@enterprise 8.0

System Administration

December 2017

Groiss Informatics GmbH

Groiss Informatics GmbH

Strutzmannstraße 10/4
9020 Klagenfurt
Austria

Tel: +43 463 504694 - 0
Fax: +43 463 504594 - 10
Email: support@groiss.com

Document Version 8.0.22989

Introduction

This manual describes the administration of the Workflow-Management-System **@enterprise** . It is written for readers, who administrate the system, define users or the organization structure, or define workflows.
The manual is structured as follows:

- **System Architecture:** The architecture of **@enterprise** is described.
- **The HTML Interface:** The structure and usage of the HTML interface for administration is described.
- **Ids, Names and internationalization:** Here you can find information about which attributes of an object class are used as ids and how the conventions for ids look like in **@enterprise**. Furthermore you can find information about the internationalization of master data and object classes.
- **Definition of the Organizational Structure:**

  Modelling the objects and the structure of the organization is necessary for modelling workflows. The following object classes are maintained in **@enterprise** :

  - **Server:** An **@enterprise** installation can consist of several servers, which cooperate for workflow execution.
  - **Roles:** Roles define groups of participants exhibiting a specific set of attributes, qualifications and/or skills.
  - **Rights:** Rights are used to restrict some operations to selected users.
  - **Permission Lists:** It is possible to combine several rights to permission lists. This permission lists can be assigned to users or roles.
  - **Users:** All persons, which work with **@enterprise** , must be registered as "users".
  - **Organizational units:** The structure of the organization is modeled with organizational units and the hierarchy between them. Organizational units are abbreviated by *OU*.
  - **Organization Class:** Organization classes are used to classify the organizations.

  - **Organization Hierarchies:** OUs can form hierarchies, i.e. one OU can be subordinate to another one and vice versa. The hierarchy of OUs is defined by restoring the corresponding OUs into the organization hierarchy. In doing so one superordinate OU can own several subordinate OUs, but a subordinate OU (in one organization hierarchy) can only belong to one superordinate OU. A OU can be arranged in several organization hierarchies (in this way it is possible to map OUs belonging to several divisions).

- **The @enterprise Right System:** This chapter describes the right system of **@enterprise**, which enables you to assign the required rights to users.

- **Workflow Modelling:** Using the organizational structure we can define processes (workflows). The following object classes are described in the respective chapters:

  - **Applications:** Applications group processes.
  - **Tasks** are the elementary activities of processes.
  - **Functions** are representations of interactive Java-methods used for execution of activities.
  - **Forms** are the data containers for local data of processes.
  - **Processes** describe the structure of a business process.
  - **Interfaces** allow the start of process instances by submitting an HTML form.

- **Process definition:** In this chapter, the definition of processes is described. It contains two sections, the definition with the script language WDL and the definition using the graphical process designer. It is also possible to define processes with XWDL – an extension of WDL – which is described in the *XWDL Handbook* of **@enterprise**.

- **Searching in @enterprise:** Here you can find cross references to those documents which are describing the possibilities to find certain information within **@enterprise**.

- **Administration Tasks:** The search facility and a set of common administration functions is described.

- **Configuration:** This chapter describes the configuration of **@enterprise**–server.

- **Dashboard:** This chapter describes how you can use the dashboard of **@enterprise**.

Contents

1 System Architecture

The workflow system **@enterprise** is build for using in the intranet and internet and based on the technologies of the World Wide Web. We briefly describe these concepts before explaining the architecture of the system.

1.1 The World Wide Web

Three concepts make up the World-Wide Web (WWW): uniform addressing of information in the Internet via the Uniform Resource Locator (URL), presentation of information in the Hypertext Markup Language (HTML), and transmission of data using the Hypertext Transfer Protocol HTTP.
The HTML format allows the integration of different media type into a document. So-called hyper-links enable the integration and connection to other documents or media types. Important for using the WWW for workflow systems is the feature of fill-in forms in HTML, which allows a form based interaction between the user and a program.
HTTP is a simple protocol for transmitting information over the net. The client (browser) requests a document from a server, by opening a socket connection and sending the URL of the document to the server. The server sends back the content of this document together with some status information. If the URL points to an executable program the server executes this program and sends the output back to the client. Moreover, the HTTP protocol provides a mechanism for user authorization allowing to restrict access to a group of users or hosts.

1.2 The System Components

Fig. 1.1 shows the components of the system. The components in detail:

- *Database:* The database contains all data relevant for process execution, process definition, organizational hierarchy, roles, as well as the dynamic data of the process instances.
- *Workflow Engine:* This component contains the interpreter for the defined processes, it is called whenever a process is started or an activity is finished through the user interface. Additionally, the engine comprises services like timers, import-export mechanisms, the monitoring component, etc.

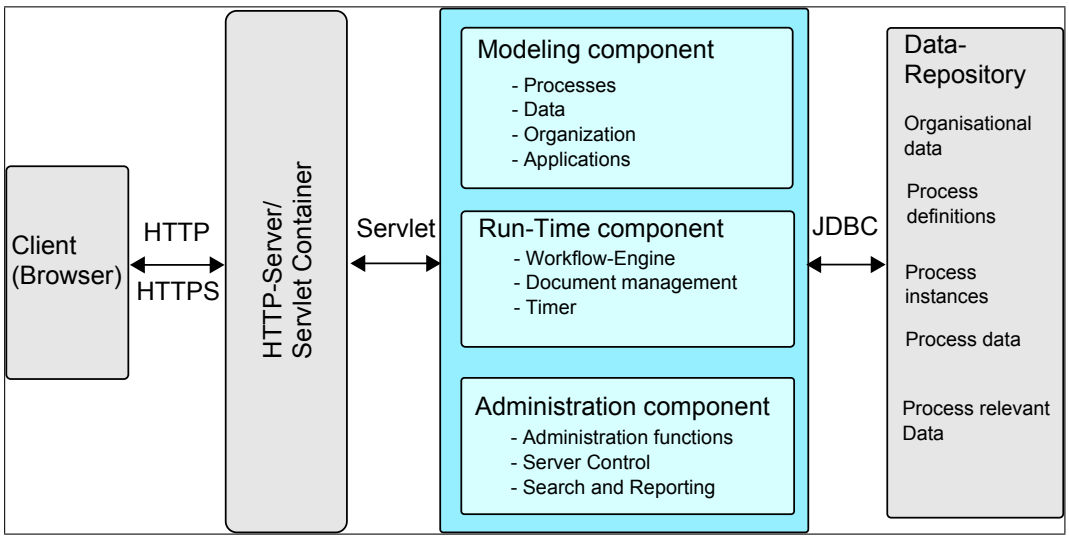


Figure 1.1: **@enterprise System Architecture**

- *HTML Interface:* The HTML interface creates the HTML pages of the user interface. It is triggered from the HTTP server whenever a user clicks on a link or a button. On the back end it communicates with the @enterprise engine via the API. The HTML interface consists of the following parts:

  – *Workflow client:* It generates the HTML pages and forms used for interaction with the user (not administrator) of **@enterprise** . The main page is the user worklist, which contains links to the other relevant information, i.e. the forms, process descriptions, history, etc.

  See the User Manual for a description of this interface.

  – *Administration and Monitoring Tool:* It contains functions for creating, modifying and deleting users, roles, and organizational units. It also allows the inspection and modification of running processes, like terminating instances, reassigning steps, etc. Like the other components communicating with the HTTP Server, the interactions with the user are done by creating and receiving HTML pages and forms.

  Two interfaces are available for process definition: Workflows defined as WDL scripts can be compiled and loaded into the system. Additionally, the process editor allows graphical definition of processes. Both components are accessible using a Web browser.

  Forms are created using a standard HTML editor. A parser extracts all input fields from the form and presents the user with a suggestion for the definition of the corresponding database table. The user can alter the data-types and creates the form table. The HTML form is stored in the database.

- *HTTP Server:* The HTTP server is the interface between the Web and the workflow system. It translates the requests from the users to calls of the corresponding procedures of the workflow system.

- *Browser:* Every interaction with the system is done by a Web browser. This allows wide availability and platform independence and made system implementation easier.

2 The HTML–Interface

For using the **@enterprise** administration component you must have a web browser installed on your machine (Internet Explorer, Netscape, Mozilla, etc.).

Login to the system either as sysadm or as another user. In the latter case you will be redirected to the worklist component. Click the @enterprise menu and "Administration" to enter the system administration (you will see this link only if you have the right "admin"). Fig. 2.1 shows the structure of the main window.

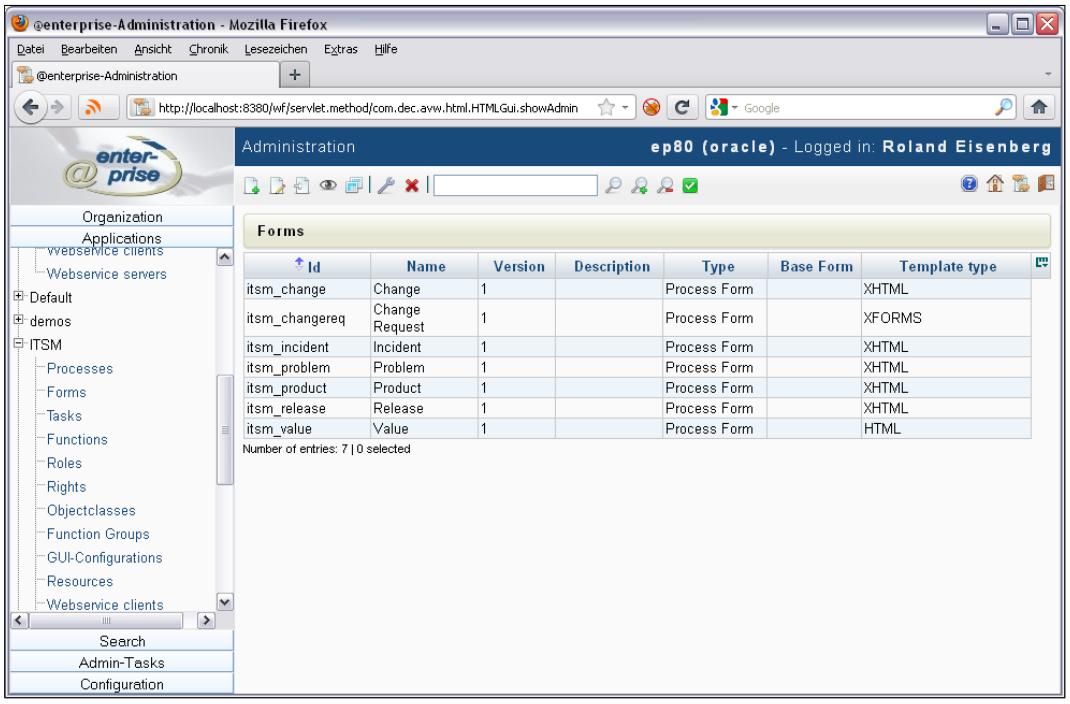


Figure 2.1: **System Administration**

The interface is split up in the following parts:

- **Information:** The top frame contains information about the logged in user and actual running server.

