

Editors' Manual

Table of contents

	Editors' Manual.....	1
1	System requirements.....	1
2	Login to the editor.....	1
3	Structure of the editor.....	2
3.1	Frame.....	2
3.2	Navigation.....	3
3.2.1	ChangeSet selection menu.....	4
3.2.2	Structure area.....	5
3.2.3	Context menu.....	6
3.2.4	Moving and sorting of objects.....	6
3.3	Object window.....	6
4	The Dashboard.....	7
5	Editing system.....	8
5.1	Object structure window.....	9
5.2	Object detail window.....	9
5.2.1	Object action menu bar.....	10
5.2.2	Object tabs.....	10
5.2.2.1	Contents.....	10
5.2.2.1.1	Progressive forms.....	11
5.2.2.1.2	Rich text editor.....	11
5.2.2.2	Object properties.....	11
5.2.2.3	Versions.....	12
5.2.2.4	Rights.....	12
5.2.2.5	Referenced by.....	13
5.2.2.6	XML view.....	13
5.3	Search.....	13
5.3.1	Recycle bin.....	14
5.4	Preview.....	15
6	Data administration.....	15
7	User and group administration.....	15
7.1	Navigation window.....	15
7.2	User administration.....	15
7.2.1	Creating a user.....	16
7.2.2	Editing a user.....	16
7.2.2.1	Access data.....	16
7.2.2.2	Roles.....	16
7.2.2.3	Group memberships.....	17
7.2.2.4	User profiles.....	17
7.2.3	Function for limiting the amount of users.....	18
7.3	Group administration.....	18
7.3.1	Creating a group.....	18
7.3.2	Editing a group.....	18
7.3.2.1	Rights.....	18
7.3.2.1.1	Adding and deleting rights.....	21
7.3.2.2	Inheriting rights of other groups.....	21
7.3.2.3	Members.....	21
8	Module system.....	21

8.1	Module overview.....	24
8.2	Working in a module.....	26
8.3	Module configuration.....	28
8.4	Administration of schemata.....	28
8.5	Data objects.....	31
8.6	Users / groups.....	31
8.7	Transformations.....	32
8.8	Object structure window.....	32
8.8.1	Configuration.....	32
8.8.1.1	<detailView>.....	33
8.8.1.1.1	<columns>.....	34
8.8.1.1.1.1	<column>.....	34
8.8.1.1.2	<childType>.....	35
8.8.1.1.2.1	<column>.....	36
8.8.1.2	<symbolView>.....	36
8.8.1.2.1	<childType>.....	37
8.8.1.2.1.1	<label>.....	37
8.8.1.2.1.2	<source>.....	37
8.9	Interfaces.....	38
8.10	Editor functions.....	38
8.10.1	Binding.....	38
8.10.1.1	Server-side Javascript.....	43
8.10.2	Implementation of a workflow.....	46
8.10.2.1	Structure of the workflow XML.....	47
8.10.2.1.1	<serverActivity>.....	48
8.10.2.1.2	<widgetActivity>.....	48
8.10.2.1.3	<dialogActivity>.....	49
8.10.2.1.3.1	<components>.....	49
8.10.2.1.3.1.1	<dialogActivity>.....	49
8.10.2.1.4	<uiActivity>.....	50
8.10.2.1.4.1	<script>.....	50
8.10.2.2	Designer.....	51
8.10.2.3	Widget component.....	51
8.10.2.3.1	Widget-Extension.....	52
8.10.2.3.1.1	chooseString.....	52
8.10.2.3.1.2	format.....	52
8.10.2.3.1.3	formatDate.....	52
8.10.2.3.1.4	formatDateTime.....	52
8.10.2.3.1.5	formatTime.....	52
8.10.2.3.1.6	settings.....	52
8.10.2.3.2	Javascript prototype.....	52
8.10.2.3.3	Configuration.....	53
8.11	Components.....	54
8.11.1	Selectors.....	55
8.11.2	Extensions.....	55
8.11.2.1	http://onion.net/genericforms	56
8.11.2.1.1	chooseString.....	56
8.11.2.1.2	format.....	56
8.11.2.1.3	hasLabel.....	56
8.11.2.1.4	id.....	56

8.11.2.1.5	label.....	56
8.11.2.1.6	serialize.....	56
8.11.2.1.7	settings.....	56
8.11.2.1.8	split.....	56
8.11.2.2	http://onion.net/genericforms/common.....	56
8.11.2.2.1	actionUrl.....	56
8.11.2.2.2	binaryLength.....	56
8.11.2.2.3	binaryMimeType.....	56
8.11.2.2.4	binaryUrl.....	56
8.11.2.2.5	childrenSchemata.....	56
8.11.2.2.6	configuration.....	56
8.11.2.2.7	dataObjectIcon.....	56
8.11.2.2.8	dataObjectPath.....	56
8.11.2.2.9	resourceUrl.....	56
8.11.2.2.10	schemalcon.....	56
8.11.2.2.11	schemaLocalization.....	56
8.11.2.2.12	schemaLocalizationFromObject.....	56
8.11.2.2.13	users.....	56
8.11.3	Javascript prototype.....	56
8.11.4	Configuration.....	57
9	Rich text editor.....	58
9.1	Tool bars.....	59
9.2	Properties' dialogues.....	61
9.3	Correct use of text tagging.....	61
9.3.1	Headings.....	62
9.3.2	Lists.....	62
9.3.3	Emphasis.....	62
9.3.4	Inserting images.....	62
9.3.5	Links.....	64
9.3.6	Special characters.....	65
9.3.7	Tables.....	68
10	Enterprise ChangeSets.....	69
10.1	Change list.....	69
10.1.1	Data view.....	70
10.1.2	Communication.....	70
10.1.3	Working with ChangeSets.....	70
10.1.4	An example.....	70
10.2	Creating and configuring.....	71
10.2.1	Table Rights and Roles of a user.....	72
10.2.2	Creating.....	73
10.2.3	Configuring a ChangeSet.....	74
10.3	Working in a ChangeSet.....	75
10.3.1	Editing documents in a ChangeSet.....	75
10.3.1.1	Versioning.....	75
10.3.2	Editor tree view.....	76
10.3.3	The ChangeSet overview.....	76
10.3.3.1	Filtering options.....	78
10.3.3.2	Example workflow.....	79
10.3.4	Automatic display optimization.....	79
10.3.5	Preview of changes.....	79

10.3.6	Dialog change list.....	81
10.3.6.1	Working with “statuses”.....	81
10.3.6.2	Directly publishing / rejecting.....	82
10.4	Publishing / rejecting ChangeSets.....	83
	Figures.....	84

Editors' Manual

On the following pages, you can learn to use the editor with the aid of the onion.net editor manual .

It contains descriptions of the required system in order to work with the editor as well as a complete line through the system starting with logging onto the editor to the point of using the text editor.

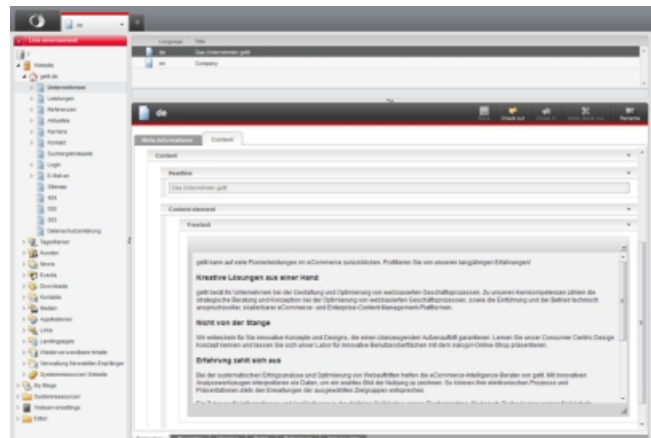


Figure 1

1 System requirements

The onion.net editor can be used with the most common browsers in their current versions. These include Windows Internet Explorer from version 8 onwards, Mozilla Firefox in the versions currently supported, Google Chrome in its current version as well as Safari from version 5.1. onwards

If Opera is used, faulty operation may occur during use depending on the version.

Cookies

onion.net sessions are saved with cookies. A unique session ID is stored in a cookie by the server here in order to recognise the logged-in user again at the time of later calls; otherwise the password would have to be re-entered for each action in the onion.net Editor. Cookies are generally activated in the browser. If this is not the case, then this can be changed in the settings of the browser. If you call the Editor from an intranet, it may well be that you do not have the authorisation to change these settings. In this case you should contact your system administration.

2 Login to the editor



Figure 2

The login page of the onion.net editor is accessible via a URL that is defined during the onion.net installation. Users who have to work with onion.net will receive this URL and their personal access data. Following entry of the user name and password and confirmation via the button "Login" opens the onion.net editor.

3 Structure of the editor

onion.net editor combines three independent administrations in one single user interface:

- > dashboard
- > content administration
- > data administration
- > user and group administration
- > model administration
- > module system

The graphic elements of the onion.net editor user interface are explained in the following paragraphs.

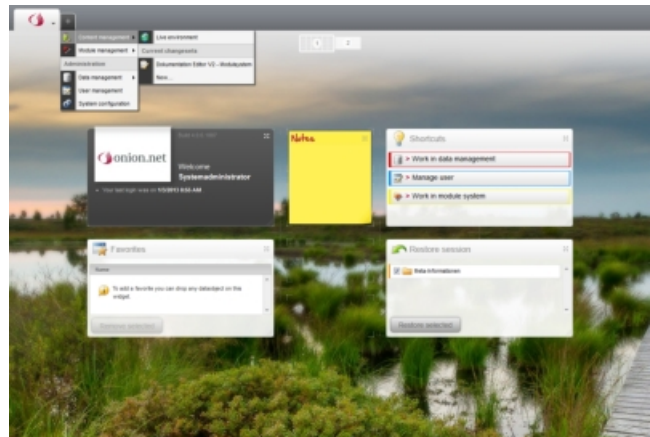


Figure 3

3.1 Frame

The frame goes around the workspace of the Editor. The workspace tabs are arranged above the workspace. After logging into the Editor, there is already a workspace open (the Dashboard) and can be seen in the tab bar. This tab cannot be closed.

To the right of the workspace tabs there is a button with a “+” on it for opening further tabs. By clicking on this button you will open a menu which, depending on the authorisations, offers ways of opening workspaces.

Simple editors are offered the point “editing system” in the context menu. Under this menu item you can access the data editor in order to maintain data objects. If you stay on the above mentioned menu item with the mouse, then a submenu will open where you can then go to the productive environment. If ChangeSets exist, these will be listed under the point “productive environment”. If you click on one of these menu items, a new workspace will open in which the desired work area is opened. A new tab is also created in the tab bar.

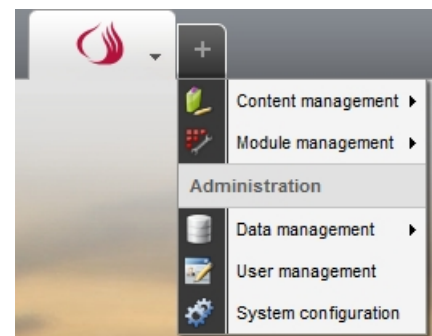


Figure 4

Widgets on the Dashboard can also offer links making it possible to open workspaces or go to documents.

If you open a work area in a new window (SHIFT + click), then a tab will also be shown in the tab bar for this. Pop-ups are treated like normal tabs in the tab bar, only with the exception that these are slightly transparent. As with normal tabs, the work area is brought into the viewing area if the tab is clicked on. In the case of pop-ups, the window is placed in the foreground.

A general rule with the tabs is that a context menu is opened when right-clicking or left-clicking on the arrow. What functions are offered in the context menu depends on the type of work area (e.g. editing system, user administration) and whether the tab is active or not.

If too many tabs are open, then buttons will appear to the left and right of the tabs for moving tabs back into the viewing area that have been moved out of it.

In order to log out, open the context menu on the tab of the Dashboard. You will now find the function “log out” there.

3.2 Navigation

The navigation through data objects in the work areas is generally divided up into the structure tree on the left, the object structure window above the object detail window and the path display. The three display options mentioned are not necessarily present in every work area and may be omitted depending on allocation.

The structure tree shows all objects in the form of a tree, which the logged-in user may work on within the respective area based on his group rights.

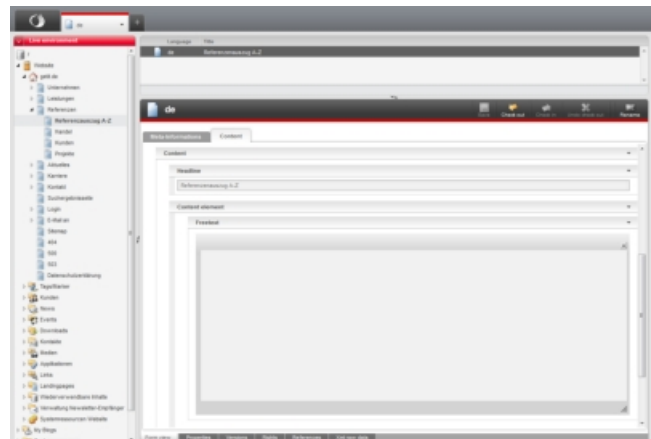


Figure 5

A tree is a special graph, and to put it simply, a hierarchically ordered set of objects which stem from a common root. Thus there is an object under which all other objects are located (the so-called systemroot object). The relationship between the objects can be pictured like that of related family members within a family tree. There are ancestors, descendants, parents, children and siblings.

The individual branches of the object tree can be expanded and collapsed. In doing so, the status of the individual branches is saved. If a branch expanded further is collapsed, then the expanded part becomes available again when the branch is reopened.

When loading a work area with a structure tree for the first time, the width adapts to the content of the structure tree. If the area should make working in the object detail window more difficult, then the frame between these two areas can be dragged with the mouse. Double clicking on the frame will instruct it to adapt to the width of the structure tree. If the frame is dragged to the left and to the edge of the browser window, then the Editor will offer to minimise this structure tree. If the structure tree is hidden, then a path display will be shown over the object detail window or over the object structure window. This path display represents the path to the document highlighted in the structure tree. Before the path display, the user will be given the option of showing the structure tree again.

In the path display, the user can navigate further through the data structure. When clicking on the path elements, the user is offered to select one of his child elements or to remove the clicked path element.

The object structure window above the data processing field is optional and is additionally configured as needed. It shows objects in a list and information can be added to it depending on column configuration. This

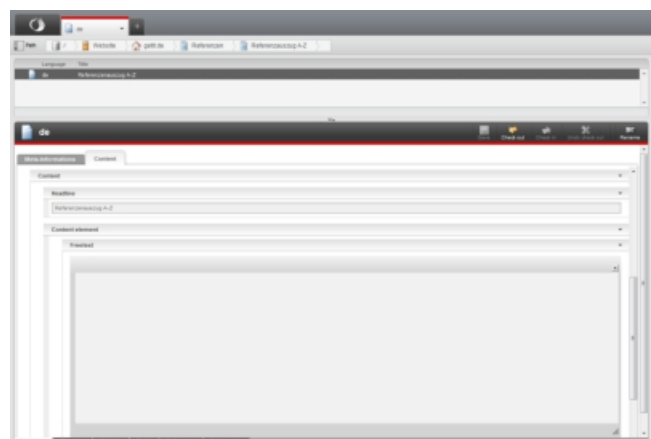


Figure 6

representation is an alternative to the representation of objects in the structure tree. The objects listed in the object structure window are therefore child elements of an object in the structure tree.

When looking at the editor for the first time you will see that, unlike with most other information or content management systems, there are very few buttons and toolbars. The user interface is not cluttered, but streamlined and functional. In order to make this possible, onion.net makes intensive use of context menus as well as the dragging and dropping of objects.

3.2.1 ChangeSet selection menu

The selection menu for ChangeSets provides two functions. The main function of the selection menu is the changing of the editing environments. Right-clicking on the element will open a menu. In the menu, the editing environment “live environment” and all existing Enterprise ChangeSet are listed. The context of the current tab can now be set to the editing environment via this menu.

The second function is the jumping to the overview page of the current editing environment. If you are in the live environment, then all checked-out documents will be indicated in the overview. In the ChangeSets, you can go, by clicking on the selection menu, to the homepage of a ChangeSet, where changes and meta information on the ChangeSet can be viewed.

3.2.2 Structure area

The structure area shows all objects in the shape of a tree that the logged-on user may edit in the respective area due to his group rights.

A tree is a special graph or, simply said, a specific number of hierarchically sorted objects coming from the same root, i. e. there is one object to which all other objects are subordinated (the so-called system root object). The interrelations of the objects could be compared to family relations in a genealogical tree. There are ancestors, descendants, parents, children and siblings.

The individual branches of the object tree can be expanded and collapsed. In doing so, the state of the individual branches will be saved. If another expanded branch is collapsed, the same expansion will be available if the branch is expanded again.

A first glance at the editor shows a major difference to most of the other information or content management systems: there are only a few buttons and menu bars. The interface is not cluttered, but streamlined and functional. In order to make this possible, onion.net makes intensive use of context menus and the drag & drop function.

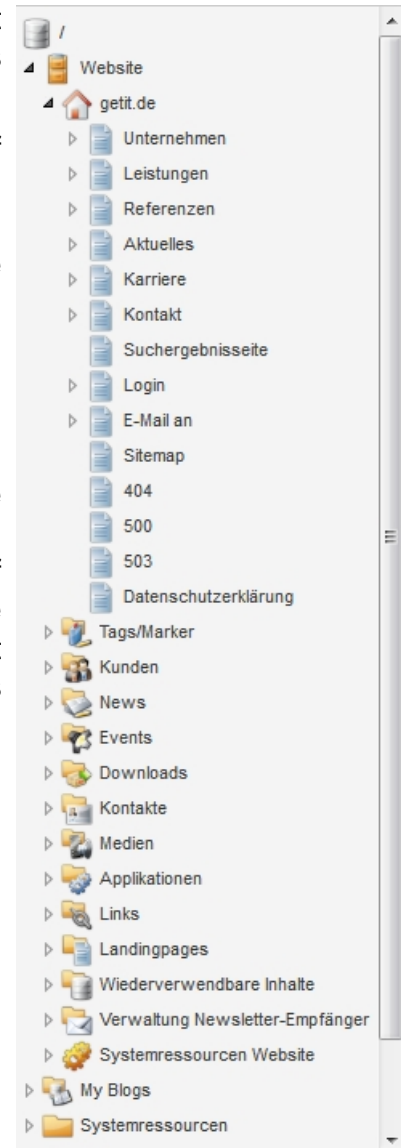


Figure 7

3.2.3 Context menu

You will know context menus from your everyday work on the computer. In the case of graphic user interfaces a context menu is an interaction object that offers the selection of different actions to the user for a certain context. In general, the context menu is opened in the proximity of the mouse pointer as a pop-up by clicking the second (right-hand) mouse button.

The context menu is subdivided into several sections. In general, it comprises the sections New and Functions and often also a section called Extras. Since onion.net is context-sensitive, special context menus are displayed depending on the clicked object. Due to this, not all context menu items are available for every object.

- All object types that can be created underneath an object appear in its context menu in the section New. In order to create a new object, you only have to call the context menu and the select the corresponding object type.
- Copy, paste, delete or rename of an object can be effected via the context menu section Functions. This section also comprises the call function of the research window within the content administration (see Research Window) and the preview (see Preview).
- For every object type, specific functions can be programmed that are listed in the section Extras.

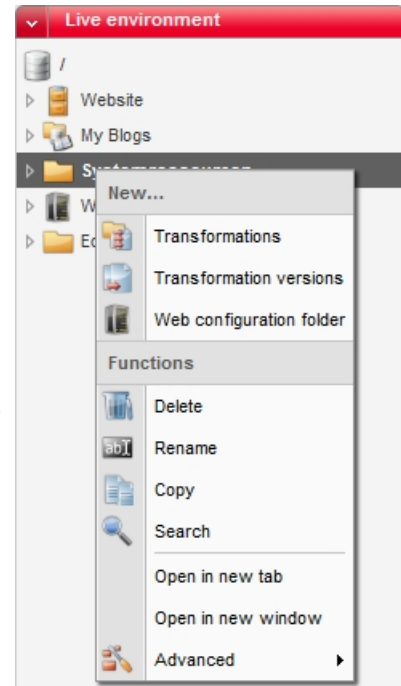


Figure 8

3.2.4 Moving and sorting of objects

Drag & drop is a method for moving graphic elements in a (web) application using a mouse. An item can be dragged and then released over a possible destination.

Drag & drop is one of the most important control elements of the onion.net editor and is used for the following two different purposes:

- Organisation of structure: Changing the order and the nesting of objects in the object tree.
- Linking: Simple placing of links within the rich text editor or for reference fields.

The system gives immediate acknowledgment of a drag & drop operation in all significant places:

- Highlighting of the object
- Visualisation of dragging
- Displaying where the object can be dropped
- Acknowledgment when dropping

3.3 Object window

If an editor has accessed an environment (productive environment or ChangeSet), the object window will show a list of all objects checked out by the editor in the first case and an output of all changes carried out so far in the ChangeSet in the second case.

If an object is clicked within the structure area, the view in the object window changes. The object can now be edited or different aspects of the object looked at. The contents and functions of the object window differ from each other in the three modes. They will be described in the coming sections.

The object window can also be opened on its own, either in a new window or in a new tab. To do this, online editors must hold down the SHIFT key or the CTRL key when clicking on the appropriate object in the structure area.

4 The Dashboard

This screen will welcome you after you have successfully logged into the Editor. This view, called “Dashboard”, provides the user with different functionalities (“widgets”). These widgets can show the user aggregated information at a glance, help him to resume work faster or provide him with functions for editing data.

You have the possibility here of dragging new widgets onto the Dashboard and also of changing the background. You can still get to all functionalities of the Editor from here.

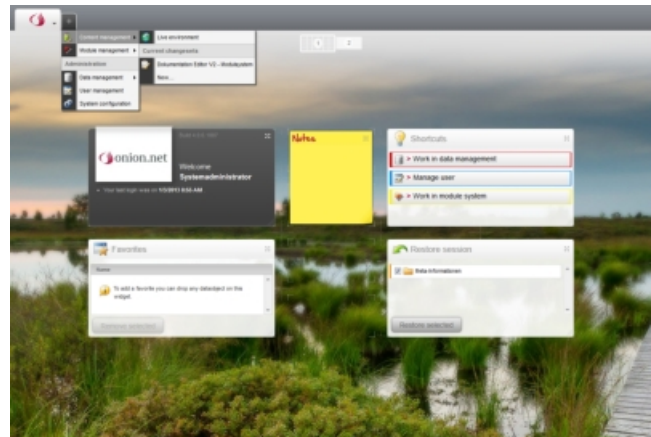


Figure 9

In order to activate a widget, please click on a free space on the Dashboard and drag the desired widget from the list open at the lower side of the screen to the desired place. While dragging, the widget to be added sticks to the cursor. At the same time, a small raster can also be seen in the upper left-hand corner of the widget, indicating the minimum amount of space the widget needs. When dragging over the Dashboard raster the rasters are highlighted which the widget would take up in the current cursor position. If the widget does not fit into this position, then the rasters are not highlighted. If there is no more space on the current page, then you can go up to the page numbers while dragging and stay on a non-activated page. After a short time the page will change and you can drop the widget there if necessary. Alternatively, you can also add a widget by clicking on the button “add” in the selection list. The system will try to find an appropriate place on the current page. If this is not possible, then it will search on the following pages, create a new page if necessary and switch to this page.

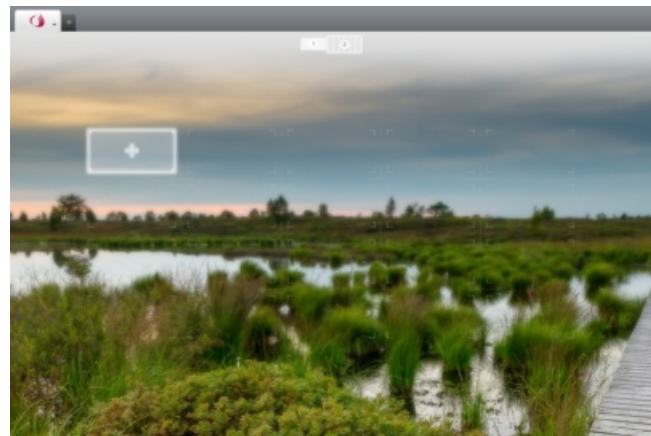


Figure 10

The position and page for each widget can be changed by dragging & dropping. You can start dragging a widget by clicking on the head area of the widget.

If you wish to remove a widget, then you need to right-click on the head area of the widget. A context menu will appear. With the context menu item “remove”, you can delete the widget from your Dashboard. If the widget is removed, then user-defined settings or data of this widget will also be deleted.

Moreover, you can personalise your start screen, where you can change the background image. You can see which background images are available and can be used through opening the context menu of the Dashboard tabs. In the context menu there is an item “Background images”. All possible background images are listed under this point and can be selected.

The configuration of the background images is performed by the administrator in the Editor configuration. Personal background images cannot be uploaded.

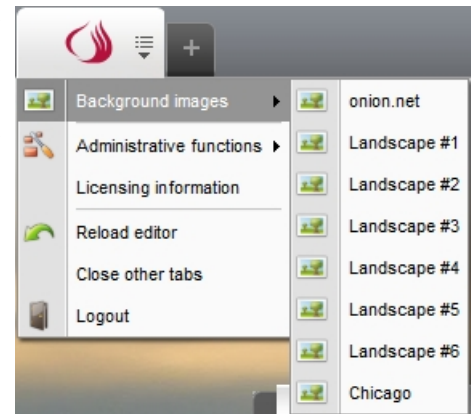


Figure 11

5 Editing system

In order to administrate contents, the user opens the area “Editing system”. If a user clicks on the button for adding new work areas, then he will be offered the editing system as the first item. Under this submenu point, the user is always offered the “productive environment” and all existing ChangeSets if necessary. If the user has the rights to create new ChangeSets, then a new ChangeSet can be created directly with the last item “New...”.

If the user has selected an editing system, then a new tab opens. The workspace for this area is arranged as follows. A structure tree can be seen on the left-hand side showing all data objects which the user may view. Above the structure tree there is a selection menu and the project logo. Using the selection menu, the user can change the work area and thus switch between the “productive environment” or the ChangeSets. If the user is in a ChangeSet, then the selection list is grey. If he is in the “productive environment”, the selection list is red. The colour of the tab also changes. While it is orange in the ChangeSets, it is red in the “productive environment”. The colour red is intended to signal to the user in a simple manner that he is in an environment where the changes carried out will have effects on all connected productive environments, such as a web page for example.

Next to the structure tree there is an area which can show two different views. The first view is the entry page to the work area. This entry page is opened if the work area is opened, a new work area is switched to or the selection list for changing the work area is clicked on. Depending on the work area, this entry page shows different information and functions. In the ChangeSets, the changes to the ChangeSet are shown to the user and the user has the possibility of publishing, discarding or configuring the ChangeSet. If the user is in the “productive environment”, then all documents are listed which are currently checked out by the user.

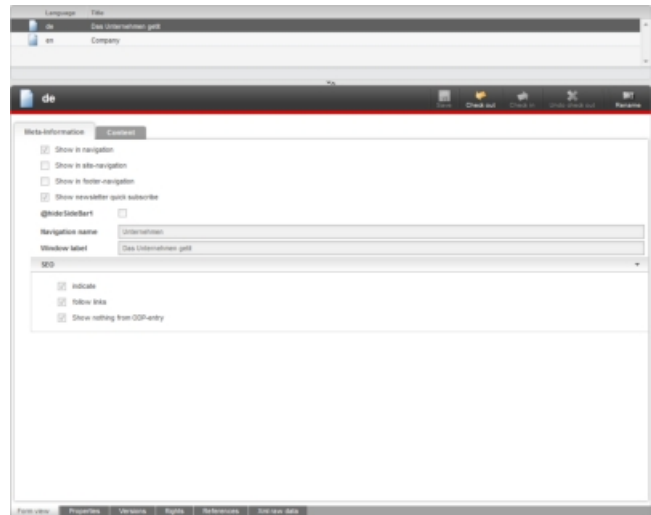


Figure 12

Depending on Editor configuration, the second possible view can be divided up into object structure window and object detail window.

5.1 Object structure window

In the structure area, objects can be displayed as part of the tree or separately in the shape of a table in the object detail window. Via the configuration files, administrators can define which objects are shown at which places. Fig. 6 shows an object structure window and presents the following advantages of such a view:

- If an object has a large number of child objects, e. g. a graphic directory that contains several dozens of graphics, the tree soon becomes confusing. The display of child objects in the object structure window improves the ergonomics of the editor.
- As opposed to the tree view, the object structure window offers the option to display additional information on objects, e. g. a minimised preview window of graphics.

The object whose descendants are currently displayed in the object structure window are always displayed in bold.

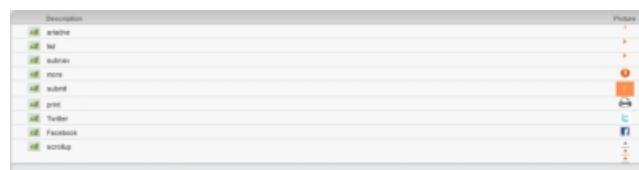


Figure 13

The object structure window is described more detailed in the section editor customizing of that manual.

5.2 Object detail window

In the object detail window, different views of the object can be seen and the object can be edited. It basically consists of three parts: object action menu bar, object tabs and object view or editing.

Information displayed in the object detail window always refer to the object that is currently activated. The activated object is highlighted by a grey background; in the structure area, the object name is displayed and the object structure shows the complete line of the table.

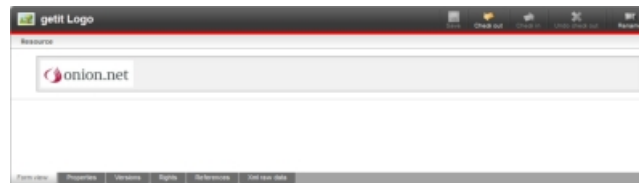


Figure 14

5.2.1 Object action menu bar

With the buttons of the object action menu bar, the onion.net editor fulfils an important function of document management: checking documents in and out to review the consistency of saved information.

- A user first has to take over an object by clicking on **Check in** to edit an object. Checking in an objects blocks editing of this object by any other user and a new object version is generated on which the user can make the desired changes.
- With the button **Save**, the user can temporarily store his changes so that they will not be lost if he changes to a different object or if the connection to the onion.net editor is interrupted, for example due to a disturbance of the internet connection.
- As soon as the user has finished his changes, he can complete the editing process of this version via the button **Check out**. At the same time, the object becomes available to other users who can then check it in it again.
- If, however, the user decides not to check out his changes, he can check out and return the object to the state it had before checking in by clicking on the button **Undo check out**. Even temporarily saved changes are irrevocably lost by this action. The object is then available for changes by other users.
- The selected object is renamed via the button "Rename" and is the only way of changing the name of an object other than through access via the context menu.



Figure 15

onion.net editor enables these buttons only if the function is available, i. e. the button **Save** is not enabled as long as the object has been checked in by you or any other user.

5.2.2 Object tabs

The object tabs can be used to call different views of the object. Six views are available that are described in the following paragraphs.

5.2.2.1 Contents

The object tab "Form view" serves for creating and editing the contents of an object. To do this, the user can make us of an input mask. Structured and weakly structured contents can be entered.

- Structured contents are data that are stored in a standardised structure or that can be made available from such a structure. Contacts within a contact administration are structured data because they usually follow a common scheme. The first name, surname, address consisting of street, house number, zip code and city, e-mail

address, telephone number etc. are to be entered. The editor who creates a contact does not have any possibility to leave this scheme because contacts are set up like this and in no other way.

- Weakly structured contents are data that are not standardised but freely combined from a certain number of information units.

onion.net editor enables users to enter structured contents via progressive forms and weakly structured contents via a rich text editor. It is possible to combine both input types with each other.

