OTRS Documentation Manual

Release 7.0

OTRS AG

Jan 28, 2019

Contents:

This documentation describes how to write documentations for OTRS 7 and its related packages.

CHAPTER 1

Sphinx Setup

Sphinx is a tool that makes it easy to create intelligent and beautiful documentation, written by Georg Brandl and licensed under the BSD license. For more information see the Sphinx website and its documentation.

1.1 Install

Sphinx can be installed via package manager on Debian/Ubuntu.

```
sudo apt-get install python3-sphinx
```

To generate locale specific PO file, this package is also required:

```
sudo apt-get install sphinx-intl
```

Some other packages is required for generating (localized) PDF output:

```
sudo apt-get install latexmk texlive-latex-extra texlive-lang-european␣
↪texlive-lang-german
```

1.2 Quick Start

To quick start, simply go to the documentation root folder, and execute the following command:

```
sphinx-quickstart
```

This will ask the following questions. Read them and answer them to your needs. The following output is for OTRSHideShowDynamicFields:

```
Welcome to the Sphinx 1.6.7 quickstart utility.

Please enter values for the following settings (just press Enter to
accept a default value, if one is given in brackets).

Enter the root path for documentation.
> Root path for the documentation [.]:

You have two options for placing the build directory for Sphinx output.
Either, you use a directory "_build" within the root path, or you separate
"source" and "build" directories within the root path.
> Separate source and build directories (y/n) [n]:

Inside the root directory, two more directories will be created; "_templates"
for custom HTML templates and "_static" for custom stylesheets and other␣
 ↪static
files. You can enter another prefix (such as ".") to replace the underscore.
> Name prefix for templates and static dir [_]:

The project name will occur in several places in the built documentation.
> Project name: OTRSHideShowDynamicFields
> Author name(s): OTRS AG

Sphinx has the notion of a "version" and a "release" for the
software. Each version can have multiple releases. For example, for
Python the version is something like 2.5 or 3.0, while the release is
something like 2.5.1 or 3.0a1.  If you don't need this dual structure,
just set both to the same value.
> Project version []: 6.0.0
> Project release [6.0.0]: 6.0.1

If the documents are to be written in a language other than English,
you can select a language here by its language code. Sphinx will then
translate text that it generates into that language.

For a list of supported codes, see
http://sphinx-doc.org/config.html#confval-language.
> Project language [en]:

The file name suffix for source files. Commonly, this is either ".txt"
or ".rst".  Only files with this suffix are considered documents.
> Source file suffix [.rst]:

One document is special in that it is considered the top node of the
"contents tree", that is, it is the root of the hierarchical structure
of the documents. Normally, this is "index", but if your "index"
document is a custom template, you can also set this to another filename.
> Name of your master document (without suffix) [index]:

Sphinx can also add configuration for epub output:
> Do you want to use the epub builder (y/n) [n]:
```

(continues on next page)

(continued from previous page)

```
Please indicate if you want to use one of the following Sphinx extensions:
> autodoc: automatically insert docstrings from modules (y/n) [n]:
> doctest: automatically test code snippets in doctest blocks (y/n) [n]:
> intersphinx: link between Sphinx documentation of different projects (y/n)␣
↪[n]:
> todo: write "todo" entries that can be shown or hidden on build (y/n) [n]:
> coverage: checks for documentation coverage (y/n) [n]:
> imgmath: include math, rendered as PNG or SVG images (y/n) [n]:
> mathjax: include math, rendered in the browser by MathJax (y/n) [n]:
> ifconfig: conditional inclusion of content based on config values (y/n) [n]:
> viewcode: include links to the source code of documented Python objects (y/
↪n) [n]:
> githubpages: create .nojekyll file to publish the document on GitHub pages␣
↪(y/n) [n]:

A Makefile and a Windows command file can be generated for you so that you
only have to run e.g. "make html" instead of invoking sphinx-build
directly.
> Create Makefile? (y/n) [y]:
> Create Windows command file? (y/n) [y]:

Creating file ./conf.py.
Creating file ./index.rst.
Creating file ./Makefile.
Creating file ./make.bat.

Finished: An initial directory structure has been created.

You should now populate your master file ./index.rst and create other␣
↪documentation
source files. Use the Makefile to build the docs, like so:

   make builder

where "builder" is one of the supported builders, e.g. html, latex or␣
↪linkcheck.
```

The following file structure have been generated:

```
.
  _build
  _static
  _templates
  conf.py
  index.rst
  make.bat
  Makefile

3 directories, 4 files
```

The `_build` directory is for the various generated outputs, like HTML, LaTeX (PDF) or gettext (POT). The content of the directory is autogenerated, so don't modify the files manually.

