
7245 A970
```

```
     uid                  Ticket System (Private pgp key for ticket
 system with addre
    ss support@example.com) <support@example.com>
    sub   2048g/52B97069 2006-02-03

    linux:~$
```

Script: Configuring GnuPG.

As shown in the script below, the default settings can be applied for most of the required parameters. Only the values for the key owner have to be entered correctly, with a proper password specified for the key.

3. Now OTRS has to be made ready to use PGP. From the Admin console, open the SysConfig interface and search for "PGP". Select the sub group Crypt::PGP from the search results.

   In the screen for the PGP settings, PGP should be activated for OTRS (first option). Also, the path to the gpg program should be set and checked.

   The next config setting (PGP::Options) may also require changing. Via this config setting, the parameters that are used for every execution of gpg by the 'otrs' user can be specified. In particular, the directory of the config files for GnuPG of the 'otrs' user is important. In the example `/opt/otrs/.gnupg` is used. This directory was created earlier during the PGP configuration.

   Via the next config option (PGP::Key::Password) it is possible to specify the pairs of key IDs and their passwords for own private keys. Because communication partners from outside write to the ticket system with their messages encrypted with your public key, OTRS can decrypt these messages with the ID/passwords specified here.

   How to get the id of your own private key? The ID of your own private key is already shown during the key generation (see step 1 from above). It is also possible to get the ID if the command specified in the following script is executed as user 'otrs':

```
    linux:~# su otrs
    linux:/root$ cd
    linux:~$ pwd
    /opt/otrs
    linux:~$ gpg --list-keys
    /opt/otrs/.gnupg/pubring.gpg
    ----------------------------
    pub   1024D/7245A970 2006-02-03
    uid                  Ticket System (Private pgp key for ticket
 system with
    address support@example.com) <support@example.com>
    sub   2048g/52B97069 2006-02-03

    linux:~$
```

Script: Getting the ID of your own private key.

The ID of the private key can be found in the line that starts with "sub". It is a hexadecimal string that is eight characters long, in the example above it is "52B97069". The password you have to specify for this key in the ticket system is the same that was given during key generation.

After this data is inserted, the "Update" button can be used to save the settings. OTRS is ready to receive and decrypt encoded messages now.

4. Finally, import a customer's public key. This ensures that encrypted messages can be sent out to this customer. There are two ways to import a public key of a customer.

   The first way is to specify the public key of a customer in the customer management interface.

   The second way is to specify the key via the PGP settings, reachable from the Admin page. On the right section of this screen, all already imported public keys of customers are displayed. After PGP has been activated and configured for OTRS, your own public key should also be listed there. In the left area of the PGP setting screen it is possible to search for keys. Also, a new public key can be uploaded into the system from a file.

   The files with the public key that need to be imported into OTRS have to be GnuPGP compatible key files. In most cases, the key stored in a file is an "ASCII armored key". OTRS can deal with this format.

Chapter 17. S/MIME

At first glance, encryption with S/MIME seems a little more complicated than with PGP. First, you have to establish a Certification Authority (CA) for the OTRS system, following which the procedures are very much like those needed with PGP: configure OTRS, install your own certificate, import other public certificates as needed, etc.

The S/MIME configuration is conducted outside the OTRS web interface for the most part, and should be carried out in a shell by the 'otrs' user. The MIME configuration under Linux is based on SSL (OpenSSL). Therefore, check first of all whether the OpenSSL package is installed on your system. The OpenSSL package includes a script called CA.pl, with which the most important steps of certificate creation can be performed. To simplify the procedure, find out where in the filesystem the CA.pl script is stored and enter the location temporarily into the PATH variable of the shell (see Script below).

```
otrs@linux:~> rpm -ql openssl | grep CA
/usr/share/ssl/misc/CA.pl
otrs@linux:~> export PATH=$PATH:/usr/share/ssl/misc
otrs@linux:~> which CA.pl
/usr/share/ssl/misc/CA.pl
otrs@linux:~> mkdir tmp; cd tmp
otrs@linux:~/tmp>
```

Script: Configuring S/MIME.

The script above shows that a new temporary directory ~/tmp has been created, in which the certificate is to be generated.

To create a certificate, perform the following operations in the command line (we assume that the OTRS administrator has to create a SSL certificate for test and learning purposes. In case you already have a certified SSL certificate for the encryption, use it and skip these steps):

1. Establish your own Certification Authority for SSL. You need it to certify the request for your own SSL certificate (see Script below).

```
otrs@linux:~/tmp> CA.pl -newca
CA certificate filename (or enter to create)

Making CA certificate ...
Generating a 1024 bit RSA private key
...++++++
......++++++
writing new private key to './demoCA/private/cakey.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be
 incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name
 or a DN.
There are quite a few fields but you can leave some blank
```

```
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:DE
State or Province Name (full name) [Some-State]:OTRS-state
Locality Name (eg, city) []:OTRS-town
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Your
 company
Organizational Unit Name (eg, section) []:
Common Name (eg, YOUR name) []:OTRS Admin
Email Address []:otrs@your-domain.tld
otrs@linux:~/tmp> ls -la demoCA/
total 8
-rw-r--r--  1 otrs otrs 1330 2006-01-08 17:54 cacert.pem
drwxr-xr-x  2 otrs otrs   48 2006-01-08 17:53 certs
drwxr-xr-x  2 otrs otrs   48 2006-01-08 17:53 crl
-rw-r--r--  1 otrs otrs    0 2006-01-08 17:53 index.txt
drwxr-xr-x  2 otrs otrs   48 2006-01-08 17:53 newcerts
drwxr-xr-x  2 otrs otrs   80 2006-01-08 17:54 private
-rw-r--r--  1 otrs otrs   17 2006-01-08 17:54 serial
otrs@linux:~/tmp>
```

Script: Establishing a Certification Authority for SSL.

2. Generate a certificate request (see Script below).

```
otrs@linux:~/tmp> CA.pl -newreq
Generating a 1024 bit RSA private key
.........................................++++++
....++++++
writing new private key to 'newreq.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be
 incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name
 or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:DE\keyreturn
State or Province Name (full name) [Some-State]:OTRS-state
Locality Name (eg, city) []:OTRS-town
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Your
 company
Organizational Unit Name (eg, section) []:
Common Name (eg, YOUR name) []:OTRS admin
Email Address []:otrs@your-domain.tld
```

```
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
Request (and private key) is in newreq.pem
otrs@linux:~/tmp> ls -la
total 4
drwxr-xr-x  6 otrs otrs  232 2006-01-08 17:54 demoCA
-rw-r--r--  1 otrs otrs 1708 2006-01-08 18:04 newreq.pem
otrs@linux:~/tmp>
```

Script: Creating a certificate request.

3. Signing of the certificate request. The certificate request can either be signed and thereby certified by your own CA, or made more credible by being signed by another external certified CA (see Script below).

```
otrs@linux:~/tmp> CA.pl -signreq
Using configuration from /etc/ssl/openssl.cnf
Enter pass phrase for ./demoCA/private/cakey.pem:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number:
            fd:85:f6:9f:14:07:16:c8
        Validity
            Not Before: Jan  8 17:04:37 2006 GMT
            Not After : Jan  8 17:04:37 2007 GMT
        Subject:
            countryName               = DE
            stateOrProvinceName       = OTRS-state
            localityName              = OTRS-town
            organizationName          = Your Company
            commonName                = OTRS administrator
            emailAddress              = otrs@your-domain.tld
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:

 01:D9:1E:58:C0:6D:BF:27:ED:37:34:14:D6:04:AC:C4:64:98:7A:22
            X509v3 Authority Key Identifier:

 keyid:10:4D:8D:4C:93:FD:2C:AA:9A:B3:26:80:6B:F5:D5:31:E2:8E:DB:A8
                DirName:/C=DE/ST=OTRS-state/L=OTRS-town/O=Your
 Company/
                CN=OTRS admin/emailAddress=otrs@your-domain.tld
                serial:FD:85:F6:9F:14:07:16:C7
```

```
Certificate is to be certified until Jan  8 17:04:37 2007 GMT (365
 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
Signed certificate is in newcert.pem
otrs@linux:~/tmp>
```

Script: Signing of the certificate request.

4. Generate your own certificate, and all data going with it, using the signed certificate request (see Script below).

```
otrs@linux:~/tmp> CA.pl -pkcs12 "OTRS Certificate"
Enter pass phrase for newreq.pem:
Enter Export Password:
Verifying - Enter Export Password:
otrs@linux:~/tmp> ls -la
total 12
drwxr-xr-x  6 otrs otrs  328 2006-01-08 18:04 demoCA
-rw-r--r--  1 otrs otrs 3090 2006-01-08 18:13 newcert.p12
-rw-r--r--  1 otrs otrs 3791 2006-01-08 18:04 newcert.pem
-rw-r--r--  1 otrs otrs 1708 2006-01-08 18:04 newreq.pem
otrs@linux:~/tmp>
```

Script: Generating a new certificate.

Now that these operations have been performed, the S/MIME setup must be completed in OTRS.

This part of the setup is carried out in the Admin page, choosing the link "SMIME". In case the general S/MIME support in OTRS has not yet been enabled, the mask points this out to the administrator and provides an appropriate link for enabling it.

With the SysConfig group "Crypt::SMIME", you can also enable and configure the general S/MIME support.

Here you can activate S/MIME support, and define the paths for the OpenSSL command and the directory for the certificates. The key file created above must be stored in the directory indicated here. Otherwise OpenSSL cannot use it.

The next step is performed in the S/MIME configuration on the OTRS Admin page. Here, you can import the private key(s) of the OTRS system and the public keys of other communication partners. Enter the public key that has been created in the beginning of this section and added to OTRS.

Obviously, all public S/MIME keys of communication partners can be imported using the customer administration tool as well.

Chapter 18. Access Control Lists (ACLs)

Introduction

From OTRS 2.0 on, Access Control Lists (ACLs) can be used to control access to tickets, modules, queues, etc., or to influence actions on tickets (closing, moving, etc.) in certain situations. ACLs can be used to supplement the existing permission system of roles and groups. Using ACLs, rudimental workflows within the system can be mapped, based on ticket attributes.

As yet, ACLs cannot be created using the SysConfig interface. They must be directly entered into the `Kernel/Config.pm` file. This chapter has some ACL examples which will walk you trough the process of defining ACL definitions, and a reference of all possible important ACL settings.

Examples

Example 18.1. ACL allowing movement into a queue of only those tickets with ticket priority 5.

This example shows you the basic structure of an ACL. First, it needs to have a name. In this case, it is "ACL-Name-2". Note that the ACLs will be numerically sorted before execution, so you should use the names carefully.

Secondly, you have a "Properties" section which is a filter for your tickets. All the criteria defined here will be applied to a ticket to determine if the ACL must be applied or not. In our example, a ticket will match if it is in the queue "Raw" and has priority "5 very high".

Lastly, a section "Possible" defines modifications to the screens. In this case, from the available queues, only the queue "Alert" can be selected in a ticket screen.

```
# ticket acl
$Self->{TicketAcl}->{'100-Example-ACL'} = {
    # match properties
    Properties => {
        # current ticket match properties
        Ticket => {
            Queue => ['Raw'],
            Priority => ['5 very high'],
        }
    },
    # return possible options (white list)
    Possible => {
        # possible ticket options (white list)
        Ticket => {
            Queue => ['Alert'],
        },
    },
};
```

Example 18.2. ACL disabling the closing of tickets in the raw queue, and hiding the close button.

Here you can see how a ticket field (state) can be filtered with more than one possible value to select from. It is also possible to limit the actions that can be executed for a certain ticket. In this case, the ticket cannot be closed.

```
$Self->{TicketAcl}->{'101-Second-Example-ACL'} = {
    # match properties
    Properties => {
        # current ticket match properties
        Ticket => {
            Queue => ['Raw'],
        }
    },
    # return possible options (white list)
    Possible => {
        # possible ticket options (white list)
        Ticket => {
            State => ['new', 'open', 'pending reminder'],
        },
        # possible action options
        Action => {
            AgentTicketBounce        => 1,
            AgentTicketClose         => 0,
            AgentTicketCompose       => 1,
            AgentTicketCustomer      => 1,
            AgentTicketForward       => 1,
            AgentTicketFreeText      => 1,
            AgentTicketHistory       => 1,
            AgentTicketLink          => 1,
            AgentTicketLock          => 1,
            AgentTicketMerge         => 1,
            AgentTicketMove          => 1,
            AgentTicketNote          => 1,
            AgentTicketOwner         => 1,
            AgentTicketPending       => 1,
            AgentTicketPhone         => 1, # only used to hide the
 Split action
            AgentTicketPhoneInbound  => 1,
            AgentTicketPhoneOutbound => 1,
            AgentTicketPrint         => 1,
            AgentTicketPriority      => 1,
            AgentTicketResponsible   => 1,
            AgentTicketWatcher       => 1,
            AgentTicketZoom          => 1,
            AgentLinkObject          => 1, # only used to hide the
 Link action
        },
    },
};
```

Example 18.3. ACL removing always state closed successful.

This example shows how it is possible to define negative filters (the state "closed successful" will be removed). You can also see that not defining match properties for a ticket will match any ticket, i. e. the ACL will always be applied. This may be useful if you want to hide certain values by default, and only enable them in special circumstances (e. g. if the agent is in a specific group).

```
$Self->{TicketAcl}->{'102-Third-ACL-Example'} = {
    # match properties
    Properties => {
        # current ticket match properties (match always)
    },
    # return possible options
    PossibleNot => {
        # possible ticket options
        Ticket => {
            State => ['closed successful'],
        },
    },
};
```

Example 18.4. ACL only showing Hardware services for tickets that are created in queues that start with "HW".

This example also shows you how you can use regular expressions for matching tickets and for filtering the available options.

```
$Self->{TicketAcl}->{'Only-Hardware-Services-for-HW-Queues'} = {
    # match properties
    # note we don't have "Ticket => {" because there's no ticket yet
    Properties => {
    Queue => {
        Name => ['[RegExp]HW'],
        }
    },
    # return possible options
    Possible => {
        # possible ticket options
        Ticket => {
            Service => ['[RegExp]^(Hardware)'],
        },
    },
};
```

Reference

In the example below there is a list of all parameters which can be used for ACLs.

Example 18.5. Reference showing all possible important ACL settings.

```
# ticket acl
$Self->{TicketAcl}->{'200-ACL-Reference'} = {
    # match properties
    Properties => {
        # current action match properties
        Frontend => {
            Action => ['AgentTicketPhone', 'AgentTicketEmail'],
        },
        # current queue match properties
        Queue => {
            Name     => ['Raw'],
            QueueID  => ['some id'],
            GroupID  => ['some id'],
            Email    => ['some email'],
            RealName => ['OTRS System'],
            # ...
            }
        },
        # current user match properties
        User => {
            UserLogin => ['some login'],
            # ...
            Group_rw => [
                'hotline',
            ],
            # ...
        },
        # current customer user match properties
        CustomerUser => {
            UserLogin => ['some login'],
            # ...
        },
        # current service match properties
        Service => {
            ServiceID => ['some id'],
            Name      => ['some name'],
            ParentID  => ['some id'],
            # ...
        },
        # current type match properties
        Type => {
            ID   => ['some id'],
            Name => ['some name'],
            # ...
        },
        # current priority match properties
        Priority = {
            ID   => ['some id'],
            Name => ['some name'],
            # ...
        },
        # current SLA match properties
        SLA = {
            SLAID    => ['some id'],
```

```
            Name     => ['some name'],
            Calendar => ['some calendar'],
            # ...
        },
        # current state match properties
        State = {
            ID       => ['some id'],
            Name     => ['some name'],
            TypeName => ['some state type name'],,
            TypeID   => ['some state type id'],
            # ...
        },
        # current ticket owner match properties
        Owner => {
            UserLogin => ['some login'],
            # ...
            Group_rw => [
                'some group',
            ],
            # ...
        },
        # current ticket responsible match properties
        Responsible => {
            UserLogin => ['some login'],
            # ...
            Group_rw => [
                'some group',
            ],
            # ...
        },
        # current dynamic field match properties
        DynamicField => {
            # keys must be in DynamicField_<field_name> format
            DynamicField_Field1          => ['some value'],
            DynamicField_OtherField      => ['some value'],
            DynamicField_TicketFreeText2 => ['some value'],
            # ...
        },
        # current ticket match properties
        Ticket => {
            Queue                => ['Raw'],
            State                => ['new', 'open'],
            Priority             => ['some priority'],
            Lock                 => ['lock'],
            CustomerID           => ['some id'],
            CustomerUserID       => ['some id'],
            Owner                => ['some owner'],
            DynamicField_Field1  => ['some value'],     # Must be the
untranslated values
                                                        #   specified
in the dynamic field
                                                        #   definition
and not the IDs
            DynamicField_MyField => ['some value'],
```