5.2.2.1.1 Progressive forms

A form is a standardised means for entering data. Forms make the collection of (mass) data easier, provide for completeness as well as data integrity and help prevent ambiguities that may arise if there is a free choice of wording or a formless request. For this reason there is no information system that can get around forms where entering contents into the system is involved. However, like in the case of official forms, a user is frequently faced with a number of fields and widgets and has to decide, without sufficient assistance, what he needs to do and fill out in order to get to where he wants.

onion.net also uses forms for data acquisition, although in a different way to comparable systems. The onion.net editor only displays the mandatory fields first of all, i. e. the fields that must be filled out in order for the document to be stored. Selection fields and optional fields must be enabled explicitly. The form therefore grows in the directions corresponding to the contents to be entered.

This should be made clearer with an example. When entering contact information, the sex as well as the first name and surname are given as mandatory fields. Additional optional information can also be entered, including the position of the contact person as well as some business contact details.



Figure 16

Which fields are mandatory and which are optional is determined in the [schema](#) that each type of object is subject to and on the basis of which the respective progressive forms generate themselves automatically.

5.2.2.1.2 Rich text editor

Part of the data acquisition in onion.net editor takes place via the flexible progressive forms. A by far greater part, however, is done using the rich text editor, which is also frequently referred to as “WYSIWYG editor”.

5.2.2.2 Object properties

The object tab “Object properties” lists seven attributes of an object. Table 1 explains these attributes.

ID	A natural number is allocated to every object created in onion.net. For every new object, the number is incremented by 1. Thanks to this, every object can be unambiguously identified in the system.
Created by	Name of the onion.net user who created the object.
Created on	Exact date on which the object was created.
Schema	Every object is of a specific type defined within the framework of the information architecture. The name of the type is indicated here.
Editor	Name of the onion.net user who last changed or checked in the object.
Last change	Exact point of time of the last change of the object.
Version	The current version. All previous versions can be seen via the tab Versions.
Changes in ChangeSet	Lists all ChangeSets, where the objects was modified.

5.2.2.3 Versions

The object tab "Versions" lists all previous versions of the object, in fact all versions from the very first one that was created to the most recent (current) one are listed. Every version can be retrieved, i. e. the current version will be replaced by the archived version, but only if the following two conditions are fulfilled:

1. The old version passes the validity check against the current schema. If new mandatory elements or attributes have been defined in the meantime or other changes of the structure have been made, this old version cannot be restored.
2. The referential integrity is not violated, i. e. the archived version does not contain any references to objects that have been deleted in the meantime.

Via the link "Compare", online editors can examine what changes have been made to the object in the meantime. The relevant version is always compared with the current version. A new tab opens for this, where the version to be compared can be seen on the left-hand side and the current version on the right. This view allows the user to compare both statuses. There is the form view here plus the XML view. If one of the conditions specified above does not apply to the old version, then only the XML view is available.

5.2.2.4 Rights

The object tab "Rights" lists all groups stored in the system including their rights concerning the object. With this view it is possible to see to which groups a user must belong in order to be able to see or edit the object or. More information on this can be found here.

5.2.2.5 Referenced by

The object tab "Referenced by" lists all objects containing a reference to the current object. One object is referenced by another if it is linked to it in one of its reference or free text fields (see Chapter and Chapter 8.3.5). As long as there are entries in the reference list, the object cannot be deleted.

5.2.2.6 XML view

All objects recorded in onion.net are available in an XML structure described by the corresponding schema. On the tab "XML view", the XML structure of the object can be seen. This view gives you an insight into the data quality and serves as information source for transformation and application developers using these pieces of information.

5.3 Search

Online editors open the onion.net Editor search via the context menu item "Search". Online editors can perform a full text search over the entire data stock using the tab "Search". The editor can choose whether to only search by name or within all object contents. Table 2 lists the possible restrictions.

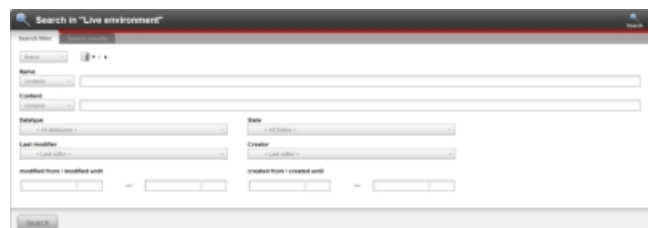


Figure 17

Restriction	Explanation
Subordinated	At this location, the object appears whose context menu was used for calling the research window. The full-text search is restricted to objects subordinated to this object. If the research window is called from the system root object, the complete database will be searched.
Referenced	Objects referencing at least one object of this path are found.
Exact path	Only objects are searched that are direct children of the selected element.
Data type	The selection menu contains all object types defined in the model administration. Abstract types appear in grey font. If a data type is selected, only objects of this type are searched.
Status	The search is restricted to objects that have been borrowed or checked in - depending on the selection.

Last editor	The selection menu lists all users stored in the system. If a user is selected, only objects will be searched that have last been edited by this user.
Editing period	The search is restricted to objects last edited within the indicated period of time. The period of time is to be entered in the format YYYY-MMM-DD or DD.MM.YYYY. If editors do not indicate the start date, the installation date of the system will be used; if they do not indicate the end date, the current date will be used.
Creator	The selection menu lists all users stored in the system. If a user is selected, only objects will be searched that have been created by this user.
Creation period	The search is restricted to objects created within the indicated period of time.

All restrictions can be combined with each other. A click on the link "Checked-in documents" on the start page (see Chapter 3.3) thus only initiates a combined restricted search for objects that

- have the status "checked" and
- that have last been edited by the user who is currently logged on.

Following a click on the button "Start search", all hits are displayed in the bottom part of the research window. The name, path, creator and last editor of the documents are displayed. A click on any object in the hit list opens this object in the onion.net editor.

5.3.1 Recycle bin

The recycle bin is only available in the productive environment and is called via the context menu of the structure area.

The research window also opens with a click on the recycle bin. The difference to the search of documents currently existing in the system is that deleting processes instead of individual documents are searched. A deleting process may concern several documents. The search restrictions refer to all documents concerned by the deleting process. Documents can be restored if they match the structure and still correspond with the model. Rights of documents in the recycle bin have expired, i. e. the user who retrieves a document will be the creator of this document.

5.4 Preview

For some onion.net projects, onion.net is a completely sufficient system to record and administrate information. In most projects, however, data is to be processed in a specific way, e. g. as HTML or PDF output. This task is performed by transformation developers via the onion.net render engine components. They create methods for transforming data into the desired type of output.

Transformation developers can create a preview for every object type. In this preview, data will be processed and displayed in any specific way defined by the transformation developer.

If a preview is available for objects of a type, then a preview button will appear over the context menu of the object. The preview opens in a new window.

6 Data administration

The data administration is available to the administrators in the Editor. This area corresponds to that of the editing system in terms of the way it works. The difference is in its use however. There is no convenient form available to the user for editing data objects as in the editing system. The data objects can be edited here in their raw form as XML.

Moreover, no Editor configurations are consulted in this view for the representation of contents, meaning that data can be seen even in the case of misconfiguration.

7 User and group administration

With the user and group administration, user administrators define groups, users and access rights. Every user who wants to work with onion.net editor or access onion.net objects via the available interfaces must be created as user in the onion.net editor and allocated to a group with the respective rights.

7.1 Navigation window

Like the content administration, the workspace consists of a structure tree, an optional object structure window and the object detail window. Only the structure area of the user and group administration's navigation window differs slightly from the one of the content administration.

- The basic structure is predefined, i. e. it cannot be changed, and distinguishes between user administration and group administration.
- Drag & drop does not work within the structure tree; the objects are sorted automatically.

The following paragraphs describe how user administrators work with the user and group administration.

7.2 User administration

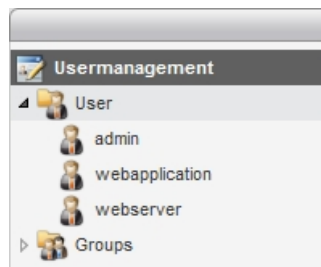


Figure 18

In the user administration, all users recorded in the system are listed in alphabetical order. If the user administrator clicks on a user, detailed information on this user will be displayed in the object detail window.

7.2.1 Creating a user

User administrators have several options to create new users: via the context menu of the object "Alphabetically" in the user administration or via that of the groups in the group administration. User administrators must at least indicate a user's access data to be able to save this user.

The object action menu bar only offers the possibility to save the user. There is no versioning and thus there are no buttons for checking in, checking out and undoing check out. A preview of a user is not possible either.

7.2.2 Editing a user

Any user has access data, roles, group memberships and, if and as required, also user profiles.

7.2.2.1 Access data

Access data comprise the user name and a password that cannot be seen. It is stored in the database in encrypted form, but it is possible to change it. This can be effected with the two fields "New password" and "Confirm password". User administrators enter the same password in both fields and then click on the button "Save". The password has now been updated.

7.2.2.2 Roles

Users can have different roles independent from their group memberships. Roles are "global rights" referring to the complete system.

- **Editor:** An editor is a user who has the right to log on to the onion.net editor. Every created user is automatically an editor.
- **Administrator:** Administrators have full access to the system. They have the right to see, edit and delete all objects. This role comprises the roles editor, user administrator and schema administrator. Administrators have the right to allocate rights to other users.
- **User administrator:** User administrators have full access to the user administration. Moreover, they have an extraordinary right: User administrators (and thus also administrators) have the right to check out objects or undo the checkout of documents checked in by other users.
- **Schema administrator:** Schema administrators have full access to the model administration.

- **Application server:** The role application server has an exceptional position because it is a technical role. No (natural) person should have this role; only applications requiring read access on the content data should operate as application server.

As a matter of principle, no user has the right to change his own role or the role allocation of users of the same rank. Example: A user with the role administrator cannot allocate to or delete the administrator role of another user. Only the system administrator (user name "admin") has the right to do this.

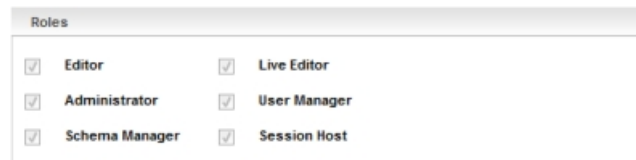


Figure 19

7.2.2.3 Group memberships

Users can be members of any number of groups. If a user is not a member of a specific group and not an administrator, he cannot log on the onion.net editor.

User administrators add a group to user memberships by dragging & dropping, i. e. the group is dragged from the group administration directly into the grey area and thus referenced. A group can be deleted via the context menu (click on the group name with the right mouse button).

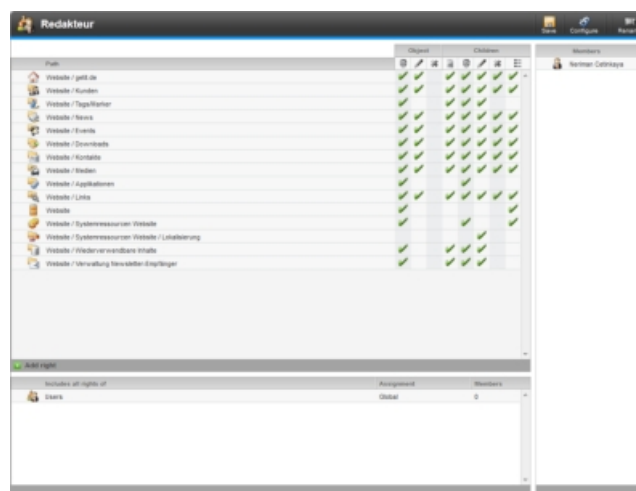


Figure 20

7.2.2.4 User profiles

User profiles are optional information which is organised into profile tabs. The information architect influences whether user profiles can be created. User profiles can also be created underneath it which are not offered in the context menu. In this case, these profiles are mostly generated.

If a new user is created, all its possible profiles are not available to start with. A new user profile can be created via the context menu of a user. A user profile can be removed again via the context menu of the profile. Profiles are edited like data objects in the editing system - a schema is also the basis from which a progressive form is generated.

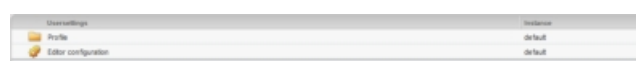


Figure 21

7.2.3 Function for limiting the amount of users

The total amount of all users can be limited or a specific user be looked for via the context menu of the user administration tab. To do a search, just enter the name or the ID of the user into the input field “Name or ID”. In this field you can look both for the ID of a user and for users beginning with the entered search word. All users fitting the search pattern will then be automatically listed under the search field.

7.3 Group administration

All groups recorded in the system are listed in the group administration. There is a distinction between Alphabetical and Hierarchical here. Both views are equivalent! Each group can be found in both views, but are laid out differently however. While all groups are shown alphabetically and on one level under Alphabetical, they can be represented in a tree structure under Hierarchical. If a group includes the rights of another group, then the former is shown under the other group in the hierarchical view.

If the user administrator clicks on a group, then its detailed information is shown in the object detail window.

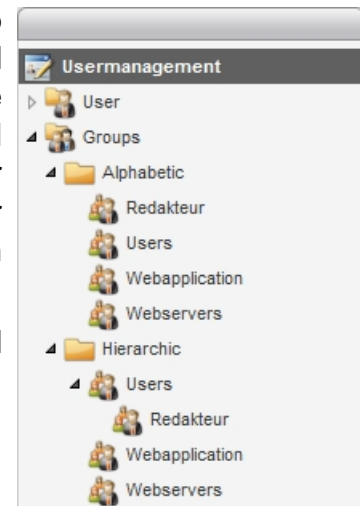


Figure 22

7.3.1 Creating a group

A group is created via the context menu. In contrast to the alphabetical view, the hierarchical view offers the option to subordinate groups to other groups. Every group must have a unique name.

The object action menu bar of the group administration is also restricted to the function Save, because groups cannot be archived like users.

7.3.2 Editing a group

A group comprises rights and members.

7.3.2.1 Rights

Rights are rules of system access control deciding if and how a user or application may perform operations with an object.









Rights can be managed for two different kinds of objects in the onion.net system. They can either refer to data, or to ChangeSets. Rights with respect to data control whether a particular data object, or its descendants can be read, edited, or deleted. Rights with respect to ChangeSets control whether a particular ChangeSet can be read, edited or published.

When new users are created in the onion.net system, they only have the rights connected to their role at first (see Chapter 6.2.2.2). The users do not have any other rights going beyond these rights of their role. A user administrator cannot allocate more rights to a user. Instead, the user administrator can allocate users to a group and allocate the respective rights to this group. The rights of this group apply to all of its members. This procedure facilitates the

administration of rights because in case of changes in the structure of rights only the rights of a group need to be adjusted and not the ones of every individual user. Users are allocated to one or several groups and thus receive their rights indirectly.




The allocation of rights is effected on the basis of objects. User administrators define group rights for specific objects. In doing so, distinctions are made between object rights and descendants' rights. Object rights refer to a specific object, descendants' rights to its descendants. Table 3.1 lists all rights for an object that can be allocated to a group. Table 3.2 lists all rights for a ChangeSet that can be allocated to a group.

Table 3.1

Symbol	Right	Description
	Reading an object	Group members have the right to see the object, i. e. they can see it but they cannot edit it.
	Editing an object	Group members have the right to edit the object, i. e. they can check it in, save it etc.
	Deleting an object	Group members have the right to delete the object.
	Create descendants	Group members have the right to create descendants of this object.
	Reading descendants	Group members have the right to see descendants of this object.
	Editing descendants	Group members have the right to edit descendants of this object.
	Deleting descendants	Group members have the right to delete descendants of this object.
	Listing descendants	Group members have the right to list descendants of this object. This is a special reading right. Group members only have the right to see the children of an object for which they were explicitly given the


right "Reading an object". All other child objects cannot be seen.


Table 3.2

Symbol	Right	Description
	Read ChangeSet	Group members have the right to see the ChangeSet, i. e. they can see it and enter it, but they cannot edit it.
	Edit ChangeSet	Group members have the right to edit the ChangeSet, i. e. they can edit the meta-data.
	Publish ChangeSet	Group members have the right to publish or delete the ChangeSet .

The object detail window gives a clear overview of the rights of a group for a specific object. Every line shows the rights of an object with its icon and path being displayed in the first column. For every right, the user administrator can indicate one of three states (see Table 4) by clicking in the table cell until the desired symbol appears.

Table 4

Symbol	State	Description
	Inherit the right by structure	Rights are inherited in a standardised way. If, for example, group members have been allocated the right to read the children of an object, this right is also inherited to their children, i. e. group members are allowed to read the children's children etc. Due to this, we speak about descendants' rights and not only children's rights. Inherited rights can be overridden if a right is explicitly allocated to or withdrawn from an object.
	Allocating a right	The group explicitly receives the corresponding right.

	Withdrawing a right	The group does explicitly not receive the corresponding right.
---	---------------------	--

7.3.2.1.1 Adding and deleting rights

User administrators add a new right via the context menu with a click on the grey area in the rights field with the right mouse button. Subsequently, they can select the path to the desired object in the menu bar appearing at the lower bottom of the rights field. With the button "Add path", user administrators can add another line to the table of rights and allocate rights to this object.

User administrators can delete a right via the context menu with a click on the respective line in the table of rights.

7.3.2.2 Inheriting rights of other groups

The field "Comprises all rights of" lists all groups who have the rights of the group that is currently being edited. In other words, you can imagine that the table of rights of this group contains all lines of the groups whose rights it comprises and that the table of rights is simply extended by more lines.

Groups are referenced by dragging a group and dropping it in the field. They are deleted via the context menu.

7.3.2.3 Members

The field Members lists all users who are members of the group. Users are added by dragging & dropping and deleted via the context menu. If a group comprises a large number of members, they appear on several pages.

If a user is dropped in the member field, the group also appears in the group membership field of the user.

8 Module system

The module system bundles the tasks for system configuration and the onion.net development in one environment.

System configuration

The system configuration can be found in the main menu ("+") under the section "Administration" after the module system has been activated. Using this configuration, administrators can make general settings for the editor.

General

A project logo and the project name can be assigned in the system configuration under the tab “General”. The logo is shown on the login screen and in the “welcome” widget. The project name can always be seen in the title or tab of the browser.

Furthermore, backgrounds for the Dashboard can be administrated in this tab.

All backgrounds are shown in the backgrounds overview which are provided by modules in the system or were created in the system configuration.

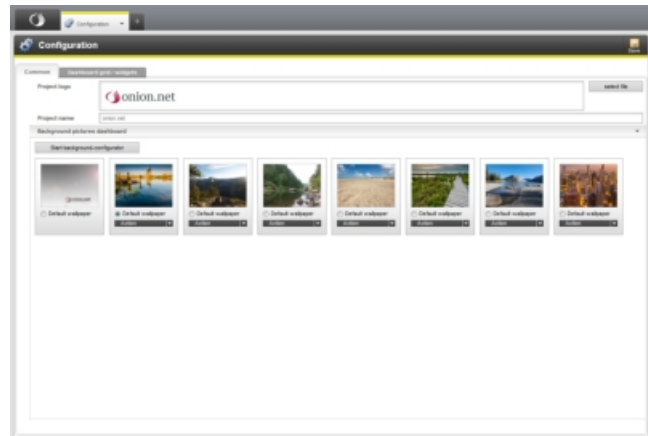


Figure 36

Under each preview picture there is the option “default background”. With this option, the default background can be configured for users who have not yet logged into the editor until now. This option no longer applies for users who have already successfully logged into the editor. After the user has logged in, the configuration of the background is stored in the profile of the user. If the user would like to use a different available background picture, then he can change it via the context menu of the Dashboard tab under the point “Background pictures”.

There is also the selection field “Actions” for background pictures. This field is not available if the background picture comes from a system module. Available actions are “Edit” and “Delete”. At this point you have the possibility of retrospectively adjusting the background picture in the configurator. If you do not wish to activate individual background pictures for use, you can simply delete the background picture.

Background configurator

The background configurator offers the possibility of creating or editing backgrounds for the Dashboard. The user has the option of either uploading a picture or indicating a colour gradient. A logo can also be uploaded, which is positioned on the lower right-hand side. When changing pages in the Dashboard the logo moves parallax to the background picture.

The configurator is divided up into two tabs. It can be decided in the configurator whether a picture or a process is to be maintained. Both at the same time is not possible for a background picture.

If a background picture is to be uploaded, then this is done in the first tab. After uploading, the picture is shown in a preview as well as the form. If you also upload a logo, then this will be shown and positioned in the preview in proportion by way of example.

When selecting the background picture you should ensure an adequate resolution. Since the picture is not scaled in the Dashboard, it should be selected in such a resolution that the editors cannot see the edges of the picture.

A further option offers the possibility of aligning pictures vertically. In the case of lower screen resolutions, this option can ensure that objects shown in the picture do not disappear from the viewing area of the editor. For example, the object may be visible on the lower third of the picture with a high screen resolution and then be no longer visible with lower screen resolutions. With the vertical adjustment configuration, the picture can be aligned lower down and the object in the picture will be in the viewing area regardless of the screen resolution.

If no suitable background pictures are available, then a colour gradient can be configured. An initial colour can be indicated which is to run into a second colour. Horizontal or vertical can be indicated as the gradient alignment. No more possibilities are offered at this point due to browser compatibilities.

Dashboard raster/widgets

This section deals with the configuration of the Dashboard. The raster size of the Dashboard can be configured once. The raster size applies for all pages in the Dashboard and cannot be adapted specifically for the user. And then an initial arrangement of widgets on the Dashboard can take place. In doing so, widgets can be configured on the pages of the Dashboard that are to be loaded when a user logs into the editor for the first time. After the login, these settings for widgets become the settings of the user, meaning he can position and configure the widgets as he requires.



Figure 37

The view of the configuration is divided into the following parts: The overview of all installed and activated widgets on the left, the control panels for the raster size of the Dashboard at the top and a simplified preview of the Dashboard in the central part.

The preview of the Dashboard is intended to show a reduced-size representation of your own window. The proportions are calculated on the basis of the selected screen resolution.

The preview works in a similar way to the live Dashboard. Widgets from the list can be dragged and dropped into a free space in the raster, and the position and size of a widget changed by dragging and dropping as usual. Since only an impression of a widget is shown in the preview, a button has been placed on the upper right-hand side of a widget for deleting it from the Dashboard.

Furthermore, the size of the raster can be configured. For this purpose, the number of lines and columns in the control panels can be entered or set via the slider. The raster in the preview adjusts according to the data entered. In this way, the optimal raster size can be determined for the lowest resolution used.

8.1 Module overview

The module overview allows you to cast your eyes over all modules present in the system. Modules can be read from an assembly, from the file system or from the onion.net data repository. The modules are listed in the overview in the following order:

- > onion.net data repository
- > Assembly
- > File system

The module overview also offers different functionalities. For example, you can call the functions “install module”, “create module” and “export modules” via the button toolbar. Modules from the onion.net data repository can also be deactivated and deleted.



Figure 38

Install module

This functionality serves for installing new modules, as well as for updating existing modules. Clicking on the associated link will open a dialogue for downloading a module package. Once this has been successfully downloaded, the contents of the package will be clearly shown and you will have the possibility of selecting the modules to be installed, with the appropriate feature.

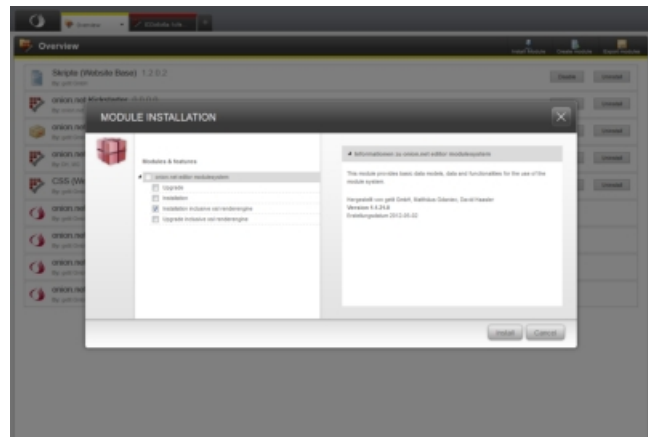


Figure 39

Create module

The dialogue for creating a module offers the possibility of maintaining a name, the author, the default language and a description. A description is not absolutely necessary for the creation of a module. After this has been successfully completed, you are taken to the module features page and can start working in a module.

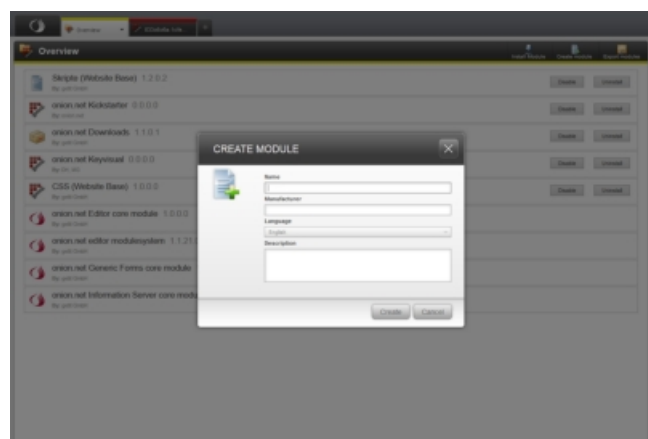


Figure 40

Export modules

This functionality serves for exporting a module package. A module package can consist of one or more modules. The modules can be secured when exporting. This means they cannot be edited in the system they are being imported into.

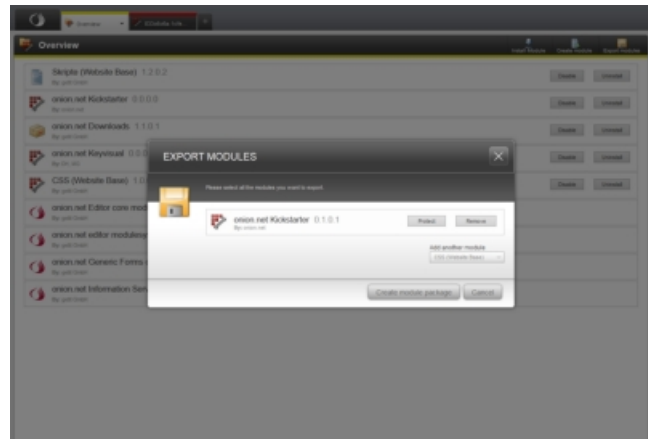


Figure 41

Deactivate module

Modules located in the onion.net data repository can be deactivated. You will find a control element for this purpose by the respective module in the module overview. In the case of a successful module deactivation, only the localization and the object structure windows of the module will be deactivated.

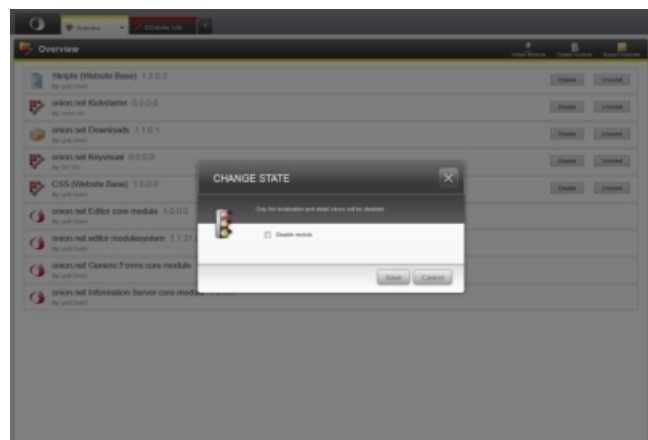


Figure 42

Delete module

Only modules can be deleted in the editor which are located in the onion.net data repository. Modules from the file system or an assembly must be deleted manually in the appropriate place. There is a control element in the overview of the respective module for deleting a module in the onion.net data repository. When deleting a module, all schemas and data objects of the module are deleted.

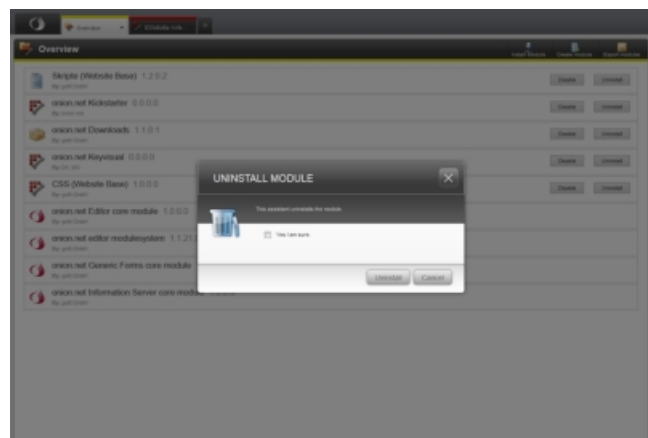


Figure 43

The starting node is the module which contains the meta information on the current module. Underneath the module, different module areas are administrated. If areas are not visible, they can be shown via the context menu.



- > Module settings
- > Configuration
- > Schema administration
- > Data objects
- > Users / groups
- > Transformations
- > Object structure windows
- > Interfaces
- > Editor functions
- > Components

Web server settings

LRU slots

The module

- › Manufacturer, authors
- › Logo
- › Version
- › Features
- › Dashboard background pictures

- Project-specific files
- Localization

Version

The structure of the version number is based on the general use of version numbers. The meaning of the individual positions is described on Wikipedia for [version numbers](#).

The creation of the version number can therefore be partly supported in the onion.net Editor.

The **major version number** is manually maintained by the editor. For this purpose there is the point “Version” in the tab “General” of the module node. The number can be increased or reduced there. If this number is changed, then the minor version number is reset to zero at the same time. The major version number is of no technical significance. It should be changed however if the module is a new implementation or includes fundamental changes.

The **minor version number** is changed if functional changes are made to the module. Functional changes in a module are changes to the schema or interfaces. Since incompatibilities with other modules can result from these changes, this number must be changed.

The **revision number** is increased when changes are made to transformations. Changes to transformations can have behavioural effects, but should not cause any incompatibilities with other modules.

The **build number** is set every time the module is exported. If a module is exported, then this number is increased by one. The module is normally exported in the “develop” mode, meaning the module is editable when imported into other systems. If the module is exported in the “release” mode however, the build number is set to zero. The mode is decided on at the time of exporting. In the dialogue there is a “protect” button, with which the module is put into “release” mode. However, the module remains on “develop” in the source system.

Features

Features are defined for a module in order to define the scope of an installation. The module developer must select at least one feature for all elements of a module, so that the element is also installed or updated for the selected feature.

So that the module developer does not have to select every feature, features can include other features. This means that the elements defined for the included feature are also available for your own feature.

Furthermore, there is the possibility, thanks to the module function “interfaces”, of defining dependencies on other modules for a feature. This means a module can only be installed if the dependencies are already available in the target system or are included in module package to be installed. So that a dependency can be created, the GUID, installed minimum version and installed feature must be indicated. The GUID is the clear identification for a module in the module system and is shown as a tooltip when you hover over the module node with the cursor.

Background pictures

Each module can provide its own background pictures for the Dashboard. The way the background configurator works is described in the point module system.

Project-specific files

Under this tab, DLLs can be integrated into the project, which provide further project-specific functions.

Localization

In the module node itself, language-independent information is maintained. Since however there is also language-dependent information on the module itself, this is listed in the structure object window below the module. In this list there is already at least one language, which, at the time of the creation of the module, was also created as a default language.

8.3 Module configuration

Finished modules are delivered write-protected. For making application-specific or client-specific settings later on there is the section “configuration”.

Here, the module developer has the possibility of defining configurations. Forms in the editor, workflows, widgets or even the templating in the Render Engine can access these configurations.

Configurations are administrated in the area “schemas”. Module configurations can be created below “schemas”. So the schema has to be directly below element “Schemata” on the root level.

The module configurations work in a similar way to user profiles. At least one instance name must be assigned for module configurations. Furthermore the schema “<http://onion.net/2010/editor/modulesystem/environment/container>” (schema location of the configuration container) has to be added as a structural parent schema.

All non-instantiated configurations of the module can now be created in the “configuration” area via the context menu. If they are all already instantiated, no context menu is displayed.