The _static directory is for the static files required by the outputs, like custom CSS or JS files or images.

The _templates directory is for document templates.

Some other folders need to create for the content and the internationalization. I suggest the following file structure:

```
.
  _build
  content
     screenshots
  locale
  _static
  _templates
  conf.py
  index.rst
  make.bat
  Makefile
```

The content of the documentation can be placed to the content directory. These are .rst files, that contain the texts. The screenshots for the content can be placed the screenshots directory inside the content directory.

The locale directory will store the translations of the documentation.

1.3 Configuration

The configuration options can be found in the conf.py file in the documentation root directory. Please check the documentation form more information about the configuration options.

1.3.1 Install Theme

However there are many themes included with Sphinx, these themes are not mobile friendly. I suggest to install the theme sphinx_rtd_theme. This is the same theme, as used by Read the Docs.

Note: Many themes are available on the Sphinx Themes website.

To install a theme, follow these instructions:

1. Download the theme as .tar.gz
2. Create a sphinx-themes directory, if necessary
3. Extract the theme to this directory
4. Add the path of the sphinx-themes directory to the conf.py in the html_theme_path setting
5. Specify the name of the theme in the conf.py with the html_theme setting

Example configuration:

```
html_logo = '_static/images/otrs-logo.png'
html_theme_path = ['/ws/sphinx-themes']
html_theme = 'sphinx_rtd_theme'
```

It is recommended to place the `sphinx-themes` directory to a shared folder to be accessible from all documentation.

Note: The theme `sphinx_rtd_theme` can be installed with the following command:

```
pip install sphinx_rtd_theme
```

1.4 Adding Content

Create some content (.rst files and screenshots) and put them to the appropriate directories. The reStructuredText syntax can be found on the Sphinx website. Sphinx can also handle the contents if they are split into separate files. Once the content have been written, add the links (without the .rst extension) to the `index.rst` file in the document root directory. The content of the `index.rst` can be the following:

```
==========================
OTRS Package Documentation
==========================

This documentation is the user manual for OTRSDummyPackage package.

This work is copyrighted by OTRS AG, Zimmersmühlenweg 11, 61440 Oberursel,␣
↪Germany.

Version 6.0.13

Build Date: 2018-06-11


.. toctree::
   :maxdepth: 3
   :caption: Contents:

   content/preface
   content/some-other-chapter
   content/one-more-chapter
```

If the content is ready, you can generate some outputs. First you have to generate the gettext (POT) file for the translations. To do this, execute the following command in the documentation root directory:

```
sphinx-build -b gettext . _build/gettext
```

This will generate the a `gettext` directory into the `_build`, and create two POT files: the `index.pot` contains the texts from the `index.rst` and the `content.pot` contains the texts from all .rst files located in `content` directory.

Note: If you prefer make file for generating contents, you can also use the following command to generate the gettext (POT) files:

```
make gettext
```

The directory structure is now the following:

```
.
  _build
      gettext
          content.pot
          index.pot
  content
      screenshots
      one-more-chapter.rst
      preface.rst
      some-other-chapter.rst
  locale
  _static
  _templates
  conf.py
  index.rst
  make.bat
  Makefile
```

Now create the locale. First you have to add some options to `conf.py`:

```
# Options for localization
locale_dirs = ['locale/']
gettext_compact = True
```

If locale don't exist yet, it will be created, otherwise it will be updated. Execute the following command for each language:

```
sphinx-intl update -p _build/gettext -l de
```

You have to specify the language with option `-l`. It is possible to specify more languages at the same time by repeating the option `-l` (i. e. `-l de -l hu`).

After generating `de` and `hu` locale, the directory structure is the following:

```
.
  _build
      gettext
          content.pot
          index.pot
  content
      screenshots
      one-more-chapter.rst
      preface.rst
      some-other-chapter.rst
  locale
      de
          LC_MESSAGES
              content.po
              index.po
      hu
          LC_MESSAGES
```

(continues on next page)

(continued from previous page)

```
              content.po
              index.po
  _static
  _templates
  conf.py
  index.rst
  make.bat
  Makefile
```

Now the translators can start translating the documentation into other languages. If the translation is done, you can generate the translated outputs.