```
            # ...
        },
    },
    # return possible options (white list)
    Possible => {
        # possible ticket options (white list)
        Ticket => {
            Queue => ['Hotline', 'Coordination'],
            State => ['some state'],
            Priority => ['5 very high'],
            DynamicField_Field1  => ['some value'],
            DynamicField_MyField => ['some value'],
            # ...
            NewOwner => ['some owner'],
            OldOwner => ['some owner'],
            # ...
        },
        # possible action options (white list)
        Action => {
            AgentTicketBounce        => 1,
            AgentTicketClose         => 1,
            AgentTicketCompose       => 0,
            AgentTicketCustomer      => 0,
            AgentTicketForward       => 0,
            AgentTicketFreeText      => 1,
            AgentTicketHistory       => 1,
            AgentTicketLink          => 0,
            AgentTicketLock          => 1,
            AgentTicketMerge         => 0,
            AgentTicketMove          => 1,
            AgentTicketNote          => 1,
            AgentTicketOwner         => 1,
            AgentTicketPending       => 1,
            AgentTicketPhone         => 1, # only used to hide the
 Split action
            AgentTicketPhoneInbound  => 0,
            AgentTicketPhoneOutbound => 1,
            AgentTicketPrint         => 1,
            AgentTicketPriority      => 0,
            AgentTicketResponsible   => 1,
            AgentTicketWatcher       => 1,
            AgentTicketZoom          => 1,
            AgentLinkObject          => 1, # only used to hide the
 Link action
        },
    },
    # remove options (black list)
    PossibleNot => {
        # See section "Possible"
        # ...
    },
};
```

Chapter 19. Stats module

The OTRS stats module holds features to track operational statistics and generate custom reports associated with OTRS usage. The OTRS system uses the term "stat" generically to refer to a report presenting various indicators.

Proper configuration of the OTRS stats module is associated with a multitude of requirements and considerations. These include the various OTRS modules to be evaluated, user permission settings, indicators to be calculated and their complexity levels, ease of configuration of the stats module, speed and efficiency of calculations, and support of a rich set of output variants.

Statistical elements, i.e. files which supplement the functionality of the stats module for specific requirements, can be integrated for calculating complex statistics.

Handling of the module by the agent

When signed on as an agent, the navigation bar displays the link "Statistics". This has various submenu options, as shown in Figure.

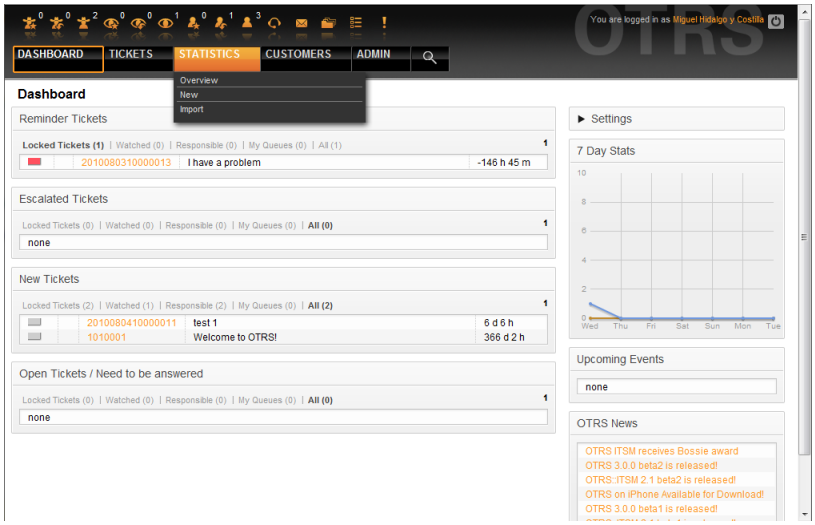

Figure: Statistics menu options.

The different options provided in the statistics menu are:

- *Overview*. Presents a list of different pre-configured reports.
- *New*. Requires rw rights.
- *Import*. Requires rw rights.

Overview

Selecting the "Statistics" link in the navigation bar, and then the submenu link "Overview", calls up the Overview screen. The Overview screen presents a list of all pre-configured reports the agent can use (see Figure below).

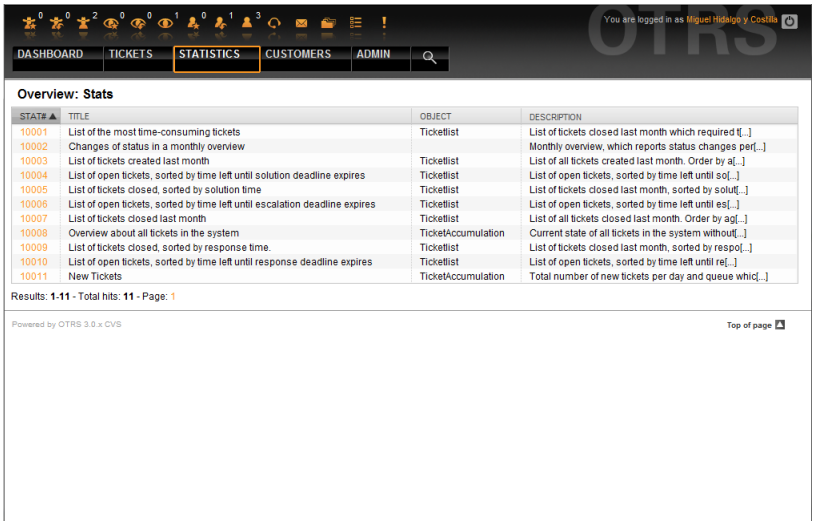

Figure: Overview of the standard reports.

The following information is provided for each of the standard reports listed in the Overview:

- *Stat#*. Unique report number.
- *Title*. Title of the report.
- *Object*. Object used for generating the statistic. In the case of a static statistic, no object is displayed as no dynamic object is used for its generation.
- *Description*. A brief description of the report.

When the stats module is installed, it comes preloaded with a few sample reports imported into the system. These are shown as a list on the Overview page. If the Overview list extends to more than a single page, the agent can browse through the different pages. The list of reports can be sorted as desired, by clicking the desired column header in the list. To generate a particular report, click on the stat number associated with the report in the Overiew list. This brings up the "View" interface for the report.

Generate and view reports

The view user interface provides the stat's configuration settings (see Figure below).

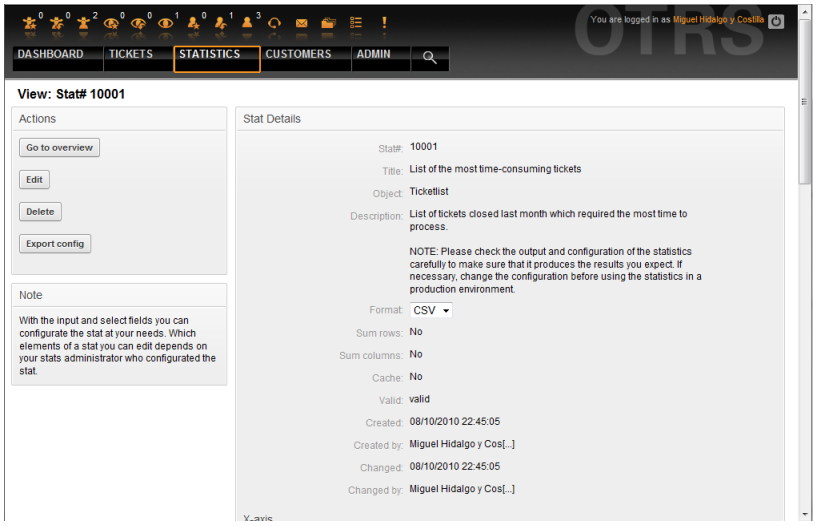

Figure: Viewing a specific report.

Configuration settings for a particular report can be set within the range of options in the View screen. Either the report creator or any others with the appropriate permissions can make the settings.

The page shows the following:

- Possible actions:
  - *Go to overview*. Link back to the Overview list of reports.
  - *Edit*. Edit the current report structure (rw rights required).
  - *Delete*. Delete the current report (rw rights required).
  - *Export config*. Export a report configuration, via file download (rw rights required).

    Usage: Export and Import functions allow for the convenient creation and testing of reports on test systems and subsequent easy integration into the production system.
- Report details:
  - *Stat#*. Number of the report.
  - *Title*. Title of the report.
  - *Object*. Object used for generating the report.
  - *Description*. Description on the report's purpose.
  - *Format*. Report output format which, depending on the configuration, can be any of the following output formats:
    - CSV.
    - Print.
    - Graph-lines.

    - Graph-bars.
    - Graph-hbars.
    - Graph-points.
    - Graph-lines-points.
    - Graph-area.
    - Graph-pie.

- *Graphsize*. Size in pixels for the graphic / chart. This option is only given when the report configuration allows a chart. All generally usable graphic sizes are configured by the OTRS administrator in SysConfig. The agent can then pre-select all relevant formats, while configuring the report.
- *Sum rows*. Indicates whether the report is amended by a column, whose cells state the sums of the respective rows.
- *Sum columns*. Indicates whether the report is amended by a row, whose cells state the sum of the respective columns.
- *Cache*. Indicates whether the generated report is cached in the filesystem.
- *Valid*. This can be set to "invalid" if a report must not be run temporarily for any reason. The "Start" button in the bottom of the right panel is then no longer displayed. The report can no longer be generated.
- *Created*. Creation time of the report.
- *Created by*. Name of the agent who created the report.
- *Changed*. Time when the report was last modified.
- *Changed by*. Name of the agent who altered the report last.
- *X-axis*. Using this function, the agent can switch the x and y axes (only when activated by the OTRS administrator).
- The general information is followed by information about the report itself. There are two different report (or stat) views:
    - *Static stat view*. Static report generators can be integrated into the stats module (see Figure below).

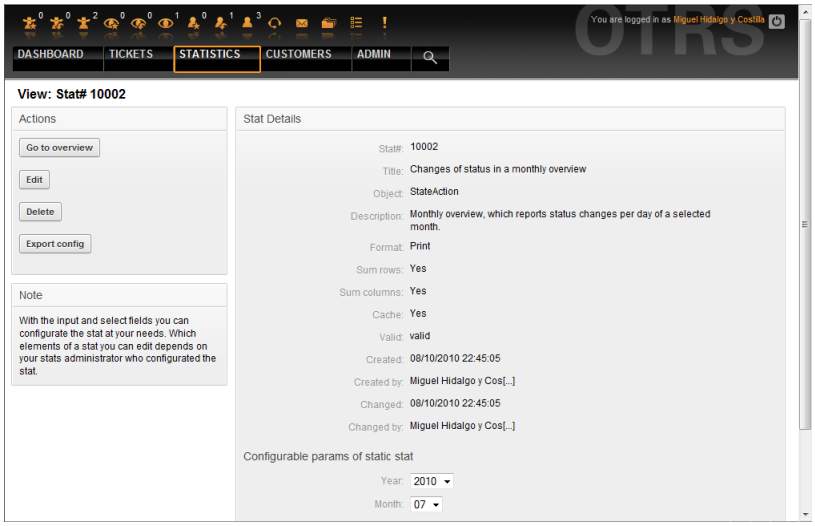

Figure: Viewing a static report.

- *Dynamic stat view* (see Figure above). They can be displayed in two different ways:
  - *Unchangeable settings*. The originator of the report has no permission for modifying this fields.
  - *Changeable settings*. The configuration settings of such reports can be changed by the agent.

Pressing the "Start" button (at the bottom of the screen) is the last step to generate the report. There are two possible reasons for this button to not be displayed:

1. The report was set to invalid and thus, deactivated.
2. The report was not configured cleanly and is, therefore, not yet executable. In this case, the necessary information can be found in the OTRS notification section (below the navigation bar).

If the settings on the View page are incorrect, this page is shown again after the "Start" button was pushed, and information about which input was incorrect is provided in the notification section.

Edit / New

Agents with write rights can edit an existing report configuration by calling up the edit user interface of the stats module. Alternately, they may create a new report. The associated screens can be reached in the following manner:

1. Edit: Via the "Edit" button in the stat view.
2. New: Via the "New" link in the Statistics menu from the navigation bar, or the "Add" button from the Overview page.

The stats are edited with a wizard in four steps:

1. General specifications.
2. Definition of the element for the X-axis.

3. Specification of the value series.

4. Selecting the restrictions to limit the report.

Steps 2 through 4 are only needed for the generation of reports with dynamic stats. For a static stat, only the general information (point 1) is required.

Information about how to handle the page is provided on each of these screens, below the Actions panel in a Hints panel.

If incorrect inputs are entered, the previously processed user interface is displayed again and with information about the incorrect input. This information can be found in the OTRS notification section. The next input user interface is only displayed after the current form has been filled out correctly.

1. *General specifications*. It is the first page of the Edit wizard (see Figure below).

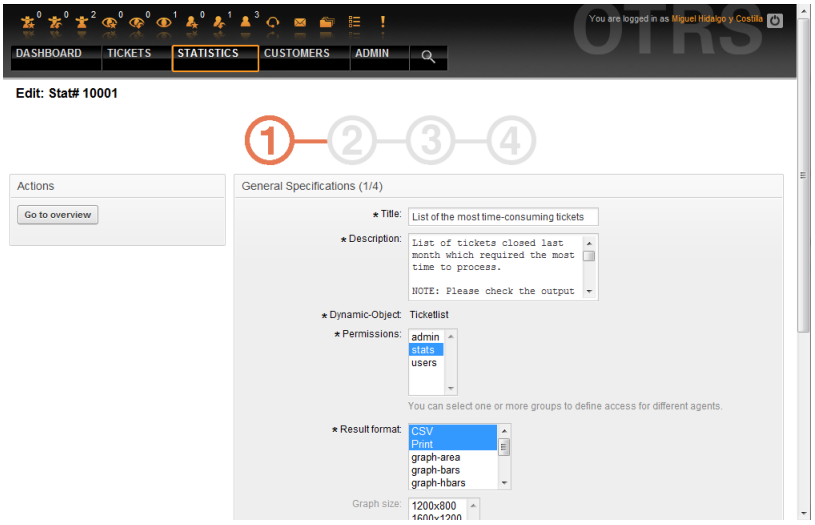

Figure: Editing the general specifications of a report.

In the screen showed in Figure, there are a great number of common specifications and settings that can be edited:

- *Title*. Should reflect the stat's purpose in a concise manner.

- *Description*. More descriptive information about the report definition, type of configuration parameters, etc.

- *Dynamic object*. If the OTRS installation provides various dynamic objects, one of them can be chosen. The objects meet the requirements of the particular modules.

- *Static file*. Usually this selection is not shown, as only static files which are not yet assigned to any reports are displayed. If "Static file" is displayed, however, it is important to tick the option field and select a generation mode (dynamic with a dynamic object or static with a file). If a static file is selected, the input user interfaces 2 through 4 are not shown as the static file contains all required configuration settings.

- *Permission settings*. Facilitate a restriction of the groups (and therefore, agents) who can later view and generate the preconfigured reports. Thus the various reports can be allocated

to the different departments and work groups who need them. It is possible to allocate one report to various groups.

Example 1: The "stats" group was selected. The report is viewable for all users having at least ro rights for the "stats" group. This access is available by default.

Example 2: A group named "sales" was selected. All users with ro rights for the "sales" group can see the stat in the view mode and generate it. However, the report will not be available for viewing by other users.

- *Format*. Output format of the stat: Depending on the configuration, one or more of the following formats can be chosen:
  - CSV.
  - Print.
  - graph-lines.
  - graph-bars.
  - graph-hbars.
  - graph-points.
  - graph-lines-points.
  - graph-area.
  - graph-pie.
- *Graphsize*. Select the chart size in pixels. This selection is only necessary if a graphical output format has been chosen under "Format". All graphic sizes that can generally be used are defined by the OTRS administrator in SysConfig. When configuring the report, the agent can pre-select all relevant formats.
- *Sum rows*. Indicates whether the report is amended by a column, whose cells contain the sum of the respective row.
- *Sum columns*. Indicates whether the report is amended by a row, whose cells contain the sum of the respective column.
- *Cache*. Specifies if the generated report should be cached in the filesystem. This saves computing power and time if the report is called up again, but it should only be used if the report's content is no longer changing.

  Caching is automatically prevented if the report contains no time designation values, or if a time designation value points to the future.

  If a cached report is edited, all cached data is deleted.
- *Valid*. This can be set to "invalid" if a pre-configured report must not be run temporarily for any reason. The "Start" button in the bottom of the right panel is then no longer displayed. The report can no longer be generated.

2. *Definition of the element for the X-axis*. It is the configuration of the element used for the depiction of the X-axis or, if tables are used, of the column name applied to the X-axis (see Figure).

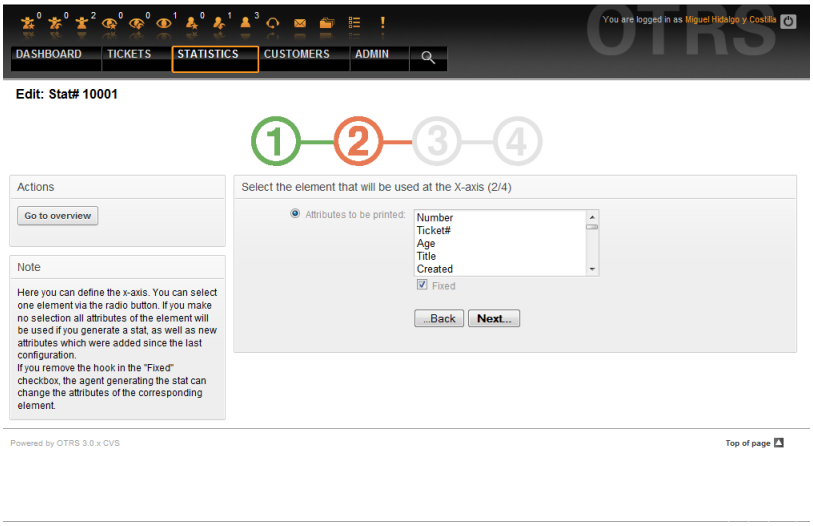

Figure: Definition of the element for the X-axis.

First of all, an element is selected using the option field. Then two or more attributes of the element must be selected. If no attributes are selected, all attributes are used including those added after the configuration of the report.

If the "Fixed" setting is disabled, the agent generating the report can change the attributes of the respective element in the "View" user interface.

Time elements are different as time period and scale have to be stated. Type and number of elements result from the used dynamic object and vary depending on it.

If all input is correct, the "Next" button leads to the "Value series" form. It is also possible to go back to editing earlier sections.

3. *Specification of the value series*.

In the third step of the report configuration, the value series are defined (see Figure below). They will later form the individual graphs or the various series within a tabular view.

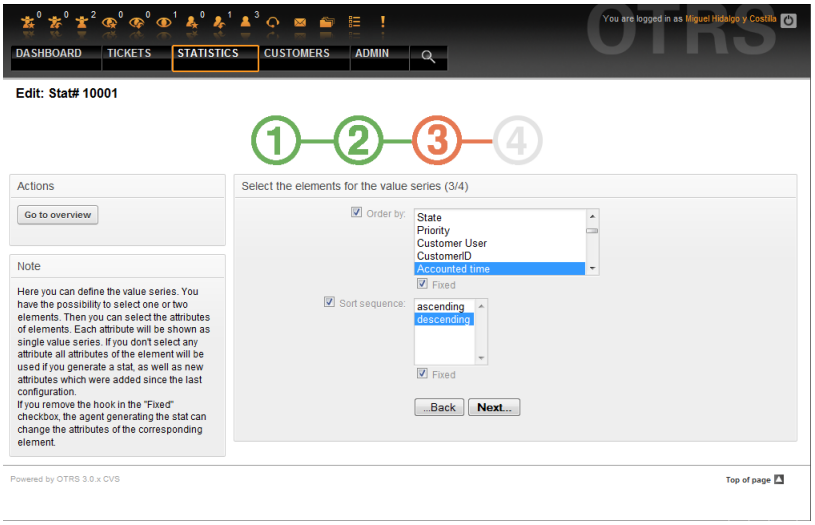

Figure: Definition of the value series.

If an element is selected, each chosen attribute will correspond to a value series (see the Example 19-1 below).

Example 19.1. Definition of a value series - one element

Element Queue:

- Value series 1 = Raw
- Value series 2 = Junk
- ....

If two elements are selected, each selected attribute of the first element is combined with an attribute of the second element to form a value series (see Example 19-2 below).

Example 19.2. Definition of a value series - two elements

Element 1 queue, Element 2 status:

- Value chain 1 = Raw - open
- Value series 2 = Raw - successfully closed
- Value series 3 = Junk - open
- Value series 4 = Junk - successfully closed

Selection of three or more elements is not allowed.

Additionally the same conditions apply to the selection of the attributes and the "Fixed" checkbox as to the "X-axis" selection:

- If no attributes of the element are selected, all attributes are used, including those added after the configuration of the report.

- If the "Fixed" setting is disabled, the agent generating the report can change the attributes of the respective element.

4. *Setting restrictions to the report*. This is the fourth and final step of the configuration (see Figure below). The restrictions serve to limit the results to the selected criteria. In many cases, no restrictions at all may be set up.

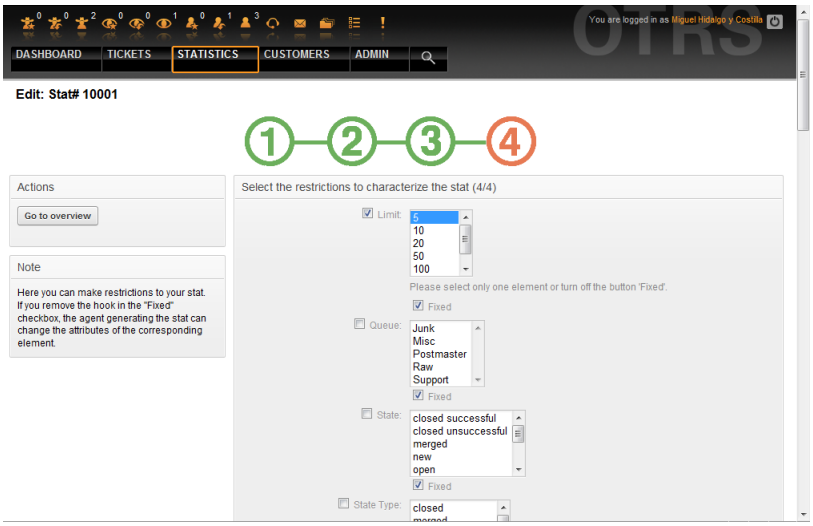

Figure: Definition of restrictions.

After all the restrictions are set up, the configuration of the report is completed by pressing the "Finish" button.

Import

The Import user interface (see Figure below) can be accessed by choosing from the navigation bar, the link "Statistics", then "Import". Alternately, pressing the Import button on the Overview screen achieves the same result. "rw" rights to the report are required.

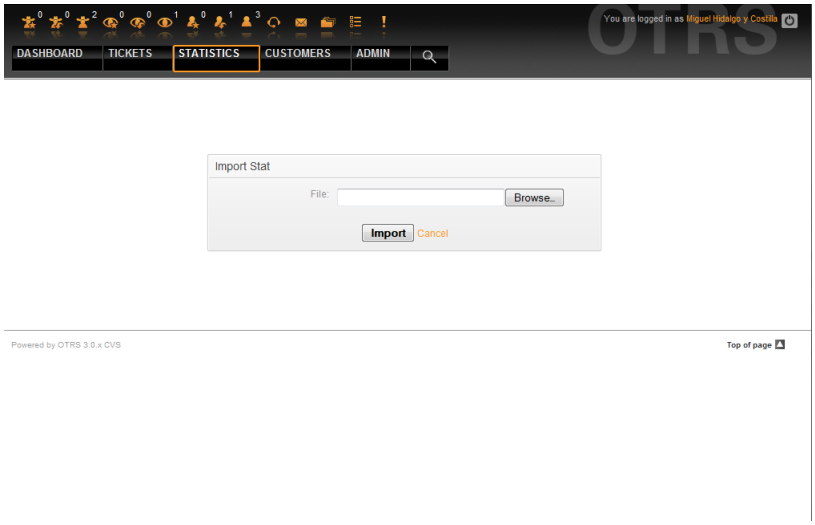

Figure: The Import user interface.

Facilitates the import of reports and is, when combined with the export function of the module, a very handy functionality. Stats can be created and tested conveniently on test systems, then imported into the production system.

The import is effected by an easy file upload. The "View" user interface of the imported report is opened automatically afterwards.

Administration of the stats module by the OTRS administrator

This section provides information about the tasks and responsibilities of the OTRS administrator dealing with the statistics module.

Permission settings, Groups and Queues

No new queues and/or groups are created when the stats module is installed.

The default configuration of the module registration gives all agents with "stats" group permissions access to the stats module.

Access according to permission settings:

- *rw*. Allows configuring statistics and reports.
- *ro*. Permits generating pre-configured statistics and reports.

The OTRS administrator decides whether agents with the permission to generate pre-configured reports are allocated ro rights in the "stats" group, or if their respective groups are added in the module registration in SysConfig.

SysConfig

The SysConfig groups Framework:Core::Stats, Framework:Core::Stats::Graph and Framework:Frontend::Agent::Stats contain all configuration parameters for the basic set-up of the statistics module. Moreover, the configuration parameter $Self->{'Frontend::Module'}->{'AgentStats'} controls the arrangement and registration of the modules and icons within the statistics module.

Administration of the stats module by the system administrator

Generally, no system administrator is needed for the operation, configuration and maintenance of the statistics module. However, a little background information for the system administrator is given at this point.

Note

File paths refer to subdirectories of the OTRS home directory (in most cases `/opt/otrs`).

Data base table

All report configurations are implemented and administrated in XML, and therefore stored in the database table "xml_storage". Other modules whose content is presented in xml format use this table as well.

List of all files

The following files are necessary for the stats module to work accurately:

- `Kernel/System/Stats.pm`
- `Kernel/Modules/AgentStats.pm`
- `Kernel/System/CSV.pm`
- `Kernel/Output/HTML/Standard/AgentStatsOverview.dtl`
- `Kernel/Output/HTML/Standard/AgentStatsDelete.dtl`
- `Kernel/Output/HTML/Standard/AgentStatsEditSpecification.dtl`
- `Kernel/Output/HTML/Standard/AgentStatsEditRestrictions.dtl`
- `Kernel/Output/HTML/Standard/AgentStatsEditXaxis.dtl`
- `Kernel/Output/HTML/Standard/AgentStatsEditValueSeries.dtl`
- `Kernel/Output/HTML/Standard/AgentStatsImport.dtl`
- `Kernel/Output/HTML/Standard/AgentStatsPrint.dtl`
- `Kernel/Output/HTML/Standard/AgentStatsView.dtl`
- `Kernel/System/Stats/Dynamic/Ticket.pm`
- `bin/otrs.GenerateStats.pl`

Caching

Whether the results of a statistic are to be cached or not can be setup in the configuration. Cached report results are stored as files in the `var/tmp`directory of the OTRS installation (in most cases `/opt/otrs/var/tmp`).

Cached stats can be recognized by the "Stats" prefix.

If the data is lost, no major damage is caused. The next time the report is called up, the stats module will not find the file any more and so will generate a new report. Of course this will probably take a little longer to run.

otrs.GenerateStats.pl

This file is saved in the `bin`directory. It facilitates the generation of report in the command line.

As an example, see the command line call in the following script.

```
bin> perl otrs.GenerateStats.pl -n 10004 -o /output/dir
```

Script: Generating a report from the command line.

A report from the stat configuration "Stat# 10004" is generated and saved as csv in the `/output/dir` directory.

The generated report can also be sent as an e-mail. More information can be called up with the command in the script below.

```
bin> perl otrs.GenerateStats.pl --help
```

Script: Getting information about the otrs.GenerateStats.pl file.

Automated stat generation - Cronjob

It usually does not make sense to generate reports manually via the command line, as the stats module has a convenient graphical user interface. However, generating reports manually does make sense when combined with a Cronjob.

Imagine the following scenario: On the first day of every month, the heads of department want to receive a report for the past month. By combining a cronjob and command line call the reports can be sent to them automatically by e-mail.

Static stats

The stats module facilitates the generation of static statistics. For every static stat a file exists in which its content is precisely defined.

This way, very complex stats can be generated. The disadvantage is that they are not particularly flexible.

The files are saved in the directory `Kernel/System/Stats/Static/`.

Using old static stats

Prior OTRS versions 1.3 and 2.0 already facilitated the generation of stats / reports. Various reports for OTRS versions 1.3 and 2.0 which have been specially developed to meet customers' requirements can be used in recent OTRS versions too.

The files must merely be moved from the `Kernel/System/Stats/` path to `Kernel/System/Stats/Static/`. Additionally the package name of the respective script must be amended by "::Static".

The following example shows how the first path is amended.