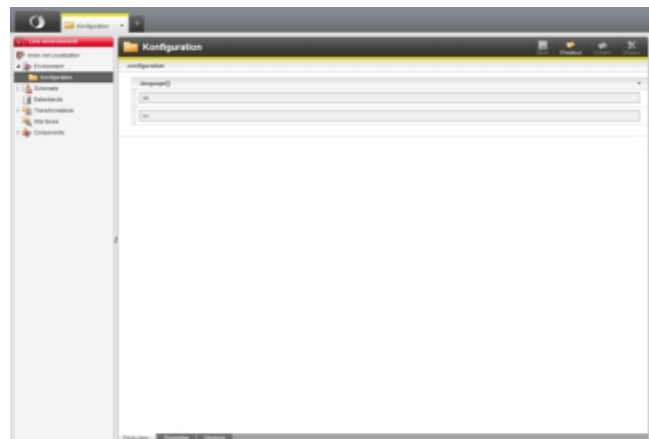


Figure 45

8.4 Administration of schemata

The schemata administration is the tool of information architects. It is used to define object types and their content models. In doing so, information architects define which objects types are available and if and where they can be created in the content administration of the structure area. They also define which data can contain object types and how they are structured. In the content administration, components of the onion.net editor use this information to generate progressive forms for data recording from the content models.

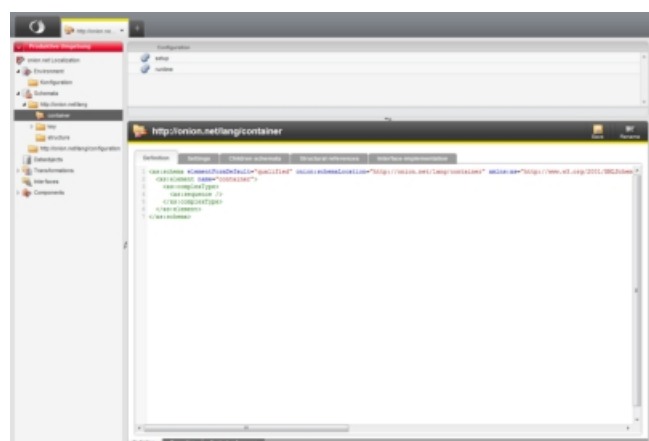


Figure 46

The workspace of the schemata administration also consists of object structure window and a object detail window.

Information architects can create objects of the type schema, user profile and module configuration in the schema administration. New schema types are created via the context menu; further schemas can be created below each schema. If a schema is clicked on, then it is displayed in the object detail window.

User profiles and module configurations extend the type “schema” to include instance names. The workspace of the schema administration consists of an object structure window and the object detail window. The object detail window contains, alongside schema definition, extensive configuration options.

➤ **Definition**

The definition of a schema is captured in the language XML Schema. This is a complex schema language for the description of XML object types.

➤ **Child schemas**

If it should be possible to create other object types underneath an object type in the content administration, these object types must be linked in the field Child schemas.

➤ **Settings**

The settings are subdivided into the tabs “General” and “Localization”.

Under “General”, fundamental settings for the schema can be made. The schema can be marked as abstract, so that no instances can be created for this schema. Furthermore, it can be decided whether the data objects of this type can be versioned and whether the data objects (child schemas) located underneath them are to be sorted alphabetically. If the schema is located in the administration directly beneath the point “Schemas”, the superordinate schema that is not located in the module will also be configured.

In the tab “localization”, the display name and icons for the schema can be assigned. If the display name is configured, then this will be displayed in the editor as a matter of priority, before the system name of the schema. The same goes for the icons. These are displayed in the structure tree or in the tabs for example.

➤ **Child schemas**

If it is to be possible to create other object types below an object type in the content administration, then these must be linked in the tab “Child schemas”. Further schema can be added by dragging and dropping or via the schema selection that can be reached by a

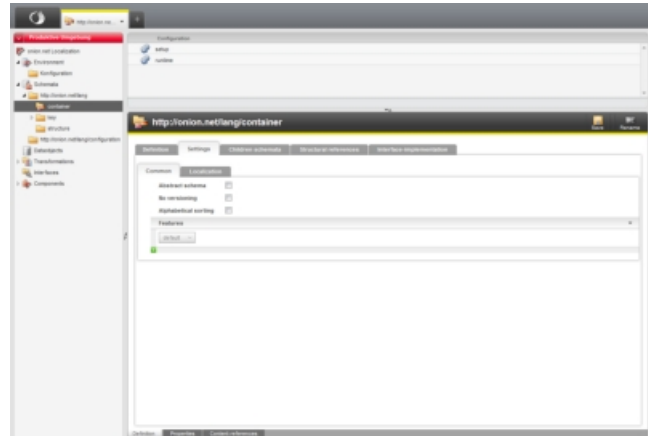


Figure 47

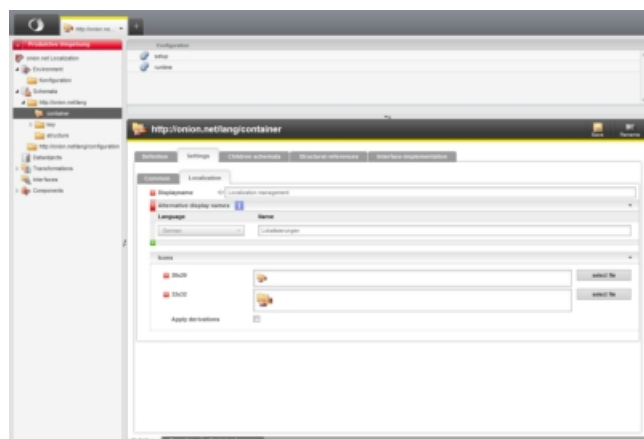


Figure 48

right-click of the mouse. Schemas are deleted via the context menu of the respective schema.

➤ Structure references

The structure references list is the counterpart to the list in the tab “Child schemas”. All schemas are listed with instances under which further instances of the current schema may be created. As in the tab “Child schemas”, further schema can be added or existing schema deleted.

➤ Interface implementation

In this tab, the interfaces are administrated which this schema is to implement. For this purpose, interfaces can either be added to the list by dragging and dropping or by right-clicking. Interfaces are deleted on the respective interface via the context menu.

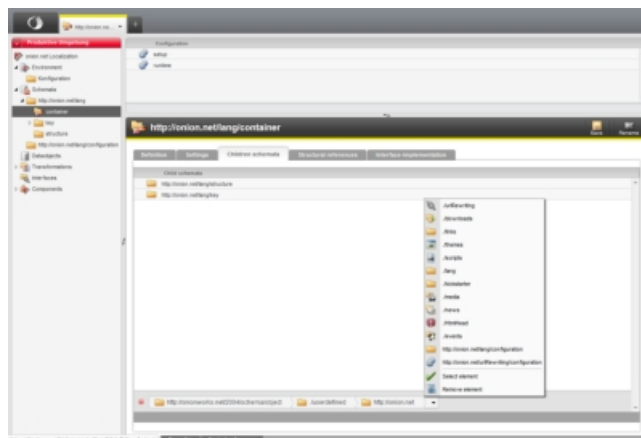


Figure 49

Further objects can be created in the object structure window.

➤ Component binding

Based on the schema definition, the form is generated in the content management system. The default components in the form can be replaced with own components using the component binding.

➤ Localization

The form components are configured via the localization objects. This means that both language-independent settings, such as the width of a selection menu, and language-dependent settings, such as the naming of a label, can be made in these objects.

➤ Update script

The use of update scripts is necessary if the module is already in use on other systems and the current schema has changed meaning that any data there may be must be revised. If the module is to now be updated on the other systems, then the script “upgrade” is executed. If the update of the module fails at a later time, then the script “downgrade” must be executed in order to restore the previous data status.

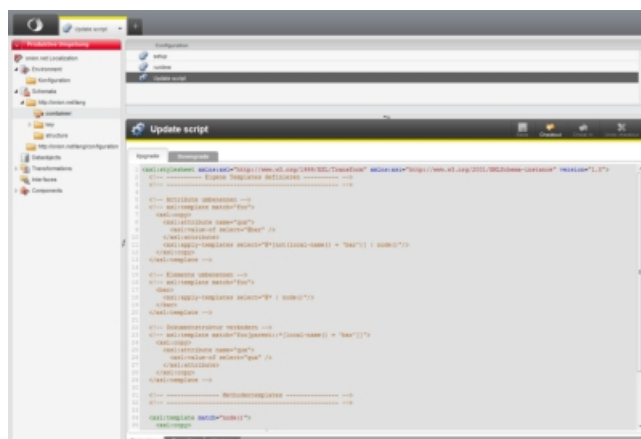


Figure 50

Name conventions

A schema is identified via its address. This must be unique across the system. It has proven successful to name schemas in the URI format.

A schema is renamed via the button “Rename” in the toolbar of the object detail window or the context menu of the schema.

8.5 Data objects

In the document “data objects”, data objects from the content management system can be referenced which are to be exported with the module. This means that required data objects or example data for instance can also be exported with the module.

In order to reference a data object from the content management system, it can simply be referenced by dragging and dropping or be selected via the path selection.

Furthermore, it can be configured for each reference whether only this data object is to be exported recursively or the data objects underneath also.

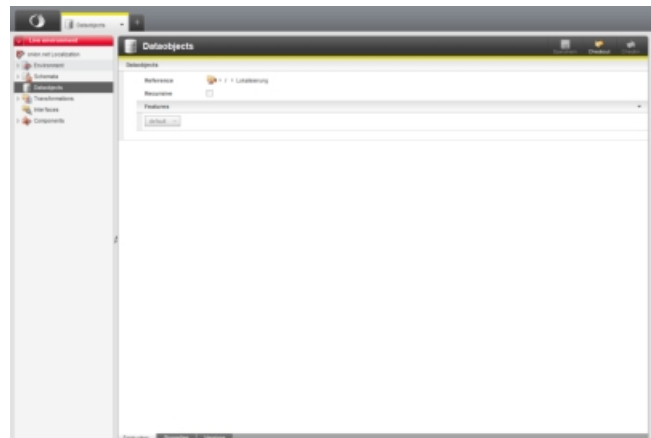


Figure 51

8.6 Users / groups

In the section “users / groups”, users, groups and user profiles can be configured for the module which are to be installed or updated when importing the module.

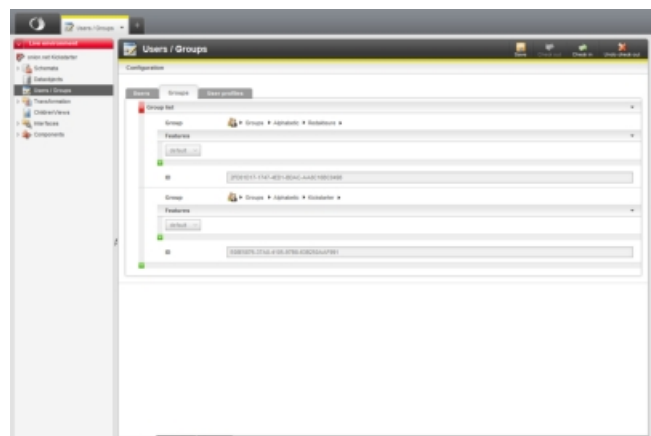


Figure 52

Users

In the “Users” tab, existing users can be selected that are intended to be part of the module. In addition, the password must be assigned so that this user can also be created in the system to be installed. The roles defined in the user are applied too, meaning the user is also given these roles in the target system. It is to be ensured here however that the user who installs the module has the necessary rights for this user to be given these roles.

Groups

In the “Groups” tab, groups are added that are intended to be part of the module. If group includes, members or data objects are present in the target system, these are added to the group.

User profiles

For users, user profiles can also be provided along with the module.

For this purpose, a data object is referenced which is to be used as a user profile. Accordingly, the instance name and the schema name of the data object have to be indicated. So that this profile can be assigned to users as well, user or group bindings can be indicated. Users indicated at the time of user binding are given this profile. In the case of group bindings, all users that are members of this group are given this user profile.

8.7 Transformations

The transformations for the module are created here. These are only valid for the module and cannot be executed by any other module. Everything concerning the module is created here (e.g. the aggregation for the structure abstraction).

It is to be **ensured** that the transformations are integrated in the respective servers, “Preview” and “Live”. This currently takes place via the context item “Configure” on a transformation container.

8.8 Object structure window

Objects can be displayed in the structure area as part of the tree or separately in the object detail window. The object detail window makes it possible to show objects with additional information. It is possible to configure a list view or symbol view for this. The following two advantages can be singled out:

- If an object has a particularly high number of child objects, such as a graphic folder containing several dozen graphics, the tree soon becomes cluttered. Displaying the child objects in the object structure window increases the ergonomics of the Editor.
- Unlike in the tree, additional information about objects can be displayed in the object structure window, such as a reduced-size preview picture for graphics.

The list view and symbol view differ in their representation. The list view is, as the name suggests, a list with additional information on an object. This information may be for example the name, data from the content or meta information. The symbol view on the other hand concentrates on the representation of pictures. Miniature pictures from the content of the individual data objects are displayed here. This means picture collections can be quickly scanned without having to open the individual objects. Of course, the usual functionalities like a sorting & dragging and dropping are available.

Configuration

Object structure windows for different schemas of the module are maintained in a central configuration. This configuration can be created on the module root node in the tree view using the context menu. The node “object structure window” will then be in the tree view.

The configuration of the object structure windows is performed using XML. You will find a detailed description on the configuration [here](#).

8.8.1 Configuration

The configuration of the object structure windows can be performed using XML. The root element “childrenViews” has the following child elements and attributes for this.

Element	Description	Namespace	Number
detailView	Configuration section for a list view	http://onion.net/modulesystem	0..*
symbolView	Configuration section for a symbol view	http://onion.net/modulesystem	0..*

Attribute	Description	Namespace	Optional
language	Default language of the object structure window	http://onion.net/common/18n	No

8.8.1.1 <detailView>

Element	Description	Namespace	Number
columns	Parent element for defining the displayable columns	http://onion.net/modulesystem	1
childType	Configuration section for a child element	http://onion.net/modulesystem	1..*

Attribute	Description	Namespace	Optional
schemaLocations	List, separated by blanks, of the SchemaLocations to which the list view is to be applied		No
applyToDerivations	Boolean value for the inheritance of the list view by the derivations of defined schemas		Yes
order	List, separated by blanks, of the value pairs columnId#sorting direction. For example: "title#ascending creator#descending"		Yes

8.8.1.1.1 <columns>

Element	Description	Namespace	Number
column	Definition of the displayable columns	http://onion.net/modulesystem	1..*

8.8.1.1.1.1 <column>

Attribute	Description	Namespace	Optional
key	Unique key for localizing the view	http://onion.net/common/18n	Yes
label	Label of the column		No
id	Id of the column		No
width	Width of the column. Possible values are %-indications, 'px'-indications or the value 'auto'. The value 'auto' is the default value.		Yes
minWidth	Minimal width of the column. Possible values are %-indications, 'px'-indications or the value 'auto'. The value 'auto' is the default value.		Yes
align	Alignment of the content. Possible values are: <ul style="list-style-type: none"> > left > center > right 		Yes
type	Data type of the content. Possible values are: <ul style="list-style-type: none"> > text > number > image > boolean > dateTime 		Yes

Element	Description	Namespace	Number
column	Definition of the column contents	http://onion.net/modulesystem	1..*

Attribute	Description	Namespace	Optional
schemaLocations	List, separated by blanks, of the SchemaLocations to which the list view is to be applied		No
applyToDerivations	Boolean value for the inheritance of the list view by the derivations of defined schemas		Yes

structureInvisible	Boolean value which controls the display in the tree view	Yes
--------------------	---	-----

8.8.1.1.2.1 <column>

In the content, the content to be displayed of the column selected with the attribute “ref” is defined. All elements from the namespace “http://www.w3.org/1999/XSL/Transform” can be used for this. Moreover, the data view “progressive” of the object and the core functions of the Renderengine are available.

Attribute	Description	Namespace	Optional
ref	Reference to the Id of the appropriate column		No

8.8.1.2 <symbolView>

Element	Description	Namespace	Number
childType	Configuration section for a child element	http://onion.net/modulesystem	1..*

Attribute	Description	Namespace	Optional
schemaLocations	List, separated by blanks, of the SchemaLocations to which the list view is to be applied.		No
applyToDerivations	Boolean value for the inheritance of the list view by the derivations of defined schemas.		Yes
maxWidth	The maximum width of the displayed picture. The default value is 100.		Yes
maxHeight	The maximum height of the displayed picture. If the value is not defined, then the		Yes

current value of the maximum width is taken.

8.8.1.2.1 <childType>

Element	Description	Namespace	Number
label	Label of the picture	http://onion.net/modulesystem	1
source	Picture source	http://onion.net/modulesystem	1

Attribute	Description	Namespace	Optional
schemaLocations	List, separated by blanks, of the SchemaLocations to which the list view is to be applied		No
applyToDerivations	Boolean value for the inheritance of the list view by the derivations of defined schemas		Yes
structureInvisible	Boolean value, which controls the display in the tree view		Yes

8.8.1.2.1.1 <label>

The picture label to be displayed is defined in the content. All elements from the namespace “<http://www.w3.org/1999/XSL/Transform>” can be used for this purpose. Moreover, the data view “progressive” of the object and the core functions of the Renderengine are available.

8.8.1.2.1.2 <source>

Attribute	Description	Namespace	Optional
referenceIdentifier	Reference identifier of the object		No
xpath	XPath to the binary reference of the picture		No

8.9 Interfaces

Interfaces serve, in the module system, for the type-safe definition of object references in the XML schema and for the definition of the type methods to be implemented. For example, a module can be built for a basic web page and make it possible, with an interface, to display user-defined content. Developers could now display their own contents by binding themselves to the interface and implementing the necessary methods.

Create

For interface definition, the creation of a container is necessary. The creation can be performed on the module data object using the context menu. It is then possible to create interfaces using the context menu and deleted them again if necessary.

Configuration

An interface is uniquely identified by its name and is also used with its name within a schema. The configuration of one or more features is necessary however for the module export or import. The definition of methods is purely for information purposes. It should always be performed however. Interface implementations should be carried out according to this definition.

Use

In order to use an interface within a schema, the interface can be dragged and dropped into the type definition of an element or attribute. Interface implementations are performed in the respective schema. You will find further information here.

8.10 Editor functions

Editor functions offer the options of equipping the editor with additional functionalities. Workflows and widgets are available for this, whereby a widget is merely an extended workflow. As a general rule, the implementation and integration of editor functions are distinguished between. The implementation is performed in data objects underneath the container data object. The integration of workflows or widgets is performed in the container data object. In addition to the connection of editor functions, the container data object offers the possibility of integrating additional APIs or script data objects for user-defined displaying.

8.10.1 Binding

Binding workflows and connecting widgets are different. Workflows can be used at a great number of positions in the editor. Widgets however can only be used on the Dashboard. To bind a widget, it is sufficient to reference the widget data object in the tab "Widget" of the editor function data object and set a GUID for the widget. Workflows however need to be bound to a pre-defined position e.g. the context menu. The binding targets described in the two following tables are available for this. The first table lists binding targets that can be used for extending the editor. The second table lists binding targets for system functions, which can be overwritten however.

The display of bound workflows can be filtered further. If a workflow is bound to the data object in the ContentManagement for example, it may be wished to display the item in the case of data objects with a certain schema. For this purpose, there is the possibility of binding a server-side Javascript and defining a function there which performs the filtering. The function must then be called in the configuration point "On" (e.g. "return isRoot();"). The global variable "context" is available for assistance and is described in more detail here.

Binding tables

Binding

changeset.overview.buttontoolbar.left
changeset.overview.buttontoolbar.right
contentmanagement.dataobjects.buttontoolbar.left
contentmanagement.dataobjects.new.buttontoolbar.left
contentmanagement.dataobjects.buttontoolbar.right
contentmanagement.dataobjects.new.buttontoolbar.right
contentmanagement.dataobjects.contextmenu.*
contentmanagement.dataobjects.contextmenu.detailview
contentmanagement.dataobjects.contextmenu.treeview
contentmanagement.index.buttontoolbar.left
contentmanagement.index.buttontoolbar.right
dashboard.paginator
datamanagement.dataobjects.buttontoolbar.left
datamanagement.dataobjects.buttontoolbar.right
datamanagement.dataobjects.contextmenu
datamanagement.index.buttontoolbar.left
datamanagement.index.buttontoolbar.right
schemamanagement.schema.buttontoolbar.left
schemamanagement.schema.buttontoolbar.right
schemamanagement.schema.contextmenu
workspacemanager.add
usermanagement.root.buttontoolbar.left
usermanagement.root.buttontoolbar.right
usermanagement.root.contextmenu
usermanagement.container.users.contextmenu
usermanagement.container.groups.contextmenu
usermanagement.container.groups.alphabetic.contextmenu
usermanagement.container.groups.hierarchic.contextmenu
usermanagement.user.buttontoolbar.left
usermanagement.user.buttontoolbar.right
usermanagement.user.contextmenu
usermanagement.usersetting.buttontoolbar.left
usermanagement.usersetting.buttontoolbar.right
usermanagement.usersetting.contextmenu
usermanagement.group.buttontoolbar.left
usermanagement.group.buttontoolbar.right
usermanagement.group.contextmenu

System binding

changeset.changes
 changeset.commit
 changeset.configure
 changeset.create
 changeset.discard
 changeset.overview.changeapprovalstate
 contentmanagement.dataobjects.checkin
 contentmanagement.dataobjects.forcecheckin
 contentmanagement.dataobjects.checkout
 contentmanagement.dataobjects.compareversion
 contentmanagement.dataobjects.copy
 contentmanagement.dataobjects.create
 contentmanagement.dataobjects.delete
 contentmanagement.dataobjects.destroyversion
 contentmanagement.dataobjects.move
 contentmanagement.dataobjects.order
 contentmanagement.dataobjects.paste
 contentmanagement.dataobjects.rename
 contentmanagement.dataobjects.renametemp
 contentmanagement.dataobjects.restoreversion
 contentmanagement.dataobjects.save
 contentmanagement.dataobjects.spellcheck
 contentmanagement.dataobjects.undocheckout
 contentmanagement.dataobjects.preview
 contentmanagement.spellchecker.execute
 contentmanagement.spellchecker.apply
 contentmanagement.spellchecker.abort
 contentmanagement.spellchecker.dictionary
 contentmanagement.search
 contentmanagement.trash
 contentmanagement.navigatetoschema
 contentmanagement.navigatetodatamanagement
 contentmanagement.workspace
 dashboard.widgets
 dashboard.workspace

datamanagement.dataobjects.checkin
datamanagement.dataobjects.forcecheckin
datamanagement.dataobjects.checkout
datamanagement.dataobjects.compareversion
datamanagement.dataobjects.copy
datamanagement.dataobjects.create
datamanagement.dataobjects.delete
datamanagement.dataobjects.destroyversion
datamanagement.dataobjects.move
datamanagement.dataobjects.order
datamanagement.dataobjects.paste
datamanagement.dataobjects.rename
datamanagement.dataobjects.renametemp
datamanagement.dataobjects.restoreversion
datamanagement.dataobjects.save
datamanagement.dataobjects.undocheckout
datamanagement.dataobjects.preview
datamanagement.search
datamanagement.trash
datamanagement.navigatetoschema
datamanagement.workspace
schemamanagement.schema.create
schemamanagement.schema.delete
schemamanagement.schema.rename
schemamanagement.schema.addtochangeset
schemamanagement.schema.save
schemamanagement.workspace
workspacemanager.activate.modulesystem
usermanagement.user.create
usermanagement.user.rename
usermanagement.user.delete
usermanagement.user.save
usermanagement.usersetting.create
usermanagement.usersetting.delete
usermanagement.usersetting.save
usermanagement.usersetting.navigatetoschema
usermanagement.group.create
usermanagement.group.rename

usermanagement.group.configure
usermanagement.group.delete
usermanagement.group.save
usermanagement.workspace
modulesystem.overview.install
modulesystem.overview.create
modulesystem.configuration.save
modulesystem.module.localization.add
modulesystem.localization.add
modulesystem.localization.fill
modulesystem.localization.delete
modulesystem.localization.rename
modulesystem.interface.implement
modulesystem.environment.create
modulesystem.module.navigateto
modulesystem.module.save
modulesystem.module.setstate
modulesystem.module.uninstall
modulesystem.schema.save
modulesystem.schema.rename
modulesystem.schema.move
modulesystem.schema.newschema
modulesystem.schema.newsetup
modulesystem.schema.newcustomizing
modulesystem.schema.add.existing
modulesystem.schema.delete
modulesystem.schema.addtoschangeset
modulesystem.schema.updatescript.new
modulesystem.schema.updatescript.rename
modulesystem.search
modulesystem.workflow.create
modulesystem.workflow.designer.create.comment
modulesystem.workflow.designer.create.serverscript
modulesystem.workflow.designer.create.clientscript
modulesystem.workflow.designer.create.component
modulesystem.workflow.designer.create.widgetcomponent
modulesystem.workflow.designer.navigate
modulesystem.workflow.designer.modify.serverscript


```

modulesystem.workflow.designer.modify.clientscript
modulesystem.workflow.designer.modify.component
modulesystem.workflow.designer.modify.widgetcomponent
modulesystem.workflow.designer.line.connect
modulesystem.workflow.designer.line.modify
modulesystem.workflow.designer.save
modulesystem.export
modulesystem.webserversettings.add.transformation
modulesystem.webserversettings.configure
modulesystem.lruslots.configure

```

8.10.1.1 Server-side Javascript

In order to filter the display of a workflow, the global variable “context” is available for the Javascript function. The variable “context” supplies the methods described in the following table.

Function	Return value	Description
getTarget()	object	Returns the target of the current workflow.
getApi(string name)	object	Returns a bound API.
getArgument(string key)	object	Returns the given arguments.
log(string message)	void	Writes a log entry.
log(string message, string category)	void	Writes a log entry.
getSession()	JOnionSession	Returns the onion session.

Targets and Arguments

Name	getTarget()	getArgument(name)
changeset.changes	DataObject.ReferenceIdentifier	null
changeset.commit	ChangeSet.Id	
changeset.configure	null	
changeset.create	ChangeSet.Id	
changeset.discard	ChangeSet.Id	
changeset.overview.buttontoolbar.left	ChangeSet.Id	
changeset.overview.buttontoolbar.right	ChangeSet.Id	
changeset.overview.changeapprovalstate		

contentmanagement.dataobjects.buttonbarleft	DataObject.ReferenceIdentifier	
contentmanagement.dataobjects.buttonbarright	DataObject.ReferenceIdentifier	
contentmanagement.dataobjects.checkin	DataObject.ReferenceIdentifier null	
contentmanagement.dataobjects.forcecheckin	DataObject.ReferenceIdentifier	
contentmanagement.dataobjects.checkout	DataObject.ReferenceIdentifier	
contentmanagement.dataobjects.comparison		
contentmanagement.dataobjects.contextmenu*	DataObject.ReferenceIdentifier	
contentmanagement.dataobjects.contextmenuallow	DataObject.ReferenceIdentifier	
contentmanagement.dataobjects.contextmenureview	DataObject.ReferenceIdentifier	
contentmanagement.dataobjects.copy	DataObject.ReferenceIdentifier	
contentmanagement.dataobjects.create	DataObject.ReferenceIdentifier	schema: Schema.Id
contentmanagement.dataobjects.delete	DataObject.ReferenceIdentifier	
contentmanagement.dataobjects.destroyversion	DataObject.ReferenceIdentifier	version: version
contentmanagement.dataobjects.move	DataObject.ReferenceIdentifier	target: DataObject.ReferenceIdentifier
contentmanagement.dataobjects.order	DataObject.ReferenceIdentifier	target: DataObject.ReferenceIdentifier, positioningMode: Before After
contentmanagement.dataobjects.paste	DataObject.ReferenceIdentifier	
contentmanagement.dataobjects.rename	DataObject.ReferenceIdentifier	
contentmanagement.dataobjects.renameprompt	null	
contentmanagement.dataobjects.restoreversion	DataObject.ReferenceIdentifier	version: version
contentmanagement.dataobjects.save	DataObject.ReferenceIdentifier null	
contentmanagement.dataobjects.undocheckout	DataObject.ReferenceIdentifier	
contentmanagement.search	null	
contentmanagement.trash	null	
contentmanagement.workspace	null	
dashboard.paginator	null	
dashboard.widgets	null	
dashboard.workspace	null	
datamanagement.dataobjects.buttonbarleft	DataObject.ReferenceIdentifier	
datamanagement.dataobjects.buttonbarright	DataObject.ReferenceIdentifier	
datamanagement.dataobjects.checkin	DataObject.ReferenceIdentifier null	
datamanagement.dataobjects.forcecheckin	DataObject.ReferenceIdentifier	
datamanagement.dataobjects.checkout	DataObject.ReferenceIdentifier	
datamanagement.dataobjects.comparison	null	

datamanagement.dataobjects.contextmenu	DataObject.ReferenceIdentifier	
datamanagement.dataobjects.copy	DataObject.ReferenceIdentifier	
datamanagement.dataobjects.create	DataObject.ReferenceIdentifier	schema: Schema.Id
datamanagement.dataobjects.delete	DataObject.ReferenceIdentifier	
datamanagement.dataobjects.destroyversion	DataObject.ReferenceIdentifier	version: version
datamanagement.dataobjects.move	DataObject.ReferenceIdentifier	target: DataObject.ReferenceIdentifier
datamanagement.dataobjects.order	DataObject.ReferenceIdentifier	target: DataObject.ReferenceIdentifier, positioningMode: Before After
datamanagement.dataobjects.paste	DataObject.ReferenceIdentifier	
datamanagement.dataobjects.rename	DataObject.ReferenceIdentifier	
datamanagement.dataobjects.rename temp	null	
datamanagement.dataobjects.restoreversion	null	
datamanagement.dataobjects.save	DataObject.ReferenceIdentifier null	
datamanagement.dataobjects.undocheckout	DataObject.ReferenceIdentifier	
datamanagement.search	null	
datamanagement.trash	null	
datamanagement.workspace	null	
schemamanagement.schema.buttoncolorleft	Schema.Id	
schemamanagement.schema.buttoncolorright	Schema.Id	
schemamanagement.schema.contextmenu	Schema.Id	
schemamanagement.schema.create	Schema.Id	
schemamanagement.schema.delete	Schema.Id	
schemamanagement.schema.rename	Schema.Id	
schemamanagement.schema.save	Schema.Id	
schemamanagement.workspace	null	
workspacemanager.add	null	
usermanagement.user.create	Group.Id null	
usermanagement.user.save	User.Id	
usermanagement.user.rename	User.Id	
usermanagement.user.delete	User.Id	
usermanagement.usersetting.create	User.Id	
usermanagement.usersetting.delete	DataObject.ReferenceIdentifier	
usermanagement.group.create	Group.Id null	
usermanagement.group.rename	Group.Id	
usermanagement.group.configure	Group.Id	

```
usermanagement.group.delete Group.Id
usermanagement.group.save Group.Id
```

8.10.2 Implementation of a workflow

The only difference in the implementation of a widget is one additional activity and additional meta information from a workflow.

Create

For creating a workflow and a widget, assistants are available which create a functional workflow or a functional widget. The assistants offer the possibility of creating a workflow or widget by merely setting the name. However, they also offer the possibility of configuring extended settings and of thus setting the meta data of the workflow.

Meta information

Meta information can be maintained directly in the XML of the workflow or widget, as well as in the node “Meta information” below the workflow or widget. Localizations are displayed in the object structure window and localizable settings can be made there.

Activities

Activities can be edited using the Designer or directly in the XML of the workflow. Editing the workflow using the Designer has a number of advantages. For example, if a server activity is created, then the appropriate function for this is automatically defined in the script. The structure of the XML data object can be seen here.