For the German HTML output, execute the following command in the documentation root directory:

```
sphinx-build -b html -D language=de . _build/html/de
```

You can not generate PDF output directly. First you have to generate LaTeX output, then the PDF from the LaTeX source. Sphinx will create a makefile for this. For the German PDF output, execute the following commands:

```
sphinx-build -b latex -D language=de . _build/latex/de
cd _build/latex/de
make
```

It is possible to generate EPUB format using the `-b epub` option with the same way.

Note: It is also possible to generate the output with the make command, but in this case you can not specify the language. To do this, execute the following commands:

```
make html
make latexpdf
```

1.5 Troubleshooting

The following issues encountered during the using of Sphinx.

1.5.1 Missing Locale Setting

Symptom:

```
perl: warning: Please check that your locale settings:
   LANGUAGE = (unset),
   LC_ALL = (unset),
   LANG = "en"
are supported and installed on your system.
```

Solution: execute the following commands in the terminal:

```
export LANGUAGE=hu_HU.UTF-8
export LC_ALL=hu_HU.UTF-8
```

1.6 Pandoc

Pandoc is an other useful tool to convert or generate documentation. To convert the existing docbook to reStructuredText, execute the following command:

```
pandoc doc/en/PACKAGE_NAME.xml --columns 78 -f docbook -t rst -o PACKAGE_NAME.
↪rst
```

1.7 Required Packages on Ubuntu 18.04

```
sudo apt-get install python3-sphinx sphinx-intl latexmk texlive-latex-extra␣
↪texlive-lang-european texlive-lang-german make
```

CHAPTER 2

reStructuredText Primer

This short tutorial will guild you through to create or update documentations for OTRS using reStructuredText as source format.

2.1 Source Format

The new documentation format name is **reStructuredText** (one word, this is the correct spelling). This is an easy to read documentation format using plain text and small inline markers. The following examples will show the usage of the documentation elements.

2.2 Elements

2.2.1 Headings

To use heading in the documentation, you have to underline the titles with special characters. The underline must start from the first letter of the title and end at the last letter of the title. The hierarchy of the special characters are the following: =, -, ~, ^, .

The following example shows the usage of the headings:

```
==============
Document title
==============

This the title of the document. Use only once in the index.rst file.

Chapter title
=============
```

(continues on next page)

(continued from previous page)

```
This is the heading 1 title. It has numbering like 1.

Section title
-------------

This is the heading 2 title. It has numbering like 1.1.

Subsection title
~~~~~~~~~~~~~~~~

This is the heading 3 title. It has numbering like 1.1.1.

Subsubsection title
^^^^^^^^^^^^^^^^^^^

This is the heading 4 title. It has numbering like 1.1.1.1.

Subsubsubsection title
......................

This is the heading 5 title. It has numbering like 1.1.1.1.1. Please don't␣
↪use this level of heading.
```

2.2.2 Paragraphs

For writing paragraphs, you have to start sentences at the beginning of the line. To create a new paragraph, just leave a blank line between the paragraphs. Example:

```
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed dictum imperdiet␣
↪enim. Curabitur
nisi diam, lobortis facilisis quam ut, porttitor consequat lectus. Nam␣
↪elementum, ipsum id
feugiat vestibulum, dolor ante dictum quam, ac bibendum ipsum felis in orci.

Vestibulum maximus egestas orci, eget consequat nibh imperdiet eget.␣
↪Suspendisse sagittis tempus
sapien, sit amet tincidunt tortor efficitur et. Etiam ac lacus sem. Sed ut␣
↪magna imperdiet,
viverra quam vitae, consequat mauris.
```

2.2.3 Inline Markup

The standard inline markup is quite simple: use

- one asterisk: *text* for *emphasis* (italics),
- two asterisks: **text** for **strong emphasis** (boldface), and
- backquotes: ‘‘text‘‘ for `literal texts` (code samples).

If asterisks or backquotes appear in running text and could be confused with inline markup delimiters, they have to be escaped with a backslash, like *.

2.2.4 Lists

To create unordered lists, start a line with asterisk (*) or slash (-). To create ordered list, start a line with numbers or hash mark (#). If you need nested lists, leave a blank line between the list items and use indentation with 3 spaces. Example:

```
* This is a bulleted list.
* It has two items, the second
  item uses two lines.