- **Toolbar:** Directly positioned under the information frame is the toolbar which contains different functions for manipulating the informations displayed in the workingarea. Four functions are always visible on the right end of the toolbar:

  - **Help:** opens **@enterprise** help in a new window.
  - **Dashboard:** shows your dashboard in the workingarea.
  - **Worklist:** switch to the worklist component of **@enterprise** .
  - **Logout:** logout from **@enterprise** .
  - **Note:** If this symbol appears, a modification at the **@enterprise**-system was made. By clicking the symbol you will get nearer information, if you have to restart the server or have to refresh the cache-structures.
  - **Information:** If this symbol appears , news are available. By clicking the symbol a popup will be opened, which contains the news.

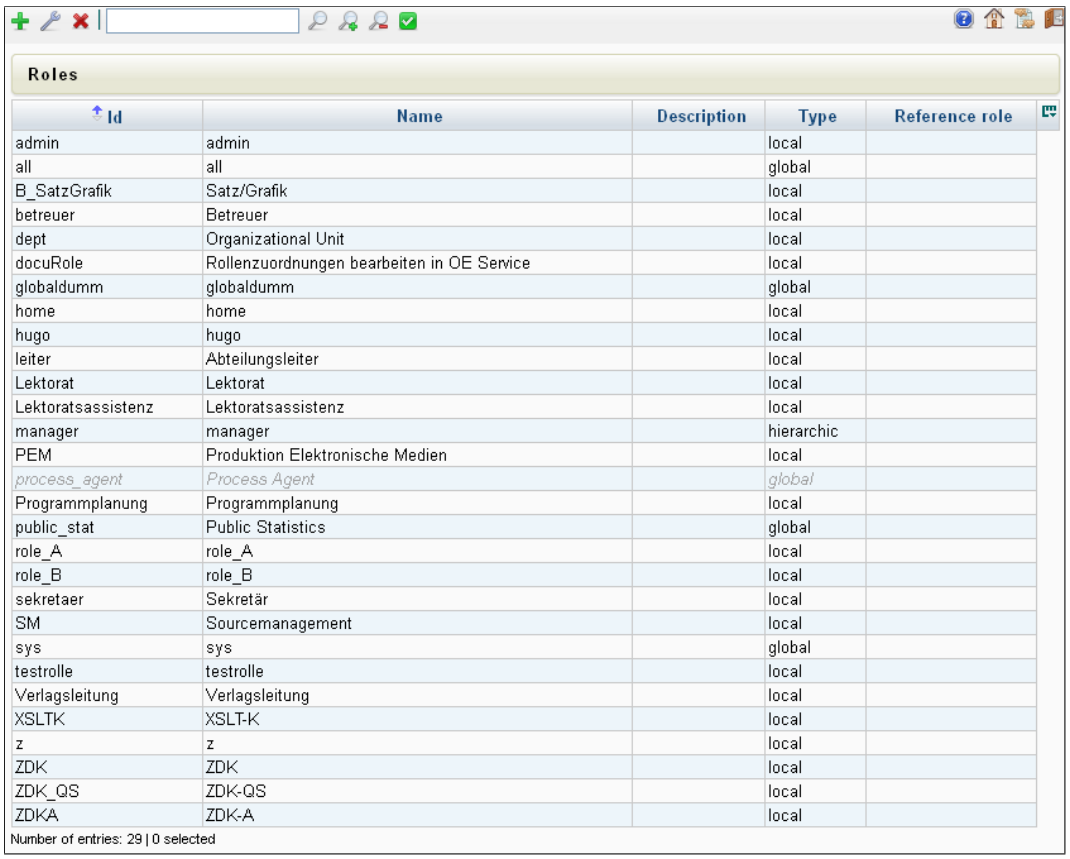
- **Navigation:** the navigation frame on the left contains the following elements:
  - **Organization:** Contains links for administration of the application-independent information: *User*, *Organizational Units*, *Organizational Classes*, *Organization Hierarchies*, *Servers*, *Permission Lists*, *Interface* and *Keywords*.
  - **Applications:** This area contains subtrees for every application. For each application a link to its *Processes*, *Forms*, *Tasks*, *Functions*, *Roles*, *Rights*, *Objectclasses*, *Function Group* and *GUI-Configuration* is shown. This area also contains a link to the overview of all applications, called *Application list*.
  - **Search:** This folder contains links to the various search functions (*Process Search*, *Document Search*, *Extended Search* and *Stored Queries*).
  - **Admin-Tasks:** Shows a list of administration tasks, for example for restarting the server, exporting data, etc.
  - **Configuration:** All functions for configuring your installation are placed here.

- **Working area:** The working area is the main part of the interface. It contains different masks and tables for manipulating the master data, configuration etc. After opening the administration your dashboard is displayed here. You can change the content of the working area by activating a link of the navigation area.

2.1 Tables

Master data are displayed in tables initially. The table contains the different objects in its rows and the columns show different information of the respective object.
Detailed information and additional functions for the object are displayed in an own window (see chapter 2.2). You can open this window by double-clicking a row in the table or selecting the row first and activating the toolbar-function *edit* secondly.

Before the table is shown, the system checks the length of the table. If it exceeds the defined limit, the system asks the user whether he will view the full table. The limit can be



Roles

Id	Name	Description	Type	Reference role
admin	admin		local	
all	all		global	
B_SatzGrafik	Satz/Grafik		local	
betreuer	Betreuer		local	
dept	Organizational Unit		local	
docuRole	Rollenzuordnungen bearbeiten in OE Service		local	
globaldumm	globaldumm		global	
home	home		local	
hugo	hugo		local	
leiter	Abteilungsleiter		local	
Lektorat	Lektorat		local	
Lektoratsassistenz	Lektoratsassistenz		local	
manager	manager		hierarchic	
PEM	Produktion Elektronische Medien		local	
process_agent	*Process Agent*		*global*	
Programmplanung	Programmplanung		local	
public_stat	Public Statistics		global	
role_A	role_A		local	
role_B	role_B		local	
sekretaer	Sekretär		local	
SM	Sourcemanagement		local	
sys	sys		global	
testrolle	testrolle		local	
Verlagsleitung	Verlagsleitung		local	
XSLTK	XSLT-K		local	
z	z		local	
ZDK	ZDK		local	
ZDK_QS	ZDK-QS		local	
ZDKA	ZDK-A		local	

Number of entries: 29 | 0 selected

Figure 2.2: **Example for Tabledisplay (Roles)**

configured in the system configuration (parameter group Localization).

Following formats are used to display the tablerows:

- **Last Changed:** The row which is changed at last is colored.
- **Inactive Entries:** Inactive objects are displayed with grey and *italic* letters. Additionally forms, where the formclass can not be loaded are marked as inactive entries, too.
- **Selected Entries:** Actually selected entries are colored.

2.1.1 Column picker, Sorting and Filter

You can change the number of displayed columns by using the column picker. The column picker is placed rightmost of the table header. Activate the functions and a popup-window containing the names of all actually visible and possible columns opens.

Already visible columns are displayed with a small checkmark. To add a new column to the table, activate a column name (without the checkmark). The table refreshes and the selected

column is displayed. To remove a column from the table, activate a column name (with the checkmark). The table refreshes without the removed column.

You can change the sorting column and sorting direction by activating a column header. Which column and direction is actually used for sorting is marked by a small arrow left of the column name.

The link *Filter* helps you to keep an overview if your table contains a lot of entries. The filter can be seen as selection criteria to mask certain entries in your table.

By clicking on the corresponding column header of your table a context sensitive filter menu with the following entries is shown:

- Order Ascending: The entries of the table are ordered in ascending order by the current column.
- Order Descending: The entries of the table are ordered in descending order by the current column.
- All Entries: The use of the column filter of the current column becomes nullified.
- User Defined: By selecting this menu item a HTML–page is shown where you can enter a certain value. If you confirm your entries in this page by clicking the button ”Ok” the table is filtered by the corresponding value.
- The first 20 different column entries; if you select one of these entries the column becomes sorted by this entry.

If you want to save the current combination of filters you have to click the link ”Filter” in the heading of the table. The filter menu is shown:

- Save Filter: By selecting this menu item you save the current combination of column filters under a name defined by you. You can also enter a description for the filter.
- Delete Filter: By selecting this menu item you delete the filter which is currently active. There is no undo function for deleting a filter!
- All Entries: The use of the saved filter is nullified.
- A list of all saved filters. If you select one of these entries the table is filtered by this filter. The list can also contain filter which have been defined by the system administrator. These filters can only be used but not deleted by you.

When a filter is selected only those entries of the table are displayed which match all the criteria specified by the filter.

2.1.2 Standard Functions

Following functions are displayed for manipulating most of the tables in the administration:

- **New:** opens object-details for creating a new object.