Server activity

A server activity serves for the execution of server-side functionalities. The [Jint](#) engine is available for this. When creating a server activity using the Designer, an associated function is defined automatically. If the Designer is not used, then the function must be created in one of the existing server-side Javascript data objects. The function name must be identical to the Id of the activity. Server-side Javascript data objects can be created underneath the “resources” data object.

Dialogue activity

A dialogue activity serves for the output of information and for interaction with a user in the form of a dialogue. When creating a dialogue activity using the Designer, several automatisms are performed. First, the model is created if there is not one already. A model is necessary for the execution of a dialogue activity. Components for displaying the dialogue are bound to this model. Then, bindings to the model are performed and a component associated with the dialogue is created. The default dialogue component is bound to the document and the newly created component is bound to the root element. If the root element is renamed or if additional components are to be bound, then the XML of the dialogue activity must be changed manually. The structure the XML node is described here.

Widget activity

A widget activity serves for the output of information and for interaction with a user in the form of a widget. When creating a widget activity using the Designer, the necessary widget component is automatically created and bound. The structure of a widget component is slightly different from the structure of the Genericform component and is described in more detail here.

Client activity

A client activity serves for the execution of client-side functionalities. When creating using the Designer, the XML element for the client activity is created. The structure of the XML node is described here.

8.10.2.1 Structure of the workflow XML

The elements underneath the “workflow” root node can occur in any order and number.

Element	Description	Namespace	Number
serverActivity	Configuration section for an activity for calling a server-side Javascript functionality	http://onion.net/system/workflow	0..*
widgetActivity	Configuration section for an activity for displaying a widget	http://onion.net/system/workflow	0..*
dialogActivity	Configuration section for an activity for displaying a dialogue	http://onion.net/system/workflow	0..*
uiActivity	Configuration section for an activity for calling a Javascript functionality	http://onion.net/system/workflow	0..*

Attribute	Description	Only for one widget	Optional
defaultHeight	Default height of a widget	Yes	Yes
defaultWidth	Default width of a widget	Yes	Yes
maxHeight	Maximum height of a widget	Yes	Yes
maxWidth	Maximum height of a widget	Yes	Yes

minHeight	Minimum height of a widget	Yes	Yes
minWidth	Minimum height of a widget	Yes	Yes
icon-20	Icon in the size 20x20	No	Yes
icon-32	Icon in the size 32x32	No	Yes
skin	Appearance of a widget	Yes	Yes
start	Starting activity	No	No

8.10.2.1.1 <serverActivity>

Attribute	Description	Namespace	Optional
id	Id and function name of the activity		No
result:success	Next activity, if the current activity has been successfully executed	http://onion.net/workflow/activity	Yes
result:error	Next activity, if the current activity has not been successfully executed	http://onion.net/workflow/activity	Yes
result:*	Any attribute, which points to an activity that can be called after the current activity	http://onion.net/workflow/activity	Yes

8.10.2.1.2 <widgetActivity>

Attribute	Description	Namespace	Optional
id	Id of the activity		No
definition	XLink to the widget component		No
result:success	Next activity, if the current activity has been successfully executed	http://onion.net/workflow/activity	Yes
result:error	Next activity, if the current activity has not	http://onion.net/workflow/activity	Yes

	been successfully executed		
result:*	Any attribute, which points to an activity that can be called after the current activity	http://onion.net/workflow/	Yes

8.10.2.1.3 <dialogActivity>

Element	Description	Namespace	Number
components	Contains bindings for components	http://onion.net/workflow/	1

Attribute	Description	Namespace	Optional
id	Id of the activity		No
result:success	Next activity, if the current activity has been successfully executed	http://onion.net/workflow/	Yes
result:error	Next activity, if the current activity has not been successfully executed	http://onion.net/workflow/	Yes
result:*	Any attribute which points to an activity that can be called after the current activity	http://onion.net/workflow/	Yes

8.10.2.1.3.1 <components>

Element	Description	Namespace	Number
component	Binding for a component	http://onion.net/workflow/	1..*

8.10.2.1.3.1.1 <dialogActivity>

Attribute	Description	Namespace	Optional
match	Selection of the binding target in the schema for the component (see here)		No
definition	Path or Xlink to a component		No

8.10.2.1.4 <uiActivity>

Element	Description	Namespace	Number
script	Contains the Javascript	http://onion.net/system/work	1

Attribute	Description	Namespace	Optional
id	Id of the activity		No
result:success	Next activity, if the current activity has been successfully executed	http://onion.net/work/activity	Yes
result:error	Next activity, if the current activity has not been successfully executed	http://onion.net/work/activity	Yes
result:*	Any attribute which points to an activity that can be called after the current activity	http://onion.net/work/activity	Yes

8.10.2.1.4.1 <script>

In this element, the Javascript is defined for calling. For assistance, the variable “context” can be used with the following functions.

Function	Description
finishActivity(value)	Changes the activity next called
getModel()	Supplies the model

<code>getSourceWindow()</code>	Supplies the <code>HtmlWindow</code> by which the workflow was executed
<code>getSourceElement()</code>	Supplies the <code>HtmlNode</code>
<code>updateModel(newModel)</code>	Updates the edited model

8.10.2.2 Designer

Using the Designer you can arrange the activities in a clear fashion and functionally interconnect them. The Designer offers certain automatisms which are performed automatically when creating an activity or when connecting activities. You can arrange all activities and connections by dragging and dropping.

Creating an activity

An overview of all instantiable activities can be found in the left-hand area of the Designer. These activities can be dragged and dropped onto the workspace and will thus be created after the name has been successfully entered.

Deleting an activity

In order to delete an activity, it must first be marked. To mark just click on the appropriate activity. If an activity is marked, then you will be able to see an “x” for deletion in the upper right-hand area of the activity. After clicking on this control element the activity will be deleted.

Editing an activity

Right-clicking on an activity opens a context menu with the available functions for an activity. The point “Edit” enables the activity to be edited.

Creating a connection

In order to create a connection, the sender activity may be not marked. When the cursor is moved over an activity, control elements for the creation of a connection are displayed in the centres of the pages. Dragging from these control points will create a temporary connection. If the temporary connection is dropped on an activity, a dialogue will appear for entering the connection name. After the input, the connection will be permanently created.

Deleting a connection

To delete a connection, this must first be marked. To mark, just click on the connection. After clicking, the connection will be shown with a frame and a control element with an “x” will be added for connection. Clicking on this control element will delete the connection.

Configuring of a connection

Right-clicking on a label of a connection will open a context menu with available functions for a connection. The point “Edit” enables the connection to be edited.

8.10.2.3 Widget component

As opposed to a `Genericform` component, a widget component is not bound to a node of the model. The widget component is automatically bound to the entire model. The structure of the widget component differs only slightly from the `Genericform` component. The maintenance is divided into four areas:

General

This tab only contains the GUID of the widget component.

Server component

In this section, the output of the component can be arranged using HTML and XSLT. In addition, the extensions described here are available.

Client component

This section makes the integration of CSS and Javascript possible. There is the possibility of binding a Javascript instance to an HTML element. The definition of a prototype and the binding to the HTML element in the server component is necessary for this. The prototype string consists of the class name of the Javascript class and the text section “.prototype” (e.g. datePicker.prototype). The binding in the server component is ensured by the attribute “data-component”. The GUID of the component must be set as the content of the attribute. You will find a more detailed description of the Javascript prototype here.

Configuration

This section serves for the definition of user-defined configuration values. These can serve for example for transferring a localized text value. The definition is performed using XML. You will find a more detailed description of the XML structure here. Configurations can be created underneath the widget component data object.

8.10.2.3.1 Widget-Extension

8.10.2.3.1.1 chooseString

8.10.2.3.1.2 format

8.10.2.3.1.3 formatDate

8.10.2.3.1.4 formatDateTime

8.10.2.3.1.5 formatTime

8.10.2.3.1.6 settings

8.10.2.3.2 Javascript prototype

Several features and methods are available to the Javascript instance.

Property	Description
context	Context of the widget
htmlNode	Javascript node on which the component is bound. See also http://de.selfhtml.org/javascript/objekte/node.htm
settings	Object with the transferred properties.

Function	Description
init()	If this function is defined, it is called when loading the widget.

resize(width, height)	If this function is defined, it is called when changing the size of the widget.
-----------------------	---

The feature “context” contains the following functions:

Function	Description
finishActivity(value)	Changes the activity next called
getModel()	Supplies the model
getSourceWindow()	Supplies the HtmlWindow by which the workflow was executed
getSourceElement()	Supplies the HtmlNode
updateModel(newModel)	Updates the edited model

8.10.2.3.3 Configuration

<properties>

The element “properties” can contain the following child elements.

Element	Description	Namespace	Number
property	Parent element for defining the configurations	http://onion.net	1..*

<property>

The “property” element can contain the following attributes:

Attribute	Description	Optional
name	Indicates the name of the feature.	No
dataType	Indicates the XMLSchema data type of the feature.	Yes
mode	Indicates whether the feature is available in the client component, in the server component or in both modes. The following values are available for this: <ul style="list-style-type: none"> > Server > Client > ClientServer 	No

localizable	Indicates whether the feature is localizable.	Yes
defaultValue	Indicates the default value of the feature.	Yes
type	Indicates the type of the feature. Features can be user-related or system-related. The following values are for definition: <ul style="list-style-type: none"> > user > system 	Yes

8.11 Components

Components serve for the user-defined output of information in a form generated from an XML schema. Examples for components are a calendar which show up when focussing a date field oder a colorpicker which is displayed when filling in a form field for a color code. A component can be defined for elements and attributes as well as for simple and complex types. There is the possibility of executing server-side transformations for this purpose and of arranging these on the client side with CSS or equipping them with additional functionalities using Javascript. The server-side transformation is necessary for a component. The CSS and Javascript are optional however.

The maintenance of the individual components is performed underneath the container “Components”, which can be seen in the tree view. The binding of a component is performed in the component binding object of the respective schema. The selectors described here are available for this.

The maintenance screen of a component is divided up into the following four categories.

General

The definition of a display name and description is possible here. Moreover, the validity of a component is defined in this configuration section. The component can be valid for the entire set as well as for elements, attributes or simple or complex types. Using the template aspect it can be controlled whether a component is to overwrite only the content area or the label as well. An array and an individual section are differentiated between here. Moreover, a GUID is automatically defined in this section for identifying the component.

Server component

In this section, the output of the component can be arranged using HTML and XSLT. In addition, the extensions described here are available.

Client component

This section makes the integration of CSS and Javascript possible. There is the possibility of binding a Javascript instance to an HTML element. The definition of a prototype and the binding to the HTML element in the server component is necessary for this. The prototype string consists of the class name of the Javascript class and the text section “.prototype”

(e.g. `dateTimePicker.prototype`). The binding in the server component is ensured by the attribute `"data-component"`. The GUID of the component must be set as the content of the attribute. You will find a more detailed description of the Javascript prototype [here](#).

The data binding mode of the component can also be selected in this section. The data binding mode provides information on the kind of further processing of the form data. `"OuterXml"` and `"NodeModel"` can be chosen between. If `"OuterXml"` is selected, then possible child elements will no longer be generated automatically. The component must set the Xml of the appropriate node. This makes good sense in the case of a WYSIWYG editor for example. If the data binding mode `"NodeModel"` is selected however, only the desired content must be set. Generating the display of child elements or attributes is still possible.

Configuration

This section serves for the definition of user-defined configuration values. These can be used for example to transfer a localized text value. The definition is performed using XML. You will find a more detailed description of the XML structure [here](#).

8.11.1 Selectors

In order to bind the component to the XML schema at a certain position, the following selectors are available.

Selector	Description
<code>#complexType</code> or <code> #(anonymous)</code>	<code>complexType</code>
<code>+simpleTypeName</code> or <code> +(anonymous)</code>	<code>simpleType</code>
<code>.elementName</code>	<code>element</code>
<code>@attributeName</code>	<code>attribute</code>
<code>?elementName</code>	<code>any element candidate</code>
<code>~attributeGroupName</code>	<code>attribute group</code>
<code>^groupName</code>	<code>group</code>
<code><.elementName></code>	<code>element reference</code>
<code><@attributeName></code>	<code>attribute reference</code>
<code>choice, sequence, any, all, content, attributes</code>	<code>literals</code>
<code>set</code>	<code>literal for root (!= root element)</code>
<code>selector@namespace</code>	<code>namespace where the selector is declared in</code>
<code>.{namespace}elementName</code> or <code>@{namespace}attributeName</code>	<code>target namespace of the item</code>
<code>selector[]</code>	<code>specifies an array</code>
<code>selector/number</code>	<code>selects the n-th iteration of the selector</code>

8.11.2 Extensions

<http://onion.net/genericforms>

<http://onion.net/genericforms/common>

8.11.2.1 <http://onion.net/genericforms>

8.11.2.1.1 [chooseString](#)

8.11.2.1.2 [format](#)

8.11.2.1.3 [hasLabel](#)

8.11.2.1.4 [id](#)

8.11.2.1.5 [label](#)

8.11.2.1.6 [serialize](#)

8.11.2.1.7 [settings](#)

8.11.2.1.8 [split](#)

8.11.2.2 <http://onion.net/genericforms/common>

8.11.2.2.1 [actionUrl](#)

8.11.2.2.2 [binaryLength](#)

8.11.2.2.3 [binaryMimeType](#)

8.11.2.2.4 [binaryUrl](#)

8.11.2.2.5 [childrenSchemata](#)

8.11.2.2.6 [configuration](#)

8.11.2.2.7 [dataObjectIcon](#)

8.11.2.2.8 [dataObjectPath](#)

8.11.2.2.9 [resourceUrl](#)

8.11.2.2.10 [schemalcon](#)

8.11.2.2.11 [schemaLocalization](#)

8.11.2.2.12 [schemaLocalizationFromObject](#)

8.11.2.2.13 [users](#)

8.11.3 Javascript prototype

Several features and methods are available to the Javascript instance.

Property	Description
ownerForm	Genericform-instance in which the component is used.
readOnly	Boolean value which provides information as to whether the Genericform is editable.
spellChecker	Spellchecker-Instance.
component	onion_GenericFormComponent Instance.
sourceWindow	Javascript window in which the component is used. See also http://de.selfhtml.org/javascript/objekte/window.htm
htmlNode	Javascript node on which the component is bound. See also http://de.selfhtml.org/javascript/objekte/node.htm
dataNode	The Genericform data node which is bound to the component.
locator	Value for unique identification in the XML.
settings	Object with the transferred properties.

Function	Description
data(key)	Enables the calling of status-wide values.
data(key, value)	Enables the setting of status-wide values.
loadState()	If this function is defined, it is called when loading and changing the status of the Genericform.
saveState()	If this function is defined, it is called when saving the Genericform.
destroy()	If this function is defined, , it is called when tidying up the Genericform.

8.11.4 Configuration

<properties>

The element “properties” can contain the following child elements.

Element	Description	Namespace	Number
property	Parent element for defining the configurations	http://onion.net/genericforms	1..*

<property>

The property element can contain the following attributes:

Attribute	Description	Optional
name	Indicates the name of the feature.	No
dataType	Indicates the XMLSchema data type of the feature.	Yes
mode	Indicates whether the feature is available in the client component, server component or in both modes. The following values are available for this: <ul style="list-style-type: none"> > Server > Client > ClientServer 	No
localizable	Indicates whether the feature can be localized.	Yes
defaultValue	Indicates the default value of the feature.	Yes

type Indicates the type of the Yes feature. Features can be user-related or system-related. The following values are for definition:

- > user
- > system

9 Rich text editor

Via the rich text editor, online editors can record contents without any HTML knowledge like they are used to it in text processing programs such as Microsoft Word.

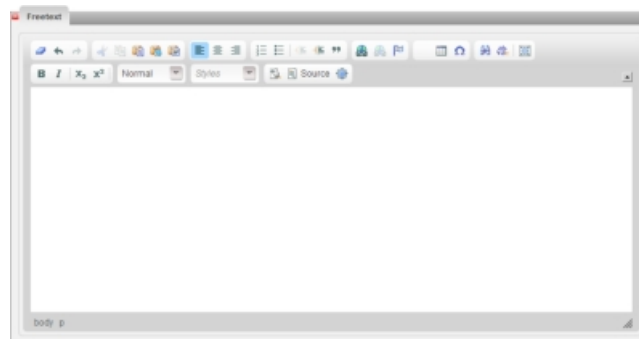


Figure 53

Rich text editors are also called "WYSIWYG editors". "WYSIWYG" is the acronym of the principle of "What You See Is What You Get". With a genuine WYSIWYG, a document is displayed on the screen during the editing processing in exactly the same way as it looks following output via another device, for example a printer. In the context of web development, a WYSIWYG program is a tool depicting an XHTML document during the editing process in exactly the same way as it will later be displayed in the browser.

Microsoft Word is one the best-known WYSIWYG programs. The user first sees an empty white page and starts to edit it. He can format texts, insert images and create tables. The desired contents can be inserted via the relevant menu commands and dialogues. The results of the user's actions can be directly reviewed on the screen.

The onion.net rich text editor works on the basis of the same principle. Transformations in the background transfer the actions of the user into source text. If the user does not want it, he does not have to leave the area of graphic depiction. Optionally, he also has the possibility to switch between layout and source text views so that the development of the page can also be followed directly in the source text; corrections and changes can also be made there.

Editing with the rich text editor offers a number of advantages:



- > Users are rapidly able to enter contents.
- > The transcription from the abstract source text level to the page depiction in the browser is not necessary.
- > The rich text editor can be used without any HTML knowledge.

9.1 Tool bars

The rich text editor has tool bars that can be configured; their buttons make all formatting options of the editor available. The standard variant of the rich text editor offers the buttons listed in Table 5. Other buttons can be added.

Icon		Description
	Delete formatting	The formatting of the highlighted text is deleted.
	Undo	The last action is undone.
	Redo last undo	The last action that has been undone is redone.
	Cut	The highlighted contents is moved to the clipboard.
	Copy	The highlighted contents is copied to the clipboard.
	Paste	Pastes contents from the clipboard.
	Insert as text	Inserts context as mere text without formatting.
	Insert from MS Word	Inserts contents from Word; formatting is maintained but will be adjusted.
	Left-aligned	Aligns the text paragraph to the left-hand margin.
	Centered	Aligns the text paragraph to the center.
	Right-aligned	Aligns the text paragraph to the right-hand margin.
	Numbered list	Inserts a sorted list.
	List	Inserts an unsorted list.
	Decrease indent	Decreases the left-hand side indent of the highlighted contents

	Increase indent	Increases the left-hand side indent of the highlighted contents
	Quotation block	Tags a text paragraph as quote.
	Add / edit link	Formats the highlighted text as hyperlink
	Delete link	Deletes the highlighted hyperlink.
	Insert / edit anchor	Inserts an anchor to be able to directly jump to a specific location via a link.
	New image	Inserts a local image.
	Table	Inserts a table.
	Insert / edit special characters	Shows a list of special characters that can be inserted in the text.
	Tag language	Tagging of the language of the highlighted contents
	Search	Searches the text
	Replace	Searches a specific text and replaces it
	Select all	Selects the complete contents of the text field
	Bold	Tags the highlighted text as bold.
	Italics	Tags the highlighted text as italics.
	Subscript	Tags the highlighted text as subscript.
	Superscript	Tags the highlighted text as superscript.
	Format	Selects a paragraph template for the current template.
	Show blocks	Highlights block elements in the text with a frame.

 Quellcode	Source code	Changes to the source text view of the text.
	Maximise	Increases the editor's view to the maximum available surface.

9.2 Properties' dialogues

For some formatting, for example tables or images, properties' dialogues can be called. Online editors can indicate additional properties and attributes. The offered options are manifold and range from the definition and formatting of font types and colors, frames, spaces and positioning. Such optional configurations are generally not supported in the standard setting of the rich text editor. The reason for this is the separation between visual and semantic formatting that is explained in the following chapter. In this chapter, we also explain which properties and attributes are available to which type of formatting.

9.3 Correct use of text tagging

Online editors can make use of numerous formatting options to tag and structure texts in a reasonable way. In doing so, editors should always stick to the following rule: the rich text editor does not format contents visually, but semantically.

Semantics are a subdiscipline of linguistics dealing with the analysis and description of the meaning of language and linguistic expressions. With their formatting, online editors make a statement on the meaning of contents at this specific location. How the content is optically presented in the rich text editor plays a minor role. The information content of a text is separated from its appearance. Highlights make a statement on the type of highlighted location, for example "this is a heading" or "this is a quotation".

Every online editor should always ask himself the following questions when he wants to record contents:

- What do I want to communicate?
- Which type of information do I have here?
- How can I structure my information in the easiest and clearest way?
- What is the meaning or function of my contents?
- What type of tagging is the best way to describe the contents?

The question "What should my contents look like?" should not be posed!

The advantage of this procedure is an improved processing of the recorded contents. These contents are usually transferred into a target format, for example HTML for the output on a website or as PDF. This transfer is effected by transformation developers. They are independent from semantic structures in order to correctly transform the recorded data into the corresponding output format without any losses. If an editor tags a heading as such, the transformation developer can see this and also transfer the semantic information into the target format. A text recorded as heading will then also appear as heading on the website or in the PDF.

Editors can use the following information to reasonably use the available formatting options.

9.3.1 Headings

A heading provides an introductory description of the topic dealt with in a section. Documents have one or several main headings (headings of the first order). The headings of the second order are headings of minor importance for a subsection or several subsections. Headings of the third order subdivide these subsections. This principle can be hierarchically continued to the heading of the sixth order.

The rich text editor presents headings in different sizes and formats (bold or italics) depending on their order. The visible output can, however, be completely different. Whether the online editor deems individual font sizes as too large, too small or exactly right should not refrain him from using headings in a logical and correct way related to contents. Headings on websites are the most important structural elements of a document for users of assisting technologies, such as [Screenreader](#). It is thus possible to generate a table of contents from the headings or to jump from heading to heading - these are navigation helps that should not be underestimated. Headings represent a hierarchy. It is thus considered as bad style to leap heading levels. A heading of the first order, for example, should never be followed by a heading of the third order but only by headings of the second order. Disordered heading hierarchies primarily confuse users of screen readers who follow headings.

9.3.2 Lists

Lists are a type of display format for data in horizontal or (preferably) vertical form. Lists are used to organise information. The rich text editor offers two types of lists: structured and unstructured lists.

- In a structured list, list items are subject to a specific order. Structured lists usually have numbered items. A chart list (Top 10, for example) is a typical example of a structured list.
- In an unstructured list, the order of the list items does not play any role. The list items are usually preceded by a list element marker (bullet). A shopping list is a typical element of an unstructured list.

Online editors should always use lists if similar contents are to be recorded. It is possible to cascade lists with the buttons "Decrease indent" and "Increase indent".

9.3.3 Emphasis

Online editors can emphasise text with the buttons "B" (bold) and "I" (italics). These names are confusing because their focus is on visual properties, i. e. on possible types of emphasis. The emphasis is, however, semantic and not visual. The button B depicts the highlighted text with a strong emphasis, the button I with a weak emphasis. The (visual) depiction depends on the output medium.

9.3.4 Inserting images

Online editors can upload images in the editor via the button "New image". Subsequently, the image is displayed at the respective location.

A double click on the image opens a properties' dialogue. Table 5 shows the properties and attributes that can be edited in the standard settings of the editor.

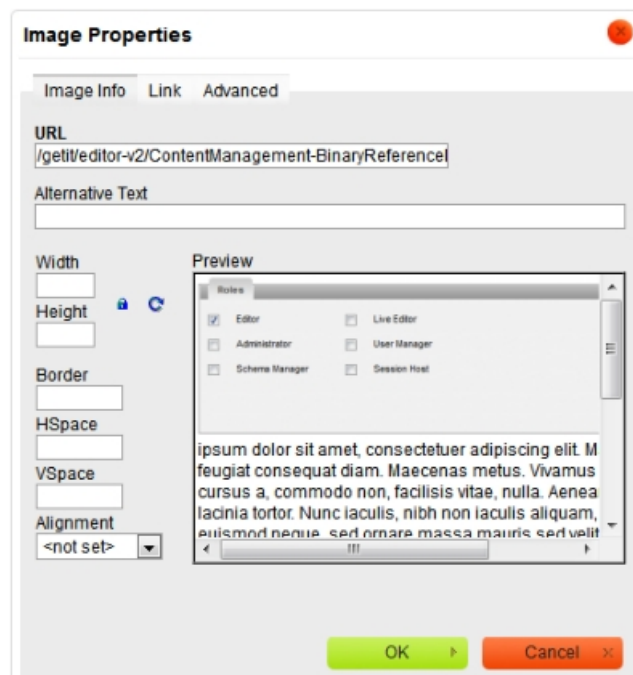


Figure 88

Image info

Width, height - The width and the height of the image can be indicated in pixels. This is, however, not necessary because these values are automatically available in the system.

Image info	Alignment	Alignment of the image; only the values "not Set", "Left" and "Right" are supported.
Link	URL	A link target can be indicated here if the image is to be linked.
Link	Target	The behaviour of the browser when the image is clicked is described here.
Extended	ID	Here you can add an ID to the image.
Extended	Text direction and language identifiers	Here you can indicate if an image differs from the standard system language in these aspects.
Extended	Long URL form	Indicates the URL of a website where the graphic is explained in more detail (in the form of text).
Extended	Style sheet class	Here you can add a class to the image.
Extended	Title description	Adds a title to the image.
Extended	Style	Here you can indicate CSS styles that are added inline to an image.

9.3.5 Links

A link is a reference from a source anchor to a target anchor, in most cases from one document to another. Distinctions are made between internal and external links of the system.

An internal system link is a link to another object in the onion.net editor. To add such a link, online editors have to highlight the content that is to serve as source anchor and drag & drop the object to be linked into the rich text editor.

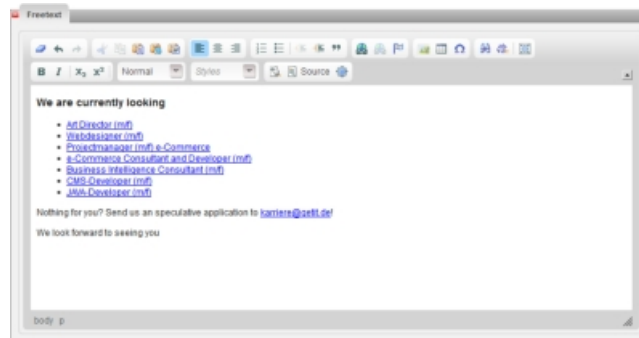


Figure 89

Online editors add an external link via the button "Add / edit link". The link text needs to be highlighted first. Following a click on the button, a properties' dialogue opens. The properties and attributes listed in Table 7 can be edited in the standard configuration.

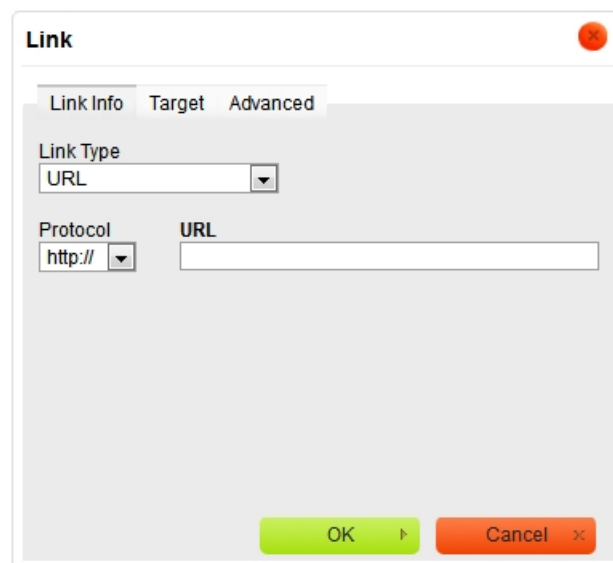


Figure 90

Link	Link type	Indicates the URL schema; depending on the type of indicated URLs
------	-----------	---

Link	Url	Indicates the target anchor of the URL; in case of internal links, the URL always corresponds to the ID of the linked object.
Link	Protocol	Here you can indicate the log of the link.
Target page	Target	The behaviour of the browser when the link is clicked is described here.
Extended	ID, Style sheet class	Indicates an ID or a class of this link.
Extended	Title description	Adds a title to the link describing the link target.
Extended	Style sheet class	Here you can add a class to the link.

9.3.6 Special characters

It is possible to use Unicode characters with onion.net editor. In 1991, the Unicode Consortium set the goal to register, index and provide virtually an character used in any language (plus a large number of additional symbols such as mathematical operators, phonetic symbols, geometric shapes, arrows and musical notes) in suitable codes that today comprise more than one million characters. Online editors can only enter a very small part of these characters directly via the keyboard. Special characters must be entered via specific short cuts or character tables. The rich text editor provides such a table containing some of the most frequently used special characters. Table 8 lists these characters and explains when they are used.

Character name	Character	Description
Non-breaking space		The non-breaking space is used when no line break is desired, it is frequently used between numbers and units (for example: 20.00 €).
Short dash	–	According to the Duden, the short dash is the only dash permissible in German. There is a blank on the left and right of the dash respectively. It is recommended to enter a non-breaking space between the dash and the word preceding it so that the dash does not appear at the beginning of a line. The short dash is also used to designate

a distance between two locations (Hamburg – Berlin, with spaces) or two opponents (Germany-Hungary 3:2, without space).

Long dash	—	In English texts, the long dash is used. In German, it is used to replace "00" of currency values in tables and price lists (for example: 20,— EUR).
Minus	–	The sign in the upper right-hand corner of the numeric pad (-) is not a minus symbol, but a simple hyphen! The "genuine" minus symbol is used as mathematical operator and as negative sign.
Omission marks	...	Omissions and continuations are often depicted by three subsequent dots. If the omission marks appear at the end of a sentence, no full stop will follow, however, there might be other punctuation marks. Omission marks have a space on their right and left side respectively, unless they replace letters of the preceding word.
Single bottom quotation mark	,	According to the Duden, the German character set generally uses two variants of quotation marks or inverted commas:
Single left quotation mark	‘	
Single right quotation mark	’	
Single angle mark pointing to the right	›	
Single angle mark pointing to the left	‹	
Double bottom quotation mark	„	<p>➤ Double quotation marks, consisting of double bottom and top quotation marks („foo“).</p> <p>➤ Double angle marks (also called guillemets or French quotation marks) consist of double</p>
Double quotation marked inclined to the left	“	

Double quotation marked " inclined to the right

angle marks pointing to the right and to the left («foo»).

Double angle mark pointing to the right

Double angle mark pointing to the left

Additionally, there are so called single quotation marks (‘foo’ and ‘foo’) used for proper names, definitions of terms or quotations within a quotation.

Anglo-Saxon countries follow different rules. The opening quotation mark is not typed at the bottom but at the top, just like the closing one and turned towards the word (“Foo”).

Apostrophe

,

In German, the apostrophe is used as omission mark for one or several letters (»Wie geht's?«) or to mark the genitive case of names ending in ss, ß, tz, z or x having no article (»Das ist Lars' Aufgabe.«). The correct character to be used is the right quotation mark (').

Cent ¢

Copyright ©

Registered trademark ®

Trademark TM

One half ½

One quarter ¼

Three quarters ¾

Per mill ‰

Center point ·

9.3.7 Tables

Tables contain structured data presented in lists of parallel columns or right-angled arrangements related to each other. Tables can be used to organise information in a (visually) sensible way.

With a click on the button "Insert table", online editors can create a table. An assistant opens enabling editors to select up to 4 lines and 6 columns. Following selection of the desired number of lines and columns, the table is inserted as configured.

The following buttons are self-explanatory and always refer to the cell that is currently highlighted, i. e. the cell in which the cursor is.

Table cells can contain two types of information: header information or data. This distinction helps user programs to present header and data cells in different ways: Text in header cells could be formatted in bold or in a different font. Online editors decide what type of cell they use via the buttons "header cell" and "data cell".

A properties' dialogue opens with a click of the right mouse button in the table and the following selection of the menu item "Table properties". Table 10 lists the properties and attributes available to online editors.