```
package Kernel::System::Stats::AccountedTime;
```

```
package Kernel::System::Stats::Static::AccountedTime;
```

Default stats

"It is not always necessary to reinvent the wheel..."

The stats module provides various default reports. Reports which are of interest for all OTRS users will in future be added to the default reports set of the stats module package. Default reports are saved in the stats module xml format in the`scripts/test/sample/` directory.

Chapter 20. Generic Interface

The OTRS Generic Interface consists of a multiple layer framework that lets OTRS communicate with other systems via a web service. This communication could be in two different directions:

- *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

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.

The list of provided Generic Interface modules shipped with OTRS will be updated and increased over time.

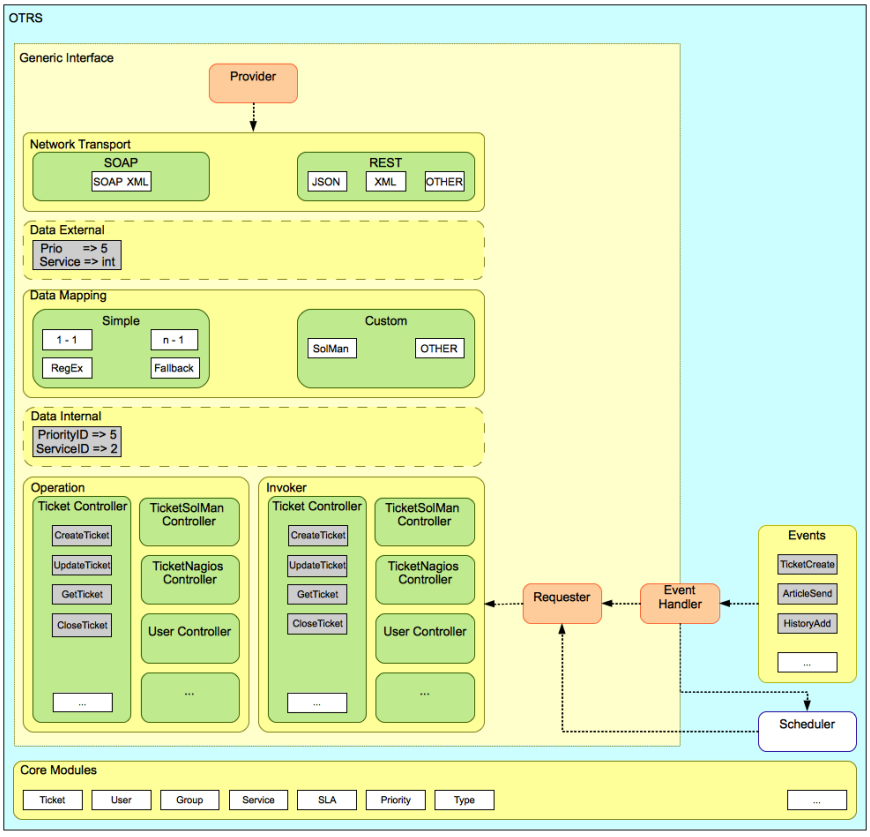

Figure: The graphical interface layers.

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.

Provider communication is handled by a new web server handle called "nph-genericinterface.pl".

Requester communication could be initiated during an event triggered by a Generic Interface module or any other OTRS module. This event is catched 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 deamon 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 Low" (OTRS) should be mapped to "Information" on 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 Remote System's interface.

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

Generic Interface has a defined flow to perform actions as a provider and as a requester.

This flows are described below:

OTRS as Provider

Remote Request:

1. HTTP request
   - OTRS receives HTTP request and pass it through the layers.
   - The provider module in in charge to execute and control this 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 pass it through the layers.
     - Asynchronous
       - Create a new Generic Interface (Requester) task for the OTRS Scheduler (by delegating the request execution to the Scheduler, the user experience could be highly improved, otherwise all 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 Scheduler process reads the new task and creates a call to the Requester that will create a new request and pass 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 form 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 by specifically by each Invoker (included error handling if any).
    - Return the Invoker result and data to the Requester.
6. Event Handler or Scheduler
    - Receives the data from the Requester, in the case of the Scheduler this data might contain information to Re-Schedule the task immediately or in the future.

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 is defined what actions the web service can perform internally (Operation), what can 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 pre-defined way, using already made building blocks that are independent from each other and interchangeable.

Web Service Graphical Interface

The web service graphical user interface (GUI) is a tool that allows to construct complex web service configurations in a friendly and nice interface. It allows to:

- Create and Delete web services.
- Import and Export configurations (in YAML file format) for existing web services.
- View, Revert and Export old configurations for existing web services in the Web Service History screen.
- Track all communication logs for each web service in the Debugger screen.

Web Service Overview

The "Web Services" link in the main screen of Admin Interface (in the System Administration box) leads to the web services overview screen, where you are able to manage your web service configurations. You can add new web services or change the configuration of the existing ones from this screen.

Every web service configuration screen has in the upper part of the screen a navigation path in a "bread crumbs" style. This navigation path is useful to know exactly in which part of the web service configuration we are, and also we can jump back to any level of the configuration at any time (this action will not save any changes).

Note

To create a new web service, press the button "Add web service", and provide the needed information.

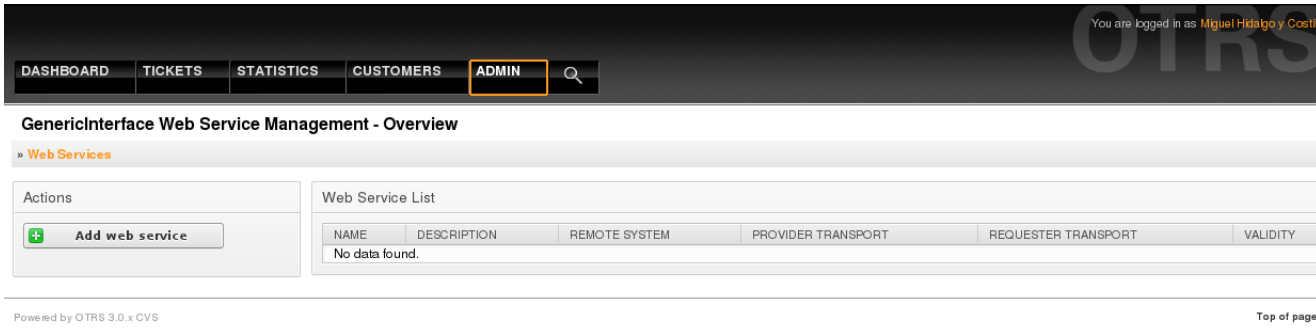

Figure: Web services overview.

Web Service Add

The only required field in this part is the web service "Name" that needs to be unique in the system and non empty. Other fields are also necessary for the configuration like the "Debug Threshold" and "Validity" but these fields are already filled with the default value for each list.

The default value for "Debug Threshold" is "debug", under this configuration all communication logs are registered in the database, each Debug Threshold value is more restrictive and discard communication logs set for lower values.

Debug Threshold levels (from lower to upper)

- Debug
- Info
- Notice
- Error

It is also possible to define the network transport protocol for "OTRS as Provider" and "OTRS as requester".

Click on the "Save" button to register the new web service in the database or click "Cancel" to discard this operation. You will now be returned to the web service overview screen.

If you already have a web service configuration file in YAML format you can click on the "Import web service" button on the left side of the screen. For more information on importing web services please check the next section "Web Service Change".

Note

To change or add more details to a web service, click on the web service name in the web service overview screen.

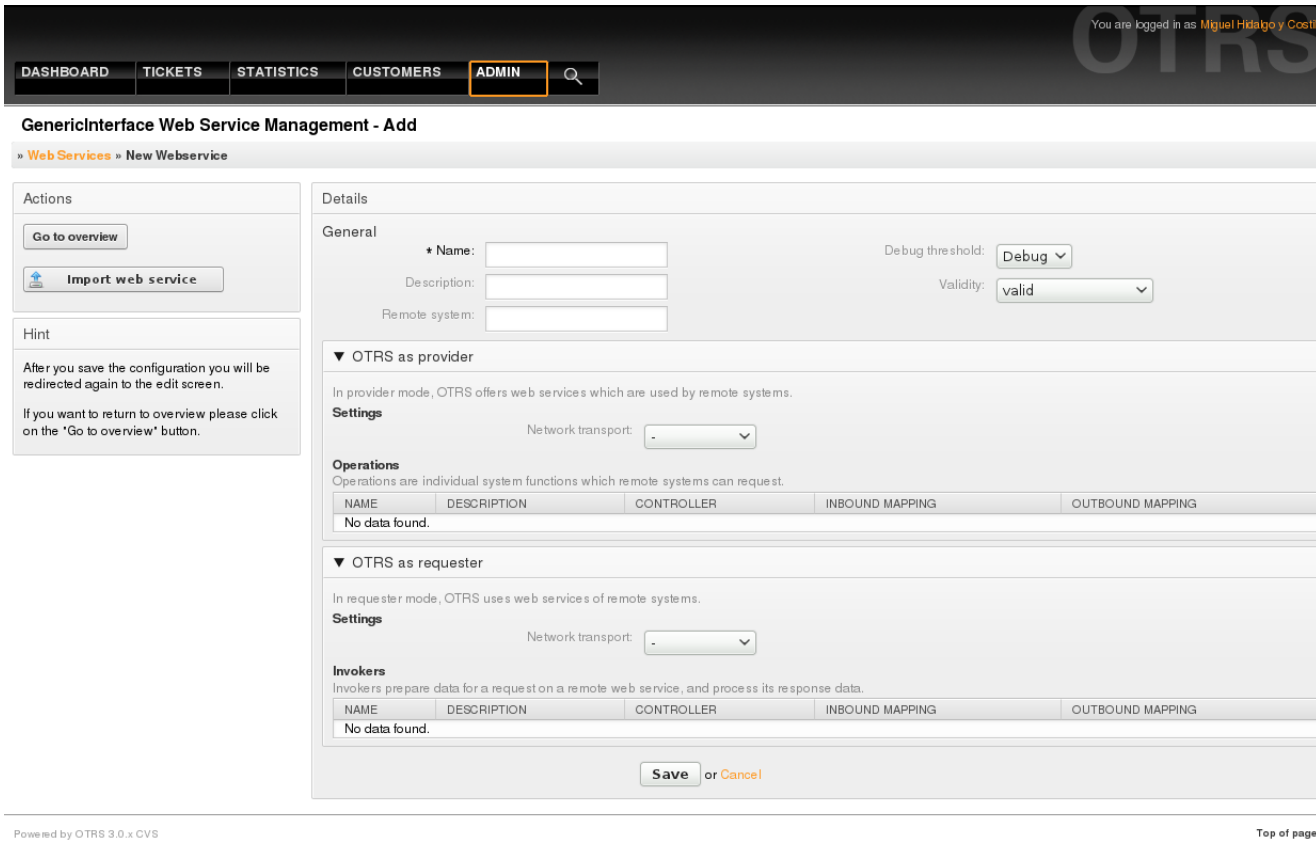

Figure: Web services add.

Web Service Change

On this screen you have a complete set of functions to handle every part of a web service. On the left side in the action column you can find some buttons that allows you to perform all possible actions on a web service:

- Clone web service.
- Export web service.
- Import web service.

- Configuration History.
- Delete web service.
- Debugger.

Note

"Configuration history" and "Debugger" will lead you to different screens.

Web Service Clone

To clone a web service, you need to click on the "Clone web service" button, a dialog will be shown where you can use the default name or set a new name for the (cloned) web service.

Note

Remember the name of the web service must be unique within the system.

Click on "Clone" button to create the web service clone or "Cancel" to close the dialog.

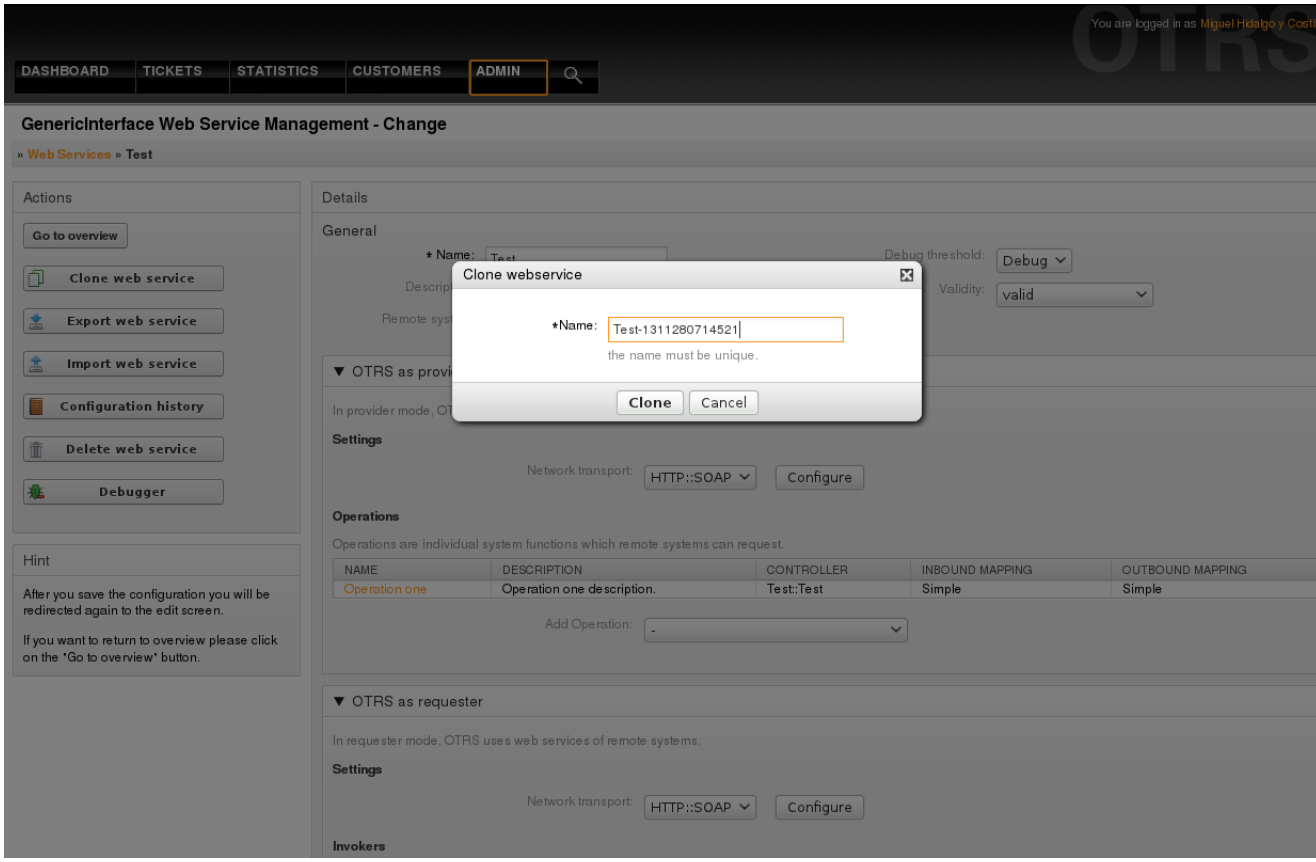

Figure: Web service clone.

Web Service Export

The "Export web service" button gives you the opportunity to dump the configuration of the current web service into a YAML file, download it and store it on your file system. This can be specially

useful if you want to migrate the web service from one server to another, for example from a testing environment to a production system.

Warning

All stored passwords in the web service configuration will be exported as plain text.

Right after clicking the "Export web service" button a save dialog of your browser will appear, just like when you click on a file download link on a web page.

Note

Each browser on each operating system has its own save dialog screen and style, depending on the browser and its configuration it is possible that no dialog is shown and the file is saved to a default directory on your file system. Please check your browser documentation for more specific instructions if needed.

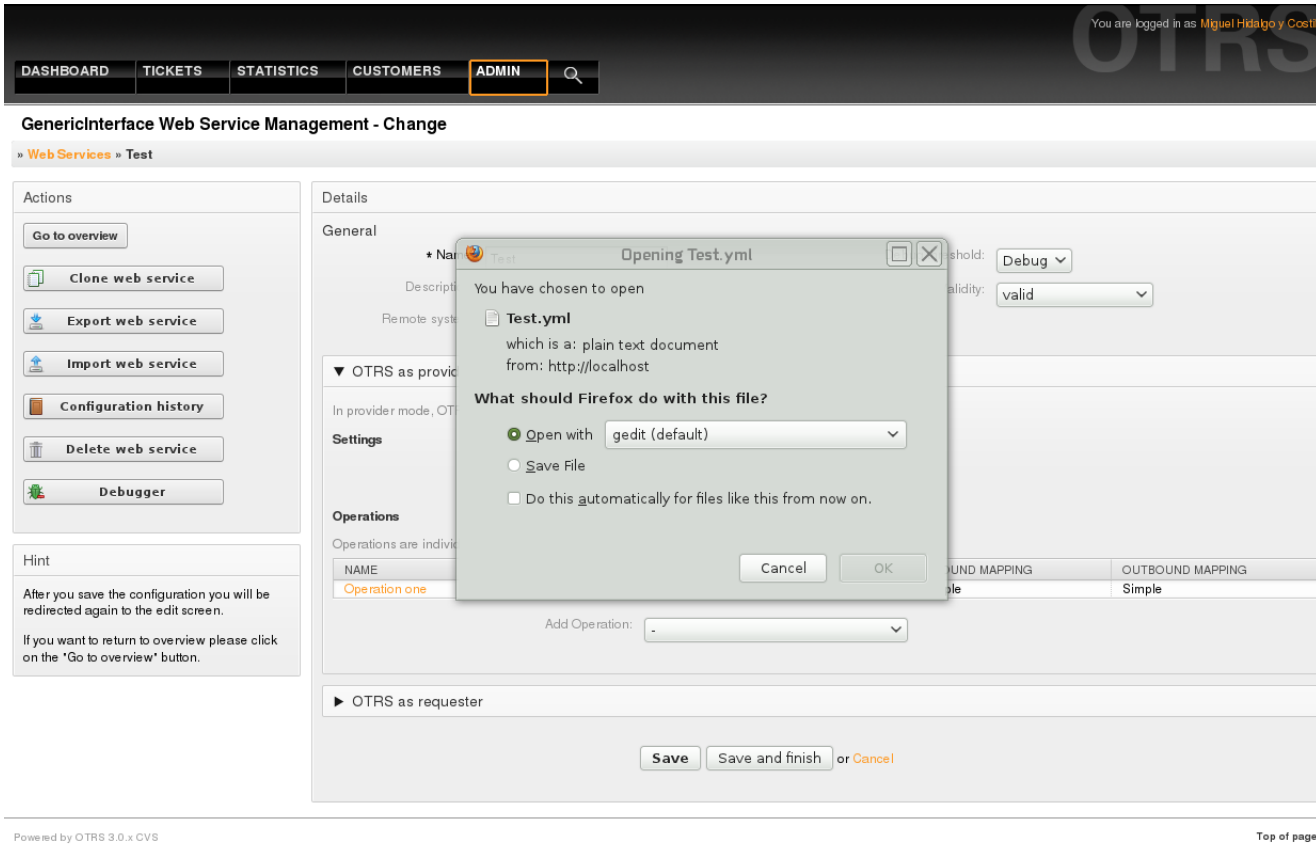

Figure: Web services export.

Web Service Import

A valid web service configuration YAML file is required to use the import web service feature. Click on the "Import web service" button, browse for the configuration file or provide the complete path in the input box.

Click "Import" button to create a new web service from a file or "Cancel" to close the dialog.

Note

The web service name will be taken from the configuration file name (e.g. if the file name is MyWebservice.yml the resulting web service will be named MyWebservice). If a web service is registered in the system with the same name as the web service that you want to import, the system will lead you to the web service change screen to let you change the name of the imported web service.

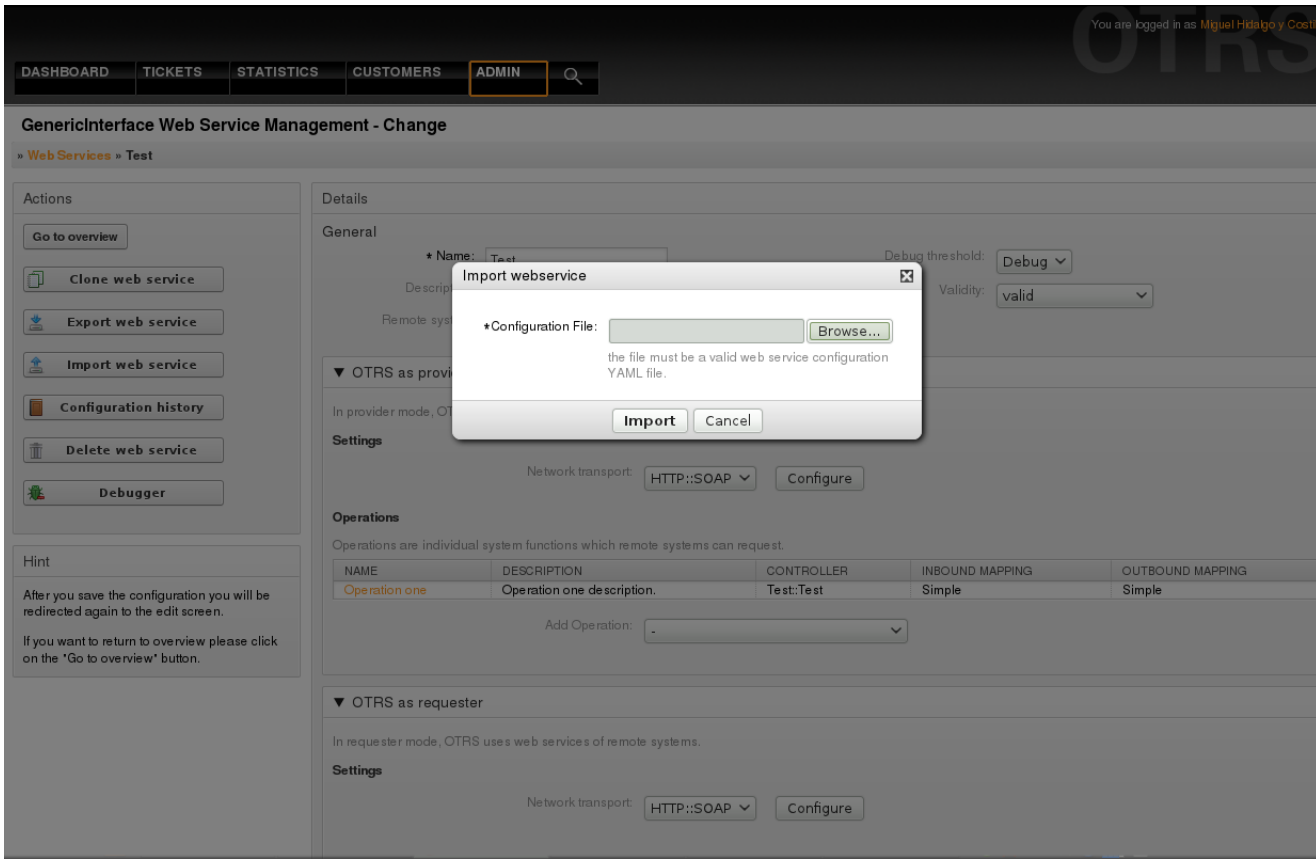

Figure: Web services import.

Web Service History

Every change to the web service configuration creates a new entry in the web service history (as a journal). The web service history screen displays a list of all configuration versions for a web service. Each row (version) in the "Configuration History List" represents a single revision in the web service history.

Click on one of the rows to show the whole configuration as it was on that particular date / time. The configuration will be shown in the "History details" section of this screen. Here you are also able to export the selected web service configuration version or to restore that version into the current web service configuration.

The "Export web service configuration" behaves exactly as the "Export web service" feature in the web service change screen. For more information refer to that section.

If changes to the current web service configuration does not work as expected and it is not easy to revert the changes manually, you can click on the "Revert web service configuration" button.

This will open a dialog to ask you if you are sure to revert the web service configuration. Click "Revert web service configuration" in this dialog to replace the current configuration with the selected version, or click "Cancel" to close the dialog.

Warning

Remember that any passwords stored in the web service configuration will be exported as plain text.

Please be careful when you restore a configuration because this can't be undone.

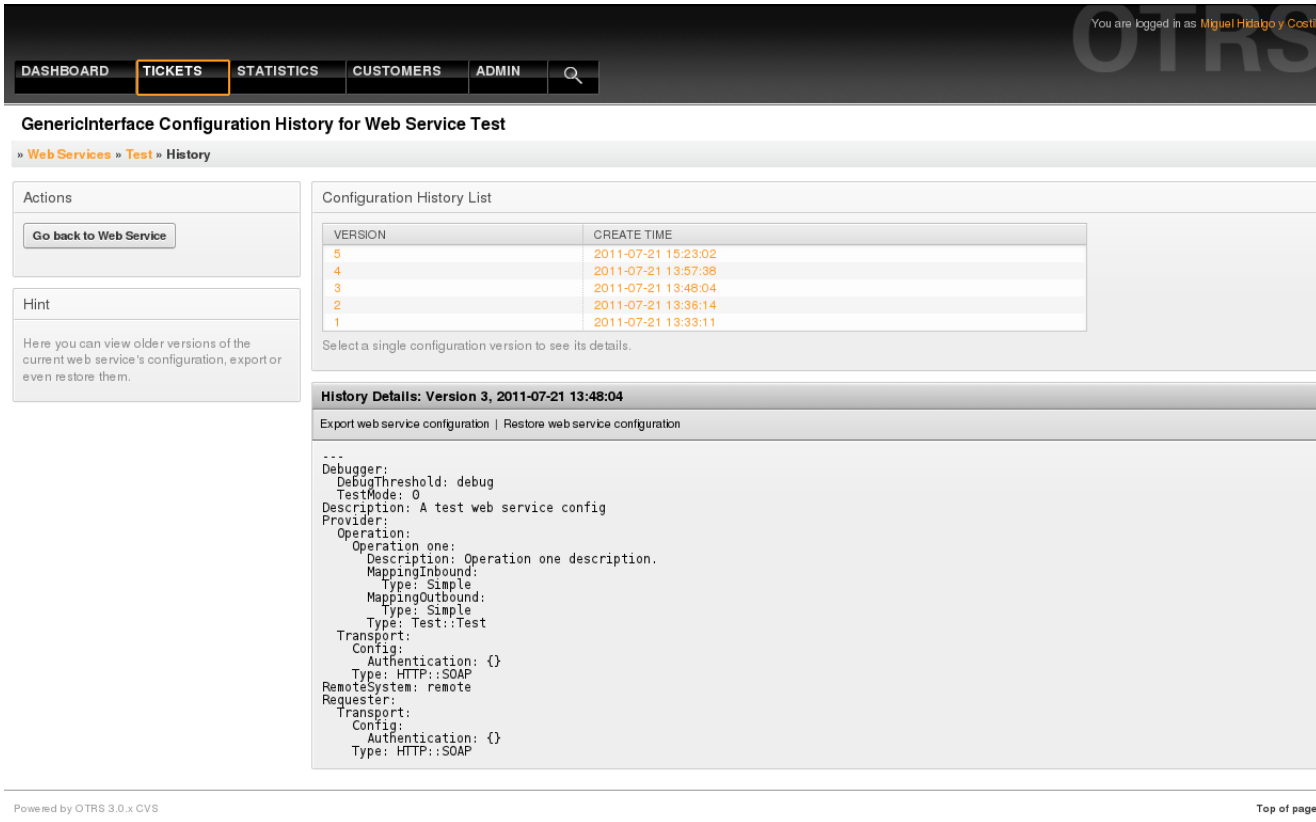

Figure: Web service history.

Web Service Delete

Sometimes it is necessary to delete a web service completely. To do this you can press on the "Delete web service" button and a new dialog will appear asking for confirmation.

Clink on "Delete" to confirm the removal of the web service or on "Cancel" to close the dialog.

Warning

Delete a web service can't be undone, please be careful when deleting a web service.

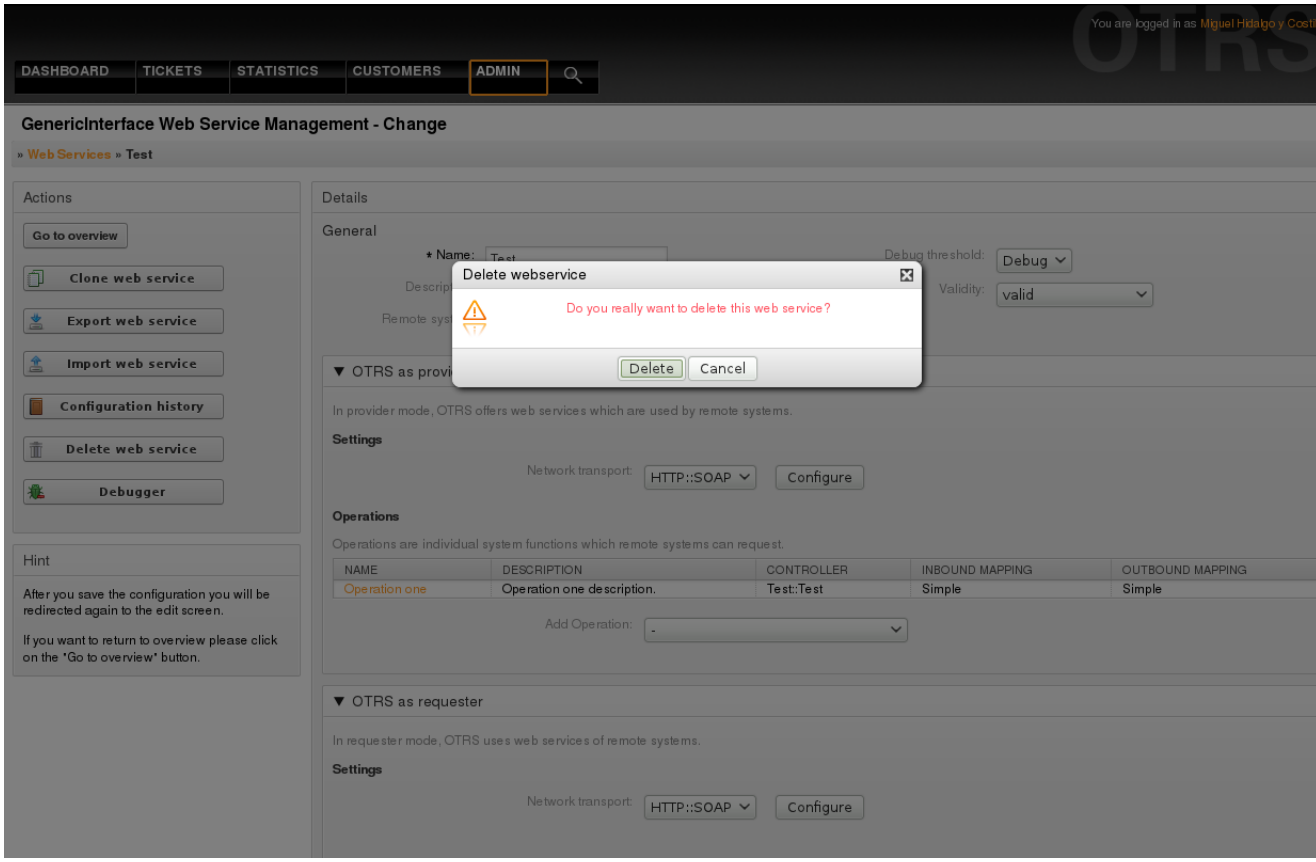

Figure: Web service delete.

Web Service Debugger

The Debugger stores the log of a web service. In the debugger screen you can track all the web service communications for either provider or requester types.

When this screen is shown the request list starts to load. After the list is fully filled you can choose one of the rows (that means a communication sequence) to check its details. This details will appear in a box below.

You can narrow the communication list using the filter on the right part of the screen. You can filter by:

- Communication type (provider or requester)
- Date: before and / or after a particular date
- The remote IP Address
- A combination of all.

After filter settings are set, push the "Refresh" button and a new list will be displayed meeting your search criteria.

Note

Depending on the search criteria for the filters the new list could return no results.

On the left part of the screen under the action column you can select "Go back to the web service" or clear the debugger log by pushing the "Clear" button. This will open a dialog that ask you to confirm erasing of the log. Click "Clear" in the dialog button to perform the action or click on "Cancel" to close this dialog.

In the "Request details" section you can see all the details for the selected communication. Here you can track the complete flow and check for possible errors or confirm success responses.

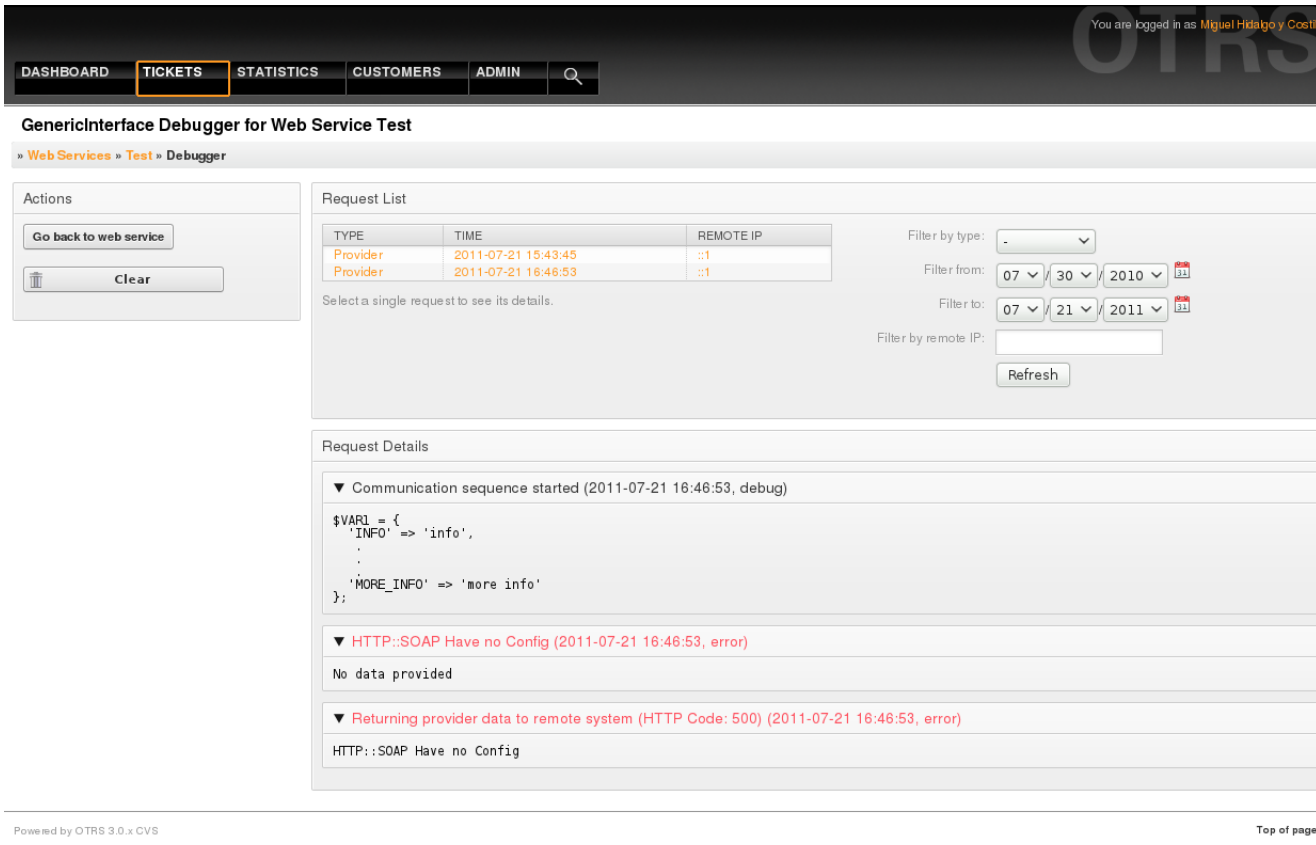

Figure: Web service debugger.

Web Service Configuration Change