- **Edit:** opens object-details for updating, deleting, viewing the history etc. the information. Depending on the class of the object further functions may be available on this page.

- **Delete:** deletes selected objects.

- **View:** opens object-details in read-only-mode, excepting forms and processes

- **Search:** If you insert a search string and click to "Search" button the result list will contain all objects matching the search string. Normally, the string is matched against the id and name of the objects, the text left of the input field names the search attributes.

- **Extended Search:** With the button "Extended Search" you can search in all attributes of the object.

- **All Entries:** views the complete list of objects of the class.

- **Select All Entries:** mark all entries as selected by activation this function.

- **Refresh:** Refresh the content of the working area.

2.2 Object details

The detail view of an object can be opened by double-clicking the entry in the table, or selecting the table row and activating the *edit*-function in the toolbar. The object-details are buildup as tabbed pane. Each tab has its information and function to the actual object (see Fig. 2.3).
The main functions of the object details are:

- **OK:** Activating this button saves the changes in the database and closes the window. The table refreshes.
- **Apply:** Activating this button saves the changes in the database and refreshes the table. You can activate this button only if the actual tab contains a mask where you can edit the information directly.
- **Cancel:** Close the window and discard the changes.
- **Delete:** Delete this object from the database.

2.2.1 Tab: General

In general the tab *General* is the first tab of the object-details. Here you can view or edit the general settings of the actual object. After changing the attributes save them through activating the button *Ok*, *Apply* or changing the tab. In this tab the button **delete** is active, too. This function is the same as the function *Delete* in the toolbar outside.

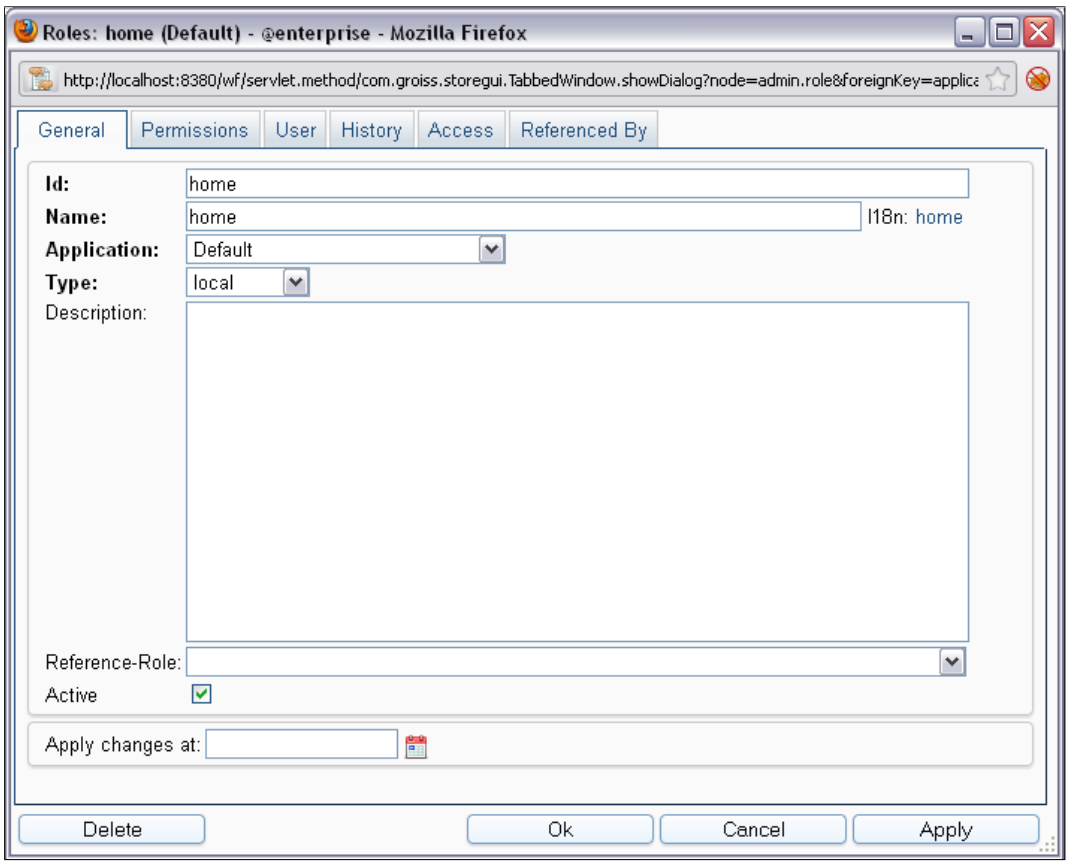


Figure 2.3: **Objectdetails: Example**

Apply changes later

Some objects can be changed so that the changes become effective at a future date. The field "Apply changes at" on the detail mask provides this functionality.

Insert in the field **Apply changes at** the date (and time) the changes should get effective. After activating the button *Ok*, *Apply* or changing the tab the deferred changes are saved.

If you view the detail mask of an object with such pending changes, you will see the date when the changes get effective in the field *Object changes at*. Activating the icon beside this field opens the detail-view of the changes. Here you can discard the changes by activating the button **Discard Changes**.

Activating / Deactivating objects

Some objects have the attribute "active" indicating whether the object is currently usable or not. In the detail mask of these objects you can manipulate this attribute with a checkbox. If the checkbox is not checked, the object is inactive. This means for:

- users: the user cannot log in and cannot receive a worklist entry.

- processes: the process cannot be started (except via the API).
- roles, role assignments: the role cannot receive a worklist entry.

In the table of objects, the inactive items have a grey background and *italic* letters.

Internationalization

The name of application-dependent objects can be translated into the available languages.

The name translated into the actually used language is displayed beside the field *Name* as link. After activating this link the internationalization for all available languages is displayed. Clicking the button *Close* closes the window. How you can change the internationalization is described in chapter 3.

2.2.2 Tab: History

This tab shows the history of changes on this object (see Fig. 2.4). You can even view the older versions of the object by activating the function *view* in the toolbar.

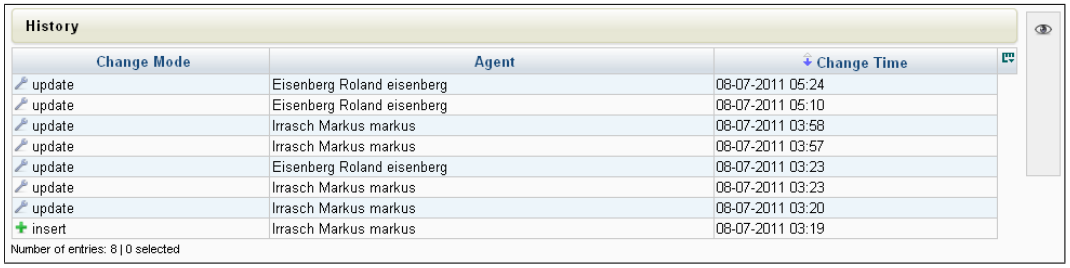


Figure 2.4: **Tab: History**

2.2.3 Tab: Access

This tab shows you who has which access to the object directly or indirectly via permission lists (see Fig. 2.5). You can edit the access rights to this object here, see chapter 5.

2.2.4 Tab: Referenced By

drilldown.png

If you select the tab **Referenced By**, an overview about all objects will be shown, which reference on the current object (see Fig. 2.6). The objects are displayed in a hierarchical structure. The symbols will be described as follows:

- *Plus-sign:* this object has one or more sub-objects, which are not shown yet. If you click on the plus-sign, the sub-objects will be shown. Furthermore the plus will be converted into a minus.

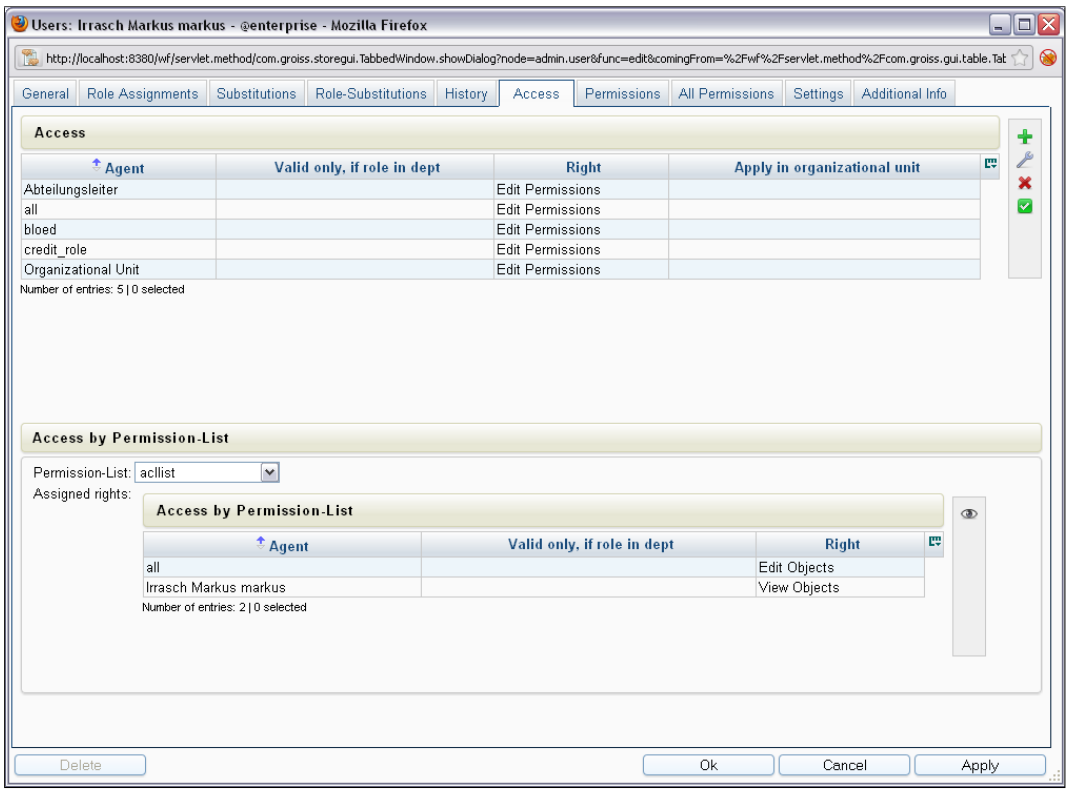


Figure 2.5: **Tab: Access**

drillup.png

- *Minus-sign:* this sign shows, that a hierarchy is already expanded. If you click on the minus-sign, all objects of this hierarchy will be hidden. Furthermore the minus will be converted into a plus.