1. This is a numbered list.
2. It has two items too.

#. This is a numbered list.
#. It has two items too.
```

Nested list example:

```
- this is
- a list

  - with a nested list
  - and some subitems

- and here the parent list continues
```

2.2.5 Literal Blocks

Literal blocks are texts that should be displayed as verbatim. To create literal blocks, do the following:

- type 2 colons (::) in a new line
- leave a blank line
- write the text with indentation of 3 spaces

Use literal blocks for code snippets, terminal outputs, configuration files, etc. Example:

```
::

   # The database host
   $Self->{DatabaseHost} = '127.0.0.1';

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

   # The database user
   $Self->{DatabaseUser} = 'otrs';
```

If the language of the code snippet is known, you can specify it for syntax highlighting:

```
.. code:: perl

   # The database host
```

(continues on next page)

(continued from previous page)

```
    $Self->{DatabaseHost} = '127.0.0.1';

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

    # The database user
    $Self->{DatabaseUser} = 'otrs';
```

```
.. code:: xml

    <Setting Name="FAQ::Agent::StateTypes" Required="1" Valid="1">
        <Description Translatable="1">List of state types which can be used in␣
↪the agent interface.</Description>
        <Navigation>Core::FAQ</Navigation>
        <Value>
            <Array>
                <Item>internal</Item>
                <Item>external</Item>
                <Item>public</Item>
            </Array>
        </Value>
    </Setting>
```

2.2.6 Tables

To create grid tables, you have to draw the table. Example:

```
+------------------------+------------+----------+----------+
| Header row, column 1   | Header 2   | Header 3 | Header 4 |
| (header rows optional) |            |          |          |
+========================+============+==========+==========+
| body row 1, column 1   | column 2   | column 3 | column 4 |
+------------------------+------------+----------+----------+
| body row 2             | ...        | ...      |          |
+------------------------+------------+----------+----------+
```

2.2.7 Hyperlinks

Hyperlinks can be used inline or referenced. For inline use, encapsulate the text of the link and the URL with back tick (‘) marks and a trailing underscore (_), like ‘OTRS website <https://otrs.com>‘_, which will display as OTRS website.

To create referenced links, you have to separate the text and the links like:

```
The documentations are available in the `OTRS documentation portal`_.

.. _OTRS documentation portal: https://doc.otrs.com/
```

2.2.8 Images

To insert an image into the documentation, first you have to put the image in the *images* folder, the create a reference to the image with:

```
.. image:: images/agent-interface.en.png
```

If you want to add caption to the image, then you can use the figure element:

```
.. figure:: images/admin-general-catalog-management-class.png
   :alt: Admin General Catalog

   Admin General Catalog
```

2.2.9 Colored Boxes

This boxes have special meanings and will be highlighted as default. Use these blocks as follows:

Warning: This is a warning box.

```
.. warning::

   This is a warning box.
```

Note: This is a note box.

```
.. note::

   This is a note box.
```

See also:

This is a see also box.

```
.. seealso::

   This is a see also box.
```

2.3 Useful Links

- On-line editor: http://rst.ninjs.org
- reStructuredText Primer: http://www.sphinx-doc.org/en/master/usage/restructuredtext/basics.html
- reStructuredText User Documentation: http://docutils.sourceforge.net/rst.html

CHAPTER 3

Style Guide

This part of the documentation is only for visual style and wording.

3.1 Writing Content

There is an internet slang **TL;DR**, which means *too long, didn't read* (see more information on Wikipedia). Many people don't like reading long texts, so please keep the documentation as short as possible. Use step-by-step tutorials instead of writing wall of text.

For example this is a **wrong** example for writing content:

```
The agents are able to change the interface language of OTRS. To change the
interface language, click on your avatar on the top left corner, then select
Personal Settings menu item. A new screen will be displayed. On this screen
click on the User Profile, and then find a widget named Language. Select the
desired language in the drop-down menu. Please make sure to click on the
Save button next to the language widget.
```

The same content in **suggested** *human understandable* format:

```
To change the interface language of OTRS:

1. Click on your avatar on the top left corner
2. Select Personal Settings
3. Click the User Profile in the new screen
4. Choose a language from the drop-down menu of the Language widget
5. Click the Save button next to the widget
```

The latter is easier to translate, because 6 short sentences will be included in the language file. If a content is changed in one of the sentences, only the changed sentence need to be reviewed and translated again. The first wrong example puts only one huge string to the language file, and if some changes will be made in the source string, the translator needs to review and re-translate the whole string.

3.2 Screenshots

Use as few screenshots as possible, because screenshots are hard to translate, hard to recreate if the content is changed and not searchable. Try to explain the needed steps by lists instead of screenshots.

If you create some screenshots, don’t use the native resolution of your machine. Usually it is full HD or bigger, so creating a screenshot with this resolution will became unreadable in some output, because all the images have to be shrink to the width of A4 paper in case of PDF. OTRS uses responsive design, so 1025 pixels is the minimum, that OTRS assumes it is a large display. Please use this as width of your screenshots.

This is an example for a **wrong** screenshot, as it has resolution of full HD. Due to the automatic shrinking the texts on the screenshot are hard to read:

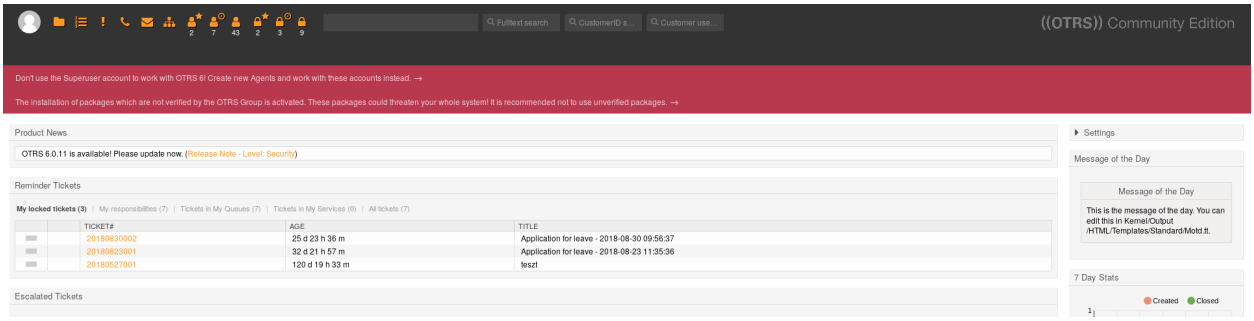

Fig. 1: Agent Dashboard (1920 pixels width)

The same screenshot with **suggested** resolution. The texts are much easier to read:

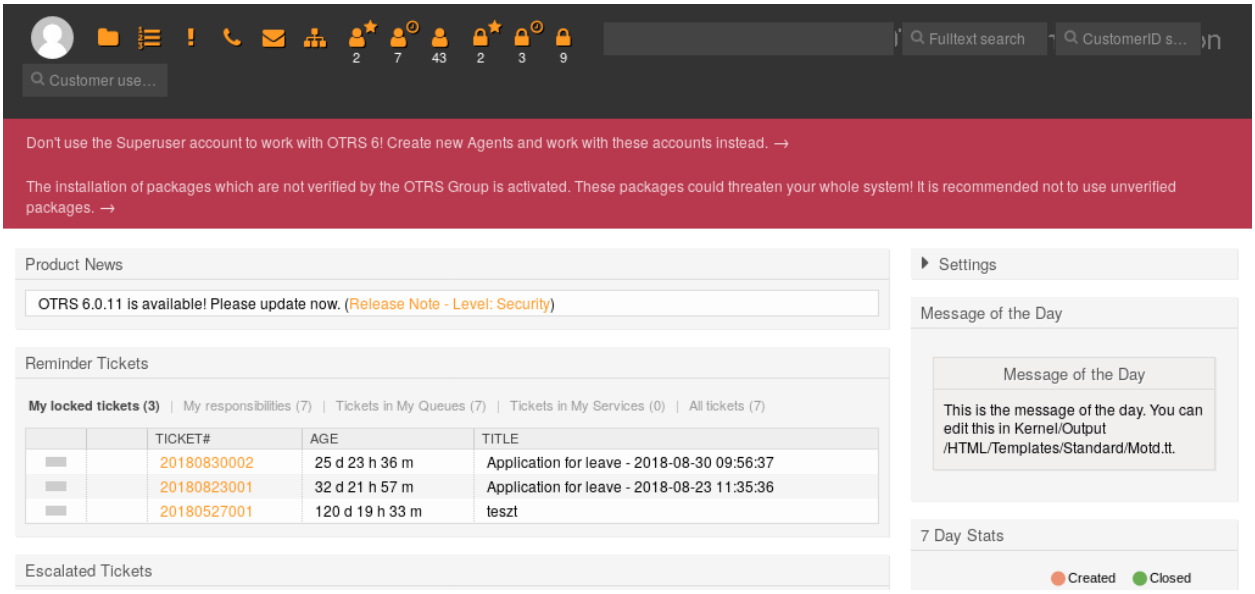

Fig. 2: Agent Dashboard (1025 pixels width)