Returning to the web service change screen, now we are going to review the right side of it. Here we have the possibility to modify all the general data for a web service such as name, description, debug threshold, etc. Also there are two more sections below that allows us to modify specific parameters for communication types "OTRS as Provider" and "OTRS as Requester".

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 to save the current configuration level. After saving the disabled links will be re-enabled again allowing you to continue with the configuration.

On the "OTRS as provider" section it is possible to set or configure the network transport protocol. Only network transport backends that are registered are shown on the list. To configure the network transport click on the "Configure" button. It is also possible to add new operations in this box. To do this select one of the available operations from the "Add Operation" list. This will

lead you to the operation configuration screen. After saving the new operation it will be listed in the table above.

"OTRS as requester" is very similar to the previous one, but instead of "operations" you can add invokers here.

Click the "Save" button to save and continue configuring the web service, "Save and finish" to save and return to the web service overview screen, or "Cancel" to discard current configuration level changes and return to web service overview screen.

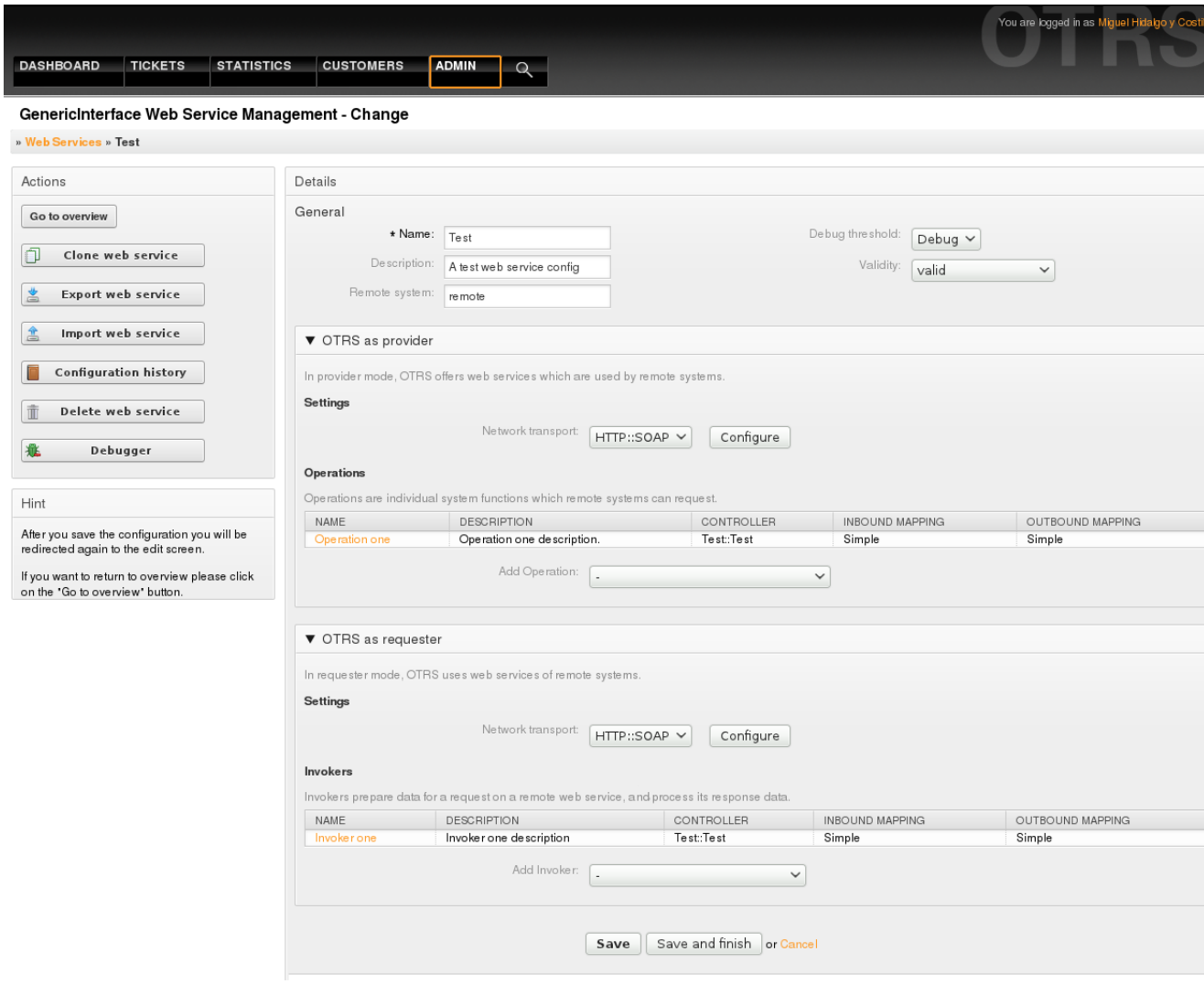

Figure: Web services change.

Note

Like the other Generic Interface configuration screens such as Network Transport, operation, Invoker and Mapping, the initial configuration (add) screen will only present two options: "Save" and "Cancel", when the configuration is re-visited then a new option "Save and Finish" will appear. The behavior of this feature is defined below.

"Save" will store the current configuration level in the database and it will return to the same screen to review your changes or to configure deeper settings.

"Save and Finish" will store the current configuration level in the database and it will return to the previous screen in the configuration hierarchy (to the immediate upper configuration level).

"Cancel" will discard any configuration change to the current configuration level and will return to the previous screen in the configuration hierarchy.

Web Service Provider Network Transport

In future the list of available network transports will be increased. Currently only the "HTTP::SOAP" transport is available. Each transport has different configuration options to setup and they might use different frontend modules to configure it, but mostly they should look similar to the "HTTP::SOAP" transport configuration module.

For "HTTP::SOAP" protocol as provider the configuration is quite simple. There are only two settings: "Namespace" and "Maximum message length". These fields are required. The first one is a URI to give SOAP methods a context, reducing ambiguities, and the second one it's a field where you can specify the maximum size (in bytes) for SOAP messages that OTRS will process.

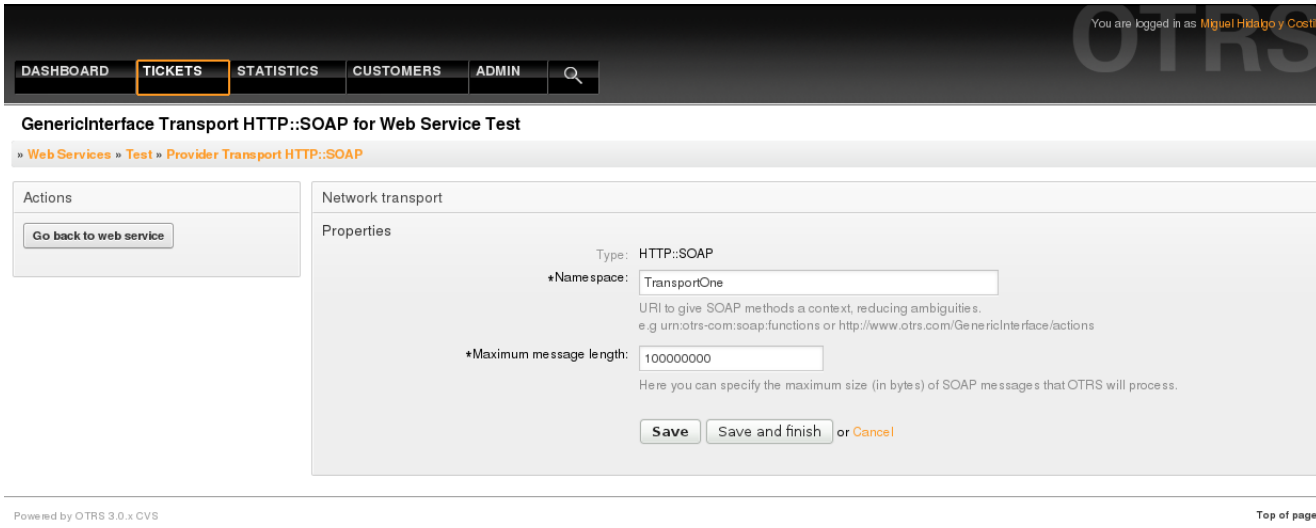

Figure: Web service provider network transport.

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 shares the same configuration dialog. But each operation can have independent configuration dialogs if needed.

Name, Description, Backend, and Mappings are fields that normally appear on every operation, other special fields can appear in non default configuration dialogs like the Remote System GUID field in SolMan Controller operations.

Normally there are two mapping configuration sections on each operation, one for the incoming data and another one for the outgoing data. You can choose different mapping types (backends) for each mapping direction, since their configuration is independent from each other and also independent from the operation backend. The normal and most common practice is that the operation uses 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.

The operation backend is pre-filled and is not editable. You will see this parameter when you choose the operation on the web service edit screen. The field is only informative.

In the left part of the screen on the action column you have the options: "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 "Delete" button to confirm the removal of the operation and it configuration or "Cancel" to close the delete dialog.

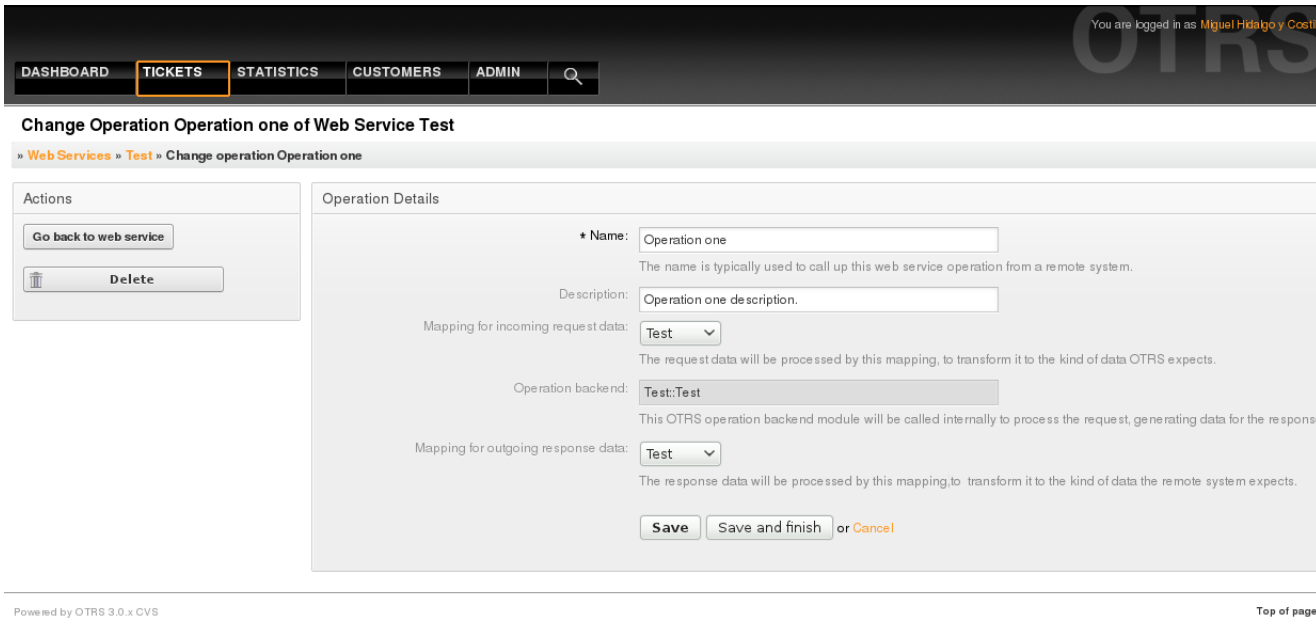

Figure: Web service operation.

Web Service Provider Transport

The network transport configuration for the requester is similar to the configuration for the provider. For the Requester "HTTP::SOAP" network transport there are more fields to be set.

Apart from the "Endpoint" (URI of the Remote System web service interface to accept requests) and "Namespace" which are required fields, you can also specify:

- Encoding (such as utf-8, latin1, iso-8859-1, cp1250, etc) for the SOAP message.
- SOAPAction Header: you can use this to send an empty or filled SOAPAction header. Set to "No" and the SOAPAction header on the SOAP message will be an empty string, or set to "Yes" to send the soap action in Namespace#Action format and define the separator (typically "/" for .Net web services and "#" for the rest).
- Authentication: to set the authentication mechanism, set to "-" to not use any authentication or select one from the list and the detail fields will appear.

Note

Currently only the "BasicAuth" (HTTP) authentication mechanism is implemented. You can decide whether or not to use it depending on the Remote System configuration. If used, you must provide the User Name and the Password to access the remote system.

Warning

If you supply a password for authentication and after you export the web service to a YAML file this password will be revealed and will be written into a plain text string inside the YAML file. Be aware of it and take precautions if needed.

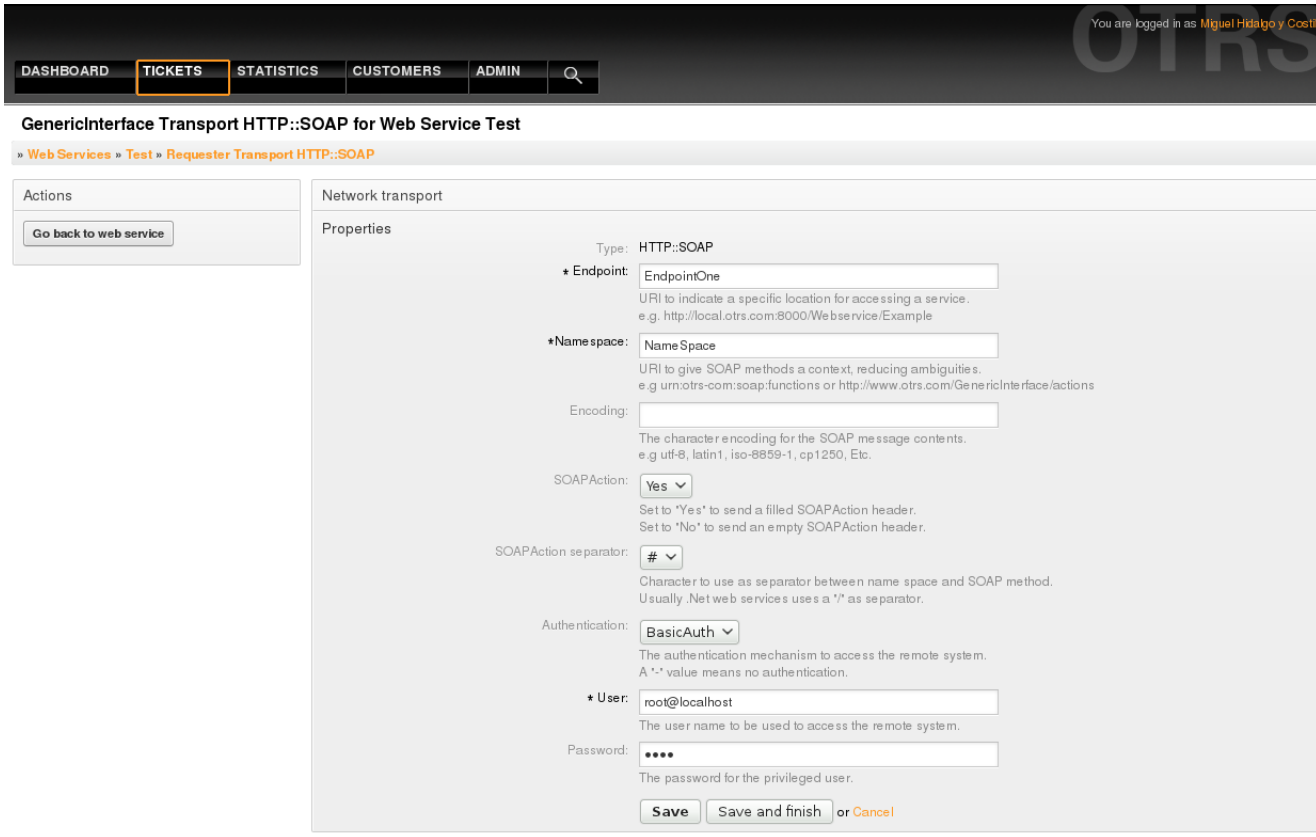

Figure: Web service requester network transport.

Web Service Invoker

The actions that can be performed when you are using OTRS as a requester are called "Invokers". Each invoker belongs to a controller (controllers are collections of operations or

invokers), normally invokers from the same controller need similar settings and share the same configuration dialogs. Each invoker can have independent configuration dialogs if needed.

Name, Description, Backend, and Mappings are fields that normally appear on every invoker, as well as the list of event triggers other special fields can appear on non default configuration dialogs like the Remote System GUID field in SolMan Controller invokers.

Normally there are two mapping configuration sections for each invoker, one for the incoming data and another one for the outgoing data. You can choose different mapping types (backends) for each mapping direction, since their configuration is independent from each other and also independent from the invoker backend. The normal and most common practice is that the invoker 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.

The invoker backend is pre-filled and is not editable. You will see this parameter when you choose the invoker on the web service edit screen. The field is only informative. informative.

Event triggers are events within OTRS such as "TicketCreate", "ArticleSend", etc. These can act as triggers to execute the invoker. Each invoker needs to have at least one event trigger registered, or the invoker will be useless, because it will never be called. The asynchronous property of the event triggers define if the OTRS process will handle the invoker or if it will be delegated to the Scheduler.

Note

The OTRS Scheduler is a separated process that executes tasks in the background. Using this the OTRS process itself will not be affected if the Remote System takes a long time to respond, if it is not available or if there are network problems. If you don't use the scheduler using web services can make OTRS slow or non-responsive. Therefore it is highly recommend to use asynchronous event triggers as often as possible.

To add an Event trigger first select the event family from the first list, then the event name from the second list, then set the asynchronous property (if unchecked means that the event trigger will not be asynchronous) and then click on the plus button. A new event trigger will be created and it will be listed on the invoker "Event Triggers" list.

To delete an Event trigger, simply locate the event trigger to be deleted in the "Event Triggers" list and click on the trash icon at the end of the row. This will open a dialog that ask you if you are sure to delete the event trigger. Click "Delete" to remove the event trigger from the list, or "Cancel" to close the dialog.

In the left part of the screen on the action column you have the options: "Go back to web service" (discarding all changes since the last save) and "Delete". If you click on the last one, a dialog will emerge and ask you if you like to remove the invoker. Click on the "Delete" button to confirm the removal of the invoker and its configuration or "Cancel" to close the delete dialog.

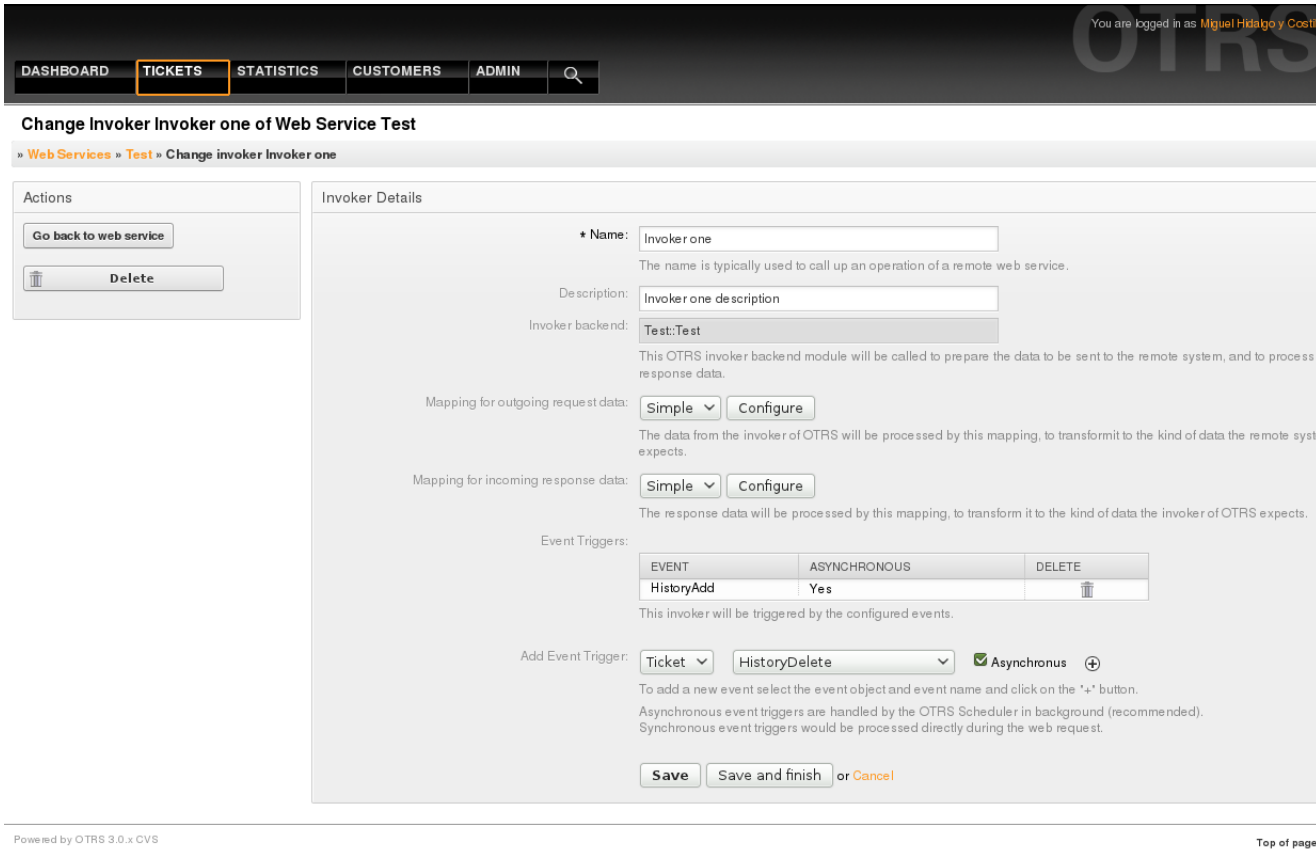

Figure: Web service invoker.

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 has to be changed, and sometimes even the names of the values (keys) or sometimes the complete structure, in order to match with the expected data on the other end. To accomplish this task the 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 shipped with OTRS), but it is not always necessary. The module Mapping::Simple should cover most of the mapping needs.

Note

When Mapping::Simple does not cover all mapping needs for a web service a new mapping module should be created. To learn more about how to create new mapping modules please consult the OTRS Development Manual.

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): doesn't 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 it 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.
- MapTo (use provided key or 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 "+" 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 "+" 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 "+" button will display a new row for a value map. Here also is 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 "-" button for each mapping value row.

Deleting the complete key mapping section (box) is possible, just push on the "-" 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 changes.

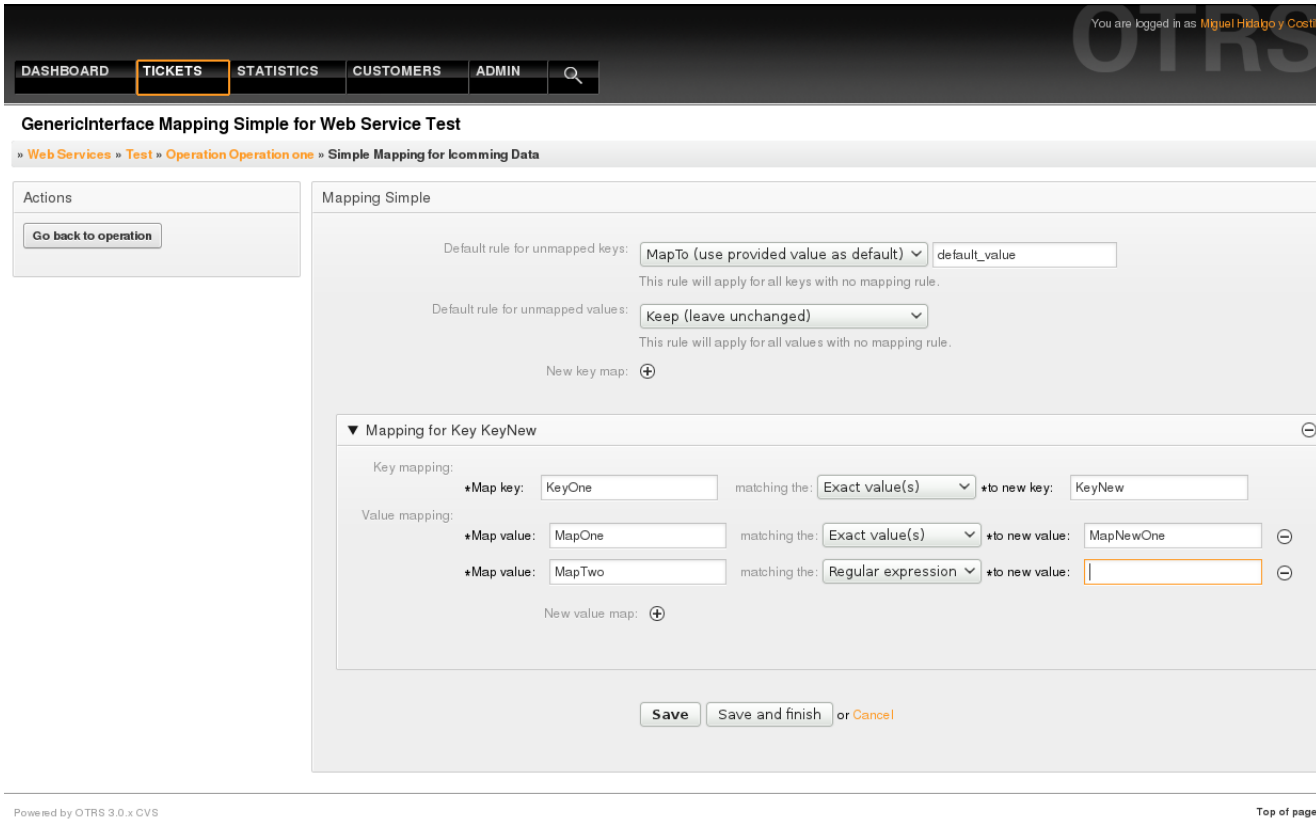

Figure: Web service mapping.

Web Service Command Line Interface

The Command Line Interface (CLI) is fast way to work with the web services. It consists of a set of tools can be use to perform basic operations like:

- Create, Update, Read, List and Delete web services based on YAML files.
- Read the Debugger log, with filter options.

Note

You don't need to use the CLI to work with web services. Integrated into the Admin interface there is a complete set of screens to interact with every part of the web services. Please read the web service GUI section included in this manual.

Web Service Configuration

The "WebserviceConfig.pl" was developed in order to create basic, but fast and powerful tool to work with web service configurations. It gives you the ability to perform the following actions:

- Add: to create web services using a YAML file as the configuration source.
- Update: to change an existing web service, the configuration can be changed using a different or modified YAML file.

- Read: to get the current web service configuration displayed on the screen.
- List: to get a complete list of all the web services registered in system.
- Delete: to delete a web service from the system. Be careful when you use it, because this action can't be undone.

Warning

A web service READ operation will display all the configuration as plain text on the screen, including any stored passwords. Please be aware of this and take the needed precautions!

Example: Creating a new web service configuration:

```
shell> OTRS_HOME/bin/otrs.WebserviceConfig.pl -a write
-n <webservice_name> -f /path/to/yaml/file
```

Also you can use 'otrs.WebserviceConfig.pl' with following options:

- **-a read -i <webservice_id>** - To read a stored configuration.
- **-a write -n <webservice_name> -f /path/to/yaml/file** - To create a new web service.
- **-a write -i <webservice_id> -f /path/to/yaml/file** - To update a web service.
- **-a list** - To list available web services.
- **-a delete -i <webservice_id>** - To delete a web service.

Web Service Debugger

Another available tool on the command line is the "otrs.GenericInterfaceDebugRead.pl" script, which is an interface to search for web service debugger log entries.

Example: Searching for debugger log entries:

```
shell> bin/otrs.GenericInterfaceDebugRead.pl
```

Optional parameters can be used for the "otrs.GenericInterfaceDebugRead.pl" script:

- **-c** - to filter by Communication ID (md5sum format).
- **-t** - to filter by CommunicationType ('Provider' or 'Requester').
- **-a** - to filter by date (At or After a date).
- **-b** - to filter by date (At or Before a date).
- **-i** - to filter by IP Address (must be valid IPv4 or IPv6 address).

- **-w** - to filter by Web Service ID.
- **-d** - to include detailed communication data.

Example: Searching for debugger log entries with all parameters:

```
shell> ./otrs.GenericInterfaceDebugRead.pl -c
a7cc4d9f5c70387a9bfbe1351bc88966 -t Provider -a '2011-07-22 00:00:00'
-b '2011-07-26 00:00:00' -i 127.0.0.1 -w 123 -d 1
```

Note

It is highly recommended to include at least one of the filter options listed above, and even more if the "-d" option is selected, because *a lot of* information can be retrieved from the data base and displayed on the screen, this could result in slow response times and much more information than what you really needed.

Web Service Configuration

From its design the web services were conceived to be portable from one OTRS system to another, e.g. from a test or development environment to a production system. Therefore it was needed to have an easy way to extract the web service configuration from the database, and import it to another. To accomplish this task Generic Interface uses YAML files as the web services configuration basis.

Why YAML? YAML is a markup language designed to be human friendly to read and write (it is easier to understand than JSON), it does not have some of the limitations of XML like numeric tags, it is open, standardized, and is complete enough to store the whole web service configuration.

Note

To learn more about YAML please visit http://www.yaml.org/.

The following is a web service configuration file example in YAML format:

```
---
Debugger:
  DebugThreshold: debug
Description: This an example of a web service configuration
Provider:
  Operation:
    CloseIncident:
      Description: This is a SolMan test operation
      MappingInbound: {}
      MappingOutbound: {}
      RemoteSystemGuid: ''
      Type: SolMan::CloseIncident
    Test:
      Description: This is a test operation
```

```
      MappingInbound:
        Config:
          KeyMapDefault:
            MapTo: ''
            MapType: Keep
          KeyMapExact:
            Prio: Priority
          ValueMap:
            Priority:
              ValueMapExact:
                Critical: 5 Very High
                Information: 1 Very Low
                Warning: 3 Normal
          ValueMapDefault:
            MapTo: 3 Normal
            MapType: MapTo
        Type: Simple
      MappingOutbound:
        Config:
          KeyMapDefault:
            MapTo: ''
            MapType: Ignore
          KeyMapExact:
            Priority: Prio
          ValueMap:
            Prio:
              ValueMapExact:
                1 Very Low: Information
                3 Normal: Warning
                5 Very High: Critical
          ValueMapDefault:
            MapTo: ''
            MapType: Ignore
        Type: Simple
      Type: Test::Test
  Transport:
    Config:
      MaxLength: 10000000
      NameSpace: http://www.example.com/actions
    Type: HTTP::SOAP
RemoteSystem: remote.system.description.example.com
Requester:
  Invoker:
    Test:
      Description: This is a test invoker
      Events:
        - Asynchronous: 1
          Event: TicketCreate
        - Asynchronous: 0
          Event: ArticleUpdate
      MappingInbound:
        Type: Simple
      MappingOutbound:
        Type: Simple
```

```
    Type: Test::Test
Transport:
  Config:
    Authentication:
      Password: '*******'
      Type: BasicAuth
      User: otrs
    Encoding: utf-8
    Endpoint: http://www.example.com:8080/endpoint
    NameSpace: http://www.example.com/actions
    SOAPAction: Yes
    SOAPActionSeparator: '#'
  Type: HTTP::SOAP
```

Configuration Details

General

- Description: a short text that describes the web service.
- RemoteSystem: a short description of the Remote System.
- Debugger: a container for the debugger settings.
- Provider: a container for the provider settings.
- Requester: a container for the requester settings.

Debugger

- DebugThreshold: the debugger level

Possible Values

- debug: all logs are stored in the database.
- info: info, notice and error level logs are stored in the database.
- notice: notice and error level logs are stored in the database.
- error: only error level logs are stored in the database.

Provider

- Operation: a container for each operation settings.
- Transport: a container for provider network transport settings.

Operation

- <OperationName>: Unique name for the operation, container for its own operation settings (cardinality 0..n, but not duplicate).

<OperationName>

This section is based on operations from type "Test::Test" other operations might contain more or different settings.

- Description: a short text that describes the operation.
- MappingInbound: a container for the mapping settings for the incoming request data.
- MappingOutbound: a container for the mapping settings for the outgoing response data.
- Type: the operation backend, in Controller::Operation format.

MappingInbound

This section is based on mappings from type "Simple". Other mappings might contain more or different settings.

- Config: a container for this mapping settings.
- Type: the mapping backend.

Config

- KeyMapDefault: a container for all non mapped keys settings.
- ValueMapDefault: a container for all non mapped values settings.
- KeyMapExact: a container for all exact key mappings (cardinality 0 .. 1).
- KeyMapRegEx: a container for all regular expression key mappings (cardinality 0 .. 1).
- ValueMap: a container for all value mappings (cardinality 0 .. 1).

KeyMapDefault

- MapTo: the new value to be used (only applicable if MapType is set to MapTo).
- MapType: the rule for the mapping.

Possible Values

- Keep: leave unchanged.
- Ignore: drop.
- MapTo: change to the MapTo value.

ValueMapDefault

Similar to KeyMapDefault.

KeyMapExact

- <oldkey>: <newkey> (cardinality 0 .. n but not duplicate).

KeyMapRegEx

- <oldkey(RegEx)>: <newkey> (cardinality 0 .. n but no duplicates).

ValueMap

- <newkey>: a container for value mappings for this new key (cardinality depends on the new keys from KeyMapExact and KeyMapRegEx).

<newkey>

- ValueMapExact: a container for all exact value mappings (cardinality 0 .. 1).
- ValueMapRegEx: a container for all regular expression value mappings (cardinality 0 .. 1).

valueMapExact

- <oldvalue>: <newvalue> (cardinality 0 .. n but not duplicate).

ValueMapRegEx

- <oldvalue(RegEx)>: <newvalue> (cardinality 0 .. n but not duplicate).

MappingOutbound

Same as MappingInbound.

Transport

This section is based on the provider network transport HTTP::SOAP, other transports might contain more or different settings.

- Config: a container for the specific network transport configuration settings.
- Type: the provider network transport backend.

Config

- MaxLength: the maximum length in bytes to be read in a SOAP message by OTRS.
- NameSpace: an URI that gives a context to all operations that belongs to this web service.

Requester

- Invoker: a container for each invokers' settings.
- Transport: a container for requester network transport settings.

Invoker

- <InvokerName>: Unique name for the invoker, container for its own invoker settings (cardinality 0..n, but not duplicate).

<InvokerName>

This section is based on invokers from type "Test::Test" other invokers might contain more or different settings.

- Description: a short text that describes the invoker

- Events: a container for a unnamed list of event trigger settings.
- MappingInbound: a container for the mapping settings for the incoming response data.
- MappingOutbound: a container for the mapping settings for the outgoing request data.
- Type: the invoker backend, in Controller::Invoker format.

Events

- *List Element*: (cardinality 0 .. n)
  - Asynchronous: to set if the invoker execution will be delegated to the Scheduler

    ### Possible Values

    - 0: not handled by the Scheduler.
    - 1: handled by the Scheduler.

  - Event: the name of the event trigger.

    ### Possible Values (for ticket events)

    - TicketCreate
    - TicketDelete
    - TicketTitleUpdate
    - TicketUnlockTimeoutUpdate
    - TicketQueueUpdate
    - TicketTypeUpdate
    - TicketServiceUpdate
    - TicketSLAUpdate
    - TicketCustomerUpdate
    - TicketFreeTextUpdate
    - TicketFreeTimeUpdate
    - TicketPendingTimeUpdate
    - TicketLockUpdate
    - TicketArchiveFlagUpdate
    - TicketStateUpdate
    - TicketOwnerUpdate
    - TicketResponsibleUpdate

- TicketPriorityUpdate
- HistoryAdd
- HistoryDelete
- TicketAccountTime
- TicketMerge
- TicketSubscribe
- TicketUnsubscribe
- TicketFlagSet
- TicketFlagDelete
- TicketSlaveLinkAdd
- TicketSlaveLinkDelete
- TicketMasterLinkDelete

Possible Values (for article events)

- Article Events
- ArticleCreate
- ArticleFreeTextUpdate
- ArticleUpdate
- ArticleSend
- ArticleBounce
- ArticleAgentNotification
- ArticleCustomerNotification
- ArticleAutoResponse
- ArticleFlagSet
- ArticleFlagDelete
- ArticleAgentNotification
- ArticleCustomerNotification

MappingInbound

Same as Operation MappingInbound

MappingOutbound

Same as Operation MappingInbound.

Transport

This section is based on the requester network transport HTTP::SOAP, other transports might contain more or different settings.

- Config: a container for the specific network transport configuration settings.
- Type: the requester network transport backend.

Config

- Authentication: a container for authentication settings.
- Encoding: the SOAP Message request encoding
- Endpoint: the URI of the Remote Server web service to accept OTRS requests
- NameSpace: an URI that gives a context to all invokers that belongs to this web service.
- SOAPAction: to send an empty or filled SOAPAction header in the SOAP Message (in "<NameSpace> <Separator> <Action>" format).

  ### Possible Values

  - YES: to send a filled SOAPAction header.
  - No: to send an empty SOAPAction header.
- SOAPActionSeparator: to set the <Separator> of a filled SOAPAction header.

  ### Possible Values

  - '/': used for .net web services.
  - '#': used for all the rest web services.

Authentication

- User: the privileged user name that has access to the remote web service.
- Password: the password for privileged user in plain text.
- Type: the type of authentication.

Connectors

A Connector is in essence set of actions called Operations if OTRS acts as a web service provider or Invokers if OTRS acts as a web service requester. But it can also include special Mappings or Transports

One Connector can have only Operations, Only Invokers or both. A connector can even use parts of other connectors like the Mappings or Transports if they are not so specific for the Connector that implements them.

In another words a Connector is not limited to just the Controller layer but it can be extended to Data Mapping or Network Transport layers if needed.

Due to the modular design of the Generic Interface a Connector can be seen as a plug-in; this means that by adding Connectors the capabilities of the generic interface can be extended using: OTRS Feature add ons, OTRS Custom modules, 3rd Party modules, and so on.

Bundled Connectors

Included with this version of OTRS the following connectors are ready to be used.

- Session
- Ticket

Session Connector

This connector is capable to create a valid SessionID that can be used in any other operation.

Provides:

- Operations:
  - SessionCreate

Operations

SessionCreate

Creates a new new valid SessionID to be used in other operations from other connectors like TicketCreate.

> **Note**
>
> To use the SessionID in other operations from other connectors is necessary that the operation implements authentication by SessionID. all the rest of the bundled operations are capable to accept a valid SessionID as an authentication method.

Possible Attributes:

```
     <SessionCreate>
        <!--You have a MANDATORY CHOICE of the next 2 items at this
level-->
        <!--Optional:-->
        <UserLogin>?</UserLogin>
        <!--Optional:-->
        <CustomerUserLogin>?</CustomerUserLogin>
        <!--Optional:-->
        <Password>?</Password>
     </SessionCreate>
```

Ticket Connector

This connector supplies the basic functionality to interact with tickets

Provides:

- Operations:
  - TicketCreate
  - TicketUpdate
  - TicketGet
  - TicketSearch

Operations

TicketCreate

Provides an interface to create ticket in OTRS, a ticket must contain an Article and can contain several attachments, all defined Dynamic Fields can be also set on TicketCreate operation.

Possible Attributes:

```
     <TicketCreate>
        <!--You have a MANDATORY CHOICE of the next 3 items at this
level-->
        <!--Optional:-->
        <UserLogin>?</UserLogin>
        <!--Optional:-->
        <CustomerUserLogin>?</CustomerUserLogin>
        <!--Optional:-->
        <SessionID>?</SessionID>
        <!--Optional:-->
        <Password>?</Password>
        <Ticket>
           <Title>?</Title>
           <!--You have a MANDATORY CHOICE of the next 2 items at
this level-->
           <!--Optional:-->
           <QueueID>?</QueueID>
           <!--Optional:-->
           <Queue>?</Queue>
           <!--You have a CHOICE of the next 2 items at this level-->
           <!--Optional:-->
           <TypeID>?</TypeID>
           <!--Optional:-->
           <Type>?</Type>
           <!--You have a CHOICE of the next 2 items at this level-->
           <!--Optional:-->
           <ServiceID>?</ServiceID>
           <!--Optional:-->
```

```
            <Service>?</Service>
            <!--You have a CHOICE of the next 2 items at this level-->
            <!--Optional:-->
            <SLAID>?</SLAID>
            <!--Optional:-->
            <SLA>?</SLA>
            <!--You have a MANDATORY CHOICE of the next 2 items at
this level-->
            <!--Optional:-->
            <StateID>?</StateID>
            <!--Optional:-->
            <State>?</State>
            <!--You have a MANDATORY CHOICE of the next 2 items at
this level-->
            <!--Optional:-->
            <PriorityID>?</PriorityID>
            <!--Optional:-->
            <Priority>?</Priority>
            <!--You have a CHOICE of the next 2 items at this level-->
            <!--Optional:-->
            <OwnerID>?</OwnerID>
            <!--Optional:-->
            <Owner>?</Owner>
            <!--You have a CHOICE of the next 2 items at this level-->
            <!--Optional:-->
            <ResponsibleID>?</ResponsibleID>
            <!--Optional:-->
            <Responsible>?</Responsible>
            <CustomerUser>?</CustomerUser>
            <!--Optional:-->
            <PendingTime>
               <Year>?</Year>
               <Month>?</Month>
               <Day>?</Day>
               <Hour>?</Hour>
               <Minute>?</Minute>
            </PendingTime>
         </Ticket>
         <Article>
            <!--You have a CHOICE of the next 2 items at this level-->
            <!--Optional:-->
            <ArticleTypeID>?</ArticleTypeID>
            <!--Optional:-->
            <ArticleType>?</ArticleType>
            <!--You have a CHOICE of the next 2 items at this level-->
            <!--Optional:-->
            <SenderTypeID>?</SenderTypeID>
            <!--Optional:-->
            <SenderType>?</SenderType>
            <!--Optional:-->
            <From>?</From>
            <Subject>?</Subject>
            <Body>?</Body>
            <!--You have a CHOICE of the next 2 items at this level-->
```

```
            <!--Optional:-->
            <ContentType>?</ContentType>
            <Charset>?</Charset>
            <MimeType>?</MimeType>
            <!--Optional:-->
            <HistoryType>?</HistoryType>
            <!--Optional:-->
            <HistoryComment>?</HistoryComment>
            <!--Optional:-->
            <AutoResponseType>?</AutoResponseType>
            <!--Optional:-->
            <TimeUnit>?</TimeUnit>
            <!--Optional:-->
            <NoAgentNotify>?</NoAgentNotify>
            <!--Zero or more repetitions:-->
            <ForceNotificationToUserID>?</ForceNotificationToUserID>
            <!--Zero or more repetitions:-->
            <ExcludeNotificationToUserID>?</
ExcludeNotificationToUserID>
            <!--Zero or more repetitions:-->
            <ExcludeMuteNotificationToUserID>?</
ExcludeMuteNotificationToUserID>
         </Article>
         <!--Zero or more repetitions:-->
         <DynamicField>
            <Name>?</Name>
            <!--1 or more repetitions:-->
            <Value>?</Value>
         </DynamicField>
         <!--Zero or more repetitions:-->
         <Attachment>
            <Content>cid:61886944659</Content>
            <ContentType>?</ContentType>
            <Filename>?</Filename>
         </Attachment>
      </TicketCreate>
```

TicketUpdate

TicketUpdate operation add the capability to modify attributes from a ticket or add a new article, including attachments and all defined dynamic fields for the ticket and the new article.

Note

It is not necessary to create a new article to modify a ticket attribute.

Possible Attributes:

```
      <TicketUpdate>
         <!--You have a MANDATORY CHOICE of the next 3 items at this
 level-->
```

```
<!--Optional:-->
<UserLogin>?</UserLogin>
<!--Optional:-->
<CustomerUserLogin>?</CustomerUserLogin>
<!--Optional:-->
<SessionID>?</SessionID>
<!--Optional:-->
<Password>?</Password>
<!--You have a CHOICE of the next 2 items at this level-->
<TicketID>?</TicketID>
<TicketNumber>?</TicketNumber>
<!--Optional:-->
<Ticket>
   <!--Optional:-->
   <Title>?</Title>
   <!--You have a CHOICE of the next 2 items at this level-->
   <!--Optional:-->
   <QueueID>?</QueueID>
   <!--Optional:-->
   <Queue>?</Queue>
   <!--You have a CHOICE of the next 2 items at this level-->
   <!--Optional:-->
   <TypeID>?</TypeID>
   <!--Optional:-->
   <Type>?</Type>
   <!--You have a CHOICE of the next 2 items at this level-->
   <!--Optional:-->
   <ServiceID>?</ServiceID>
   <!--Optional:-->
   <Service>?</Service>
   <!--You have a CHOICE of the next 2 items at this level-->
   <!--Optional:-->
   <SLAID>?</SLAID>
   <!--Optional:-->
   <SLA>?</SLA>
   <!--You have a CHOICE of the next 2 items at this level-->
   <!--Optional:-->
   <StateID>?</StateID>
   <!--Optional:-->
   <State>?</State>
   <!--You have a CHOICE of the next 2 items at this level-->
   <!--Optional:-->
   <PriorityID>?</PriorityID>
   <!--Optional:-->
   <Priority>?</Priority>
   <!--You have a CHOICE of the next 2 items at this level-->
   <!--Optional:-->
   <OwnerID>?</OwnerID>
   <!--Optional:-->
   <Owner>?</Owner>
   <!--You have a CHOICE of the next 2 items at this level-->
   <!--Optional:-->
   <ResponsibleID>?</ResponsibleID>
   <!--Optional:-->
```

```
            <Responsible>?</Responsible>
            <!--Optional:-->
            <CustomerUser>?</CustomerUser>
            <!--Optional:-->
            <PendingTime>
               <Year>?</Year>
               <Month>?</Month>
               <Day>?</Day>
               <Hour>?</Hour>
               <Minute>?</Minute>
            </PendingTime>
         </Ticket>
         <!--Optional:-->
         <Article>
            <!--You have a CHOICE of the next 2 items at this level-->
            <!--Optional:-->
            <ArticleTypeID>?</ArticleTypeID>
            <!--Optional:-->
            <ArticleType>?</ArticleType>
            <!--You have a CHOICE of the next 2 items at this level-->
            <!--Optional:-->
            <SenderTypeID>?</SenderTypeID>
            <!--Optional:-->
            <SenderType>?</SenderType>
            <!--Optional:-->
            <From>?</From>
            <Subject>?</Subject>
            <Body>?</Body>
            <!--You have a CHOICE of the next 2 items at this level-->
            <!--Optional:-->
            <ContentType>?</ContentType>
            <Charset>?</Charset>
            <MimeType>?</MimeType>
            <!--Optional:-->
            <HistoryType>?</HistoryType>
            <!--Optional:-->
            <HistoryComment>?</HistoryComment>
            <!--Optional:-->
            <AutoResponseType>?</AutoResponseType>
            <!--Optional:-->
            <TimeUnit>?</TimeUnit>
            <!--Optional:-->
            <NoAgentNotify>?</NoAgentNotify>
            <!--Zero or more repetitions:-->
            <ForceNotificationToUserID>?</ForceNotificationToUserID>
            <!--Zero or more repetitions:-->
            <ExcludeNotificationToUserID>?</
ExcludeNotificationToUserID>
            <!--Zero or more repetitions:-->
            <ExcludeMuteNotificationToUserID>?</
ExcludeMuteNotificationToUserID>
         </Article>
         <!--Zero or more repetitions:-->
         <DynamicField>
```

```
            <Name>?</Name>
            <!--1 or more repetitions:-->
            <Value>?</Value>
         </DynamicField>
         <!--Zero or more repetitions:-->
         <Attachment>
            <Content>cid:166861569966</Content>
            <ContentType>?</ContentType>
            <Filename>?</Filename>
         </Attachment>
      </TicketUpdate>
```

TicketGet

This operation is used to get all the attributes of a ticket including the dynamic fields, all the articles and all the attachments that belongs to the ticket.

Possible Attributes:

```
      <TicketGet>
         <!--You have a MANDATORY CHOICE of the next 3 items at this
 level-->
         <!--Optional:-->
         <UserLogin>?</UserLogin>
         <!--Optional:-->
         <CustomerUserLogin>?</CustomerUserLogin>
         <!--Optional:-->
         <SessionID>?</SessionID>
         <!--Optional:-->
         <Password>?</Password>
         <!--Optional:-->
         <TicketID>?</TicketID>
         <!--Optional:-->
         <DynamicFields>?</DynamicFields>
         <!--Optional:-->
         <Extended>?</Extended>
         <!--Optional:-->
         <AllArticles>?</AllArticles>
         <!--Optional:-->
         <DynamicFields>?</DynamicFields>
         <!--Optional:-->
         <ArticleSenderType>?</ArticleSenderType>
         <!--Optional:-->
         <ArticleOrder>?</ArticleOrder>
         <!--Optional:-->
         <ArticleLimit>?</ArticleLimit>
         <!--Optional:-->
         <Attachments>?</Attachments>
      </TicketGet>
```

TicketSearch

TicketSearch operation returns a list of Ticket IDs that matches a predefined criteria.

Possible Attributes:

```
     <TicketSearch>
        <!--You have a MANDATORY CHOICE of the next 3 items at this
level-->
        <!--Optional:-->
        <UserLogin>?</UserLogin>
        <!--Optional:-->
        <CustomerUserLogin>?</CustomerUserLogin>
        <!--Optional:-->
        <SessionID>?</SessionID>
        <!--Optional:-->
        <Password>?</Password>
        <!--Optional:-->
        <Limit>?</Limit>
        <!--Zero or more repetitions:-->
        <TicketNumber>?</TicketNumber>
        <!--Zero or more repetitions:-->
        <Title>?</Title>
        <!--Zero or more repetitions:-->
        <Queues>?</Queues>
        <!--Zero or more repetitions:-->
        <QueueIDs>?</QueueIDs>
        <!--Optional:-->
        <UseSubQueues>?</UseSubQueues>
        <!--Zero or more repetitions:-->
        <Types>?</Types>
        <!--Zero or more repetitions:-->
        <TypeIDs>?</TypeIDs>
        <!--Zero or more repetitions:-->
        <States>?</States>
        <!--Zero or more repetitions:-->
        <StateIDs>?</StateIDs>
        <!--Zero or more repetitions:-->
        <StateType>?</StateType>
        <!--Zero or more repetitions:-->
        <StateTpeyIDs>?</StateTpeyIDs>
        <!--Zero or more repetitions:-->
        <Priorities>?</Priorities>
        <!--Zero or more repetitions:-->
        <PriorityIDs>?</PriorityIDs>
        <!--Zero or more repetitions:-->
        <Services>?</Services>
        <!--Zero or more repetitions:-->
        <ServiceIDs>?</ServiceIDs>
        <!--Zero or more repetitions:-->
        <SLA>?</SLA>
        <!--Zero or more repetitions:-->
```

```
         <SLAIDs>?</SLAIDs>
         <!--Zero or more repetitions:-->
         <Locks>?</Locks>
         <!--Zero or more repetitions:-->
         <LockIDs>?</LockIDs>
         <!--Zero or more repetitions:-->
         <OwnerIDs>?</OwnerIDs>
         <!--Zero or more repetitions:-->
         <ResponsibleIDs>?</ResponsibleIDs>
         <!--Zero or more repetitions:-->
         <WatchUserIDs>?</WatchUserIDs>
         <!--Zero or more repetitions:-->
         <CustomerID>?</CustomerID>
         <!--Zero or more repetitions:-->
         <CustomerUserLogin>?</CustomerUserLogin>
         <!--Zero or more repetitions:-->
         <CreatedUserIDs>?</CreatedUserIDs>
         <!--Zero or more repetitions:-->
         <CreatedTypes>?</CreatedTypes>
         <!--Zero or more repetitions:-->
         <CreatedTypeIDs>?</CreatedTypeIDs>
         <!--Zero or more repetitions:-->
         <CreatedPriorities>?</CreatedPriorities>
         <!--Zero or more repetitions:-->
         <CreatedPriorityIDs>?</CreatedPriorityIDs>
         <!--Zero or more repetitions:-->
         <CreatedStates>?</CreatedStates>
         <!--Zero or more repetitions:-->
         <CreatedStateIDs>?</CreatedStateIDs>
         <!--Zero or more repetitions:-->
         <CreatedQueues>?</CreatedQueues>
         <!--Zero or more repetitions:-->
         <CreatedStateIDs>?</CreatedStateIDs>
         <!--Zero or more repetitions:-->
         <DynamicFields>
            <!--You have a MANDATORY CHOICE of the next 6 items at
this level-->
            <!--Optional:-->
            <Equals>?</Equals>
            <!--Optional:-->
            <Like>?</Like>
            <!--Optional:-->
            <GreaterThan>?</GreaterThan>
            <!--Optional:-->
            <GreaterThanEquals>?</GreaterThanEquals>
            <!--Optional:-->
            <LowerThan>?</LowerThan>
            <!--Optional:-->
            <LowerThanEquals>?</LowerThanEquals>
         </DynamicFields>
         <!--Optional:-->
         <Ticketflag>
            <!--Optional:-->
            <Seen>?</Seen>
```

```
            </Ticketflag>
            <!--Optional:-->
            <From>?</From>
            <!--Optional:-->
            <To>?</To>
            <!--Optional:-->
            <Cc>?</Cc>
            <!--Optional:-->
            <Subject>?</Subject>
            <!--Optional:-->
            <Body>?</Body>
            <!--Optional:-->
            <FullTextIndex>?</FullTextIndex>
            <!--Optional:-->
            <ContentSearch>?</ContentSearch>
            <!--Optional:-->
            <ContentSearchPrefix>?</ContentSearchPrefix>
            <!--Optional:-->
            <ContentSearchSuffix>?</ContentSearchSuffix>
            <!--Optional:-->
            <ConditionInline>?</ConditionInline>
            <!--Optional:-->
            <ArticleCreateTimeOlderMinutes>?</
ArticleCreateTimeOlderMinutes>
            <!--Optional:-->
            <ArticleCreateTimeNewerMinutes>?</
ArticleCreateTimeNewerMinutes>
            <!--Optional:-->
            <ArticleCreateTimeNewerDate>?</ArticleCreateTimeNewerDate>
            <!--Optional:-->
            <ArticleCreateTimeOlderDate>?</ArticleCreateTimeOlderDate>
            <!--Optional:-->
            <TicketCreateTimeOlderMinutes>?</
TicketCreateTimeOlderMinutes>
            <!--Optional:-->
            <ATicketCreateTimeNewerMinutes>?</
ATicketCreateTimeNewerMinutes>
            <!--Optional:-->
            <TicketCreateTimeNewerDate>?</TicketCreateTimeNewerDate>
            <!--Optional:-->
            <TicketCreateTimeOlderDate>?</TicketCreateTimeOlderDate>
            <!--Optional:-->
            <TicketChangeTimeOlderMinutes>?</
TicketChangeTimeOlderMinutes>
            <!--Optional:-->
            <TicketChangeTimeNewerMinutes>?</
TicketChangeTimeNewerMinutes>
            <!--Optional:-->
            <TicketChangeTimeNewerDate>?</TicketChangeTimeNewerDate>
            <!--Optional:-->
            <TicketChangeTimeOlderDate>?</TicketChangeTimeOlderDate>
            <!--Optional:-->
            <TicketCloseTimeOlderMinutes>?</TicketCloseTimeOlderMinutes>
            <!--Optional:-->
```

```
            <TicketCloseTimeNewerMinutes>?</TicketCloseTimeNewerMinutes>
            <!--Optional:-->
            <TicketCloseTimeNewerDate>?</TicketCloseTimeNewerDate>
            <!--Optional:-->
            <TicketCloseTimeOlderDate>?</TicketCloseTimeOlderDate>
            <!--Optional:-->
            <TicketPendingTimeOlderMinutes>?</
TicketPendingTimeOlderMinutes>
            <!--Optional:-->
            <TicketPendingTimeNewerMinutes>?</
TicketPendingTimeNewerMinutes>
            <!--Optional:-->
            <TicketPendingTimeNewerDate>?</TicketPendingTimeNewerDate>
            <!--Optional:-->
            <TicketPendingTimeOlderDate>?</TicketPendingTimeOlderDate>
            <!--Optional:-->
            <TicketEscalationTimeOlderMinutes>?</
TicketEscalationTimeOlderMinutes>
            <!--Optional:-->
            <TTicketEscalationTimeNewerMinutes>?</
TTicketEscalationTimeNewerMinutes>
            <!--Optional:-->
            <TicketEscalationTimeNewerDate>?</
TicketEscalationTimeNewerDate>
            <!--Optional:-->
            <TicketEscalationTimeOlderDate>?</
TicketEscalationTimeOlderDate>
            <!--Optional:-->
            <ArchiveFlags>?</ArchiveFlags>
            <!--Zero or more repetitions:-->
            <OrderBy>?</OrderBy>
            <!--Zero or more repetitions:-->
            <SortBy>?</SortBy>
            <!--Optional:-->
            <Permission>?</Permission>
            <!--Zero or more repetitions:-->
            <CustomerUserID>?</CustomerUserID>
         </TicketSearch>
```

Examples:

Web Service Configuration

The following is a basic but complete web service configuration file in YAML format to use all the Ticket Connector operations, in order to use it in OTRS you need to copy the content, save it into a file called GenericTicketConnector.yml, and import it into OTRS in the Web Services screen in the Admin panel by clicking in the "Add web service" action from the overview screen and then clicking in the "Import web service" action in the add screen.

```
---
Debugger:
  DebugThreshold: debug
  TestMode: 0
Description: Ticket Connector Sample
FrameworkVersion: 3.1.x CVS
Provider:
  Operation:
    SessionCreate:
      Description: Creates a Session
      MappingInbound: {}
      MappingOutbound: {}
      Type: Session::SessionCreate
    TicketCreate:
      Description: Creates a Ticket
      MappingInbound: {}
      MappingOutbound: {}
      Type: Ticket::TicketCreate
    TicketUpdate:
      Description: Updates a Ticket
      MappingInbound: {}
      MappingOutbound: {}
      Type: Ticket::TicketUpdate
    TicketGet:
      Description: Retrieve Ticket data
      MappingInbound: {}
      MappingOutbound: {}
      Type: Ticket::TicketGet
    TicketSearch:
      Description: Search for Tickets
      MappingInbound: {}
      MappingOutbound: {}
      Type: Ticket::TicketSearch
  Transport:
    Config:
      MaxLength: 100000000
      NameSpace: http://www.otrs.org/TicketConnector/
    Type: HTTP::SOAP
RemoteSystem: ''
Requester:
  Transport:
    Type: ''
```

Perl SOAP Requester

The following code is a Perl script that can connect to OTRS via the generic interface, to perform the operations provided by the Ticket Connector, it uses two Perl CPAN modules SOAP::Lite and Data::Dumper, be sure that your environment is capable to use that modules before you try to run the scipt.

```
#!/usr/bin/perl -w
# --
# otrs.SOAPRequest.pl - sample to send a SOAP request to OTRS Generic
 Interface Ticket Connector
# Copyright (C) 2001-2012 OTRS AG, http://otrs.org/
# --
# $Id: genericinterface-connectors.xml,v 1.6 2012/02/10 16:29:43 cr
 Exp $
# --
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU AFFERO General Public License as
 published by
# the Free Software Foundation; either version 3 of the License, or
# any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU Affero General Public
 License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307
  USA
# or see http://www.gnu.org/licenses/agpl.txt.
# --

use strict;
use warnings;

# use ../ as lib location
use File::Basename;
use FindBin qw($RealBin);
use lib dirname($RealBin);

use SOAP::Lite;
use Data::Dumper;

# ---
# Variables to be defined.

# this is the URL for the web service
# the format is
# <HTTP_TYPE>:://<OTRS_FQDN>/nph-genericinterface.pl/Webservice/
<WEB_SERVICE_NAME>
# or
# <HTTP_TYPE>:://<OTRS_FQDN>/nph-genericinterface.pl/WebserviceID/
<WEB_SERVICE_ID>
my $URL = 'http://localhost/otrs/nph-genericinterface.pl/Webservice/
GenericTicketConnector';

# this name space should match the specified name space in the SOAP
 transport for the web service.
```

```
my $NameSpace = 'http://www.otrs.org/TicketConnector/';

# this is operation to execute, it could be TicketCreate,
 TicketUpdate, TicketGet, TicketSearch
# or SessionCreate. and they must to be defined in the web service.
my $Operation = 'TicketCreate';

# this variable is used to store all the parameters to be included on
 a request in XML format, each
# operation has a determined set of mandatory and non mandatory
 parameters to work correctly, please
# check OTRS Admin Manual in order to get the complete list.
my $XMLData = '
<UserLogin>some user login</UserLogin>
<Password>some password</Password>
<Ticket>
    <Title>some title</Title>
    <CustomerUser>some customer user login</CustomerUser>
    <Queue>some queue</Queue>
    <State>some state</State>
    <Priority>some priority</Priority>
</Ticket>
<Article>
    <Subject>some subject</Subject>
    <Body>some body</Body>
    <ContentType>text/plain; charset=utf8</ContentType>
</Article>
';

# ---

# create a SOAP::Lite data structure from the provided XML data
 structure.
my $SOAPData = SOAP::Data
    ->type( 'xml' => $XMLData );

my $SOAPObject = SOAP::Lite
    ->uri($NameSpace)
    ->proxy($URL)
    ->$Operation($SOAPData);

# check for a fault in the soap code.
if ( $SOAPObject->fault ) {
    print $SOAPObject->faultcode, " ", $SOAPObject->faultstring, "\n";
}

# otherwise print the results.
else {

    # get the XML response part from the SOAP message.
    my $XMLResponse = $SOAPObject->context()->transport()->proxy()-
>http_response()->content();

    # deserialize response (convert it into a perl structure).
```

```
    my $Deserialized = eval {
        SOAP::Deserializer->deserialize($XMLResponse);
    };