Figure 91

Table 10: important attributes

Properties	Description
Contents	Indication of the intention and structure of a table for user programs which output the content for non-visual media such as speech or Braille browsers

Headings	Indicates the position of headings in the table (line, column or none)
Heading	Indication of a table heading

10 Enterprise ChangeSets

What are onion.net Enterprise ChangeSets?

Due to the increasing complexity and relevance of web contents, ever larger groups of editors are working on various tasks at the same time in content management systems (CMS). The editors are optimally supported during this parallel working by the onion.net Enterprise ChangeSets – referred to as “ChangeSets” for short in the following.

Each ChangeSet is presented to the editors as an independent editing environment where they can carry out any necessary steps that are necessary for the completing the task.

The editors themselves do not have to adapt when working with ChangeSets. The usual way of working on the onion.net editor does not change. Instead, additional functionalities are provided as support. In this way conflicts are directly recognized and prevented at the time of editing. A system message gives detailed information on why an action cannot be performed.

10.1 Change list

A “ChangeSet” can also be referred to as a “change list”. This means that editors’ work in a ChangeSet is automatically backed up in a change list. Each ChangeSet works independently of the other. This means for example that content in documents can be changed in a ChangeSet, while the same documents are adjusted in structure in another ChangeSet.

If the work in a ChangeSet has been completed to a satisfactory degree, these changes can be transferred to the productive area. That is to say, all activities in the ChangeSet have not yet had any effect on the productive area up to this point. These changes are applied to the productive area through the publication of a ChangeSet. Of course, ChangeSets can also be rejected. It makes sense to do this in order to safely try out changes on the data stock. For example, the documents in a ChangeSet can be completely re-sorted in order to test a new navigation structure.

Previews of the ChangeSets can be seen at any time if you need to check what the changes to the productive stock would look like. Moreover, previews from several ChangeSets can also be combined to test the interaction of tasks which have been worked on at the same time.

It is possible to make changes to the same documents in different ChangeSets as long as these changes are not conflicting. This would be the case for example if the name of a document had been changed in ChangeSet A and an editor wished to change the name of the same document in ChangeSet B. He gets a message stating that this is a conflict. Other changes to the document can be performed however, such as the re-sorting or changing of content fields which we have already talked about.

10.1.1 Data view

In a ChangeSet the editor sees the data stock of the productive area with the ChangeSet changes taken into account.

This becomes especially clear if a change has been made in the productive environment. This is then visible in all ChangeSets immediately. The change saved in the ChangeSet is then applied to the document already changed.

Example: a document is renamed in ChangeSet A. In the productive area, the same document is sorted into another position. After sorting, the document is also in the ChangeSet in its new position, but has the altered name in addition.

Conflicts are prevented by the fact that no contradictory changes can be made among ChangeSets and in the productive environment. This means for example that it is not possible to change the title of a document in ChangeSet B if it has already been changed in ChangeSet A. This is necessary for example for performing quality assurance over several ChangeSets simultaneously.

10.1.2 Communication

This case makes it clear that onion.net Enterprise ChangeSets are therefore also a communication tool which is intended to support the users of the system in their parallel completion of tasks. A conflict message (e.g. saying that a certain characteristic of a document has already been changed in another ChangeSet) lets you know that contradictory tasks are about to be performed in the editorial department. The error message loads for the purpose of direct communication with colleagues and ensures both a content-related and technical validity of the documents.

Thus all ChangeSets are consistent at all times and can be transferred to the productive environment at any time without the manual conflict elimination that would otherwise be necessary.

10.1.3 Working with ChangeSets

There is no limit as to the number of ChangeSets in a system. Since no copy of the data stock is created, which would take up time and storage space each time in the case of extensive websites, it makes sense to create a ChangeSet for many tasks before editing.

10.1.4 An example

The following example ought to give you a better understanding of how to go about using onion.net Enterprise ChangeSets.

Long-term tasks

Your website often undergoes parallel, longer-term changes in its everyday working life. In our example these are:

1. "Redesign": The website is to be modernized to coincide with to the company anniversary on 01.12. This includes a new layout as well as an altered navigation structure. Work on this will take several weeks and is accordingly started as early as in July.
2. "Christmas special": The Christmas special is to be put online just as redesign is taking place and still has to be prepared. The peculiarity is that the Christmas special will be available online in both the old and new layout.

Neither of the tasks can be postponed and have to be worked on in parallel and published at different times.

10.2 Creating and configuring

Users and rights

One way in which work can be divided up in the editorial department is differentiating between an editor in charge and a group of editors. Editors are to be able to edit documents but not to go live with unverified changes.

The editor in charge is to be able to create and edit ChangeSets. Moreover, he is to be able to perform quality assurance and assign the statuses “accepted” and “rejected” for this purpose.

The editors only need the rights to work in ChangeSets and to change the status between “in processing” and “approved”.

You will find a more detailed explanation on the individual statuses in chapter Fehler: Referenz nicht gefunden Fehler: Referenz nicht gefunden.

An “editors” group should be created if there is not yet one available. In our example it is sufficient for this group to receive the rights “only in ChangeSets”.

You can also set up separate groups for editors in charge and editors if you would only like to offer the editor in charge the possibility of deleting documents. Since the ChangeSet (and thus the deleted documents) can only be published by the editor in charge, doing this is not absolutely necessary however.

Depending on how your website is structured in the editor, it is sufficient for you to give the group access to the website folder. The transformations and editor settings do not need to be able to be changed for our example.

You can also set up different user groups for different work, each of which can only edit a smaller section of the website. So there can be a group for example which only takes care of editing news and receives no access to the remaining contents of the website. In the ChangeSet overview these users can then only see changes to documents to which they have access.



Figure 92

Configuration button for the group settings

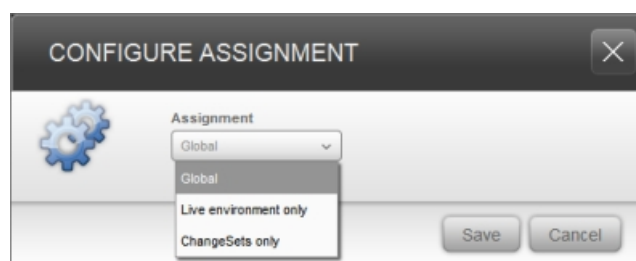


Figure 93

Change of assignment

Apart from the above mentioned settings in the group, the editor in charge also needs the “Live Editor” role. This is necessary in order to create and configure ChangeSets. Besides, he could not publish ChangeSets without this role. He does not necessarily need reading/writing rights on the productive area.

Difference between software releases

Up to release 4.0.6: ChangeSets can be seen and entered by all users with the editor role. A ChangeSet can be edited, published or deleted by the user who created it, or by users with the administrator role or the user administrator role.

Since release 4.0.7: ChangeSets can only be seen and entered by the user who created it, or by users with the administrator role or the user administrator role. Rights with respect to ChangeSets have been introduced as a new feature. These rights are associated to groups and grant all members of the group the right to see and enter, or edit, or publish and delete the respective ChangeSet.

Compatibility upon upgrading: Upon upgrading to release 4.0.7 (or subsequent) the rights for all existing ChangeSets are initially set in a way, that the editors can work as they have been used before. These initial rights can be altered at any time later by the user and group administration function.

Administrators or user administrators can set the rights to any of the following options while they create a new ChangeSet:

- > Full access: creator (default from release 4.0.7)
- > Full access: creator; read: all users
- > Full access: creator; read & edit: all users
- > Full access: all users (default up to release 4.0.6)

The access for those editors who did not create the ChangeSet, has been restricted to read & edit up to release 4.0.6. It is now possible, to grant the publishing right for any particular ChangeSet to these editors.

10.2.1 Table Rights and Roles of a user

Role / rights	Editor	Live editor
Create / configure ChangeSet		
Publish/reject ChangeSet		
Set status "in processing"		
Set status "finished"		
Set status "acceptance issued"		
Set status "rejected"		
Reject individual changes		
Publish individual changes		

10.2.2 Creating

To the right-hand side of the last tab there is a tab with a plus symbol. Clicking on this tab will open a context menu via which you can switch to a ChangeSet that has already been created or create a new one. Thanks to the coloured highlighting the editor can see at any time whether he is in a ChangeSet (orange) or in the productive environment (red).

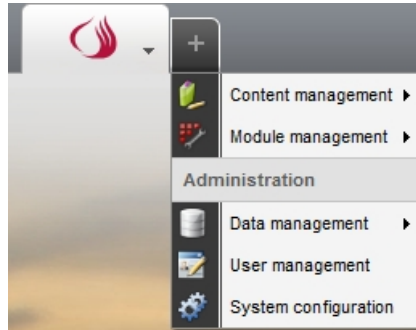


Figure 94

The “plus” sign on the upper tab opens the "Administration selection"

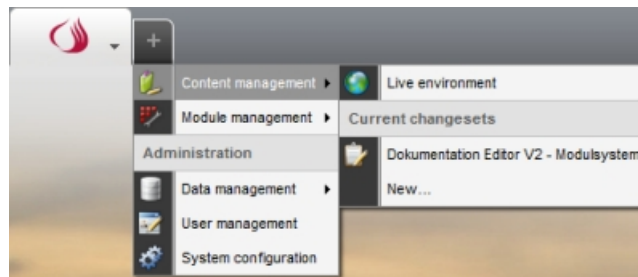


Figure 95

To switch to another ChangeSet or create new sets click on "Editing system".

Figure 96

You designate and describe your new ChangeSet in this dialogue.

10.2.3 Configuring a ChangeSet

The new ChangeSet can now be adapted to requirements by clicking on the button “Configure”. If you wish to make a change to the configuration while working, you can get to the right configuration page by clicking once on the ChangeSet name in the structure area.



Figure 97

Button “Configure”

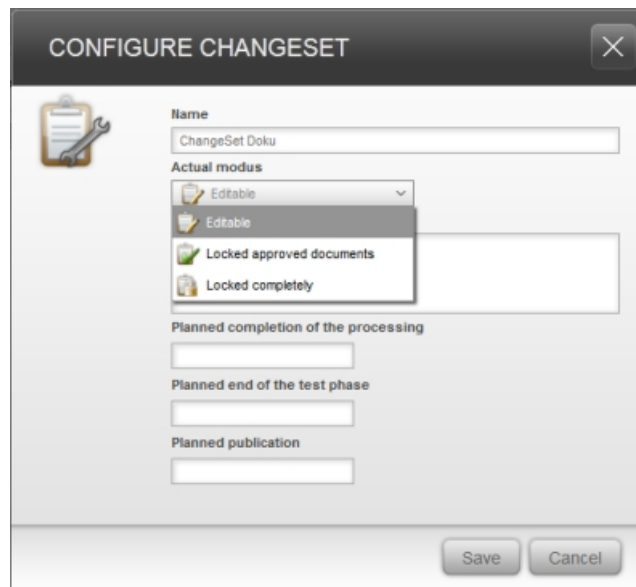


Figure 98

Adapting the ChangeSet

Three different modes can be set per ChangeSet.

Editable

Whoever works in this ChangeSet can edit all documents to which he has access. This is the default for new ChangeSets.

Accepted documents blocked

Documents which have been given the status “Accepted” can no longer be edited. This makes sense if workflows are worked with within the ChangeSet and a quality assurance phase starts. For more on the different statuses within a ChangeSet, see the chapter Working with “statuses”.

Completely blocked

No (more) changes can be made within this ChangeSet. This is particularly wise for quality assurance phases in which a “frozen zone” has been included. During this time, no more changes may be made in the contents to be tested, so that no new errors can worm their way into documents that have already been checked.

10.3 Working in a ChangeSet

As soon as you are in a ChangeSet, all changes made within the ChangeSet to the documents are shown both in the tree view with colored highlighting and in the ChangeSet overview as part of a list.

Changes to documents made by you or another editor in a ChangeSet can only be seen in the respective ChangeSet and can be undone again until the ChangeSet has been published or document changes have been published via “publish directly”, see also Directly publishing / rejecting

10.3.1 Editing documents in a ChangeSet

As soon as you make a change in the ChangeSet, you also produce a new version of the document just as in the productive environment. However, the version numbering within the ChangeSet is different.

If for example you change a document which was in Version 2 before being edited, it is now given the version number 2.1. Further changes within the ChangeSet increase the version number after the point. If the document or the complete ChangeSet is published, the version jumps to the next highest version before the point, so Version 3 in this example.

It is important to know that the intermediate versions behind the comma are scrapped at the time of publishing and cannot be restored.

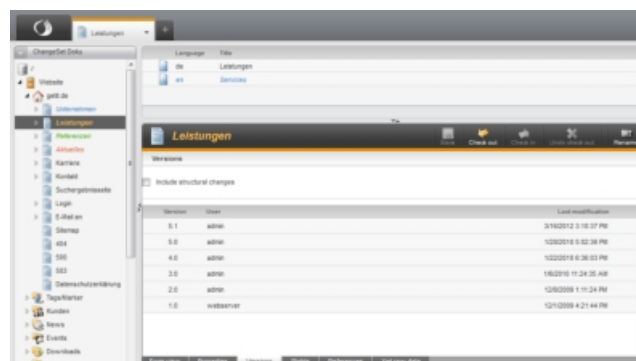


Figure 99

10.3.1.1 Versioning

Versioning has two peculiarities:

If you are at Version 2.1 in the ChangeSet and the document is checked out before publication in the productive area, a new Version 3 is created through this. If the ChangeSet version is then published, a new Version 4 will be produced in the productive area. The Version 3 changes are no longer contained in Version 4, since the Version 2.1 published is based on Version 2. However, there is still a Version 3 of the document containing the productive area changes.

The second peculiarity arises if a document has been checked out (and is thus given the name Version 2), edited and saved in the productive area, but has not yet been returned. If the document is checked out in a ChangeSet in this moment, a new Version 2.1 will be produced. The document that is still checked out is then reset to Version 1 in the productive area by pressing “reject version”. The changes from the productive area now continue to be contained in the ChangeSet in Version 2.1 and also enter the productive area at the time of publishing. In this way contents can be published which should not really be published.

In order to prevent this problem, a notice is shown when the document is checked out in the ChangeSet, indicating that the document in the productive area is also being edited at present and that a temporary data stock is being transferred to the ChangeSet.

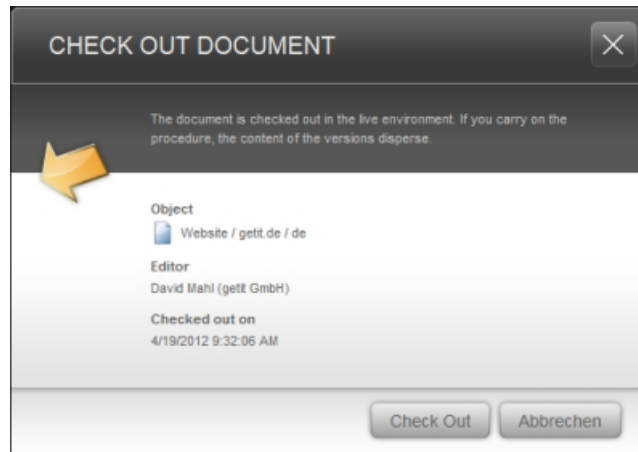


Figure 100

10.3.2 Editor tree view

The colors in the tree view reflect the status of a document. Documents that have not been changed in relation to the productive environment are shown in dark grey as usual. If documents have been changed within this ChangeSet, they are shown in blue italics.

As well as the status “in processing”, there is a further status which a document in the ChangeSet can take. These become important when workflows are worked with. These statuses are “finished” in orange and italics, “acceptance issued” in green and italics and “acceptance refused” in red and italics.

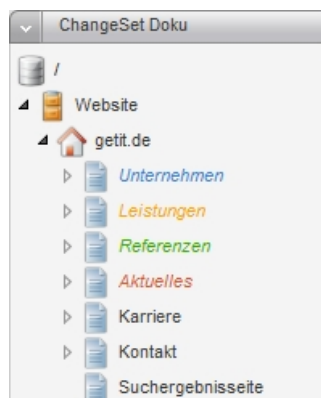


Figure 101

The work with the statuses is explained in more detail in Working with “statuses”

10.3.3 The ChangeSet overview

You can switch to the ChangeSet overview at any time by clicking once on the name of the ChangeSet. This contains lists of the changed documents with their statuses as well as possibilities of set configuration.

Tip: Click on the field with the name of the ChangeSet while holding down the Shift key. The ChangeSet will now open in a new window. This is particularly practical, since the “list of the last 25 changes in the ChangeSet” is updated as soon as you make changes in the main window.

This view is divided into three tabs.

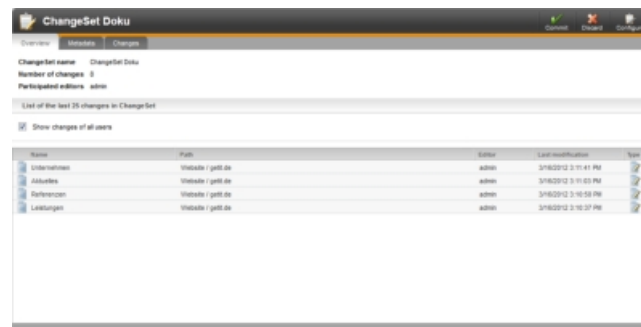


Figure 102

Overview

This view gives you both an overview of the changes in the ChangeSet and the possibility of navigating between the documents last edited.

In the area at the top you can see the meta data of the document. This includes the name of the ChangeSet, the total number of changes as well as the list of the editors involved.

Underneath you will find a list of the last 25 changes made by you. You can expand the view to include the changes by all editors involved, by putting a checkmark next to “show changes by all users”. Due to the fact that individual changes are given in this list, documents are shown several times if they contain several changes.

By right-clicking once on a line in the list, you can open a context menu where you can jump directly to the document within the ChangeSet or open the change list of this document.

Tip: If you hold down Shift when clicking on the context menu entry “open”, you can open the target in a new window here too.

If you click on “change list”, the dialog of the same name will open where you can execute further actions on this document. This is explained in chapter Dialog change list.

Changed documents are shown on the list in grey, whilst the name of deleted documents is indicated in dark red. A symbol in the last column shows which change is concerned:

- > Producing of a document
- > Renaming
- > Content-related editing
- > Sorting changed (order in a structural level)
- > Moving (into another structural level)
- > Deleting

Meta data

Here you will find further meta information on the ChangeSets, which display both information about the originator of the ChangeSet and the configuration of the latter. This tab is purely for the purpose of information.

Changes

The list displayed here contains a detailed overview of the changed documents and also indicates which changes have been made to the respective document.

Just as with the overview list, you have the possibility here of opening the document indicated or of accessing the change list with a right-click.

Moreover, columns for the different statuses of a document “in processing”, “finished”, “accepted” and “rejected” are shown in the right-hand area. These statuses do not only serve for the overview, but can - dependant on the rights of the user - also be changed directly in this list by clicking once on the appropriate field. See the example workflow in this chapter.

10.3.3.1 Filtering options

As time goes on, a great many changes to documents can accumulate in a ChangeSet. Using the filtering options in this list you can determine which changes you would like to be shown.

All documents are shown as standard which have a checkmark in at least one of the columns under “changes” and “status”.

By clicking on one or several of the symbols in the column heading, you can now deactivate (filter) these for displaying. Documents which only have a checkmark in these deactivated columns would now no longer be shown.

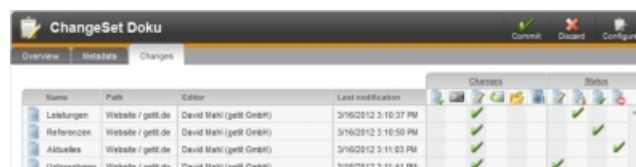


Figure 103

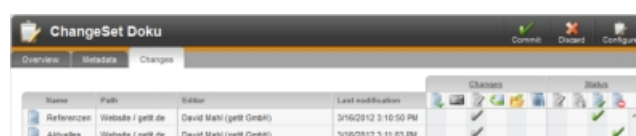


Figure 104

Alternatively, you can also select all columns in the respective area at once by clicking on the heading “changes” or “status”.

If you have deactivated a column and a document is still shown in the list because of another checkmark, the checkmark of the deactivated column is shown in grey.

The list is “OR”-linked. This means that all documents are shown which have a checkmark in at least one of the active columns.

Workflows can be easily displayed using these filter functions.

10.3.3.2 Example workflow

The editor in charge would like to examine all documents for approval which have been set to the “finished” status by the editors.

To do this, he first opens the ChangeSet in a new window and positions it in such a way that he can see both Editor windows at the same time. He goes to the change view in the ChangeSet window and hides all documents first by clicking on the headings “changes” and “status”. He then activates the column “finished” under “status” only. Now only those documents with this status are shown in the list.

The editor in charge now goes through the list and examines the document changes. It is shown which these are by the grey ticks in the columns under “changes”. Depending on the change, previewing the document or opening it in the editor may become necessary.



Figure 105

The document is opened by left-clicking on the name or path. By holding down the CTRL key and left-clicking once you will open the document in a new tab. Right-clicking in the same place will open the context menu. The changes list can now be called here.

For the examined document, the tick can now be directly changed from “Approved” to “Accepted” or “Rejected” in the changes list by clicking in the free field within the respective column.

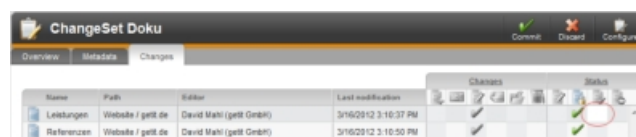


Figure 106

The edited element will now disappear automatically from the list of changes.

10.3.4 Automatic display optimization

The way the changes are displayed is optimized both in the tree view and in the tables of the ChangeSet. This means that only actual changes in relation to the productive area are shown.

If for example you sort an element in the navigation in a ChangeSet, it is marked as a change. If the same document is now moved in the productive environment to the same place in the navigation, it is recognized by the ChangeSet. Since the new positioning in the ChangeSet is now no longer a change, it is also no longer shown as such.

10.3.5 Preview of changes

If you access a preview in the ChangeSet from the changes lists or by right-clicking in the tree view, you will see exactly the status that corresponds to the current productive status with the changes of the current ChangeSet. You will not get to see changes which are made simultaneously in another ChangeSet.

However, you also have the possibility of including changes from other ChangeSets in the preview. Using checkboxes you can select the ChangeSets to be included in this preview. These settings will now also be used for future previews accessed in the normal way.

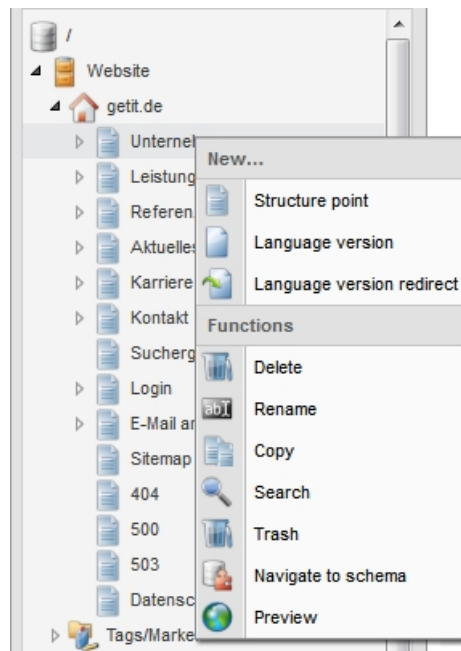


Figure 107

This setting will be applied until the session in the preview browser expires due to user inactivity. This value is set to a default of 15 minutes.

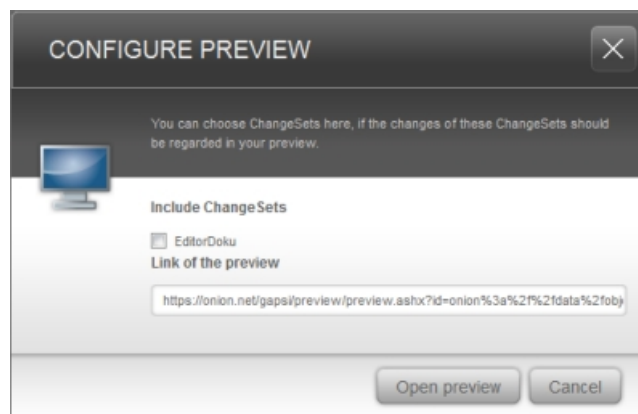


Figure 108

In this way you can look at different work statuses separately as well as a combination of several of them.

Example

Your enterprise is working on the website relaunch mentioned above. The new layout is to go online at a particular time. There is ChangeSet for this relaunch.

At the same time, a Christmas Special is being worked on which is to be put online for a certain period. The peculiarity here is that the Special is to go online before the relaunch, and is also to stay there for some time after the relaunch. This Special also has its own ChangeSet.

A preview can be created within each one of the ChangeSets during the QA phase, meaning you can see the Special with the old layout or the new layout without Special.

If you now access the preview from the productive area, you can activate the two ChangeSets for this preview in the assistant and thus check what the Special looks like together with the new layout.

10.3.6 Dialog change list

The dialog change list offers you possibilities in a ChangeSet of administering the changes to a document.

You can open the dialog in the tree view or from the two list views in the ChangeSet overview. In order to do this you need to right-click a changed document and select the context menu entry “change list”.

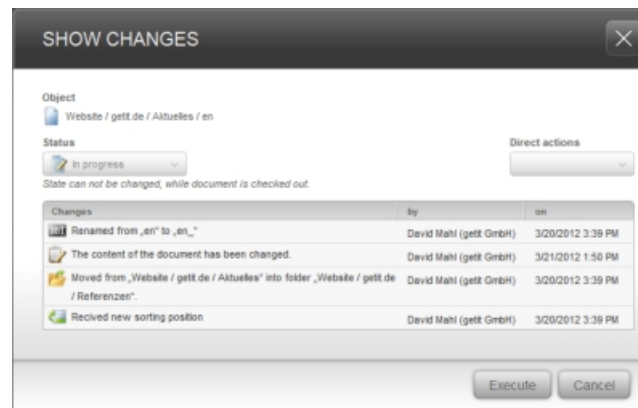


Figure 109

The change list offers two basic functionalities. A status can be given to the document in the area at the top. Underneath there is a list showing all changes which have been made to this document in the current ChangeSet.

10.3.6.1 Working with “statuses”

It is possible to display workflows using the status field. These statuses do not necessarily have to be used in the ChangeSet, although they do aid clarity.

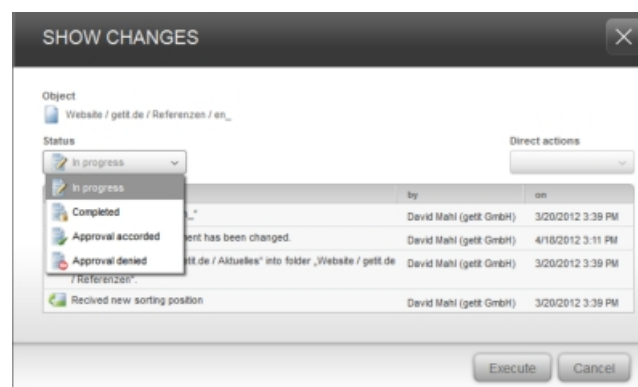


Figure 110

A document can take the following statuses:

- “In processing”
- “Finished”
- “Acceptance issued”
- “Rejected”

A document automatically switches to the status “in processing” as soon as a document has been changed. This concerns both content-related changes and re-sorting in the navigation. All further status changes must be explicitly set via the change list.

If an editor is happy with the state of a document, he sets the status to “finished”. The editor in charge can now have all finished documents listed in the ChangeSet overview and check them. Then the document can either be set to the status “accepted” or “rejected”.

In combination with the optional ChangeSet configuration to not allow accepted documents to be edited again, an efficient QA phase can be implemented.

10.3.6.2 Directly publishing / rejecting

Apart from status changing, this dialog also offers the possibility of viewing the changes to a document. Each change is represented by its own line in the list. In addition, it is shown who made the change and when.

As well as offering pure information, this list also gives you the possibility of directly publishing or rejecting all changes or just individual ones.

“Publishing” means that the selected changes pass directly into the productive area without the complete ChangeSet having to be published.

In order to do this, either press the button “publish directly” or “reject” in the area “direct actions”.

A check box will appear beside each of the changes. You can use the check box to define whether the respective change is to be published or rejected.

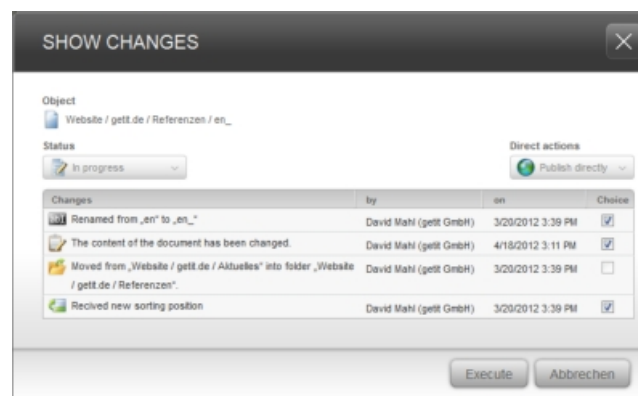


Figure 111

Thanks to this mechanism, changes from the ChangeSet can go directly online without the complete document having to be published. Other changes which are intended for a later time stay in the ChangeSet as a change after publication. The exact same thing goes for the rejecting of changes. All other changes stay but the selected changes are deleted from the ChangeSet.

With this kind of publishing or rejecting it is possible for conflicts to arise however. The ChangeSets are always valid as a whole, but older changes in documents may contradict changes in other elements.

Example: In a navigation path there is a document with the name A and a document with the name B. Document A is renamed to “D” and then to “C”. Document B is now renamed to “D”. Following all editing steps, there is now a Document C and a Document D in the ChangeSet.

Using the “publish directly” function of Document D, an editor would now like to publish the change of name “from A to C”. This will cause a conflict however, since there is (now) another document with this name.

10.4 Publishing / rejecting ChangeSets

After all work in a ChangeSet has been completed, the latter can be published. The peculiarity of the onion.net Enterprise ChangeSet is that each ChangeSet can be published without problems since no conflicts can arise at the time of the publication process.

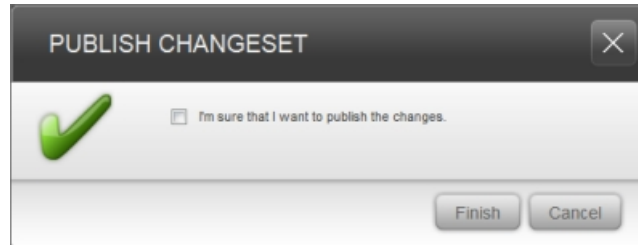


Figure 112

Following a confirmation prompt, the changes of the ChangeSet will be applied to the productive area and the empty ChangeSet is deleted.

The status the individual documents are in is not important for a publication. So even documents with the status “rejected” are published. If documents that are still checked out are in the ChangeSet at the time of publication, these are transferred to the productive environment with the changes currently checked out and are no longer checked out there.

The intermediate versions of the documents produced in the ChangeSet are scrapped when publishing.

Of course, it is also possible to reject all changes in one go. If this is done, all changes are lost and the ChangeSet is deleted.