It is also wrong, if the screenshot has good resolution in pixels, but with high DPI. For example this screenshot is **wrong**, because the texts on it is much bigger than the other texts in the documentation:

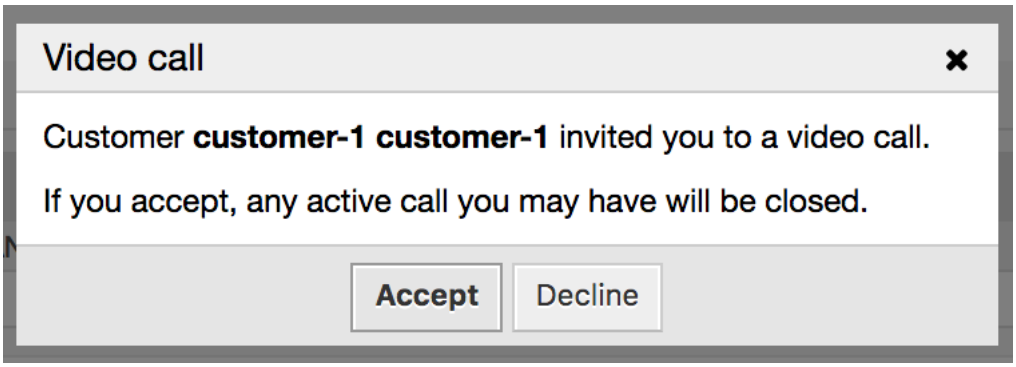

Fig. 3: Video Invitation Dialog (756 pixels width but with high DPI)

3.2.1 Create Screenshots with Firefox

If only a part of the screenshot is required, then the screenshot needs to be cropped. The administration area of OTRS consist of a left sidebar and a main content column. To create screenshots with Firefox:

1. Open the *Inspector* with `F12` or `Ctrl+Shift+C`
2. Search for `.MainBox` in the DOM
3. Right click on the node and select *Screenshot Node*
4. For the left sidebar search for `.SidebarColumn`
5. For the content column search for `.ContentColumn`

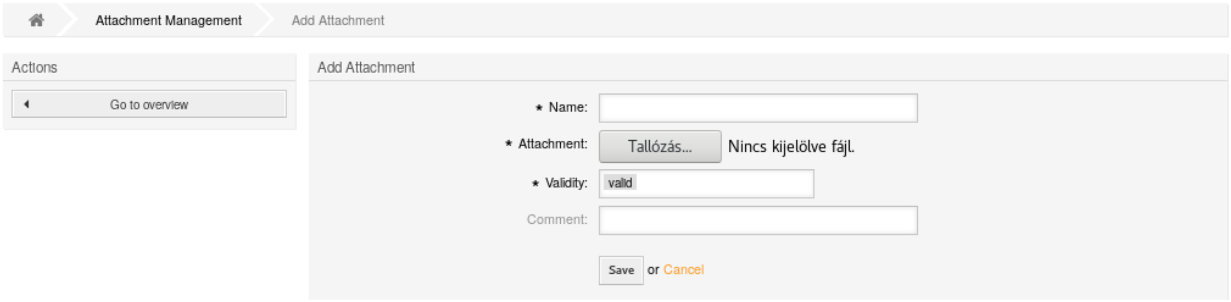

Fig. 4: Example screenshot for the main content

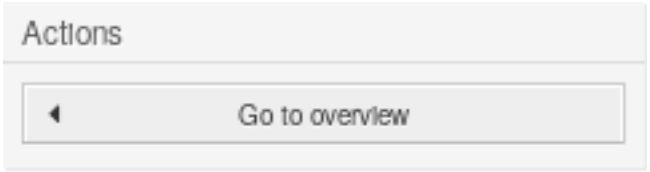

Fig. 5: Example screenshot for the left sidebar

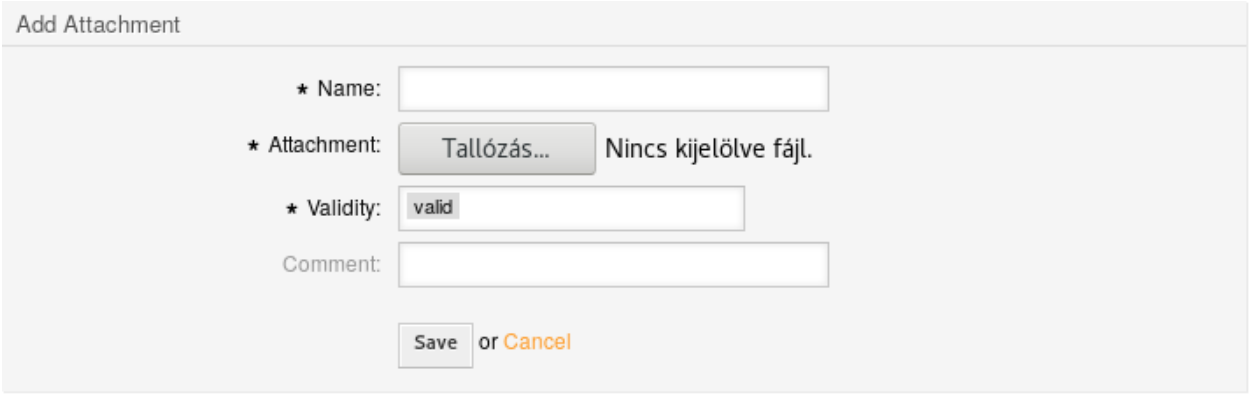

Fig. 6: Example screenshot for the main content column

3.3 Capitalization

For titles always have to use sentence case capitalization, which means, that in titles always capitalize:

- Nouns (man, bus, book)
- Adjectives (angry, lovely, small)
- Verbs (run, eat, sleep)
- Adverbs (slowly, quickly, quietly)
- Pronouns (he, she, it)
- Subordinating conjunctions (as, because, that)

In titles do not capitalize:

- Articles: a, an, the
- Coordinating Conjunctions: and, but, or, for, nor, etc.
- Prepositions (fewer than five letters): on, at, to, from, by, etc.

In normal sentences don't capitalize any words, only names and reference to titles have to be capitalized. This is a **wrong** example:

```
An Agent is a user, who handles Tickets in the Ticket Zoom screen.
```

The **suggested** sentence with proper capitalization. Besides, *Ticket Zoom* is the name of the screen, so it should be emphasized:

```
An agent is a user, who handles tickets in the *Ticket Zoom* screen.
```

3.4 Buttons and Screen Names

In the content sentences all buttons and screens should be emphasized and should be written with capital letters or in sentence case. Don't use apostrophes or quotation marks for emphasizing.

This sentence is **wrong**, because apostrophes are used for emphasizing:

```
If you click the 'Save and Finish' button, you will be redirected to the
↪'Ticket Zoom' screen.
```

The **suggested** way is to use asterisks for emphasizing:

```
If you click the *Save and Finish* button, you will be redirected to the␣
↪*Ticket Zoom* screen.
```

3.5 Wording

Don't use variable names in sentences. This sentence is **wrong**, because a variable name is meaningless for some people:

```
Add a new widget to AgentTicketZoom.
```

The same sentence without variable name, this is **suggested**:

```
Add a new widget to the Ticket Zoom screen of the agent interface.
```

3.6 Variable Names

Variable names should always marked as `literal` content. This is useful for translators, as they can exactly know, that the string mustn't be translated. If a string is not marked as literal content, it usually should be translated. For example:

The `ObjectManager` object has an `Init()` function. Additional configuration can be set in `Kernel::Config::Config.pm` file.

3.7 Useful Links

- GNOME User Manual: https://help.gnome.org/users/gnome-help/stable/