- *Expand all:* by this sign the whole objects can be expanded or the sub-objects can be collapsed.

drilloff.png

- *Blue circle:* shows, that a detail view of the object exists.

2.2.5 Further functions

Some functions are used in the masks again and again. The following chapters describes this functions.

- **Select:** Activating this function opens a new window where you can select a object. The selected value is inserted in the field beside this function. For example: selecting a user, an organizational unit.

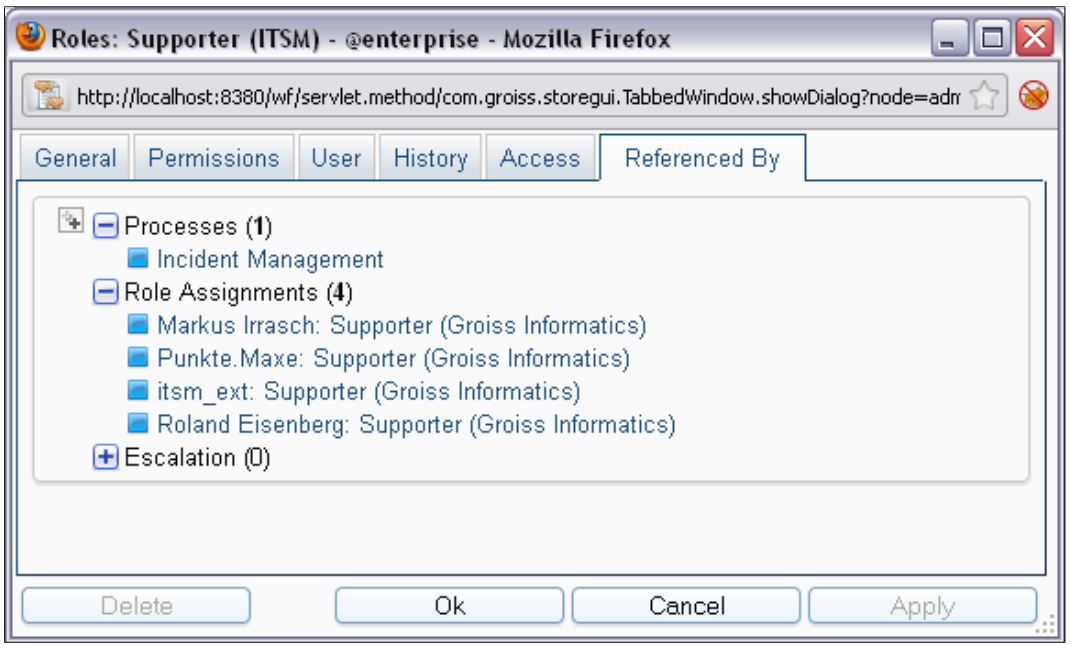


Figure 2.6: **Tab: Referenced By (Roles)**

- **Remove:** Activating this function removes the value of the field beside. This function is always combined with the *Select*-function.

Since @*enterprise* version 8.0 drop-down lists are integrated. By activating this symbol, the content of the list is displayed, where you can select the needed object.

- **Calendar:** After activating this function a calendar is displayed. The calendar helps you selecting a date. Detailed information can be found in the user manual.

- **Classpath-Checker:** With the Classpath-Checker you can check URLs. The existing class and also the existing method and the correct method-signature will be checked. In special cases will be checked, if the class implements the required interface (e.g. Logger Class must implement the interface com.groiss.log.ILogger). If the URL can be found in the classpath, the symbol of the Classpath-Checker changes its color to green. In any other case the color of the Classpath-Checker is red.

3 Ids, Names and Internationalization

3.1 Ids and Names

In **@enterprise** all master data objects are identified internally with an unique identifier (*id*). The *name* is normally used in the user interface. According to the object class the following attributes are used as identifiers:

- id
- name
- both the id and the name
- a combination of id and version
- a combination of name and version

The object classes and their corresponding identifiers are listed in table 3.1.

Within **@enterprise** the id of an object is unique and furthermore the id is also unique for all applications of **@enterprise**. Therefore it is not possible to create an object of the same class in different applications with the same ids (e.g. user *A* in application *X* and user *A* in application *Y*).

Another peculiarity of **@enterprise** is, that the user and roles are sharing their scope, i.e. it is not possible that within one **@enterprise**–server there are a user and a role which ids are identical or where the name of the user corresponds to the id of the role or vice versa.

For a syntactically correct *id* the following rules apply:

- Ids start with a letter or a $ or / or \. Then, additional characters from the described set or digits can follow.
- The complete length of an id must not exceed 80 characters.
- Ids can also contain special characters (e.g. email-addresses), but whitespaces, exclamation marks and commas are not allowed. In a WDL definition the agent-id must start with an exclamation mark, if the id is no "simple" id.
  Example:

Object class	*Identifier*
User	Id
Organizational Unit	Id
Task–Function	Id
Access List	Name
Object Class	Name
Function Group	Id
Role	Id, Name
Right	Id, Name
Organizational Class	Id, Name
Organizational Hierarchy	Id, Name
Application	Id, Name
Server	Id, Name
Task	Id+Version
Process	Id+Version AND Name+Version
Form	Id+Version AND Name+Version

Table 3.1: **Object classes and their identifiers**

```
!right.user@xy.com do_something(f);
```

3.2 Internationalization of Meta Data Objects and Object Classes

In **@enterprise** it is possible to internationalize object classes and all meta data where it makes sense. Meta data which can be internationalized are:

- Applications
- Tasks
- Task Functions
- Roles
- Rights

In implementing a corresponding *java.lang.ResourceBundle* and putting it into the corresponding application directory, it is possible to internationalize your own applications. For further details on this topic read the programming handbook of **@enterprise**. There you find also informations on how to internationalize the meta data of the default application.

4 Definition of the Organizational Structure

4.1 Roles

Roles define groups of participants exhibiting a specific set of attributes, qualifications and/or skills. Examples are *Supervisor* or *Insurance Underwriter*. To assign a role to a user you must first define the role, then assign it to one or more users (see the next section).

The object-details of roles contain the following tabs:

- General
- Permissions
- User
- History
- Access
- Referenced By

4.1.1 Tab: General

You can edit the following attributes (required fields are bold):

- **Id:** Unique identifier of the role.
- **Name:** Name of the role. By activating the I18n-link beside this field, the translations (if defined in application mask - tab *Properties*) of this key are displayed and can be edited directly by changing the values and activating the button *Save*. The changes are stored in the resource file of this application (see section 6.9).
- **Application:** Application, the role belongs to.
- **Type: @enterprise** distinguishes three role types:
  - **local:** A local role is assigned to a user in one organizational unit.
  - **global:** A global role is independent of organizational units.

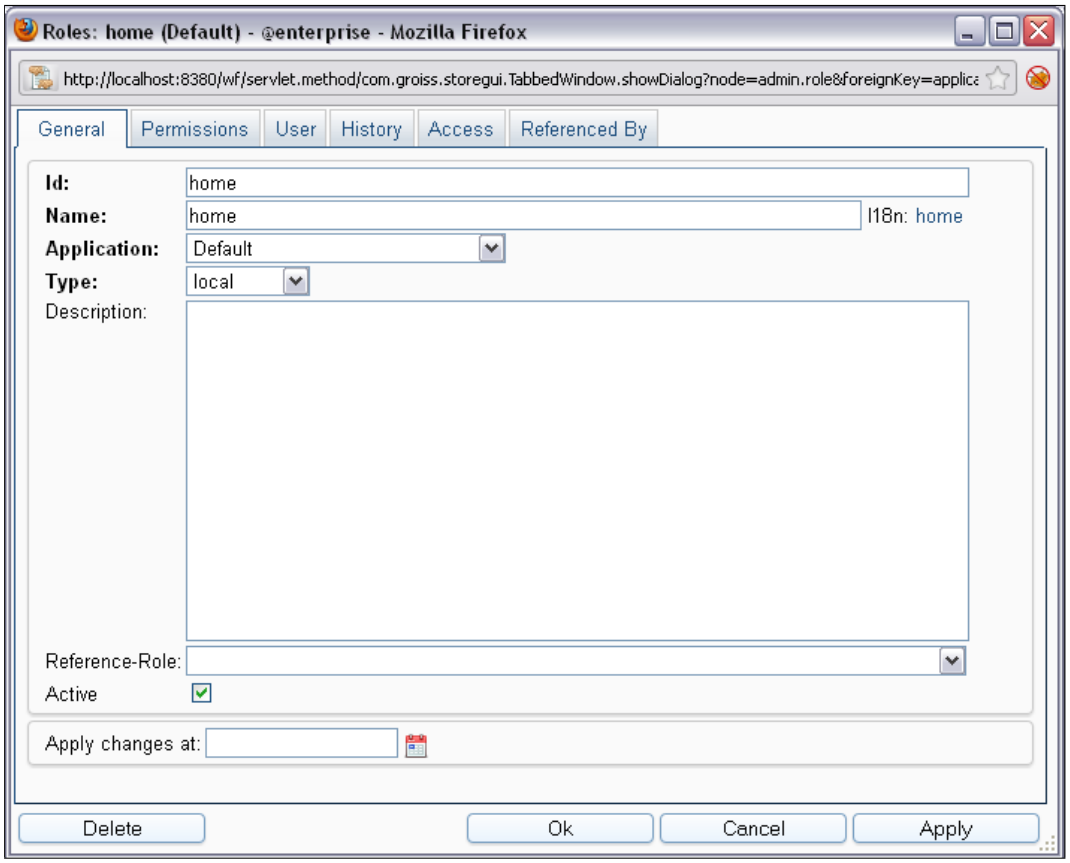


Figure 4.1: **Objectdetails: Roles**

  - **hierarchic:** A hierarchic role is assigned to a user in an organizational unit, but it is valid also for all sub-OUs, (the organizational units which are below in the organizational hierarchy).