Figures

Abbildung 1, Seite 1

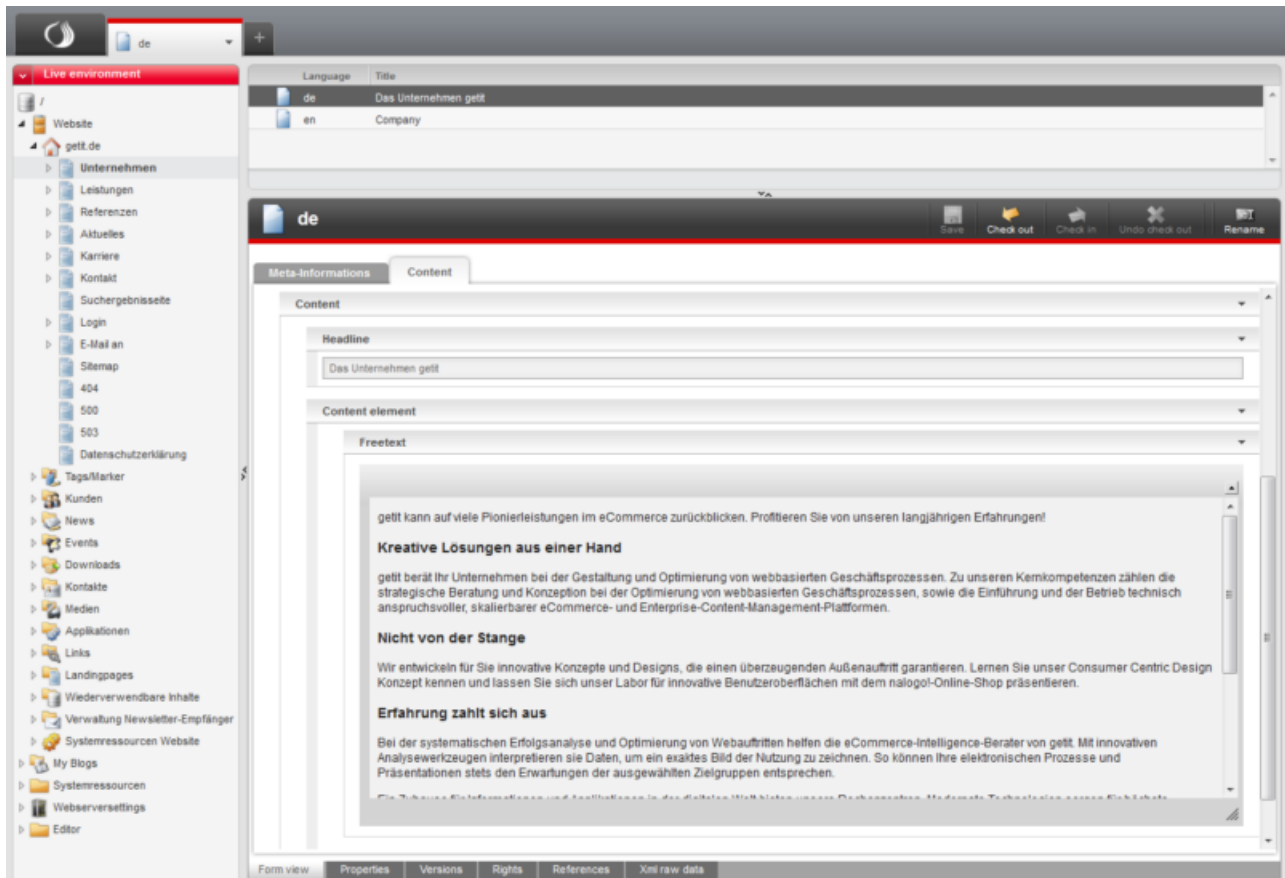


Abbildung 2, Seite 2

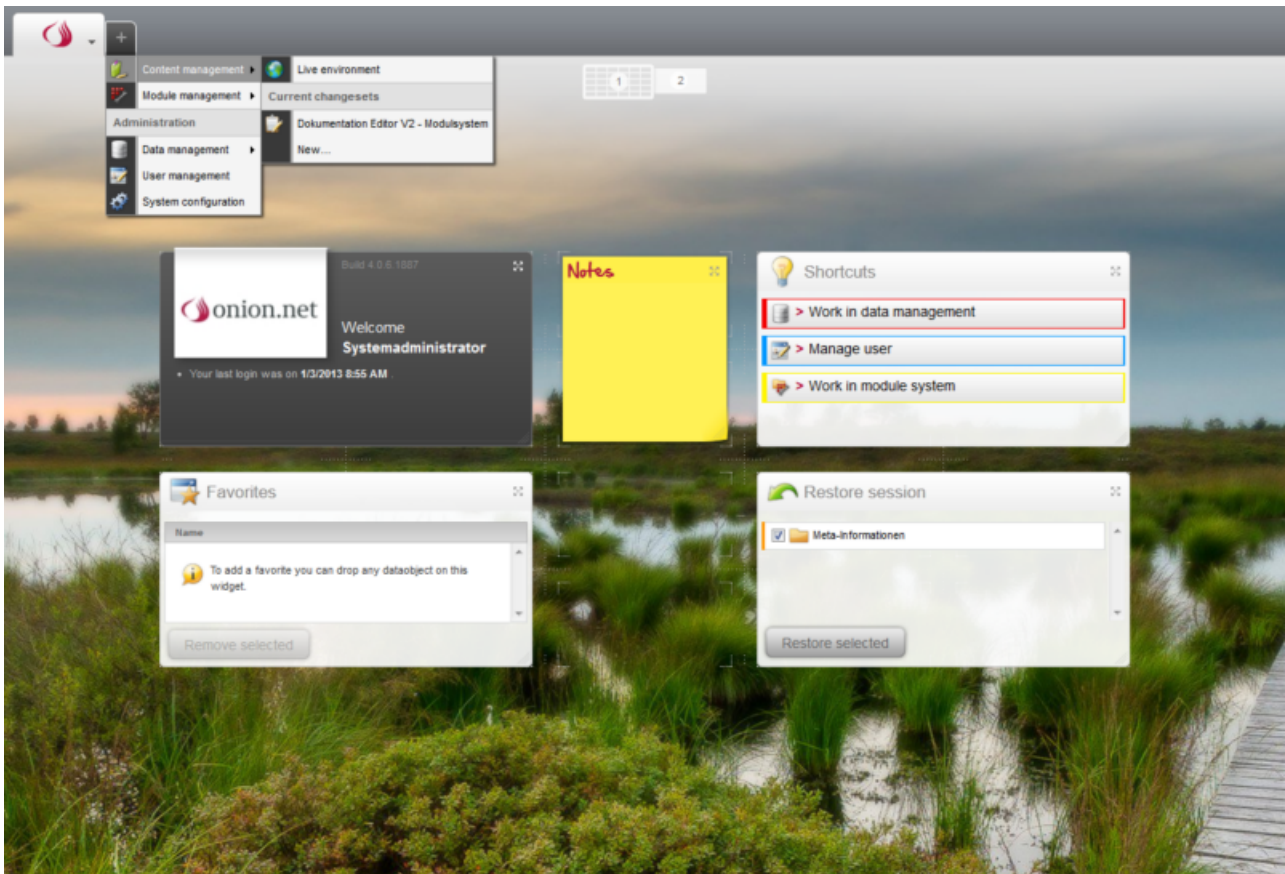


Abbildung 3, Seite 3

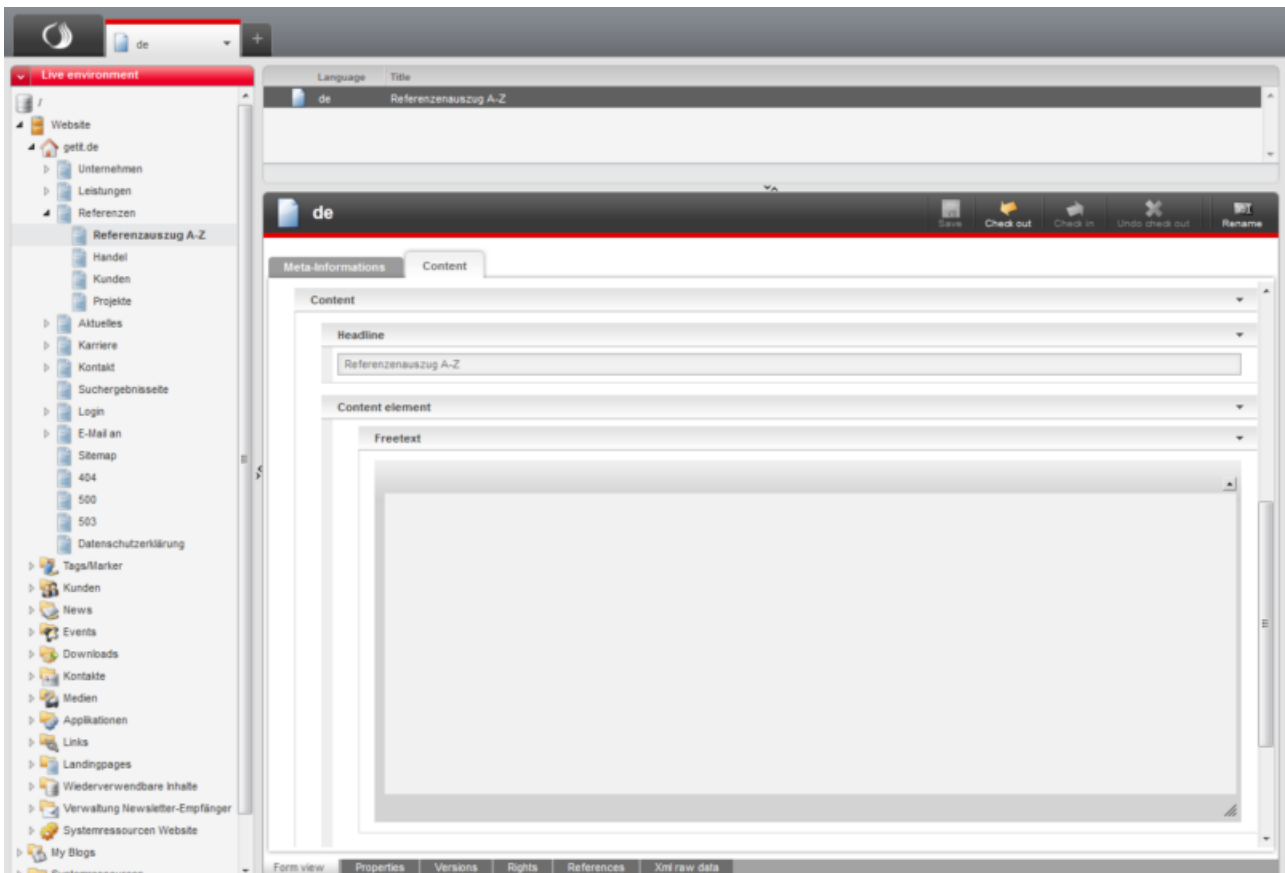


Abbildung 4, Seite 3

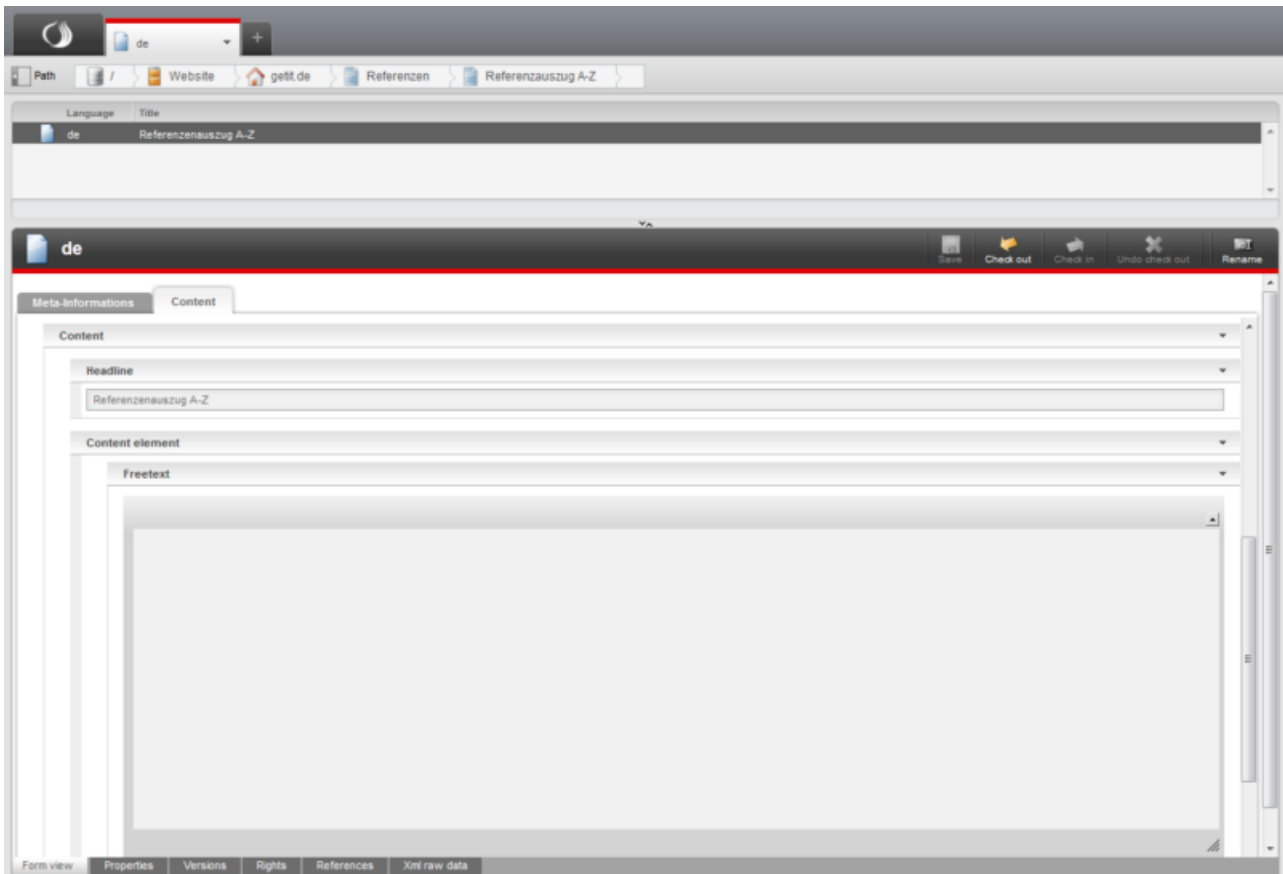


Abbildung 5, Seite 7

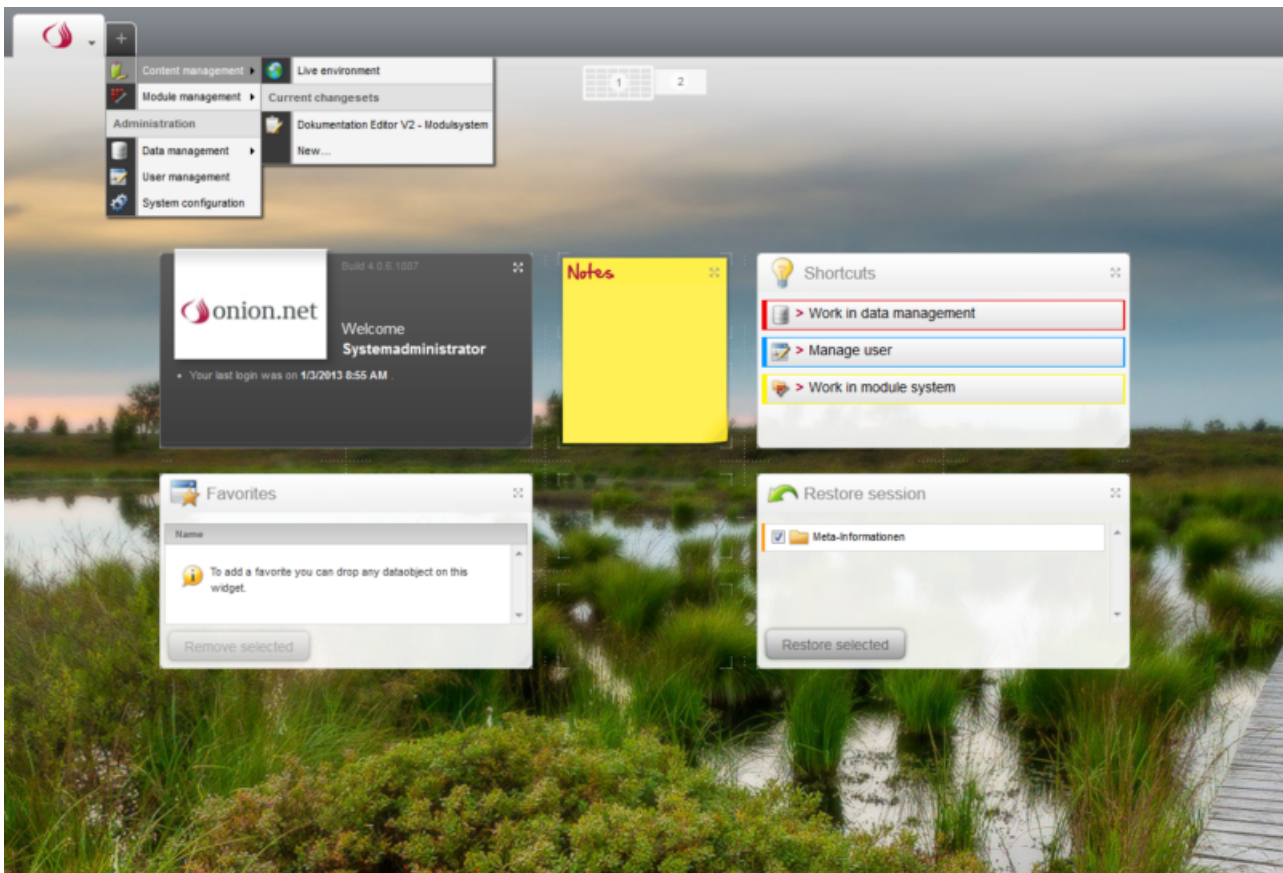


Abbildung 6, Seite 7

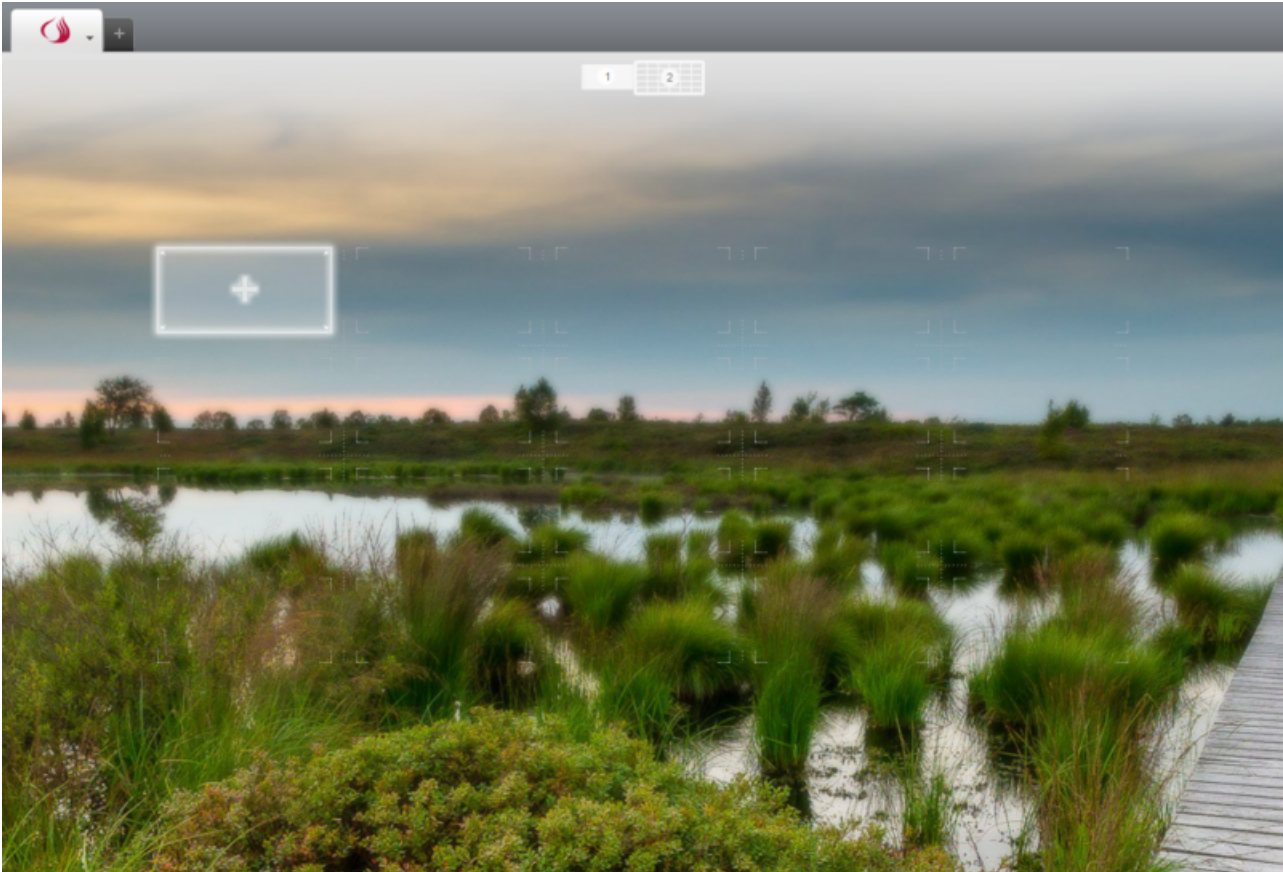


Abbildung 7, Seite 9

Language	Title
de	Das Unternehmen getit
en	Company

de

Save Check out Check in Undo check out Rename

Meta-Information

Content

☒ Show in navigation

☐ Show in site-navigation

☐ Show in footer-navigation

☒ Show newsletter quick subscribe

@hideSideBar1 ☐

Navigation name Unternehmen

Window label Das Unternehmen getit

SEO

☒ indicate

☒ follow links

☒ Show nothing from ODP-entry

Form view Properties Versions Rights References Xml raw data

Abbildung 8, Seite 9

Description	Picture
ariadne	
list	
subnav	
more	
submit	
print	
Twitter	
Facebook	
scrollup	

Abbildung 9, Seite 10

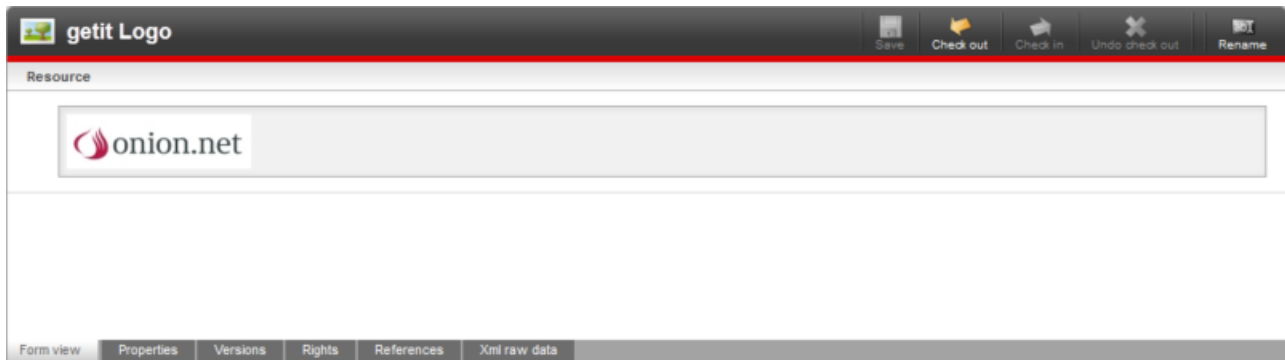


Abbildung 10, Seite 10



Abbildung 11, Seite 11

The screenshot shows the 'getit' web interface with a form titled 'Contact'. The form is displayed in a tabbed view with the following tabs: 'Form view', 'Properties', 'Versions', 'Rights', 'References', and 'Xml raw data'. The 'Form view' tab is selected. The form contains the following fields:

- Salutation:** A dropdown menu with 'Mr' selected.
- Picture:** A checkbox.
- Title:** A checkbox.
- Forename:** A text input field with 'David' entered.
- Lastname:** A text input field.
- Position:** A checkbox.
- Phone:** A checkbox.
- Fax:** A checkbox.
- E-Mail:** A checkbox.

Abbildung 12, Seite 13

The screenshot shows the 'getit' web interface with a search results page. The page is titled 'Search in "Live environment"'. It features a search filter section with the following fields:

- Search filter:** A dropdown menu with 'Search results' selected.
- Name:** A text input field with a 'contains' dropdown menu.
- Content:** A text input field with a 'contains' dropdown menu.
- Datatype:** A dropdown menu with '< All datatypes >' selected.
- State:** A dropdown menu with '< All States >' selected.
- Last modifier:** A dropdown menu with '< Last editor >' selected.
- Creator:** A dropdown menu with '< Last editor >' selected.
- modified from / modified until:** Two text input fields separated by a minus sign.
- created from / created until:** Two text input fields separated by a minus sign.

A 'Search' button is located at the bottom of the search filter section.

Abbildung 13, Seite 17

Redakteur

Save

Configure

Rename

Path	Object	Children	Members
Website / gett.de			
Website / Kunden			
Website / Tags/Marker			
Website / News			
Website / Events			
Website / Downloads			
Website / Kontakte			
Website / Medien			
Website / Applikationen			
Website / Links			
Website			
Website / Systemressourcen Website			
Website / Systemressourcen Website / Lokalisierung			
Website / Wiederverwendbare Inhalte			
Website / Verwaltung Newsletter-Empfänger			