    # remove all the headers and other not needed parts of the SOAP
 message.
    my $Body = $Deserialized->body();

    # just output relevant data and no the operation name key (like
 TicketCreateResponse).
    for my $ResponseKey ( keys %{$Body} ) {
        print Dumper( $Body->{$ResponseKey} );
    }
}
```

Chapter 21. OTRS Scheduler

The OTRS Scheduler is an independent system process that executes tasks in background. These kind of processes are know as *daemons* in Unix / Linux systems or as *services* on Windows environments. It is independent but that doesn't mean that the Scheduler does everything alone, it is fully integrated into OTRS and can use any OTRS module as needed to complete each task.

Currently the OTRS Scheduler is only able to handle Generic Interface tasks. These kind of tasks executes invokers that send requests to remote systems. Other handlers for different tasks will be added in future OTRS versions.

For sanity reasons the Scheduler process needs to be restarted from time to time. This is done automatically by the scheduler process itself once a day, but it can be adjusted as needed using the SysConfig by editing the "Scheduler::RestartAfterSeconds" setting.

The OTRS Scheduler is a set it and forget it process, the only needed human interaction is to check its status periodically and start or stop it as needed.

Note

If the Scheduler is stopped for any reason all pending tasks and new tasks registered when the Scheduler is stopped will be executed as soon as the Scheduler starts again (unless the tasks are set to be executed in the future).

Scheduler Graphical Interface

The Scheduler is not visible in the OTRS Graphical User Interface unless it stops running.

Scheduler Not Running Notification

There are two different types of notifications if the system detects that scheduler is not running. This detection is based on the Scheduler process update frequency, if the difference between current time and the last process update time is 2 times the process update frequency a warning message will be displayed in the OTRS notification area. If it is over 4 times the process frequency then an alert will be displayed instead.

The Scheduler process update time can be configured via the SysConfig in the "Scheduler::PIDUpdateTime" setting.

If you would see a warning message it is not always necessary to take an action, but is highly recommended to check if the scheduler process is running. If you see an alert, then is highly probable that the scheduler is in fact not running and should be started.

By default the Scheduler not running notification is enabled, if there is a valid web service registered in the database, and is only displayed to the users in the "admin" group.

To disable the notification (not recommended) or to change or add the notification groups, please edit the "Frontend::NotifyModule###800-Scheduler-Check" setting in the SysConfig.

Figure: Scheduler notification.

Start Scheduler

By clicking on the Scheduler not running notification link (either warning or alert) a dialog box will open to let you start the Scheduler process again. The Scheduler can be started normally or forced to start, by clicking on the appropriate check box in the dialog.

Note

A forced Scheduler start is only necessary if previous Scheduler process was terminated abnormally and the Process ID is still registered in the database.

To have full control of the Scheduler process and to check it real status please use the command line tools described below.

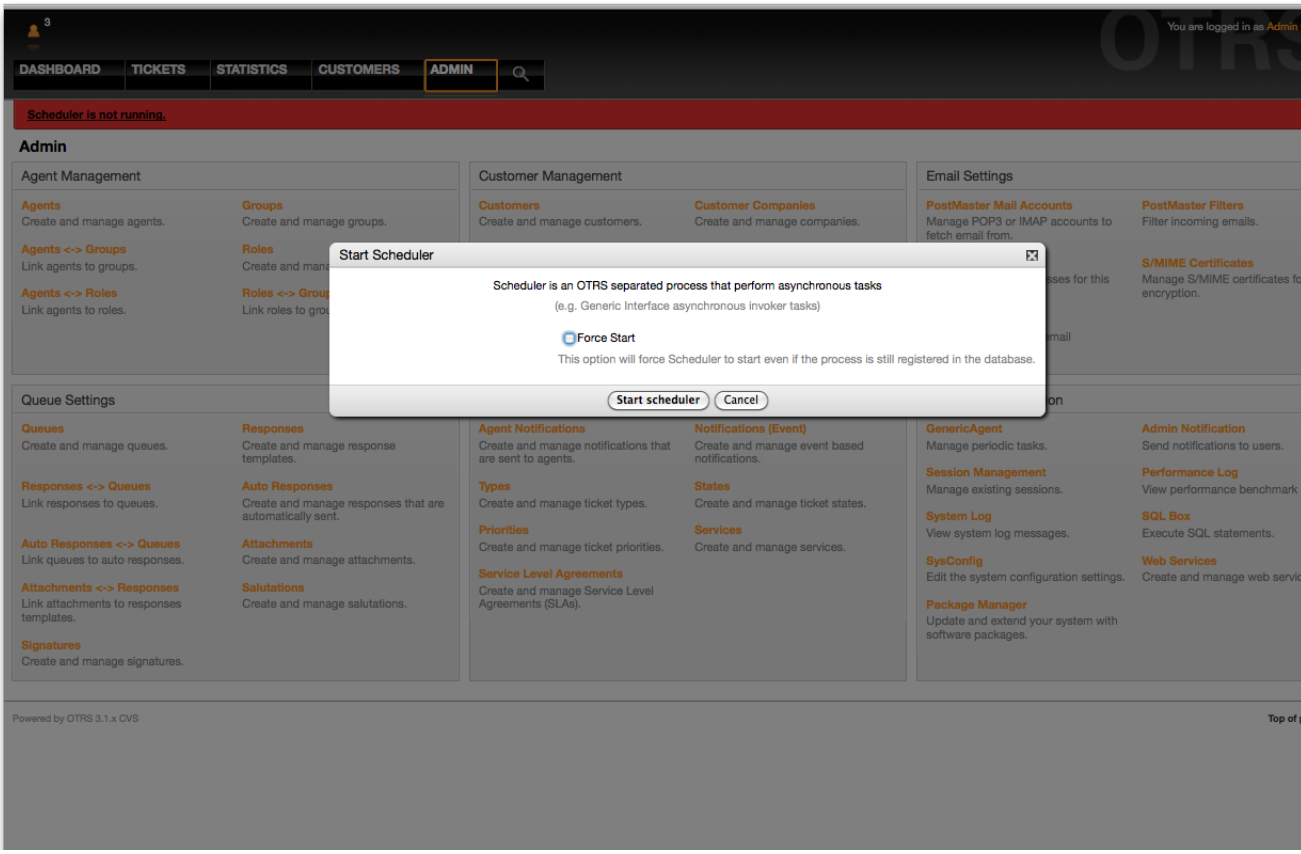

Figure: Start Scheduler.

Scheduler Command Line Interface

The Scheduler command line tools let you control the Scheduler process (Start / Stop) or query it status. There are also tools to register the process to be controlled by the operating system.

Included with OTRS there are two set of CLI tools, one for Unix / Linux OS and another for MS Windows OS.

Unix / Linux

Scheduler Init.d Files

Init.d files are special scripts that are called by the operating system at startup and shutdown (or restart) times.

OTRS provide init.d scripts to start / stop the OTRS Scheduler process automatically by the operating system, this scripts are located under OTRS_HOME/scripts.

Init.d scripts needs to be copied to the correct location for your operating system, they need to had the proper permissions and some internal variables needs to be set to work properly.

Init.d Script Internal Variables

- **OTRS_HOME** - the path of your OTRS installation.
- **User** - the apache process user name.
- **Group** - the apache process user's group name.

Note

Currently there are only init.d scripts for Linux platforms.

Table 21.1. List of Init Scripts And Supported Operating Systems

Init Script	Supported OS
otrs-scheduler-linux	Red Hat, Fedora, CentOS, SUSE, openSUSE, Debian, Ubuntu
otrs-scheduler-gentoo-init.d, otrs-scheduler-gentoo-conf.d	Gentoo

Example 21.1. Example To Start The OTRS Scheduler Form An Init.d Script

```
shell> /etc/init.d/otrs-scheduler-linux start
```

Available Actions

- **start** to start the OTRS Scheduler process.
- **stop** to stop the OTRS Scheduler process.
- **restart** to restart the OTRS Scheduler process.
- **status** to query the OTRS Scheduler process status.

The Scheduler needs the database to be available to register its Process ID, for this reason is necessary to:

- Execute the Scheduler init.d script to *start* the Scheduler process after the database process is up and running.
- Execute the Scheduler init.d script to *stop* the Scheduler before the database process shuts down.

Note

If you want the Scheduler to run at system startup, please read the documentation of the operating system for the right location to place the init.d scripts, how to configure them to run automatically and how to set the run order.

Scheduler Daemon File

This is the part of the Scheduler that stays running in the background checking for tasks to execute. It also provides the main functions to control the process.

All Unix / Linux uses the file **OTRS_HOME/bin/otrs.Scheduler.pl**.

Example 21.2. Example To Start The OTRS Scheduler

```
shell> OTRS_HOME/bin/otrs.Scheduler.pl -a
start
```

Available Options

- **-a** action.

Possible Values

- **start**- to start the Scheduler process.
- **stop**- to stop the Scheduler process.
- **status**- to query Scheduler process status.

- **-f** to force the start or stop of the Scheduler process.

Example 21.3. Example To Force Stop The OTRS Scheduler

```
shell> OTRS_HOME/bin/otrs.Scheduler.pl -a
stop -f 1
```

Note

Force stop the Scheduler is used remove the process ID from the database when scheduler is not running and the process is still registered.

Force start the Scheduler is used to start the Scheduler process if the scheduler is not running and the process is registered.

Force start or stop are only necessary if the start of the process is needed to be done before the process update time expires. Otherwise an expired entry in the database is discarded by normal start.

Windows

Scheduler Service Installer

The integration of the services into the MS Windows Operating System is done via the Windows Service Control Manager (SCM). In order to make the OTRS Scheduler process to be controlled by the SCM is necessary to register this service

OTRS provides the script **OTRS_HOME/bin/otrs.Scheduler4WinInstaller.pl** to register or unregister the OTRS Scheduler into the SCM.

Example 21.4. Example To Register The OTRS Scheduler Into the Widows SCM

```
                              shell> OTRS_HOME/bin/
otrs.Scheduler4WinInstaller.pl -a install
```

Available Options

- **-a** action.

Possible Values

- **install**- to install the Scheduler process into the Windows SCM.
- **remove**- to remove the Scheduler process from the Windows SCM.

After installing into the Widows SCM the OTRS Scheduler process can be used as any other service in Windows. It can be started, stopped and restarted and can be configured to be started manually or automatic.

Note

To learn more about Windows Services and the Windows SCM please read the Windows documentation, and Microsoft online help.

Scheduler Service File

This is the part of the Scheduler that stays running in the background checking for tasks to execute. It also provides the main functions to control the process.

Windows Operating System uses the file **OTRS_HOME/bin/otrs.Scheduler4Win.pl**.

Example 21.5. Example To Start The OTRS Scheduler

```
                              shell> OTRS_HOME/bin/otrs.Scheduler4Win.pl
 -a start
```

Available Options

- **-a** action.

Possible Values

- **start**- to start the Scheduler process.
- **stop**- to stop the Scheduler process.
- **status**- to query Scheduler process status.

- **-f** to force the start or stop of the Scheduler process.

Example 21.6. Example To Force Stop The OTRS Scheduler