- Description: Free text.
- Reference Role: Reference roles are used for defining different roles with different rights but one "reference" role used in process definitions.

  1. Example: Assume we have defined the role *assi* for assistant and use this role in process definitions. the roles *asi_no_rights* oder *assi_many_rights* are assigned to persons with no or with many extra rights, respectively. Both roles have *assi* as reference roles, so that an activity assigned to the role *assi* is also assigned to the persons with the other to assi* rights.

  2. Example: Our company has assistants and a department manger. The first agent of process definition P is the role *dm_assi*. This role is a reference role of roles *dm* and *assi*. The users have the roles *dm* or *assi*, but assistants and department managers are able to start process P and have the same rights in first process step.

Note, that it is not possible to define reference roles for reference roles.

- Active: see chapter 2.2.1.

4.1.2 Tab: Permissions

In this tab you can add and edit the permissions assigned to the role. Users who are assigned to the role have the permissions assigned to the role. Use the toolbar functions for manipulating the permissions.

4.1.3 Tab: User

This tab shows you which users are already assigned to the role. You can open the details of this relationship.

4.1.4 System-defined Roles

In **@enterprise** four system-defined roles exist:

- **all**: A useful role you can assign to all users. If you define then rights for this role, everybody has this right. Processes with *all* as agent of the first task, can be started by all workflow participants (or more exactly: by everybody, who has the role *all* assigned).
- **sys**: This role is used for system administration, it allows you to perform all system administration activities.
- **home**: The home-role connects a user to a "home" organizational unit. A user can have at most one home OU.
- **dept**: The role dept is used as "Inbox" of an organizational unit. If you want to send a process instance to a OU without knowing the specific user, you can send it to the role *dept*. Note, that you must assign this role to a user, before you can use it as agent of a task.

4.2 Rights

Rights are used to restrict some operations to selected users. The assignment of rights to users is directly or using roles. See chapter 5 for a detailed descriptions of the **@enterprise** right system.

The object-details of rights contain the following tabs:

- General
- User
- History
- Access
- Referenced By

4.2.1 Tab: General

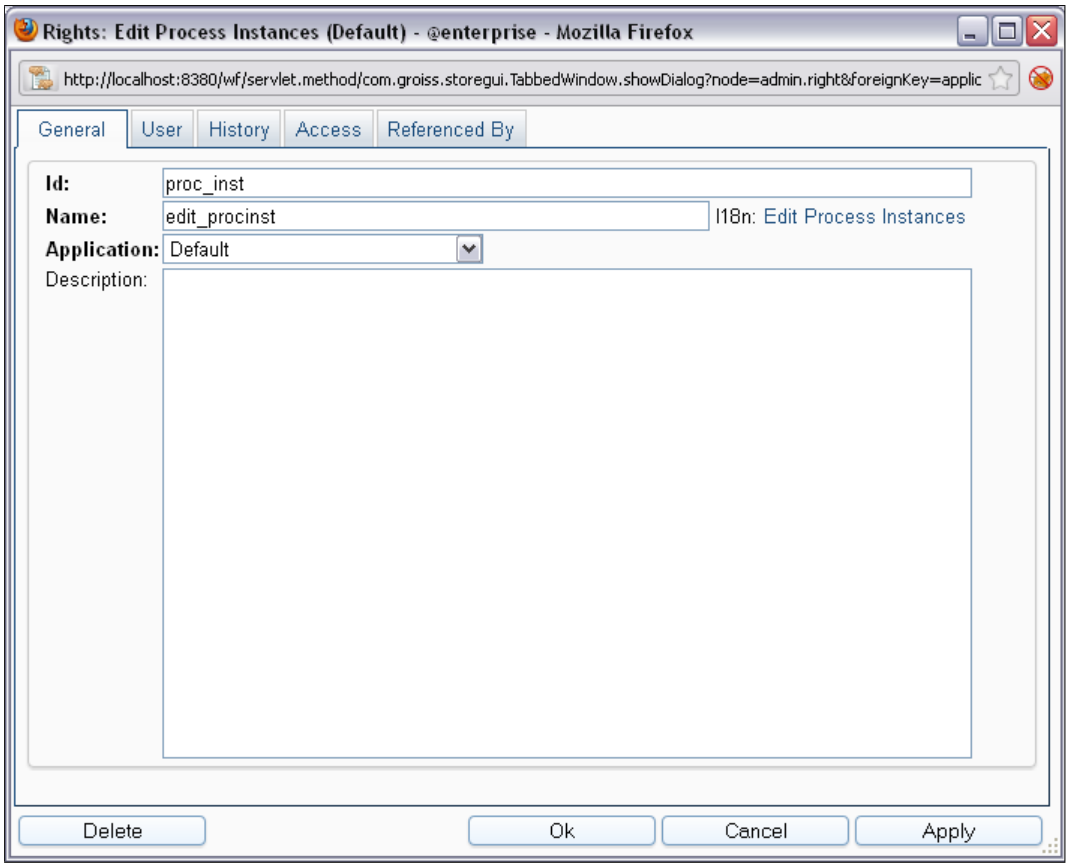


Figure 4.2: **Objectdetails: Rights**

You can edit the following attributes (required fields are bold):

- **Id:** Unique identifier of the right.
- **Name:** Name of the right. By activating the I18n-link beside this field, the translations (if defined in application mask - tab *Properties*) of this key are displayed and can be edited directly by changing the values and activating the button *Save*. The changes are stored in the resource file of this application (see section 6.9).
- **Application:** Application, the right belongs to.
- Description: Free text.

4.2.2 Tab: User

In this tab you can see a list of users who have the current right. If the right is limited to a certain object, this object is displayed in the column *Target Object*.

4.2.3 System-defined Rights

In @**enterprise** the following system-defined rights exist:

- **create:** Create an object.
- **edit:** Edit an object.
- **delete:** Delete an object.
- **edit-acl:** Edit the rights somebody has for an object.
- **view:** View something.
- **execute:** Execute the object (for example a function object).
- **conf:** Right to configure the system.
- **admin:** Right to enter the system administration. Necessary for some administration tasks, like viewing the log file.
- **view_procinst:** View process history, list of documents and notes and all process forms and versions.
- **proc_inst:** Edit Process Instances: Necessary to cancel process instances or edit the agent, view process history, list of documents and notes and all process forms and versions. The right is resolved in the context of the organizational unit of the process. If someone has this right for an OU, he may cancel all process instances that have been started in this OU. Furthermore there is the possibility to add this right to a process definition. For this purpose the object class *Processes* has to be extended by the right *proc_inst*.
- **dept_edit:** Used to edit organizational units.
- **set_agent:** Set the agent in a process instance, view process history, list of documents and notes and all process forms and versions.
- **stat:** Create statistics, except user-specific.
- **searchable:** Search in forms and list stored queries.
- **named_user:** qualify user as named user.
- **abort_step:** Abort a step in a process-instance.
- **editCal:** Used to edit calendar entries.
- **insertCal:** Used to create calendar entries.
- **viewCal:** Right to see calendar entries of other users.
- **share:** Right to allow other users to use its objects (e.g. worklist filter)

4.3 Users

All persons, which work with **@enterprise** , must be registered as "users". At the extended search the number of shown users in the user list can be influenced by different search-attributes. For example a search-attribute is the Organizational Unit, only these users will be listed, who have a role in this OU.

The object-details of roles contain the following tabs:

- General
- Role Assignments
- Substitutions
- Role-Substitutions
- History
- Access
- Permissions
- All Permissions
- Settings

4.3.1 Tab: General

You can edit the following attributes (required fields are bold):

- **Id:** unique identifier of the user.
- **Surname:** Surname of the user.
- First Name: .. of the user.
- Title: some (academic) title
- Description: Free text.
- E-Mail: Email address of the user.
- Phone Number: Phone number of the user (or some other text, we don't use this field).
- **Server:** The **@enterprise** server, where the worklist is accessible.
- **Language:** Select the language for the user interface.
- Active: see chapter 2.2.1.
- Order Attribute: free text, can be used for sorting.
- Password: Password for login.