Add right

Includes all rights of	Assignment	Members
Users	Global	0

Neriman Cetinkaya

Abbildung 14, Seite 17

Usersettings		Instance
Profile		default
Editor configuration		default

Abbildung 15, Seite 22

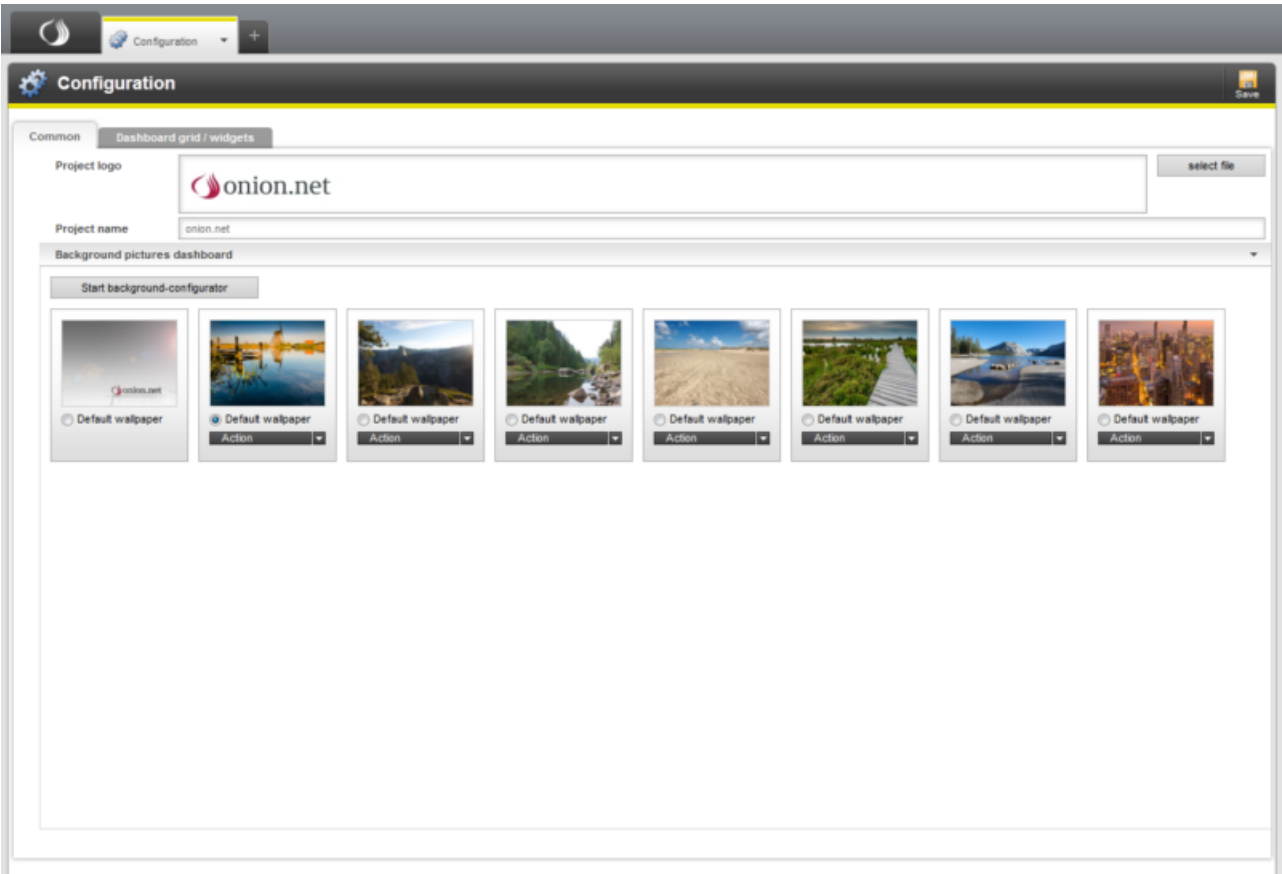


Abbildung 16, Seite 23

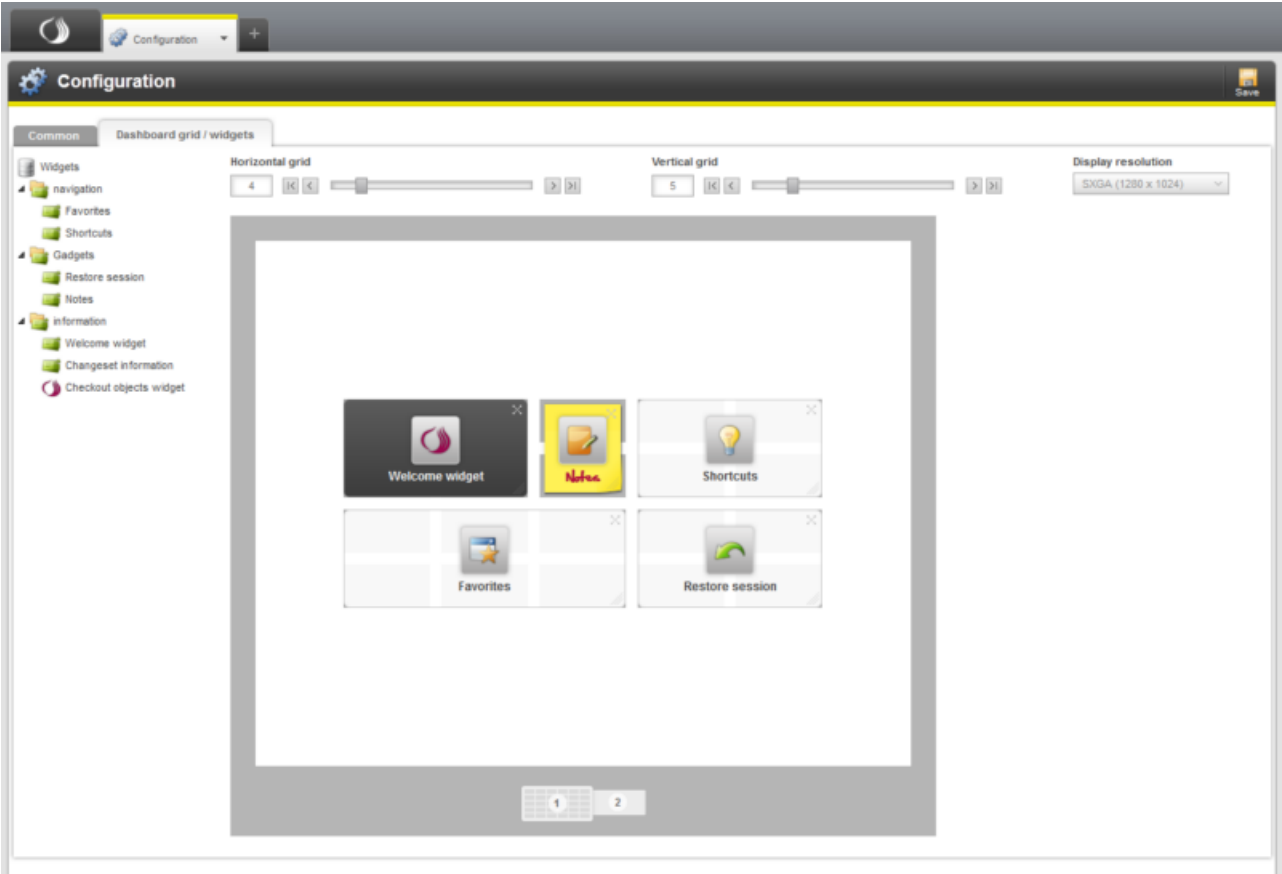


Abbildung 17, Seite 24

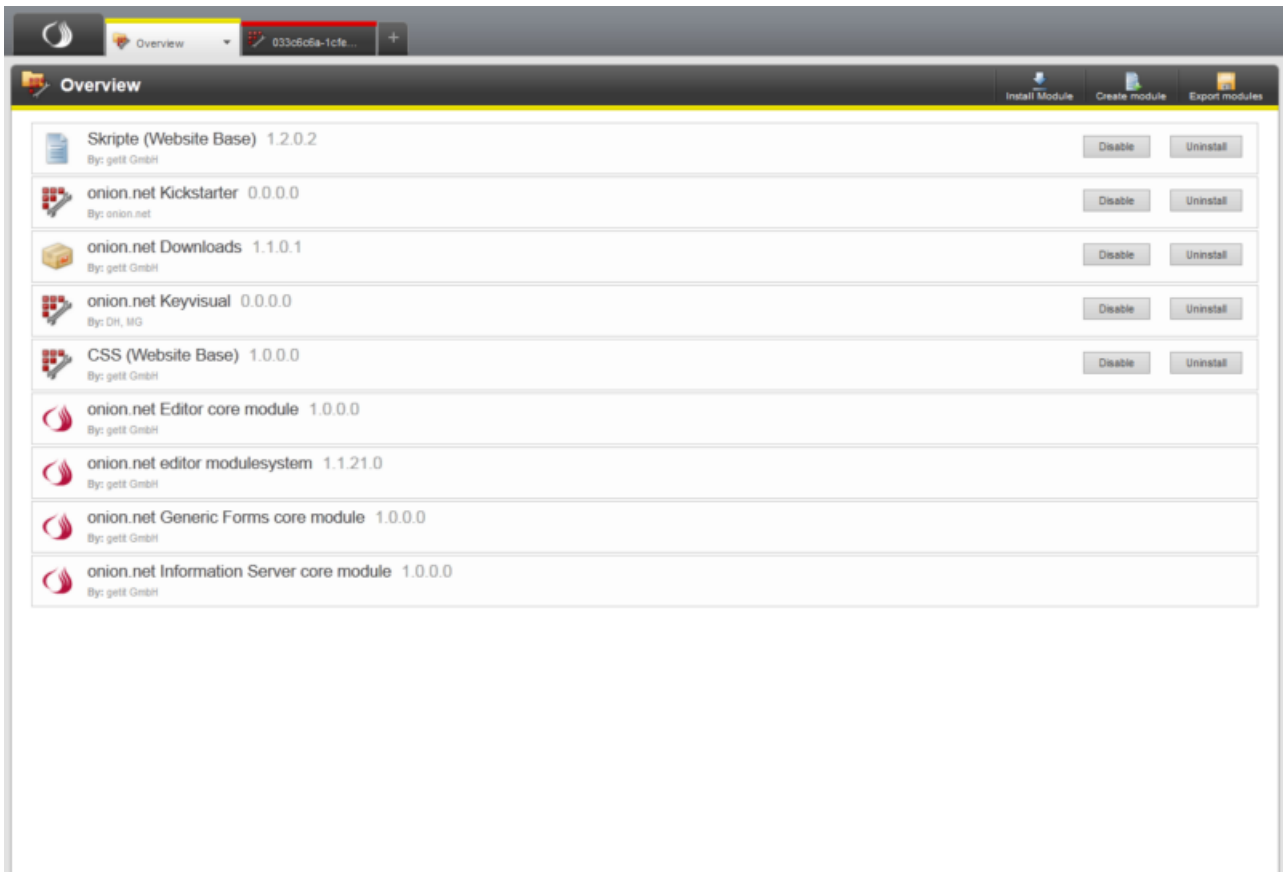


Abbildung 18, Seite 24

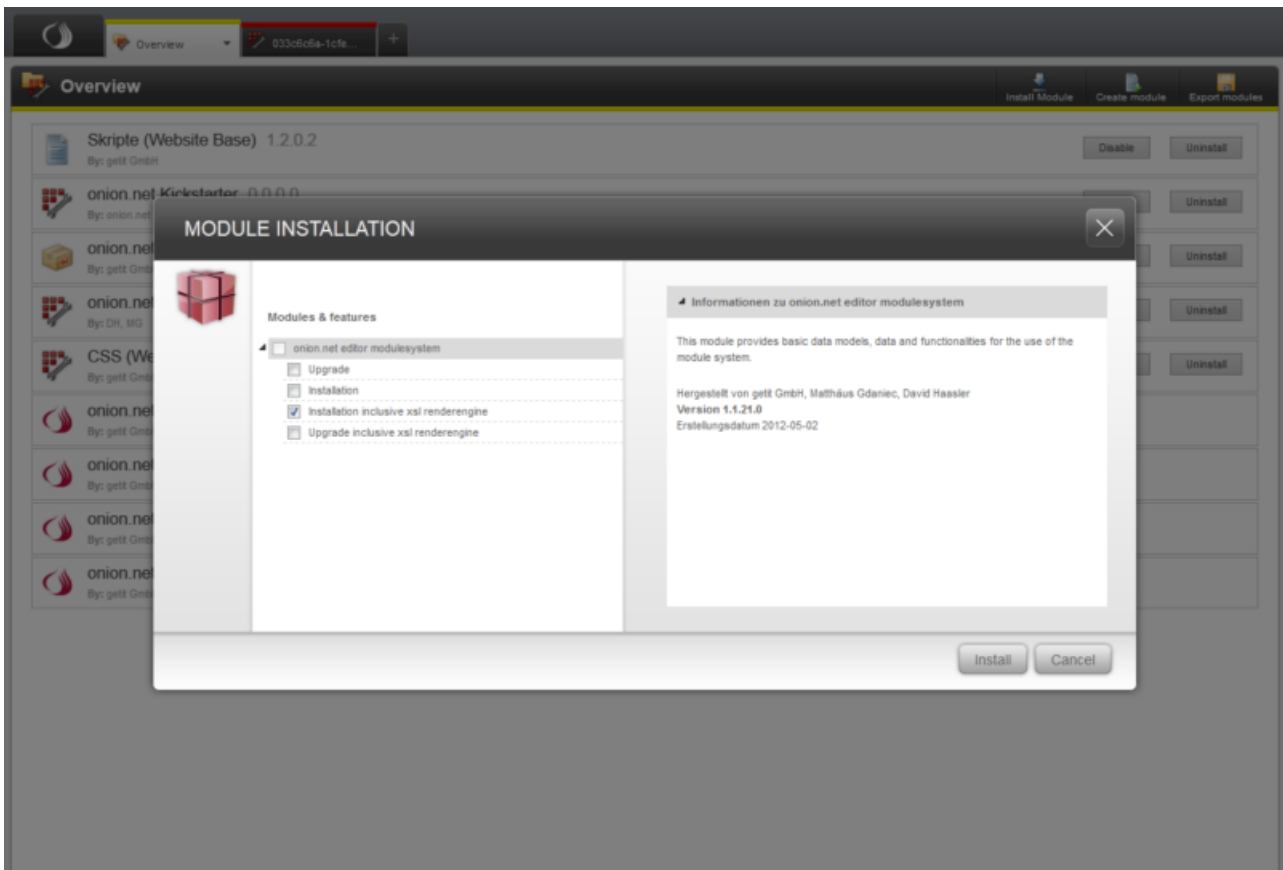


Abbildung 19, Seite 24

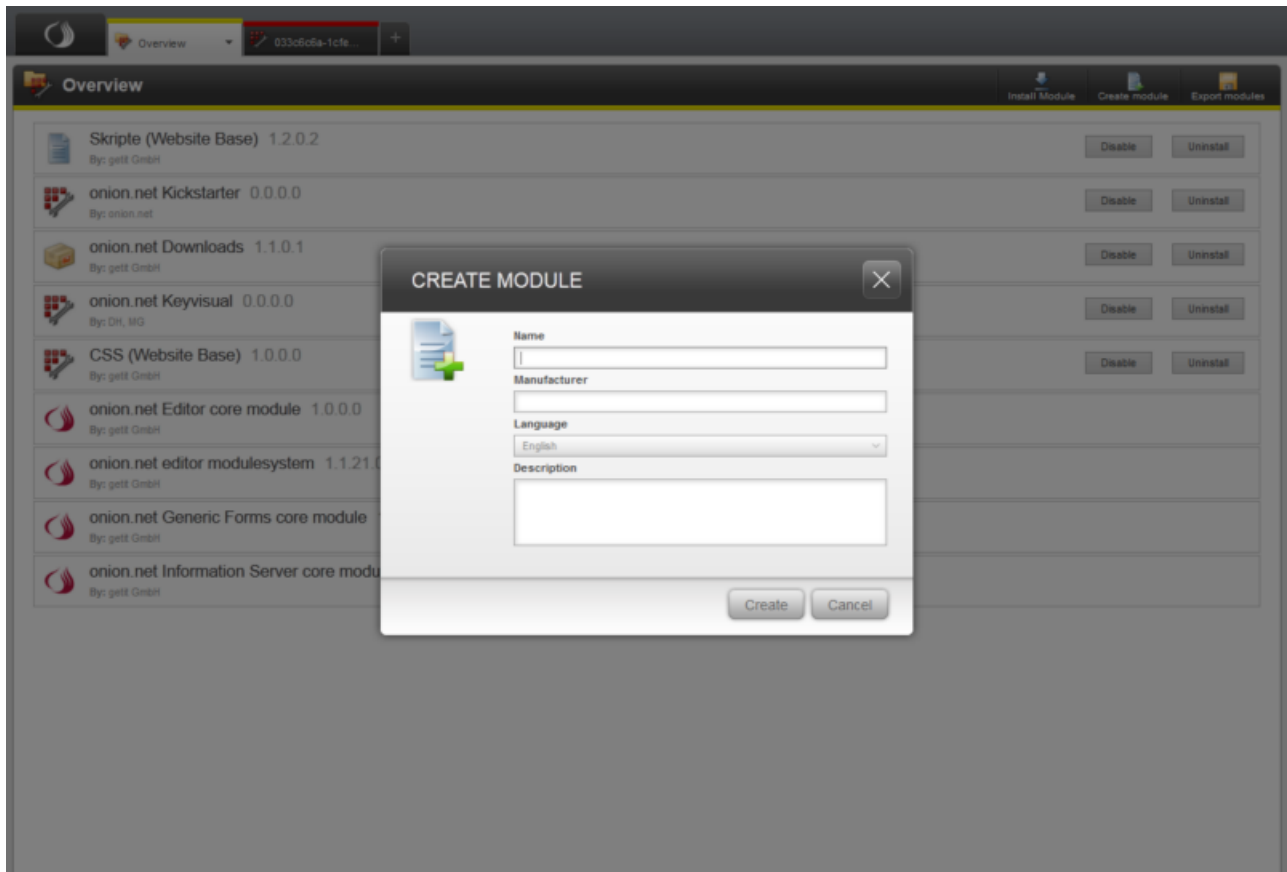


Abbildung 20, Seite 25

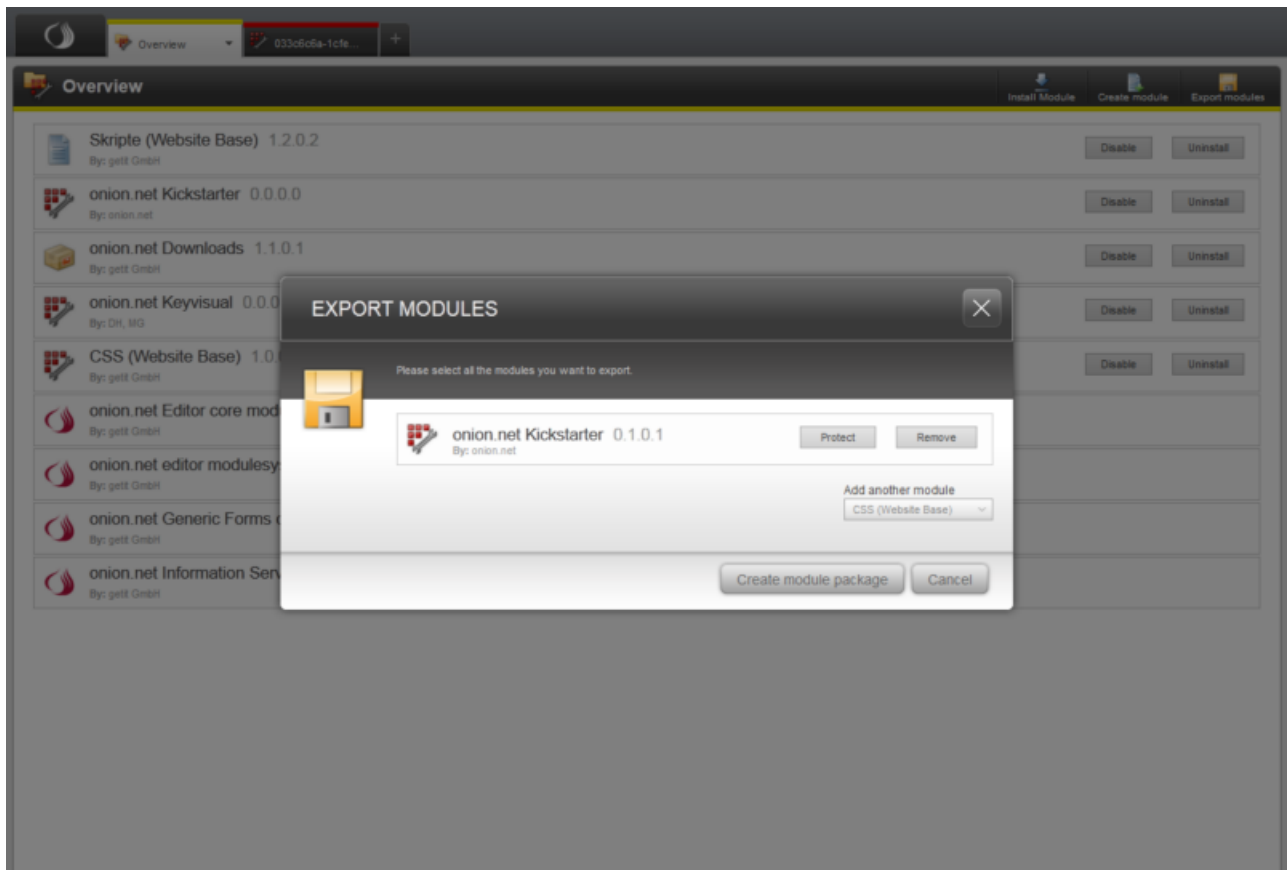


Abbildung 21, Seite 25

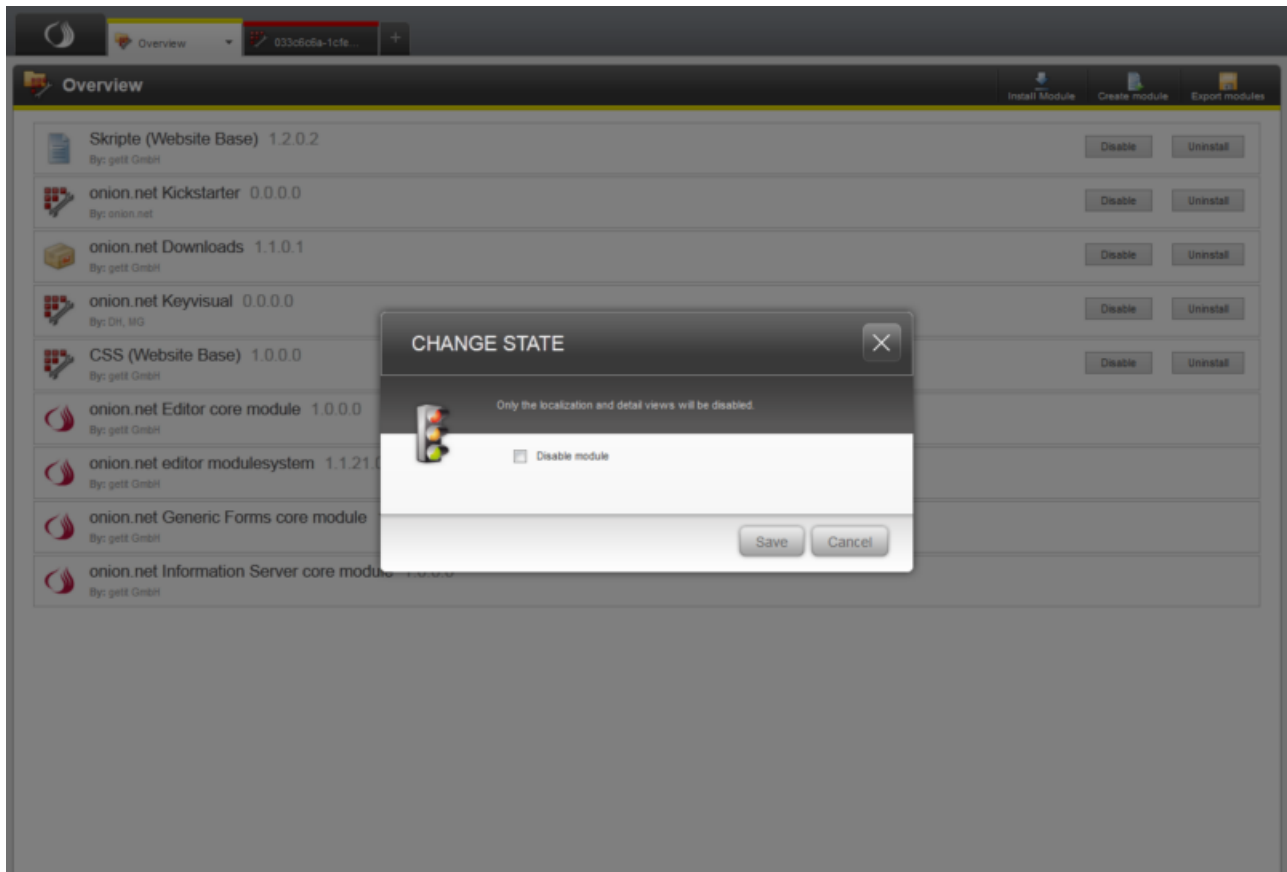


Abbildung 22, Seite 25

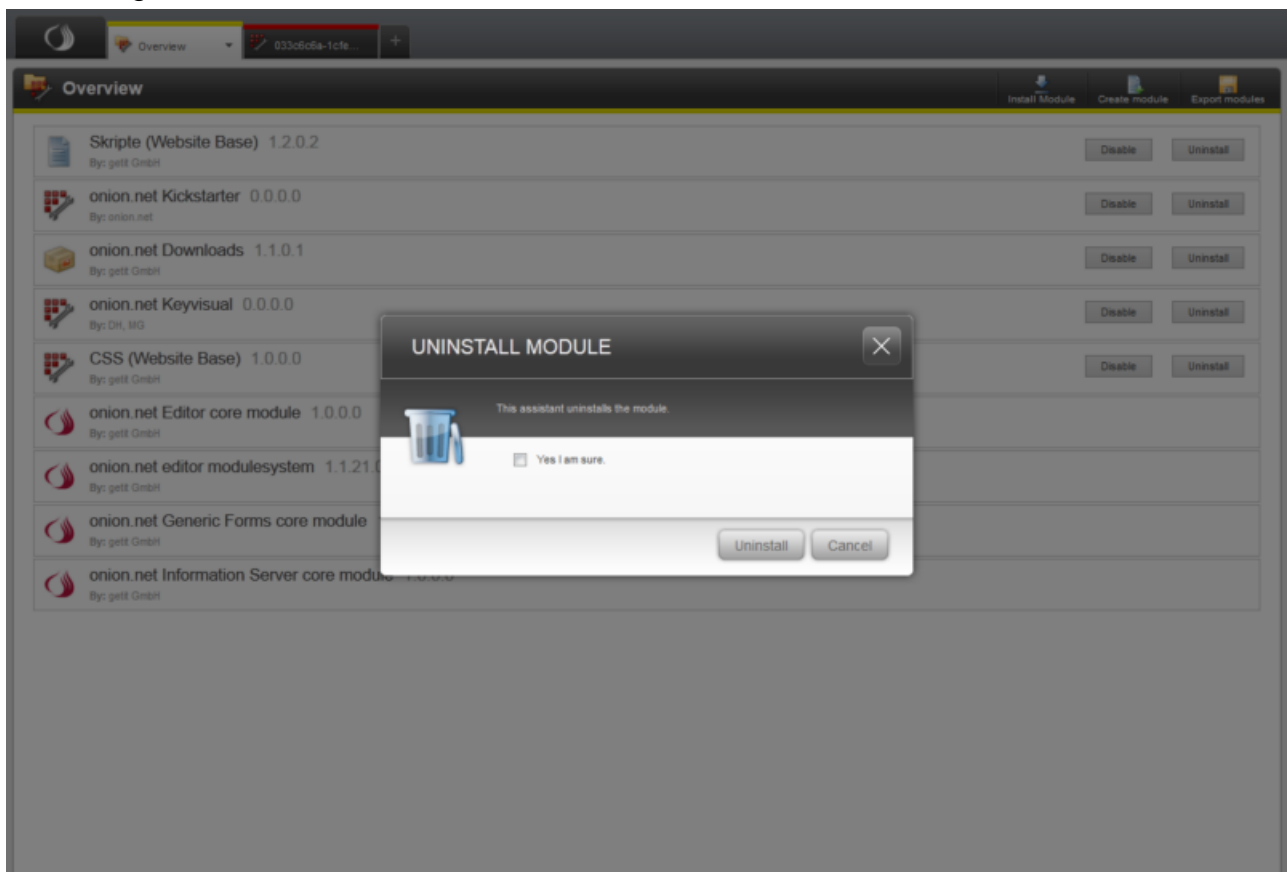


Abbildung 23, Seite 26

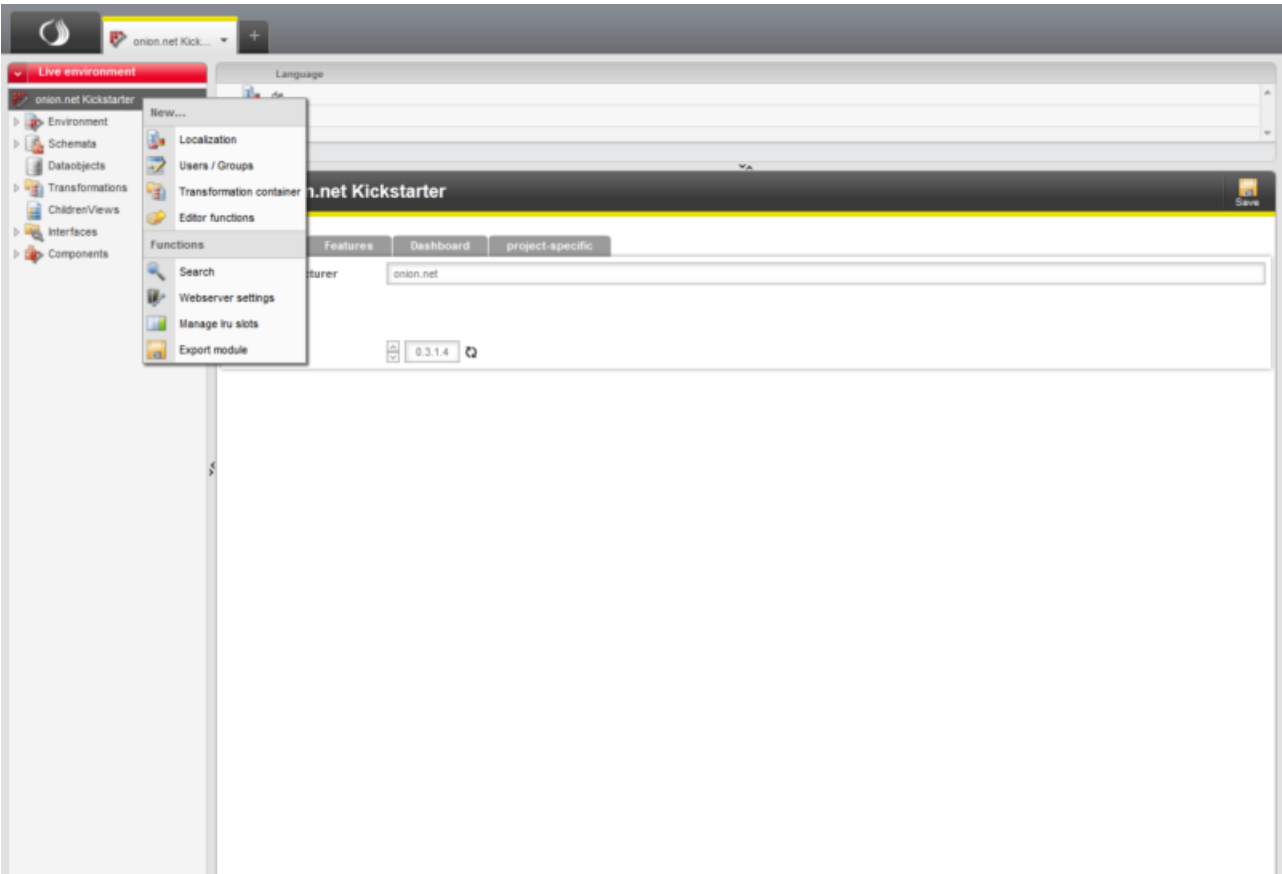


Abbildung 24, Seite 28

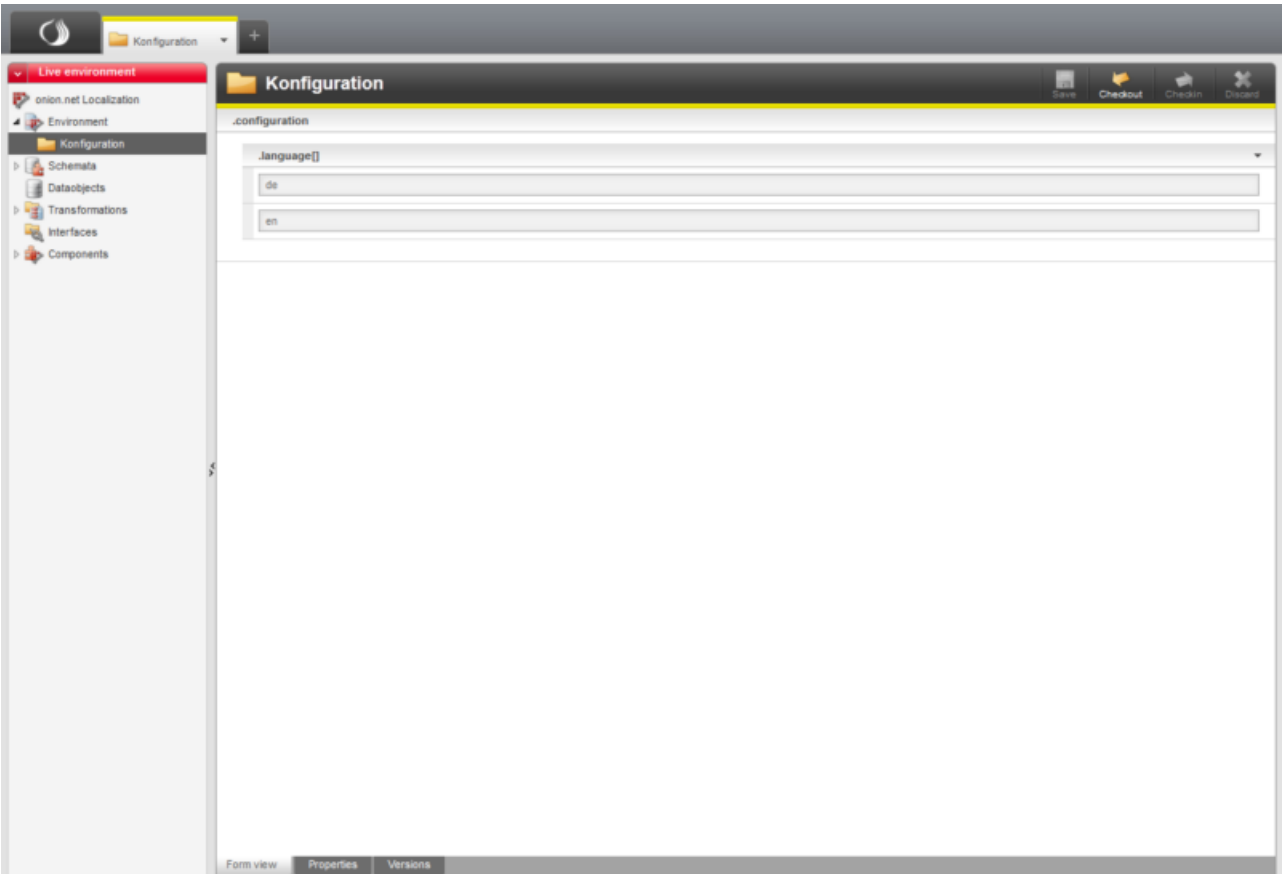


Abbildung 25, Seite 28

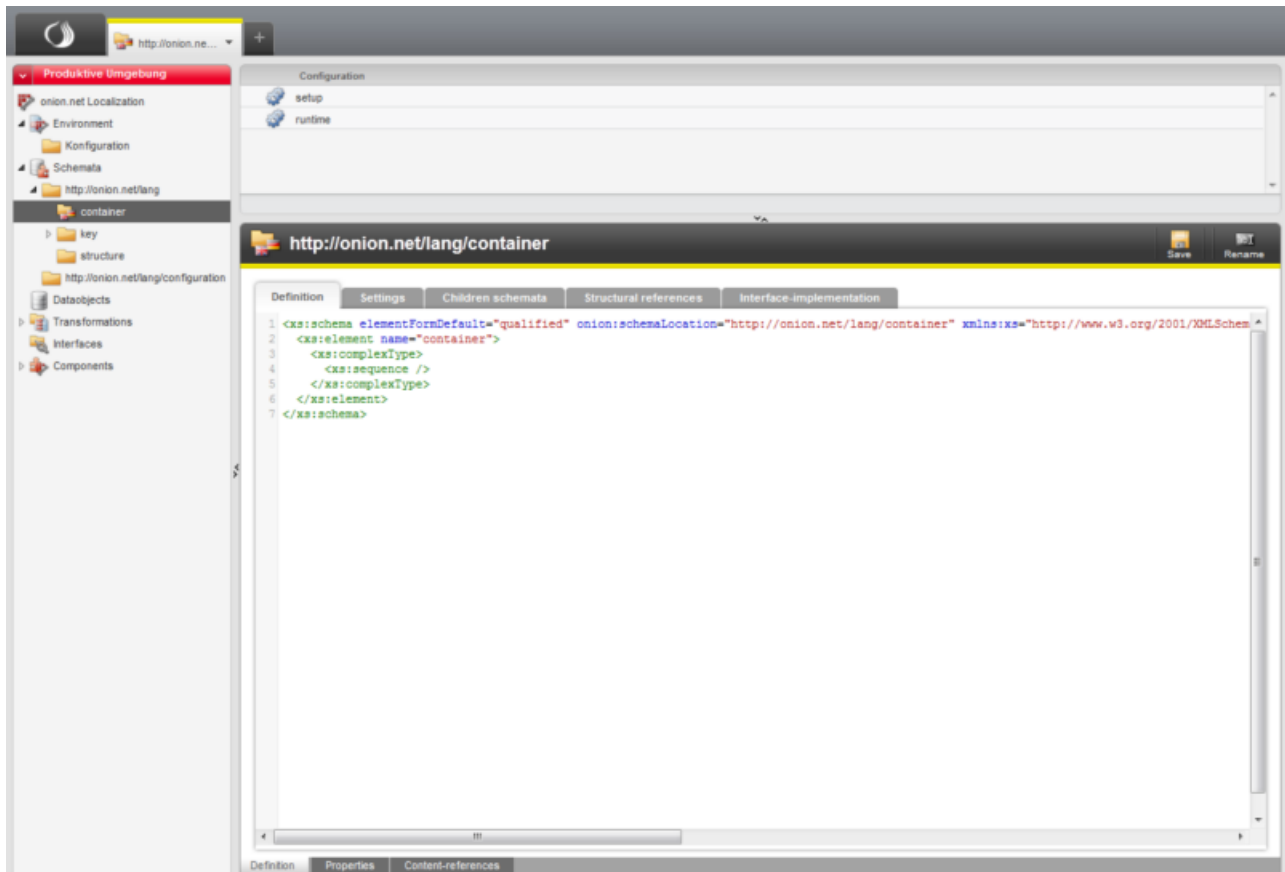


Abbildung 26, Seite 29

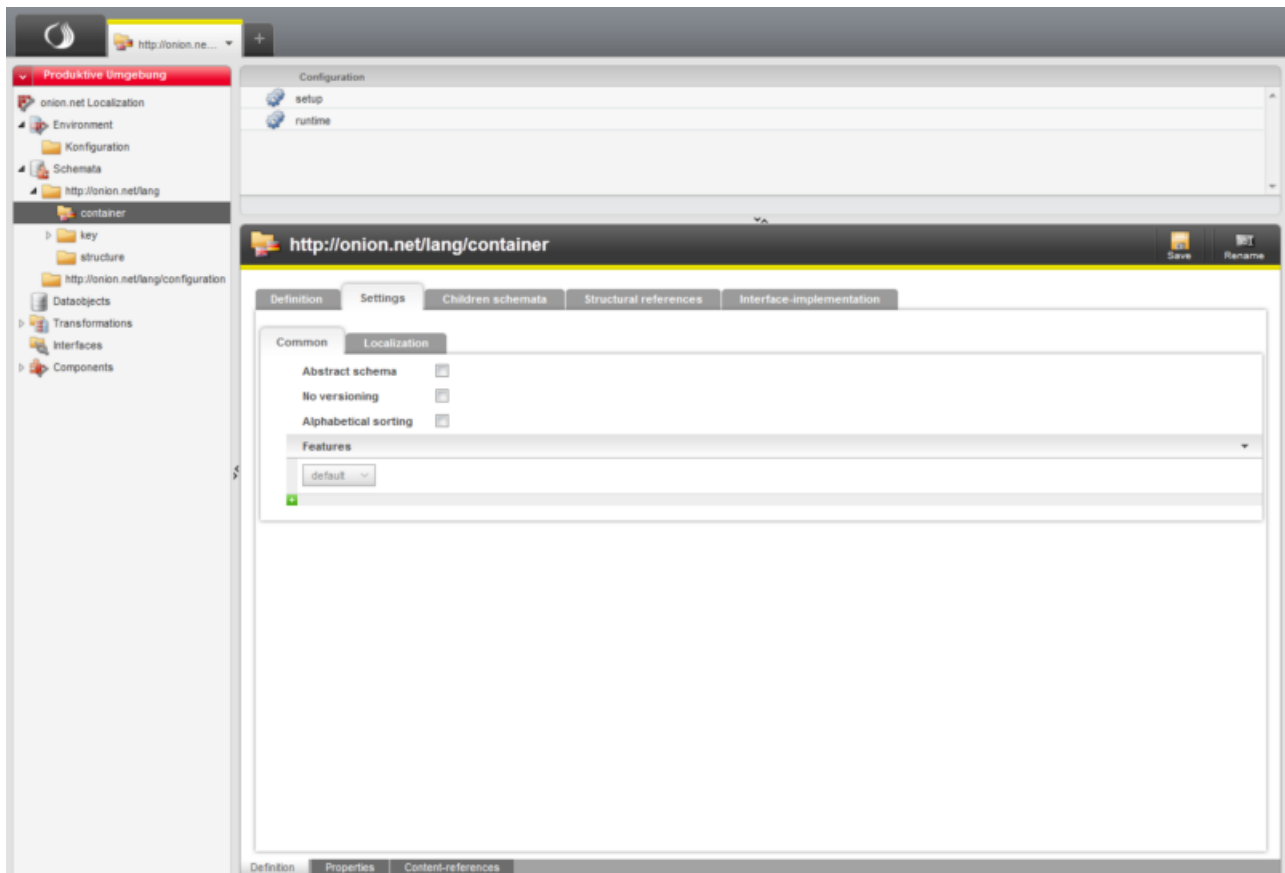


Abbildung 27, Seite 29

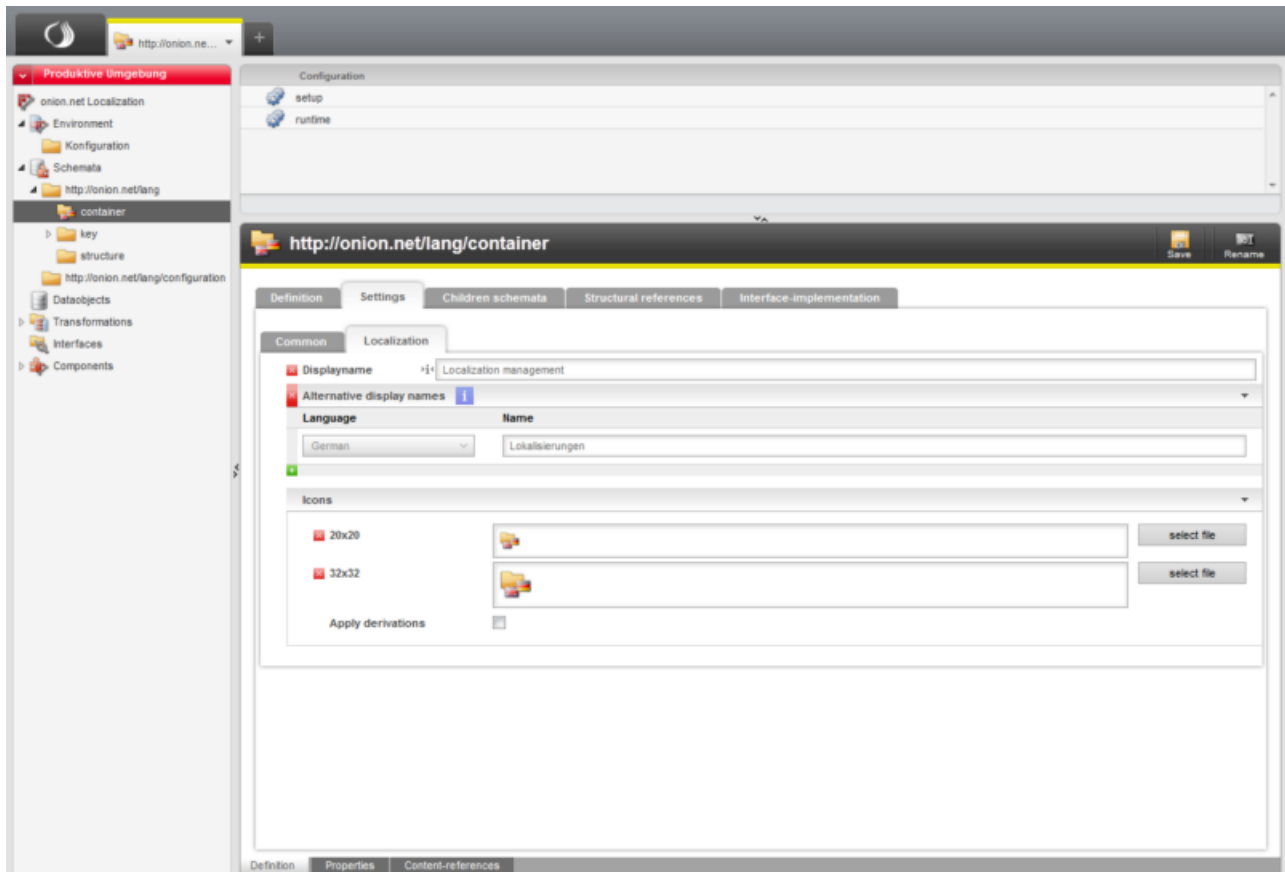


Abbildung 28, Seite 30

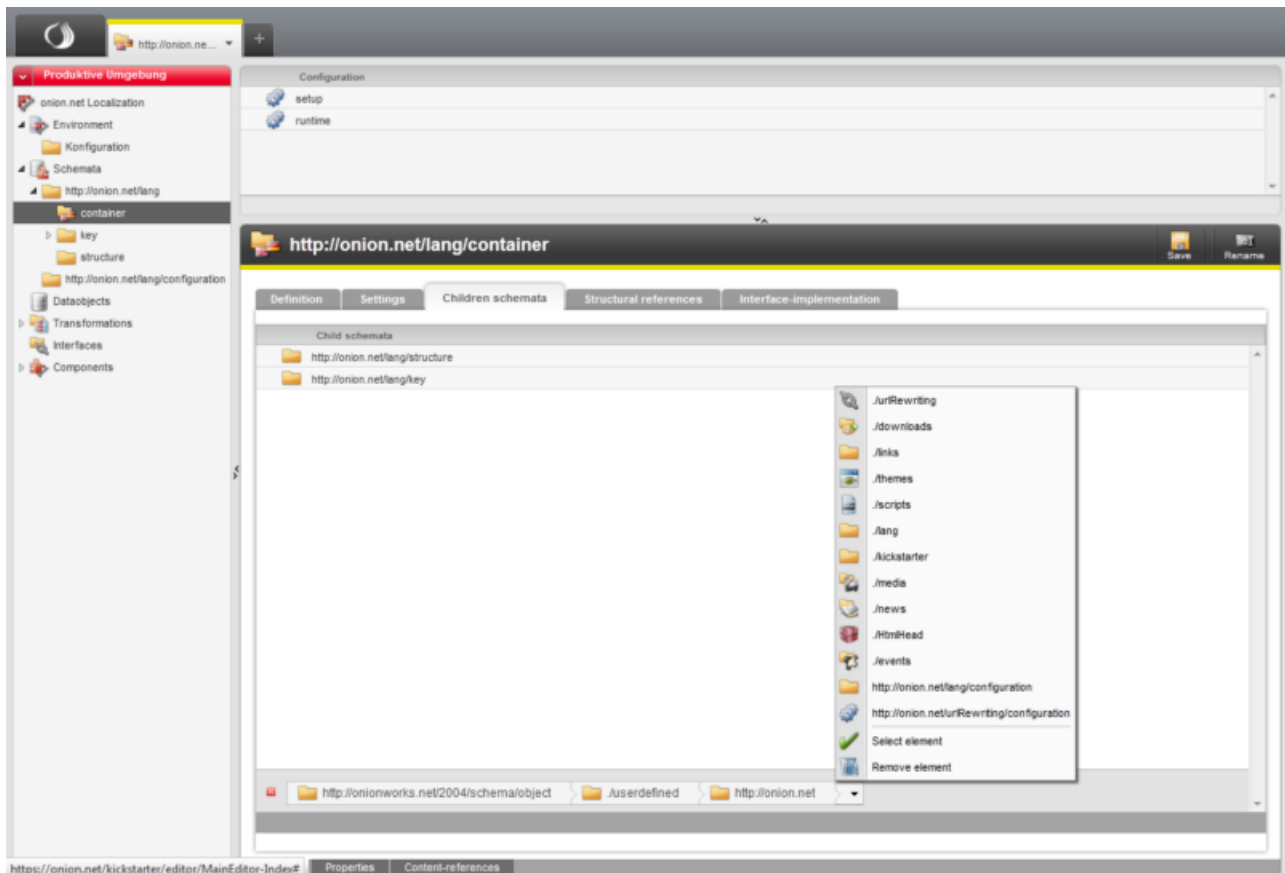


Abbildung 29, Seite 30

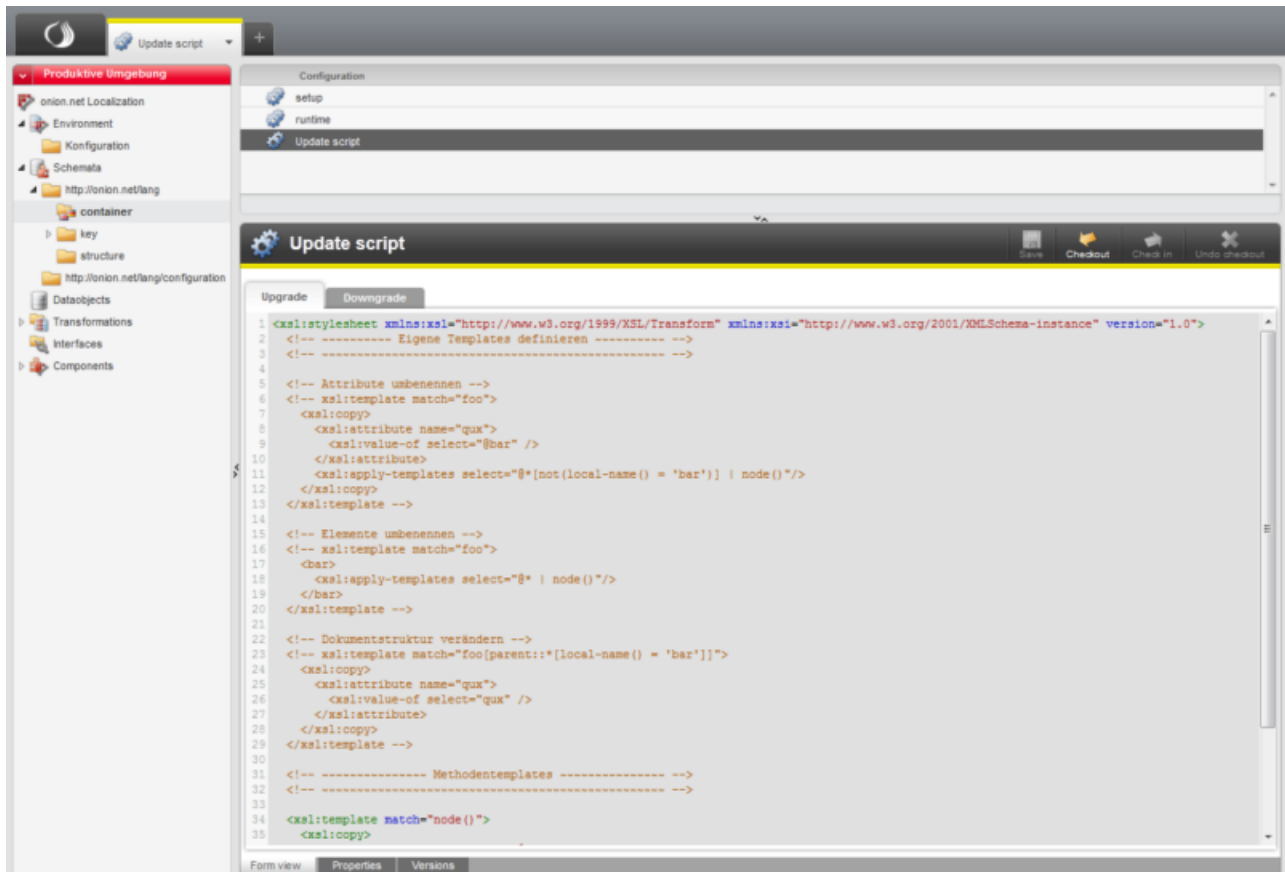


Abbildung 30, Seite 31

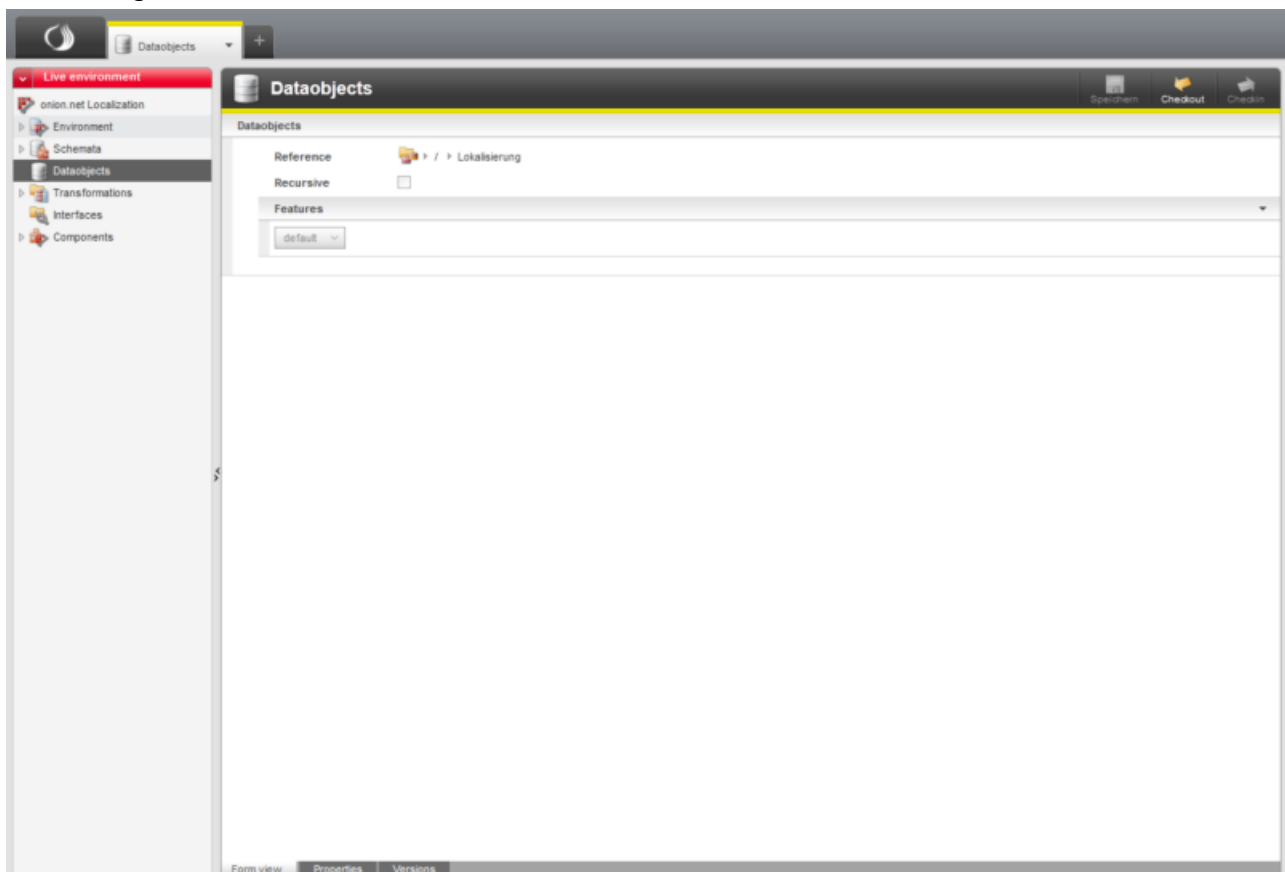


Abbildung 31, Seite 31

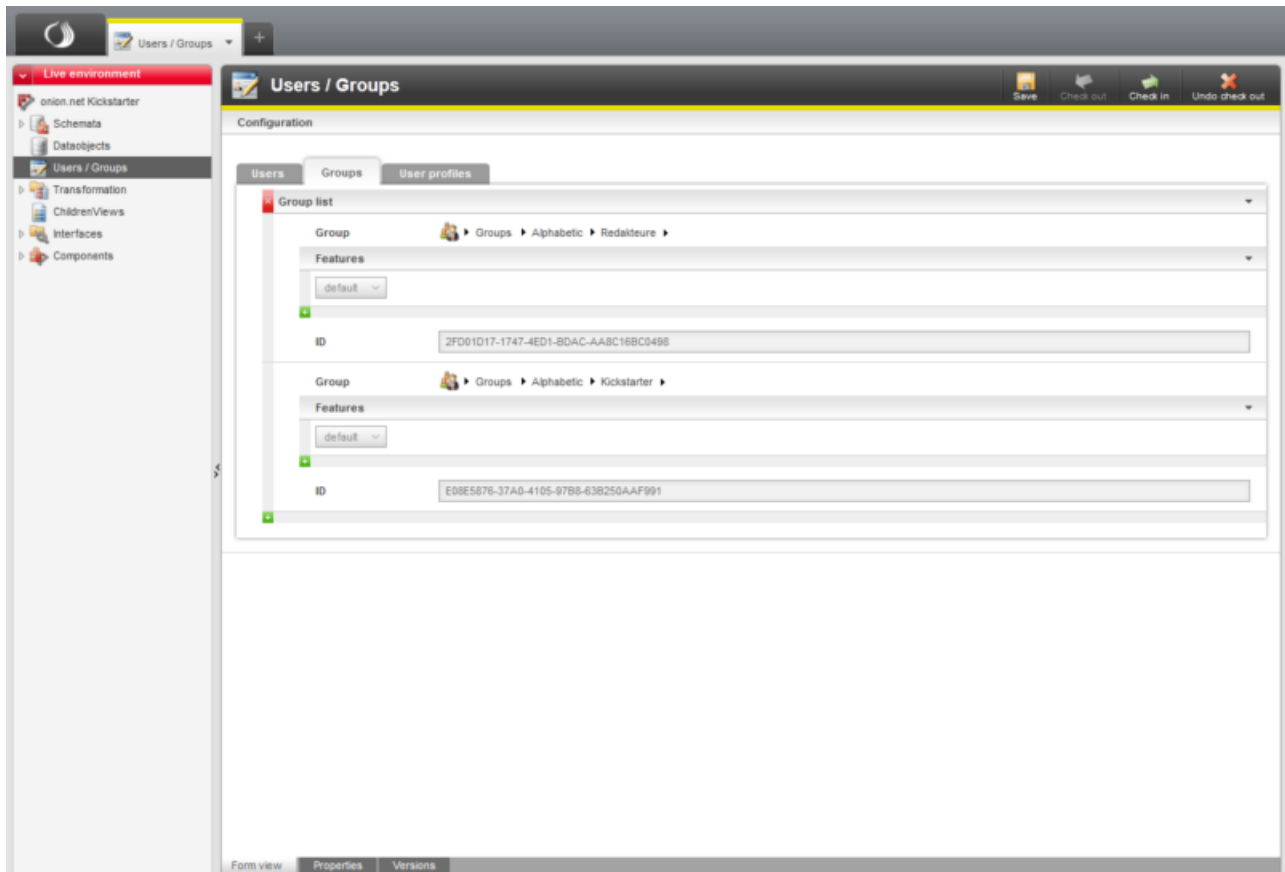


Abbildung 32, Seite 58

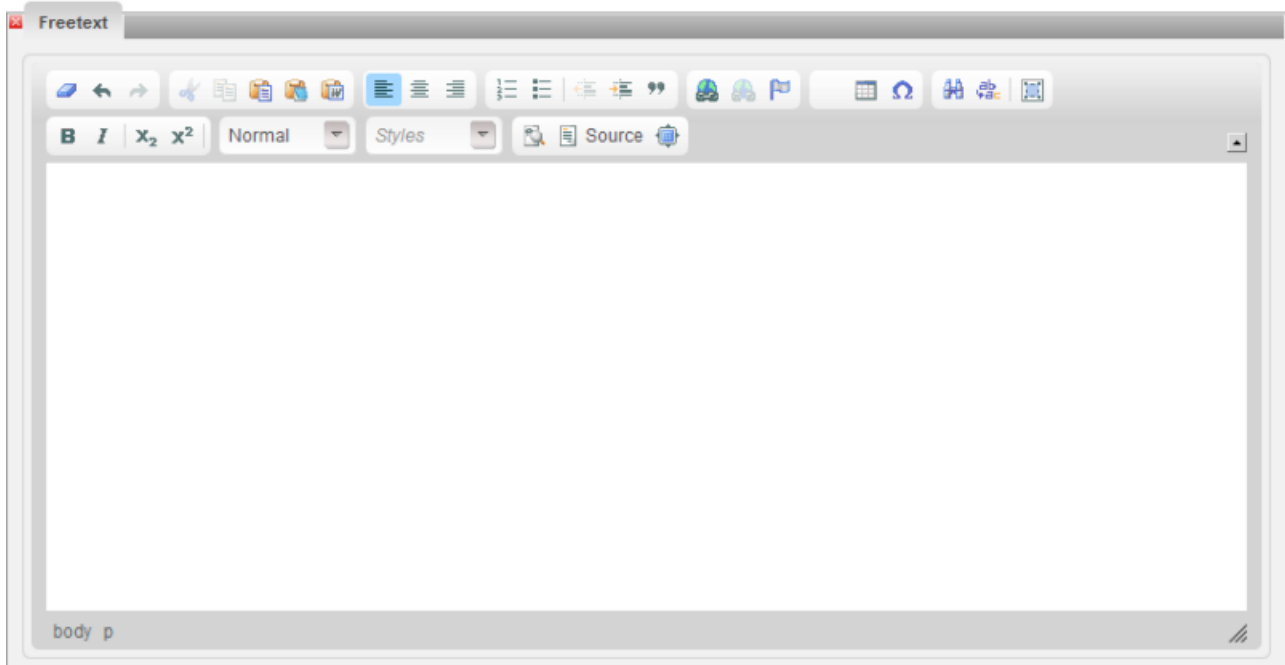


Abbildung 33, Seite 64

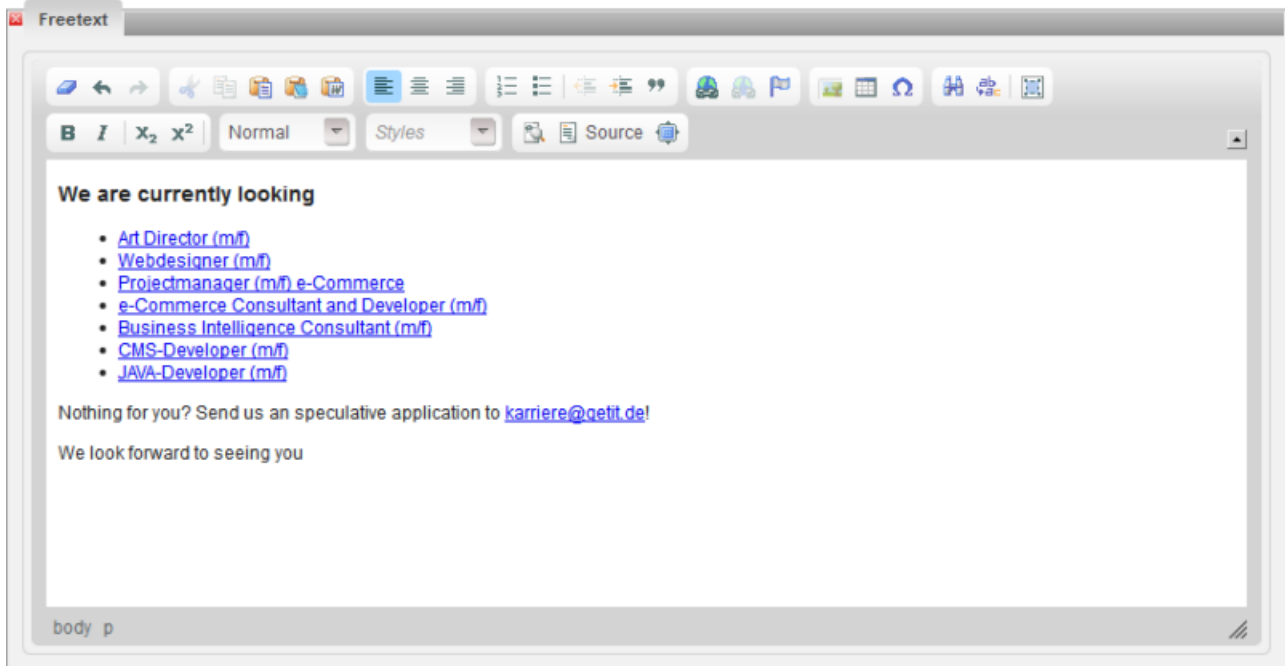


Abbildung 34, Seite 71



Abbildung 35, Seite 74



Abbildung 36, Seite 75

The screenshot shows the DokuWiki interface for the 'Leistungen' page. On the left is a sidebar with a tree view of the site structure, including 'Website', 'Unternehmen', 'Leistungen' (selected), 'Referenzen', 'Aktuelles', 'Karriere', 'Kontakt', 'Suchergebnisseite', 'Login', 'E-Mail an', 'Sitemap', '404', '500', '503', 'Datenschutzerklärung', 'Tags/Marker', 'Kunden', 'News', 'Events', 'Downloads', and 'Kontakte'. The main content area has a title 'Leistungen' and a toolbar with buttons for 'Save', 'Check out', 'Check in', 'Undo check out', and 'Rename'. Below the title is a 'Versions' section with a checkbox for 'Include structural changes'. A table lists the version history:

Version	User	Last modification
5.1	admin	3/16/2012 3:10:37 PM
5.0	admin	1/28/2010 5:02:36 PM
4.0	admin	1/22/2010 6:36:03 PM
3.0	admin	1/6/2010 11:24:35 AM
2.0	admin	12/8/2009 1:11:24 PM