```
shell> OTRS_HOME/bin/otrs.Scheduler4Win.pl
-a stop -f 1
```

Note

Force stopping the Scheduler is used to remove the process ID from the database when scheduler is not running and the process is still registered.

Force starting the Scheduler is used to start the Scheduler process if the scheduler is not running and the process is still registered.

Force start or stop are only necessary if starting the process is needed to be done before the process update time expires. Otherwise an expired entry in the database would be discarded by a normal start.

Chapter 22. Dynamic Fields

Introduction

A dynamic field is a special kind of field in OTRS, created to extend the information stored on a ticket or article. These fields are not fixed in the system and they can appear only in specific screens, they can be mandatory or not, and their representation in the screens depends on the field type defined at their creation time according to the data to be held by the field. For example, there are fields to hold a text, a date, a selection of items, etc.

Dynamic fields are the evolution of TicketFreeText TicketFreeKey TicketFreeTime, ArticleFreeText and ArticleFreeKey fields that where commonly used in OTRS 3.0 and before. The limitation of these "Free Fields" was that they can be defined up to 16 (text or dropdown) fields and 6 time fields for a ticket and 3 (text or dropdown) fields for each article only, not more.

Now with dynamic fields the limitation in the number of fields per ticket or article is removed, you can create as many dynamic fields you like either for ticket or articles. And beyond that, the framework behind the dynamic fields is prepared to handle custom fields for other objects rather than just ticket and articles.

This new framework that handles the dynamic fields is build using a modular approach, where each kind of dynamic field can be seen as a plug-in module for the framework. This means that the variety of dynamic fields can be easily extended by public OTRS modules, OTRS Feature Add-ons, OTRS custom developments, and other custom developments.

The following dynamic field types are included with this release:

- Text (one line of text)
- Textarea (multiple lines of text)
- Checkbox
- Dropdown (single choice, multiple values)
- Multiselect (multiple choice, multiple values)
- Date
- Date / Time

Configuration

By default, a clean installation of OTRS 3.1.x does not include any dynamic fields. If you plan to use such fields in tickets or articles you need to create dynamic fields.

An updated installation from OTRS 3.0.x will have all the old "free fields" created as dynamic fields for compatibility and data preservation. The following is the list of dynamic fields that are created during the migration from OTRS 3.0.x to 3.1.x.

- TicketFreeKey[1-16] (TicketFreeKey1, TicketFreeKey2 ... TicketFreeKey16)

- TicketFreeText[1-16]
- TicketFreeTime[1-6]
- ArticleFreeKey[1-16]
- ArticleFreeText[1-16]

Note

During the migration procedure from OTRS 3.0.x to OTRS 3.1.x all the old "free fields" data and configuration are migrated to the new dynamic fields architecture. Any custom development around the old "free fields" has to be updated to use the new dynamic field framework.

The migration of the configuration include the field itself and the screen configurations to hide, show or show field as mandatory for each screen.

The configuration of a dynamic field is split in two parts, to add a new dynamic field or manage an existing one you need to navigate into the "Admin" panel in the "Dynamic Fields" link. To show, show as mandatory or hide a dynamic field in one screen you need to change the OTRS settings in the "SysConfig" screen.

Adding a Dynamic Field

Click in the "Admin" button located in the navigation bar, then click on the "Dynamic Field" link inside "Ticket Settings" box located in the lower center of the screen. The dynamic fields overview will display as follows:

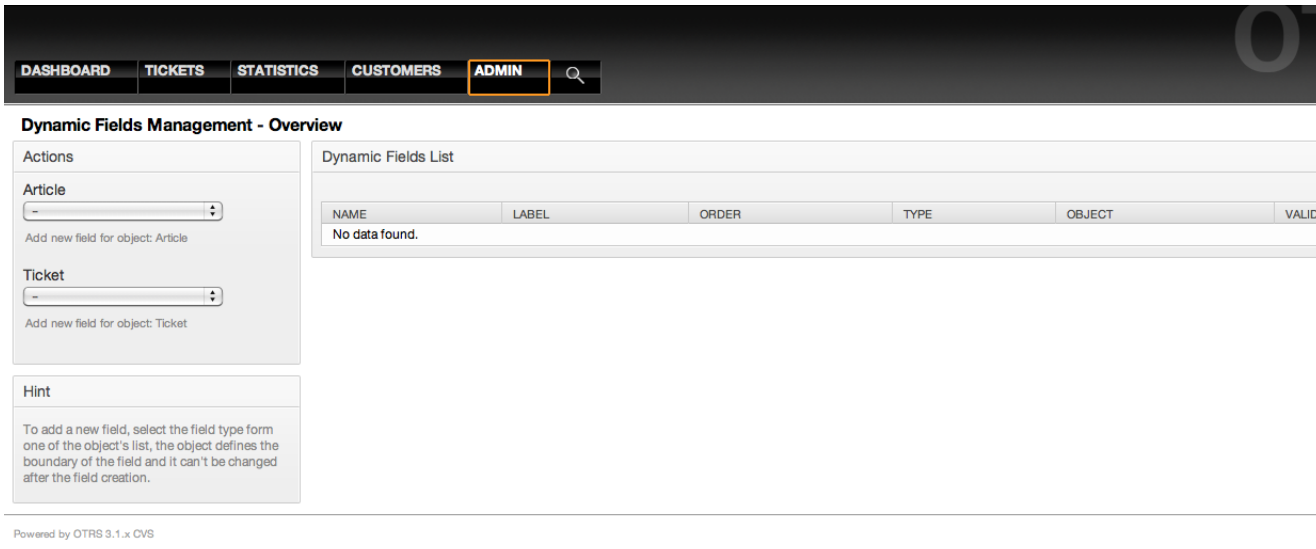

Figure: Dynamic fields overview screen, empty.

Notice that this screen will change as you add more dynamic fields to list all created dynamic fields. This screen might already have some fields if the installation was updated from an older version of OTRS.

The Actions in the side bar at the left of the screen describes two possibilities: Article and Ticket, each one has it's own dropdown selection of dynamic fields.

Note

The installation of an OTRS package could add more objects to the Action side bar.

The general procedure to create a dynamic field is:

- Click on the desired dynamic field object dropdown in the Action side bar.
- Click on the dynamic field type that you want to add from the list.
- Fill the configuration.
- Save.

The configuration dialogs for the dynamic fields are split in two parts, the upper section is common among all the fields and the lower part might be different from one type of dynamic field to another.

General dynamic field settings:

- Name: Mandatory, unique, only letters and numbers are allowed.

  This is the internal name of the field, used for example to show or hide a field in one screen. Any modification on field name (not recommended) will need manual a update on the "SysConfig" settings where the field is referenciated.

- Label: Mandatory.

  This is field name to be displayed in the screens, it supports translations.

  ## Note

  Label translations have to be added manually to language translations files.

- Field order: Mandatory.

  Defines the relative order in which the field will be displayed in the screen, by default each new field has the last position, a change in this setting will affect the other of the other created dynamic fields.

- Validity: Mandatory.

  An invalid dynamic field will not be displayed in any screen, no matter if is configured to displayed.

- Field type: Mandatory, Read only.

  Shows the current selected field type.

- Object type: Mandatory, Read only.

  Shows the scope of field.

  ### Note

  To illustrate each specific field type settings a few fields will be added in our example. These new fields will be referenciated in later sections.

  For the following examples all the dynamic fields will be created for the Ticket object if you need to create a dynamic field for Article object, just chose the field from the Article dropdown list.

Table 22.1. The following fields will be added into the system:

Name	Label	Type
Field1	My Field 1	Text
Field2	My Field 2	Textarea
Field3	My Field 3	Checkbox
Field4	My Field 4	Dropdown
Field5	My Field 5	Multiselect
Field6	My Field 5	Date
Field7	My Field 6	Date / Time

Text Dynamic Field Configuration

Text dynamic field is used to store a single line string.

Text dynamic field settings:

- Default value: Optional.

  This is the value to be shown by default in the edit screens (like New Phone Ticket or Ticket Compose).

- Show link: Optional.

  If set, the field value will be converted into a clickable link for display screens (like ticket zoom or overviews).

  For example, if "Show link" is set to "http://www.otrs.com", clicking on the filled value will make your browser to open the OTRS web page.

  ### Note

  The use of $LQData{"NameX"} in the Set link value, where NameX is the name of the field will add the field value as part of the link reference.

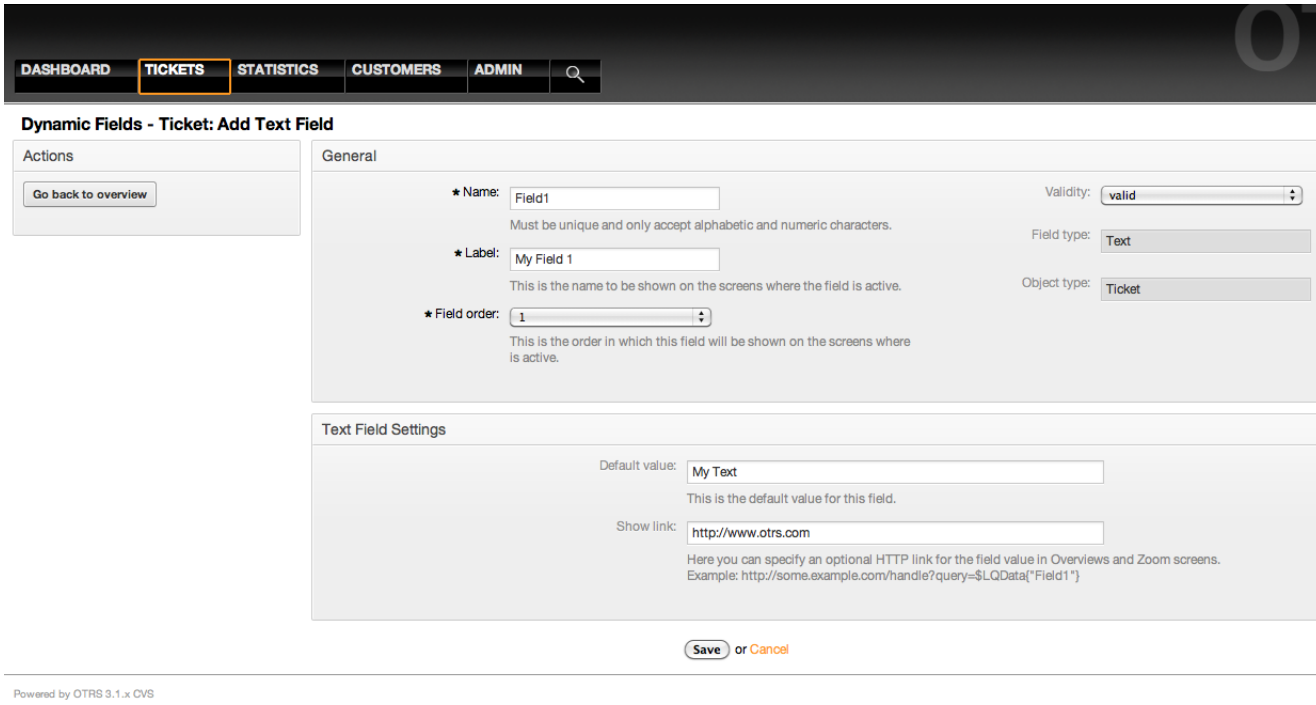

Figure: Dynamic field Text configuration dialog.

Textarea Dynamic Field Configuration

Textarea dynamic field is used to store a multiple line string.

Textarea dynamic field settings:

- Number of rows: Optional, integer.

  Used to define the height of the field in the edit screens (like New Phone Ticket or Ticket Compose).

- Number of cols: Optional, Integer.

  This is value is used to define the width of the field in the edit screens.

- Default value: Optional.

  This is the value to be shown by default in the edit screens (it can be a multiple line text).

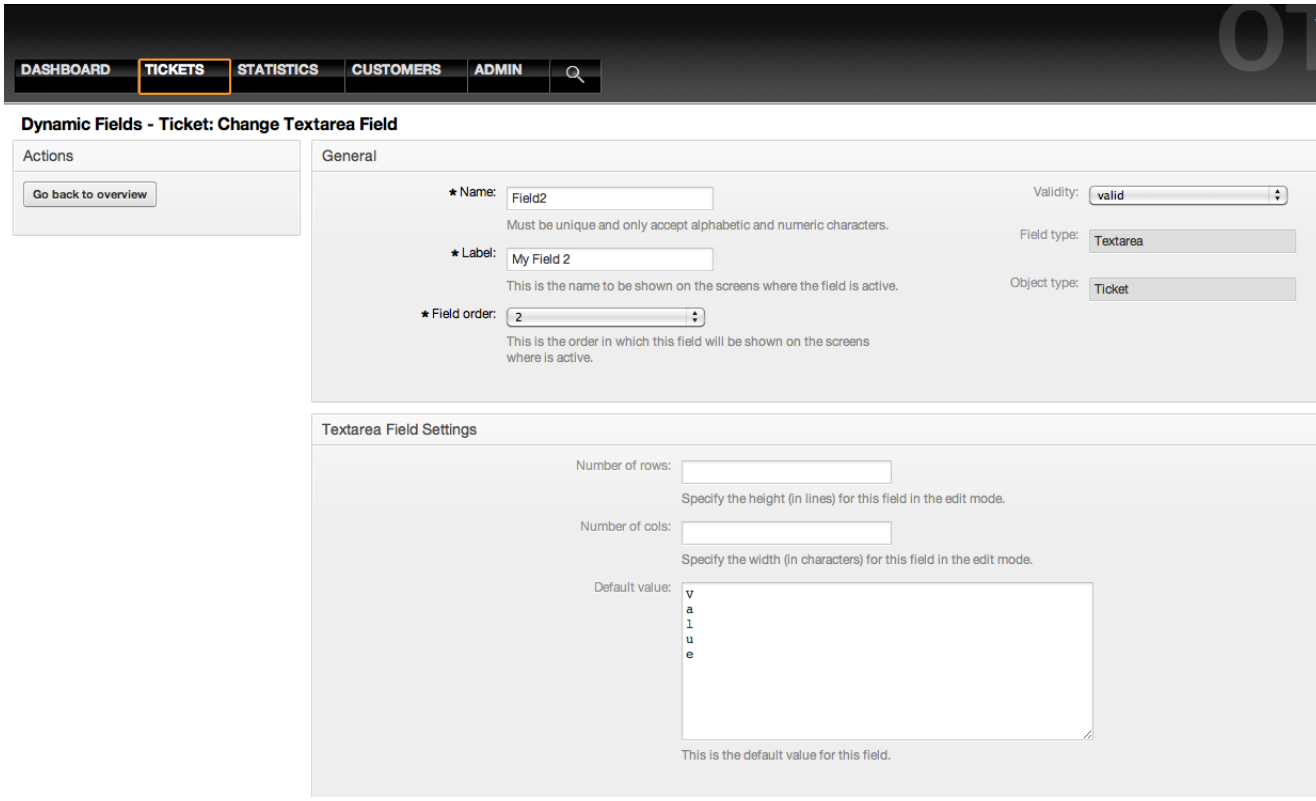

Figure: Dynamic field Textarea configuration dialog.

Checkbox Dynamic Field Configuration

Checkbox dynamic field is used to store true or false value, represented by a checked or unchecked check box.

Checkbox dynamic field settings:

- Default value: Mandatory.

  This is the value to be shown by default in the edit screens (like New Phone Ticket or Ticket Compose), the default value for this field is closed selection that can be Checked or Unchecked.

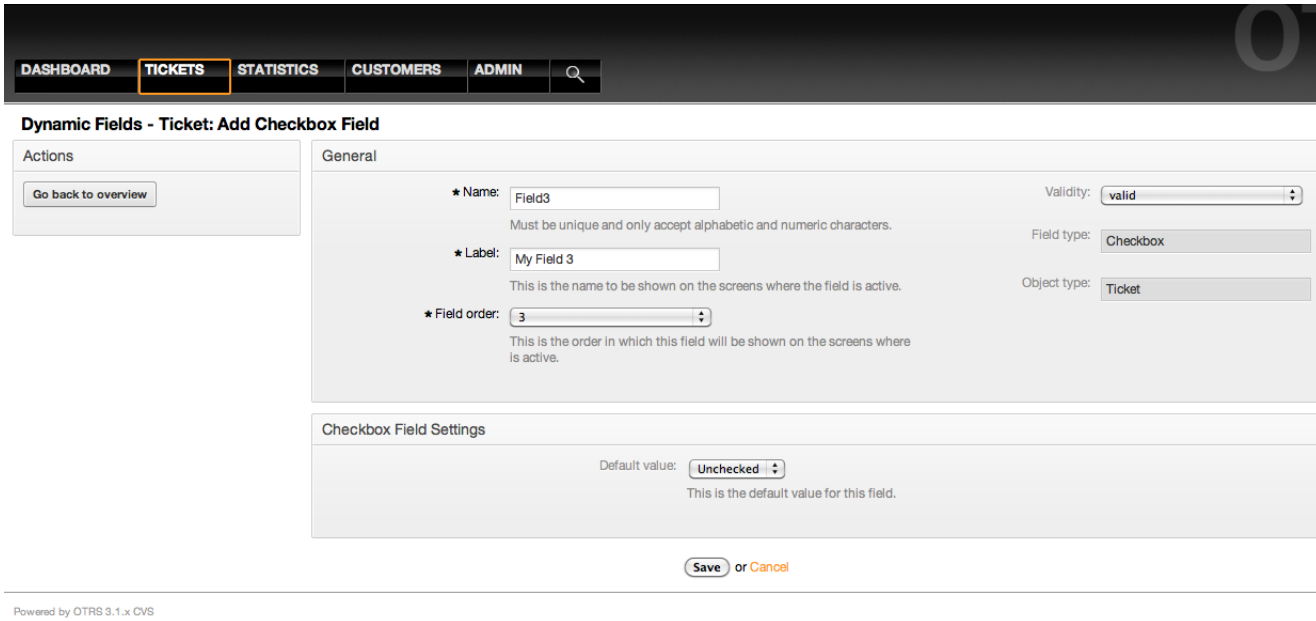

Figure: Dynamic field Checkbox configuration dialog.

Dropdown Dynamic Field Configuration

Dropdown dynamic field is used to store a single value, from a closed list.

Dropdown dynamic field settings:

- Possible values: Mandatory.

  List of values to choose. when add a new value is necessary to specify the Key (internal value) and the Value (display value).

- Default value: Optional.

  This is the value to be show by default in the edit screens (like New Phone Ticket or Ticket Compose), the default value for this field is closed selection defined by the Possible values.

- Add empty value: Mandatory, (Yes / No).

  If this option is activated an extra value is defined to show a "-" in the list of possible values, this special value is empty internally.

- Translatable values: Mandatory, (Yes / No).

  This setting is used mark the possible values of this field to be translated. Only the display values are translated, internal values are not affected, the translation of the values needs to be manually added to the language files.

- Show link: Optional.

If set, the field value will be converted into a clickable HTP link for display screens (like Zoom or overviews).

For example, if Show link is set to "http://www.otrs.com", clicking on the filed value will make your browser to open the OTRS web page.

Note

The use of $LQData{"NameX"} in the Set link value, where NameX is the name of the field will add the field value as part of the link reference.

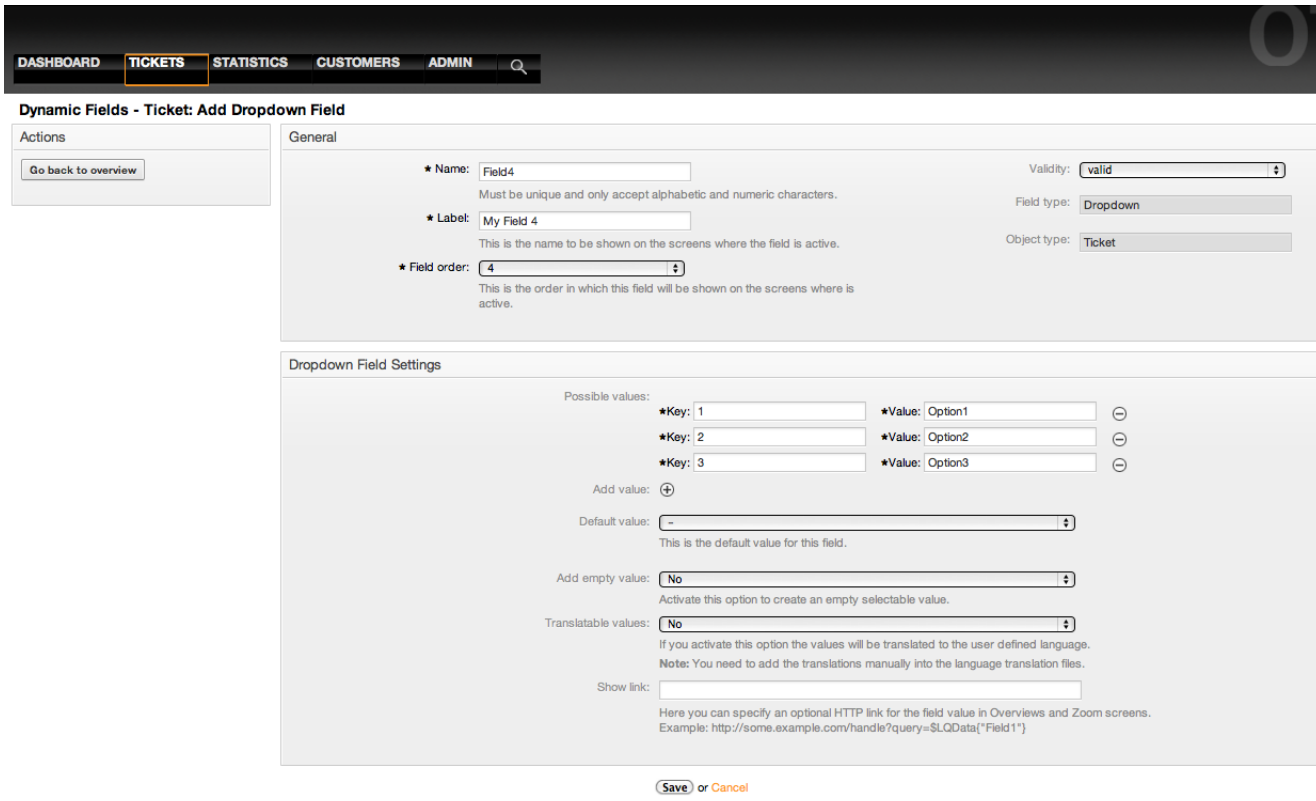

Figure: Dynamic field Dropdown configuration dialog.

Multiselect Dynamic Field Configuration

Multiselect dynamic field is used to store a multiple values, from a closed list.

Multiselect dynamic field settings:

- Possible values: Mandatory.

  List of values to choose. when add a new value is necessary to specify the Key (internal value) and the Value (display value).

- Default value: Optional.

  This is the value to be show by default in the edit screens (like New Phone Ticket or Ticket Compose), the default value for this field is closed selection defined by the Possible values.

- Add empty value: Mandatory, (Yes / No).

  If this option is activated an extra value is defined to show a "-" in the list of possible values, this special value is empty internally.

- Translatable values: Mandatory, (Yes / No).

  This setting is used mark the possible values of this field to be translated. Only the display values are translated, internal values are not affected, the translation of the values needs to be manually added to the language files.

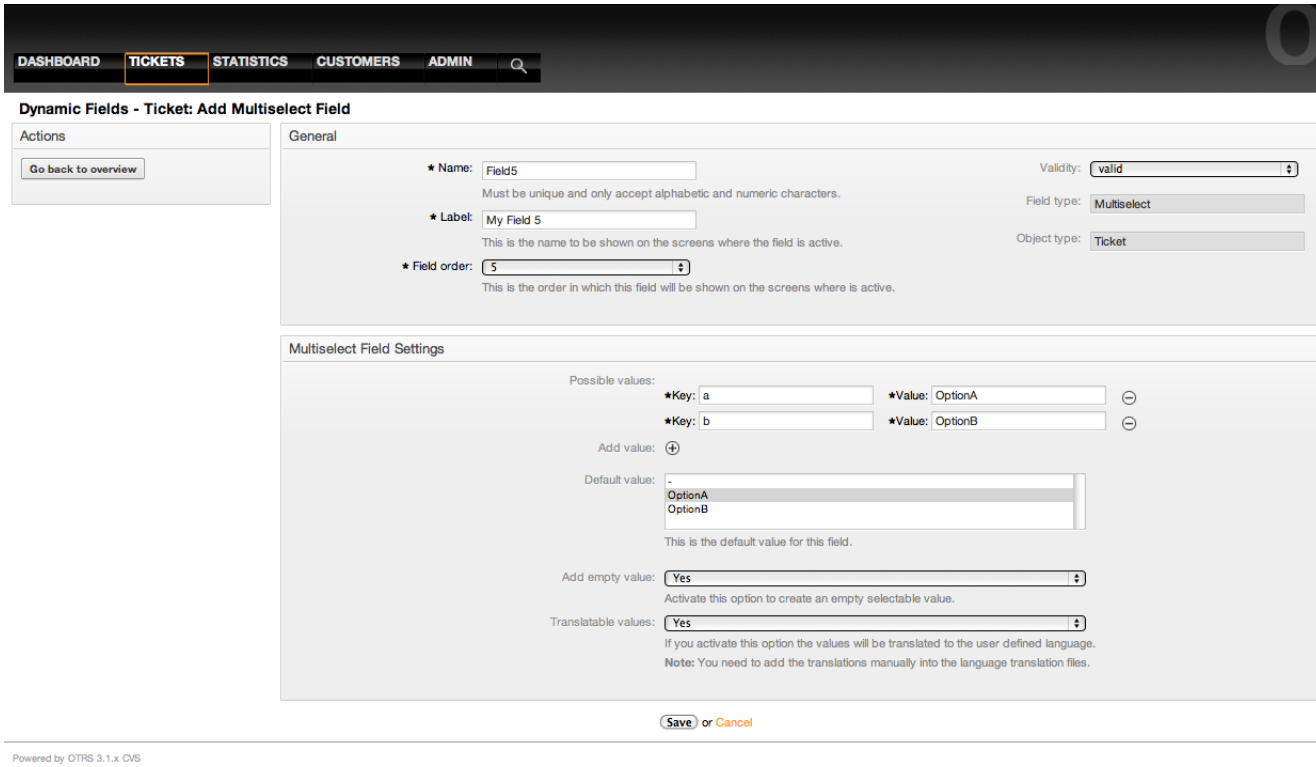

Figure: Dynamic field Multiselect configuration dialog.

Date Dynamic Field Configuration

Date dynamic field is used to store a date value (Day, Month and Year).

Date dynamic field settings:

- Default date difference: Optional, Integer.

  Number of seconds (positive or negative) between the current date and the selected date to be show by default in the edit screens (like New Phone Ticket or Ticket Compose).

- Define years period: Mandatory (Yes / No).

  Used to set a defined number of years in past and future from current date in the year select of this field, If set to Yes the following options are available:

- Years in the past: Optional, Positive integer.

  Define the number of years in past from current day to display in the year selection for this dined in edit screens.

- Years in the future: Optional, Positive integer.

  Define the number of years in future from current day to display in the year selection for this dined in edit screens.

- Show link: Optional.

  If set, the field value will be converted into a clickable HTP link for display screens (like Zoom or overviews).

  For example, if Show link is set to "http://www.otrs.com", clicking on the filed value will make your browser to open the OTRS web page.

  ### Note

  The use of $LQData{"NameX"} in the Set link value, where NameX is the name of the field will add the field value as part of the link reference.

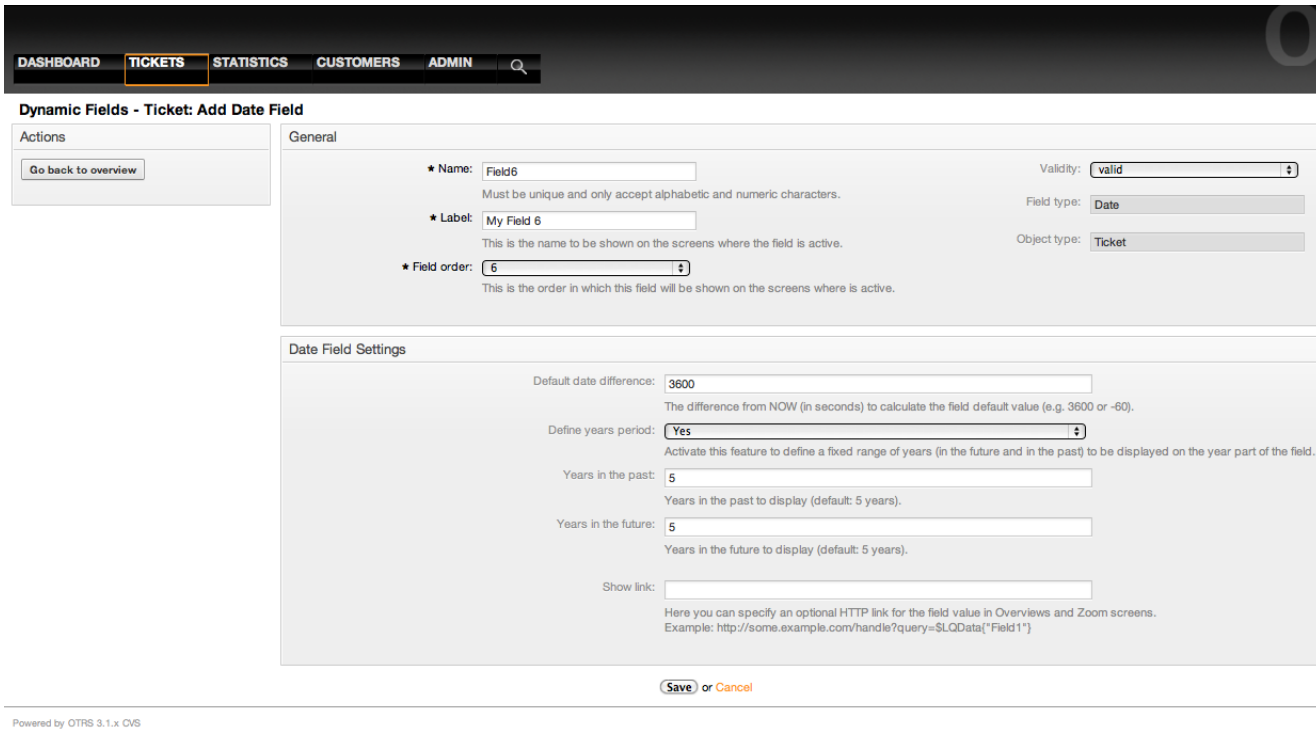

Figure: Dynamic field Date configuration dialog.

Date / Time Dynamic Field Configuration

Date / Time dynamic field is used to store a date time value (Minute, Hour, Day, Month and Year).

Date / Time dynamic field settings:

- Default date difference: Optional, Integer.

  Number of seconds (positive or negative) between the current date and the selected date to be show by default in the edit screens (like New Phone Ticket or Ticket Compose).

- Define years period: Mandatory (Yes / No).

  Used to set a defined number of years in past and future from current date in the year select of this field, If set to Yes the following options are available:

  - Years in the past: Optional, Positive integer.

    Define the number of years in past from current day to display in the year selection for this dined in edit screens.

  - Years in the future: Optional, Positive integer.

    Define the number of years in future from current day to display in the year selection for this dined in edit screens.

- Show link: Optional.

  If set, the field value will be converted into a clickable HTP link for display screens (like Zoom or overviews).

  For example, if Show link is set to "http://www.otrs.com", clicking on the filed value will make your browser to open the OTRS web page.

  ## Note

  The use of $LQData{"NameX"} in the Set link value, where NameX is the name of the field will add the field value as part of the link reference.

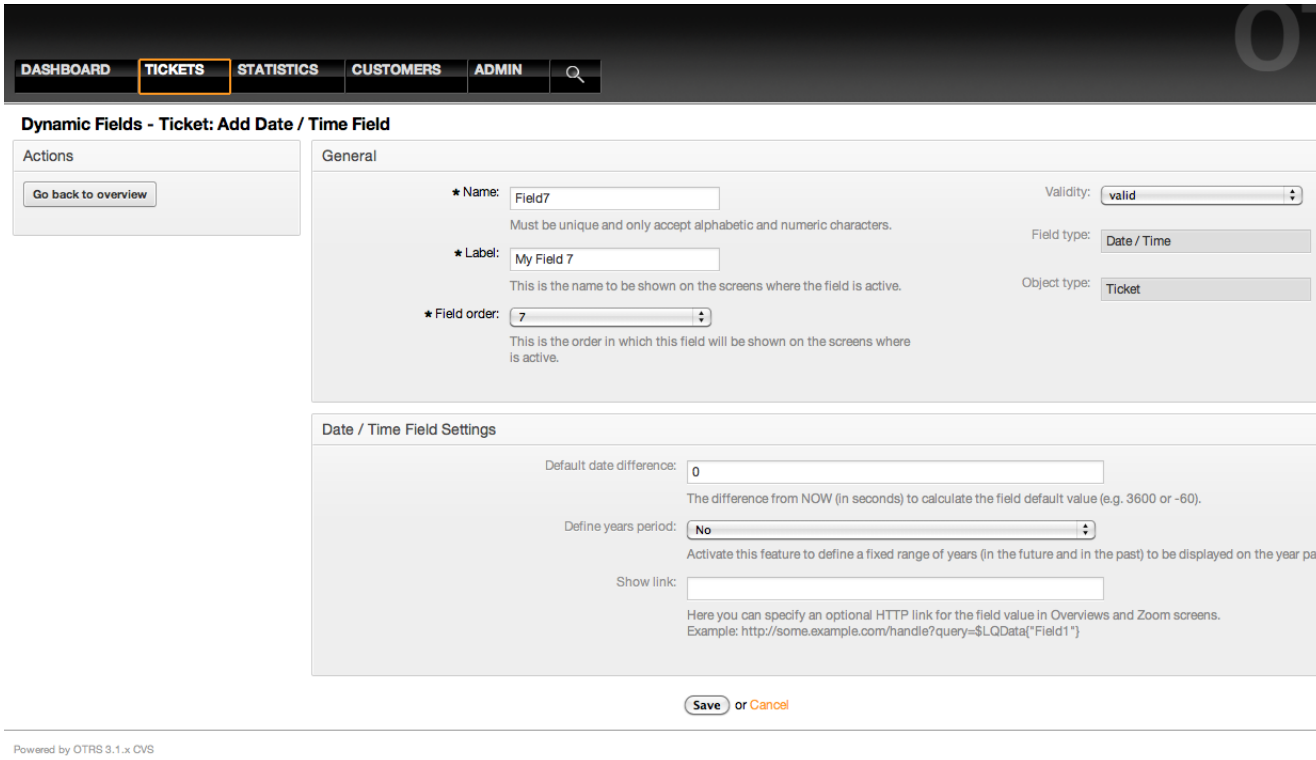

Figure: Dynamic field Date / Time configuration dialog.

Editing a Dynamic Field

A filled dynamic field overview screen (with the previous examples) should look like:

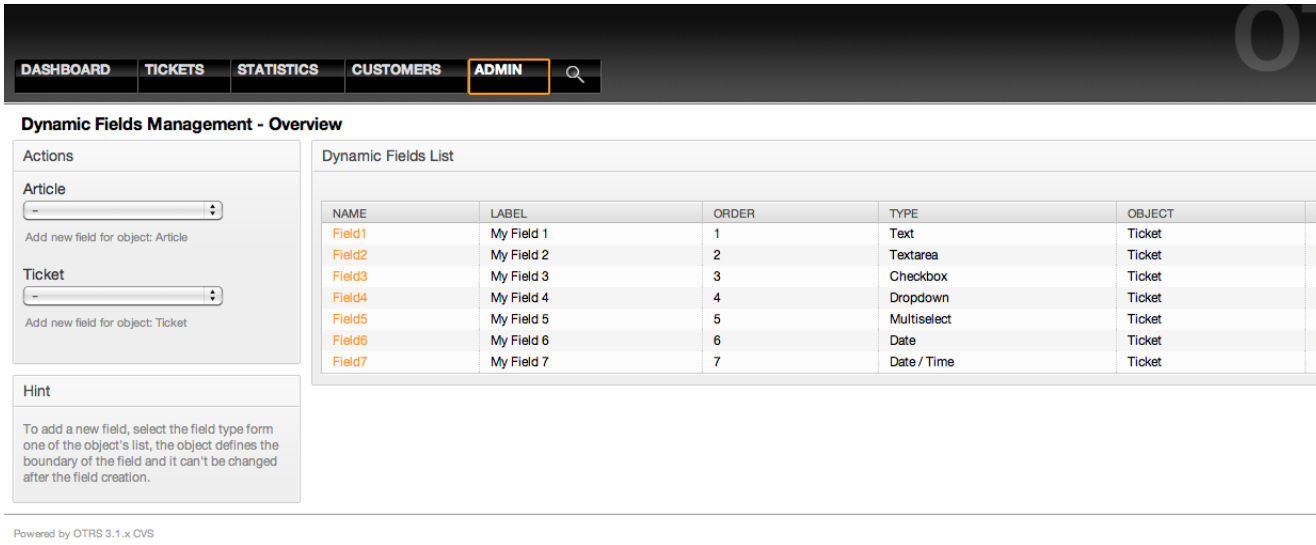

Figure: Dynamic field overview screen filled with sample data.

To change or edit a dynamic field you must have at least one field defined, select an already added field from the dynamic fields overview screen and update it's settings.

Note

Not all the dynamic field settings can be changed, the Field type and Object type are fixed from the selection of the field and they can't be changed.

It is not recommended to change the field internal name, but the label can be changed at any time. If internal name is changed all "SysConfig" settings that has a reference to that particular field needs to be updated as well as user preferences (if defined).

Showing a Dynamic Field on a Screen

To display a dynamic field on a particular screen there are two mandatory conditions:

1. The dynamic field must be valid.
2. The dynamic field must be set to 1 or 2 in the configuration of the screen.

Follow this steps to show a dynamic field in a screen

- Be sure that the dynamic field is set to valid, you can see the validity of the field from the dynamic field overview screen. Set to valid by editing the field if necessary.
- Open the "sysconfig" and select "Ticket" from the dropdown list in the Actions side bar located in the left part of the screen.

Note

You can also search for "DynamicField" in the search box above or the "sysconfig" key directly if you already know it.

- Locate the setting sub-group for the screen that you are looking for and click on it. For example "Frontend::Agent::Ticket::ViewPhoneNew".
- Search for the setting that ends with "###DynamicField". For example "Ticket::Frontend::AgentTicketPhone###DynamicField".
- If the setting is empty or does not have the required dynamic filed name, click on the "+" button to add a new entry. For example Key: Field1, Content: 1.

  If the setting already has the dynamic field name listed be sure that is set to "1" to display the field or to "2" to display it as mandatory.
- Save the configuration by clicking in the "Update" button and the bottom of the screen and navigate to the screen where you want the field to be displayed.

Show Examples

The following are "sysconfig" configurations examples to show or hide dynamic fields on different screens.