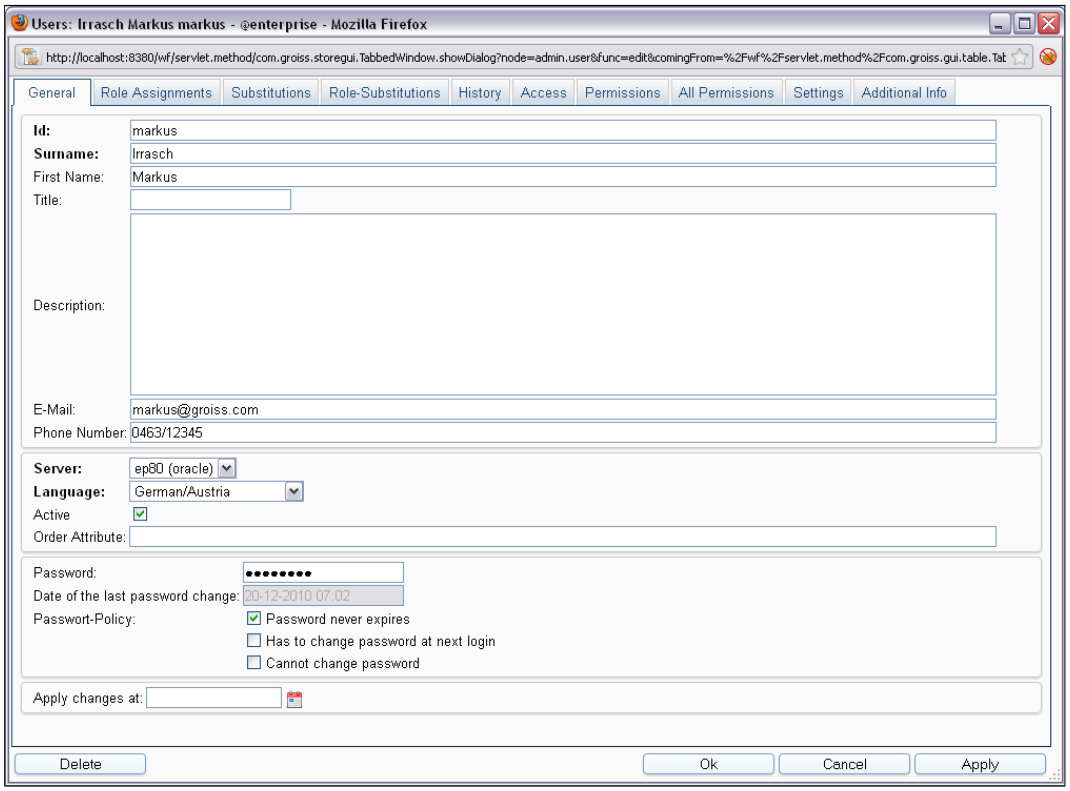


Figure 4.3: **Objectdetails: Users**

- Date of the last password change: Date, when the password was changed.
- Password-Policy:
  1. Password never expires: The password of this user never expires.
  2. Has to change password at next login: The user has to change his password at the next login.
  3. Cannot change password: The user should not able to change his password.
- Apply changes at: see section 2.2.1.

4.3.2 Role Assignments

In the role assignment mask you can specify the following attributes:

- User: The user you want to give a role.
- Role: The role you want to give to the user.
- Organizational Unit: the organizational unit where the role should be assigned. Note, that this should be left blank for global roles but is mandatory for local and hierarchic roles.
- Acitve: see chapter 2.2.1.

Define a Substitute of a Role of a User

To make substitutions more fine-grained, it is possible to define one or more substitutes for each role-assignment. Use the following steps to define such a substitution:
Activate the tab *Substitutions* of the role-assignment to add role substitutes.

Hint: The timer *CurrentSubstitutes* activates/deactivates the substitution, if a from- and/or to-date has been entered (see section 9.1.6).

4.3.3 Tab: Substitutions

For each user you can define several substitutes, each of them with an optional substitution interval.
In this tab you can define the personal substitutes. A popup window for the administration of the substitutes will be opened.

Hint: The timer *CurrentSubstitutes* activates/deactivates the substitution, if a from- and/or to-date has been entered (see section 9.1.6).

4.3.4 Tab: Role-Substitutions

The tab *Role-Substitutions* provides information about role substitutions which concern you.

This HTML–page is divided into two sections:

1. The first section, called *Users who substitute my roles*, lists all users, who substitute you in a certain role. If you are substituted in a certain role and a task is forwarded to this role, then this task also appears in the role–worklist of your substitute.
2. The second section, called *Users whose roles I'm substituting*, lists all roles you got due to a substitution. Tasks that are assigned to these roles will appear in your role–worklist.

The table *Users who substitute my roles* contains the following information:

- **Active:** Indicates, if a role is active (= green point) or inactive (= red point).
- **Role:** Name of the role your substitute have got due to his substitution.
- **Organizational Unit:** Name of the organizational unit in which your substitute have got the corresponding role.
- **User:** Here you find the name of the user who substitutes you in a certain role.
- **From:** This column shows the point in time when your substitute start having the role substitution for you.
- **Until:** This column shows the point in time until when your substitute stops having the role substitution for you.

The table *Users whose roles I'm substituting* contains the following information:

- **Active:** Indicates, if a role is active (= green point) or inactive (= red point).
- **Role:** Name of the role you have got due to a substitution.
- **Organizational Unit:** Name of the organizational unit in which you have got the corresponding role.
- **User:** Here you find the user whose role substitution you have got.
- **From:** This column shows the point in time when you start having the role substitution for this user.
- **Until:** This column shows the point in time until when you have the role substitution for this user.

4.3.5 Tab: Permissions

You can assign rights to users either directly or via roles. See chapter 5 for an introducton to the **@enterprise** right system.
Edit the personal rights of a user in this tab. A HTML–page is shown which enables you to update the actual right.

4.3.6 Tab: All Permission

The overview shows all rights, either assigned directly to the user or via a role assignment. Furthermore this tab contains a view of rights (of the user) at a specified time stamp.

4.3.7 Tab: Settings

With the help of this function the system administrator is able to update the settings of the current user. The mentioned settings are described in the user manual of **@enterprise**.

4.4 Organizational Units

The structure of an organization can be modeled under the links *Organizational Units* and *Organizational Hierarchy*. The first allows the creation and administration of the units of your organization, the second is used to define the hierarchy between them.

Note, that it is possible to define more than one organizational hierarchy. Each application uses exactly one of these hierarchies, but one hierarchy can be used in several applications.

The object-details of organizational units contain the following tabs:

- General
- Super Organizational Units
- Roles

- History
- Access
- Referenced By

4.4.1 Tab: General

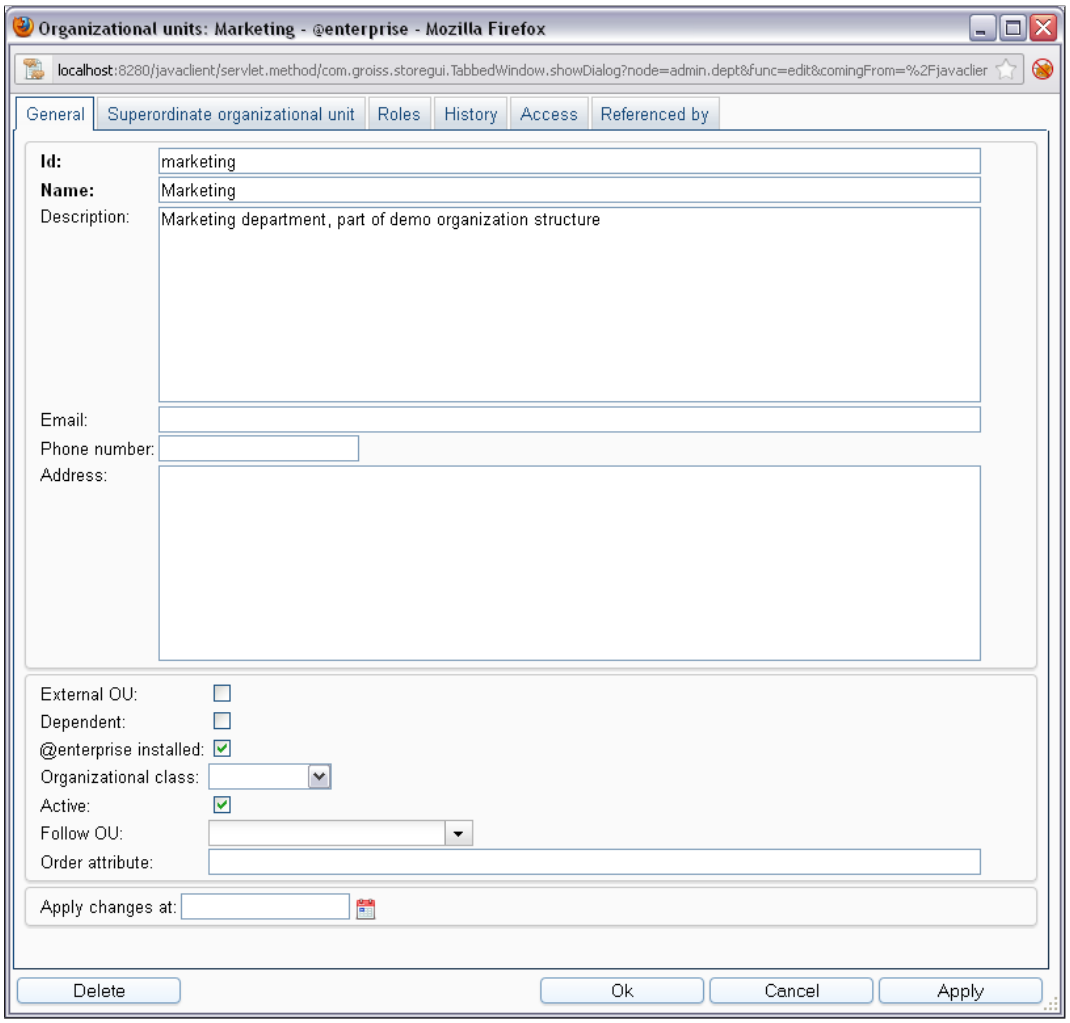


Figure 4.4: **Objectdetails: Organizational Units**

You can edit the following attributes (required fields are bold):

- **Id:** Unique identifier of the OU.
- **Name:** Name of the OU.
- Description: Free text.
- E-Mail: Email address of the OU.

- Phone Number: Phone number of the OU (or some other text, we don't use this field).
- Address: Address of the OU.
- External OU: When checked, the OU is external.
- Dependent: This attribute is used in the right system.
- @enterprise installed: Specifies whether the OU has **@enterprise** installed.

  External OUs and OUs where @enterprise is not installed can not be used in process instances.
- Organization Class: The organization class the OU belongs to.
- Active: see chapter 2.2.1.
- Follow–OU: It is possible that some organizational units are replaced by other organizational units due to some reorganization of your company. Through this field it is possible to adhere by which OU the current OU has been replaced during the reorganization.
- Order Attribute: Here a free text can be entered. At the implementation of own application this text can be used for sorting organizational units independent of the available attributes.