1.0	webserver	12/1/2009 4:21:44 PM

At the bottom of the main area are tabs for 'Form view', 'Properties', 'Versions' (active), 'Rights', 'References', and 'Xml raw data'.

Abbildung 37, Seite 77

The screenshot shows the 'ChangeSet Doku' page in DokuWiki. The top navigation bar includes 'Overview', 'Metadata', and 'Changes' (selected). On the right, there are buttons for 'Commit', 'Discard', and 'Configure'. The page displays the following information:

- ChangeSet name:** ChangeSet Doku
- Number of changes:** 8
- Participated editors:** admin

Below this is a section titled 'List of the last 25 changes in ChangeSet' with a checkbox 'Show changes of all users' which is checked. A table lists the recent changes:

Name	Path	Editor	Last modification	Type
Unternehmen	Website / getit.de	admin	3/16/2012 3:11:41 PM	
Aktuelles	Website / getit.de	admin	3/16/2012 3:11:03 PM	
Referenzen	Website / getit.de	admin	3/16/2012 3:10:50 PM	
Leistungen	Website / getit.de	admin	3/16/2012 3:10:37 PM	

Abbildung 38, Seite 78

ChangeSet Doku

Commit

Discard

Configure

Overview

Metadata

Changes

				Changes						Status			
Name	Path	Editor	Last modification										
Leistungen	Website / getit.de	David Mahl (getit GmbH)	3/16/2012 3:10:37 PM										
Referenzen	Website / getit.de	David Mahl (getit GmbH)	3/16/2012 3:10:50 PM										
Aktuelles	Website / getit.de	David Mahl (getit GmbH)	3/16/2012 3:11:03 PM										
Unternehmen	Website / getit.de	David Mahl (getit GmbH)	3/16/2012 3:11:41 PM										

Abbildung 39, Seite 78

ChangeSet Doku

Overview

Metadata

Changes

Name	Path	Editor	Last modification	Changes					Status		
Referenzen	Website / getit.de	David Mahl (getit GmbH)	3/16/2012 3:10:50 PM								
Aktuelles	Website / getit.de	David Mahl (getit GmbH)	3/16/2012 3:11:03 PM								

Abbildung 40, Seite 79

ChangeSet Doku

Commit

Discard

Configure

Overview

Metadata

Changes

Name	Path	Editor	Last modification	Changes					Status			
Leistungen	Website / getit.de	David Mahl (getit GmbH)	3/16/2012 3:10:37 PM									
Referenzen	Website / getit.de	David Mahl (getit GmbH)										

Functions

List of modifications

Abbildung 41, Seite 79

ChangeSet Doku

Overview

Metadata

Changes

Name	Path	Editor	Last modification	Changes						Status			
Leistungen	Website / getit.de	David Mahl (getit GmbH)	3/16/2012 3:10:37 PM										
Referenzen	Website / getit.de	David Mahl (getit GmbH)	3/16/2012 3:10:50 PM										