Here you can also use the functions *Apply changes later* and *Activate*.

4.4.2 Tab: Super Organizational Units

It is possible to add an organizational unit to several organizational hierarchies. Therefore a organizational unit can have more than one super (parent) OU, namely one per organizational hierarchic.

4.4.3 Tab: Roles

Here you can view the role assignments in the OU.

4.5 Organization Hierarchy

After installation the system contains one hierarchy with name *default*. The *default* application uses this hierarchy.

The object-details of organization hierarchies contain the following tabs:

- General
- Organization Hierarchy
- History
- Access
- Referenced By

4.5.1 Tab: General

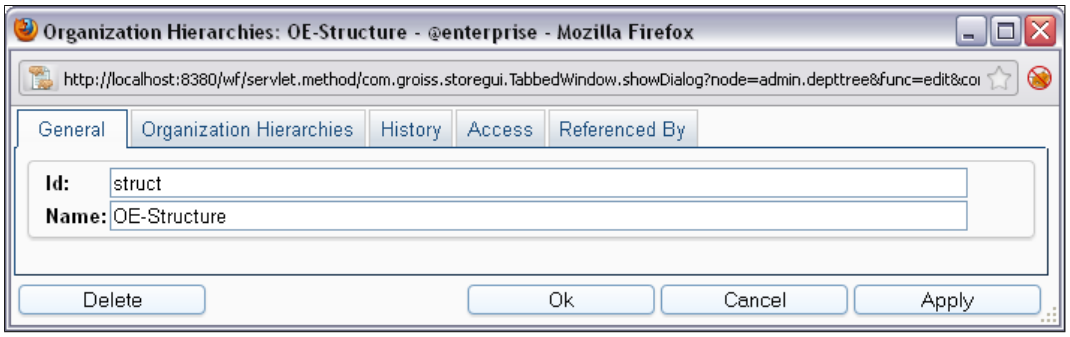


Figure 4.5: **Object-Details: Organizational Hierarchies**

You can edit the following attributes (required fields are bold):

- **Id:** Unique identifier of the OU.
- **Name:** Name of the OU.

4.5.2 Tab: Organization Hierarchy

In this tab you can see the buildup of the hierarchy. Every organizational unit attached to the tree is displayed with its name.

Navigation Through The Hierarchy

The buildup of the hierarchy uses three symbols:

- *blue circle:* the actual organizational unit has no further sub-units.
- *plus-sign:* If a plus is displayed in front of a organizational unit in the current hierarchy this means that it has at least one subordinate organizational unit. If you click onto the plus the organization hierarchy becomes expanded at this position and all subordinate OUs of the next level are displayed. Furthermore the plus is converted into a minus.

- *minus-sign:* If a hierarchy is already expanded you can collapse it by clicking on the minus in front of the corresponding hierarchy. By doing so the minus becomes converted into a plus.

Functions

Functions of the hierarchy are displayed in the toolbar or in the popup-window, which opens after activating an entry of the hierarchy.

- **Expand Hierarchy:** opens the whole hierarchy under this node.

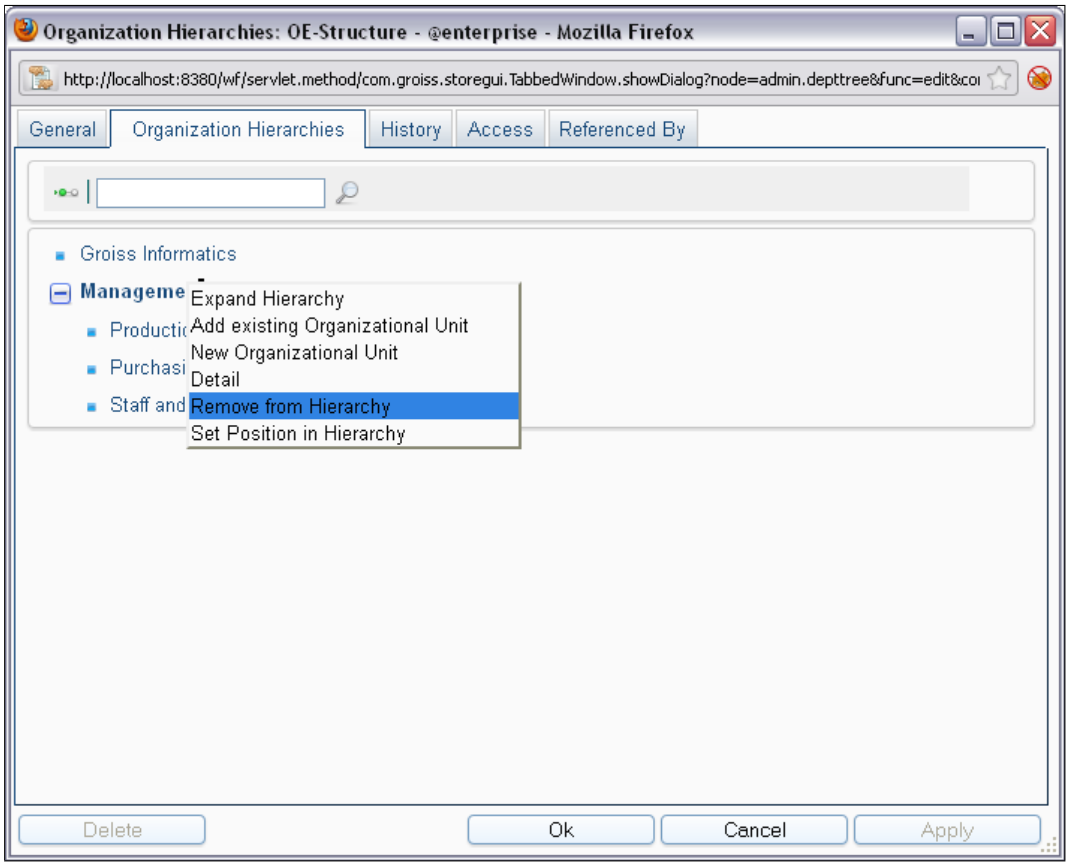


Figure 4.6: **Tab: Organizational Hierarchies**

- **Add existing Organizational Unit:** Select an OU and put it in the hierarchy as child of this OU.
- **New Organizational Unit:** Create a new OU and put it in the hierarchy as child of this OU.
- **Detail:** Edit the attributes of the OU or delete the OU.
- **Remove from Hierarchy:** removes the OU from the hierarchy.
- **Set Position in Hierarchy:** You can edit the position of this OU in the hierarchy.

The first function in the toolbar is **Add To Hierarchy**. Activating this function opens a dialog. Here it is possible to search for a organizational unit and to click the button "Apply". By doing so you add the selected OU at the top level of the hierarchy.

If you have been searching for a certain OU with the help of the function **Search** in the toolbar, those OUs which have been found in the hierarchy are displayed in bold font. In addition there are "next"– and "previous"– links (displayed by >> and <<) which make it possible to navigate through the found OUs.

4.5.3 Function *Merge organizational hierarchies*

With the toolbar function *Merge organizational hierarchies* it is possible to add an organizational hierarchy to an second one. This could be possible, if there are two **@enterprise** installations where the organizational hierarchies should be merged. In the first installation (A) the tree is managed and should be submitted to second installation (B) via XML export. Installation B contains also additional OUs with hierarchies which should be kept.
In installation B an own organizational hierarchy is created for the available OUs and relations. After synchronizing the organizational hierarchy with installation A (via XML import) the relations of installation B will be merged into the organizational hierarchy with this merge function..

Example:

Organizational hierarchy default (on A):
Dept. A
  Dept. A1
  Dept. A2

A1 and A2 are Sub-OUs of A.

In installation B the tree should look like this:
Dept. A
  Dept. A1
  Dept. A2
    Dept. X
Dept. Y
  Dept. Z

That means that X is under A2 and Z is under Y in an own tree.

For the usage of function "Merge organizational hierarchies" the private structure of B must be defined in an own organizational hierarchy:

Dept. A2
  Dept. X
Dept. Y
  Dept. Z

A merge of this structure in the default organizational hierarchy results in the desired structure.

4.6 Organization Class

Organization classes are used to classify the organizational units. This information is not used from **@enterprise** , but can be useful when modeling the structure of big organizations.

The object-details of organization classes contain the following tabs:

- General
- History
- Access
- Referenced By

4.6.1 Tab: General

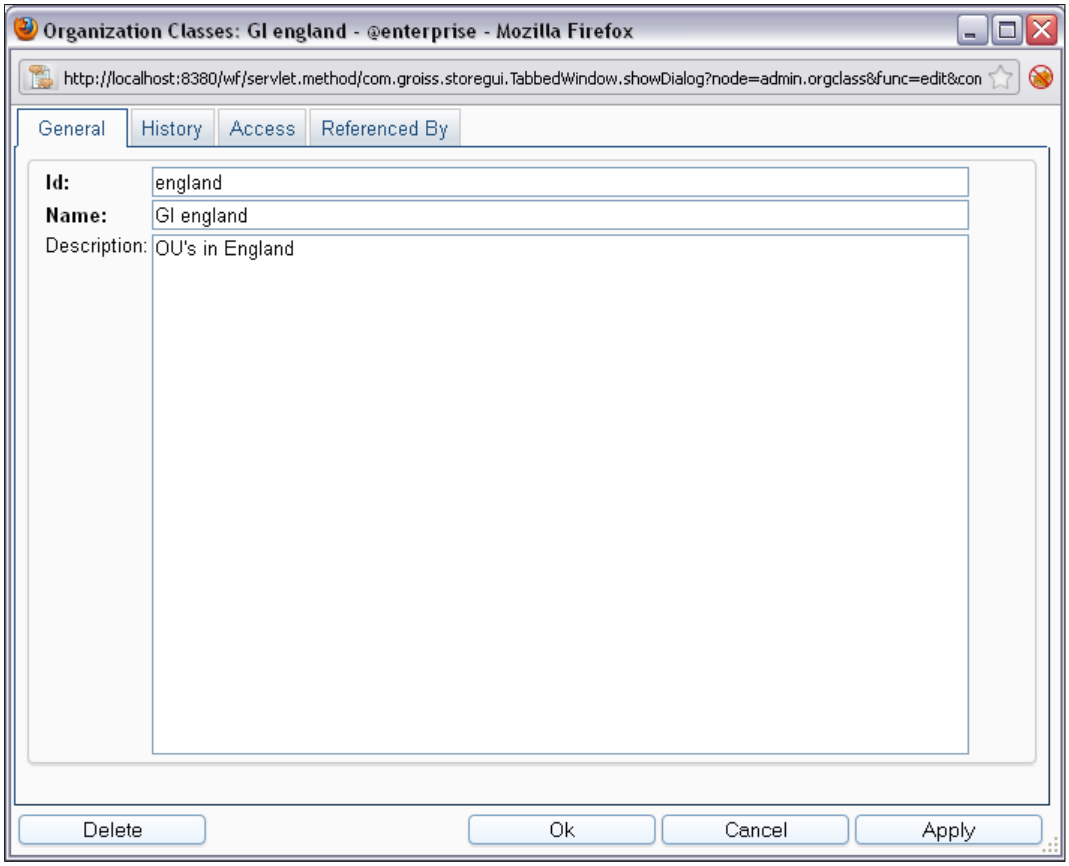


Figure 4.7: **Object-Details: Organizational Classes**

You can edit the following attributes (required fields are bold):

- **Id:** Unique identifier of the organization class.
- **Name:** Name of organization class.
- Description: Free text.

4.7 Keywords

By clicking this link a HTML–page will be opened, where you can administrate a list of keywords. The entered keywords can be assigned to individual documents in the DMS (see User Manual). These keywords can be used in the document-search.

4.8 Server

This meta data object is still offered for the reason of downward compatibility to prior versions of **@enterprise**. There it has been relevant for the distribution mechanism.
Since version 6.1 **@enterprise** has a new so called *cluster architecture* and therefore the distribution mechanism is not used any longer. However, one server object is still required. It represents the current installation of an **@enterprise**–server. If this server is deleted accidentally it has to be inserted again with the attributes which are defined in the configuration file of **@enterprise**.

Hint: The settings about HTTP-Port and RMI-Port must be set in *Configuration* of @enterprise.

5 The @enterprise Right System

5.1 Introduction

The right system of **@enterprise** allows a very flexible assignment of rights to users.
The central data structure of the right system is the *permission*. A permission describes who has which right on which object.
Permissions can be grouped to so-called *permission-lists*. They can be used to group permissions together and use them for several objects.
Standard-permissions are used to assign permissions to new objects. You define standard-permissions for an object class. If a new instance of the class is created (an object) the defined standard-permission is assigned to this object.
In the following section we describe these concepts in detail.

5.1.1 Rights

For the administration of rights see section 4.2.

5.1.2 Object Classes

Object classes define the classes which can be used in the right system. For each object class you can define two things:

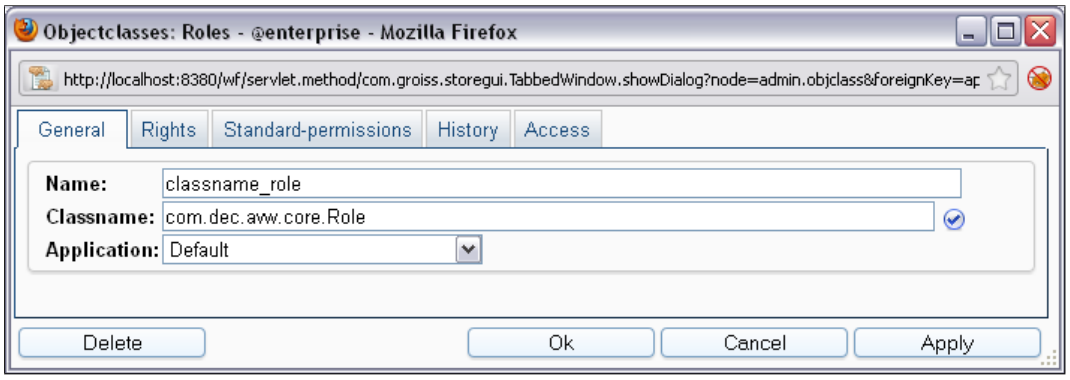


Figure 5.1: **Update Object Class**

- The rights applicable for the object class. For example the right 'execute' is useful for functions but not for persons. The rights specified here can then be selected when defining permissions.

  An additional mode can be selected: either "create", "edit", "view", or "execute".

  If one of these modes is selected together with a right *r*, the right *r* is used for the corresponding operation (create, edit, view, execute) instead of the original right.

- The standard-permissions: You can select a permission-list as standard permission for the object class. Moreover, if you select an organizational unit you can define a standard permission specific to an organizational unit. This would then be used, when a new object is created in the context of an OU, for example a document belonging to an OU.

5.1.3 Permissions

A permission describes WHO has which RIGHT on which TARGET. Therefore, it contains the following information:

- Who: a user, a role or a role together with an organizational unit.
- Right: a right
- Target: the object, on which the right will be applied or the object class, when the permission is for all objects of this class.
- The scope of the permission:
  1. The permission is for all objects, no target is specified.
  2. The permission is for all objects of a class, the target class is specified.
  3. The permission is for one object, which is specified as target.
  4. The permission is for all objects belonging to an organizational unit. As target the OU is specified.
  5. The permission is for all objects belonging to the organizational unit, where the agent has the role specified under "Who".
- The scopes 4 and 5 of the previous list can be refined with an OU-Scope. The set of OUs where the permission is valid can be modified or extended in the following ways:
  1. local: The permission is given for the specified organizational unit.
  2. hierarchic: The permission is given for the specified OU and all sub-OUs.
  3. dependent hierarchic: The permission is given for the specified OU and all dependent sub-OUs.
  4. independent: The permission is given for the next upper independent OU.
  5. independent and dependent-hierarchic: The permission is given for the next upper independent OU and all dependent sub-OUs.

6. super-OU: The permission is given for the next upper OU.

- Yes/No: The permission is given or not given.

To understand the different OU-scopes see the following example. Fig. 5.2 shows an organizational hierarchy with independent and dependent OUs. The grey circles represent the dependent OUs.

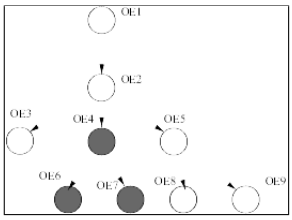


Figure 5.2: **Organizational Hierarchy with independent and dependent OUs**

A permission for the organizational unit OE2 comprises the following units in the different scopes:

- local: OE2
- hierarchic: OE2, OE3, OE4, OE5, OE6, OE7, OE8, OE9
- dependent hierarchic: OE2, OE4, OE6, OE7
- independent: OE2
- independent and dependent-hierarchic: OE2, OE4, OE6, OE7
- super-OU: OE1

5.1.4 Permission-List

Permission-lists are aggregations of permissions. They can be attached to several objects to define identical access rights to this objects. For each object one permission list can be defined.
The permissions relevant to an object are therefore the permissions where the target is the object plus the permission where the target is a permission list and this permission list is used for this object.

5.2 *Definition of Permissions*

In the **@enterprise** system administration permissions can be defined from two sides: The permissions of an agent (user or role) can be defined in the respective detail masks. The permissions applied to an object can be edited from the detail mask of the object ("Access" button). The permission-lists can be administrated from the link in the navigation frame of the administration main window. The standard-permissions can be edited via links in the tables of the object classes.

5.2.1 Permissions of Users

In the table of users there is a link to the permissions of the selected user. If you click on the link a window opens with a list of the permissions of the user. Note that you only see the permissions directly assigned to a user, the permissions assigned to the user via the role assignments can be edited in the role administration.
You can insert, edit and delete table entries in the usual manner.

5.2.2 Permissions of Roles

The permissions of roles are edited in the same way as the permissions of users.

5.2.3 Administration of Permission-Lists

Click on the link "Permission-List" in the navigation frame. You can create permission-lists when clicking at the add button, insert the name in the "General" tab and administrate the permissions in the second tab.
The permissions for the list can be created by clicking the link in the table line of a permission-list. But to assign a permission to a permission-list you must have right *edit-acl* for that list. See the next section for the usage of permission-lists.

5.2.4 Permissions for an Object

Objects underlying the right system have the tab "Access" in the detail mask. If you click the button, a window opens where you can see two frames:

- In the first frame you can edit the permissions for the object.
- In the second frame the permission-list of the object can be viewed and changed.

In the mask for the permission you can select the agent (user, role, organization) who has the permission, the right and the organization scope.

5.2.5 Permissions for Permissions

To edit permissions which refer not to a specific object a agent must have the right *edit-acl* for all objects.
If a permission refers to an object, the agent must have the right *edit-acl* for the object or the object class.
Additionally, the agent needs the right *execute* for the right which is used in the permission. This allows to restrict the permission of assigning rights to specific rights.

5.2.6 Permissions for Role-Assignments

The manipulation of role-assignments is also a special case, because a user can change his permissions by adding roles. Therefore, for changing role-assignments two rights are necessary: First, the right to edit the user, second the *execute* right for the role.

5.2.7 Administration of Object Classes

Object classes are used to define the usability of rights to object classes. Only when a right is assigned to an object class, the right is usable for objects of this class.
Furthermore, the standard-permissions (see above) of object classes are defined here.
Informations of the Object Class detail window (required fields are bold):

- **Name:** The name of the object class. Detailed information to ids and names can be found under 3.
- **Class:** The Java–class implementing the object class.
- **To Application:** Application, the object class belongs to.

For object class objects the functions available under 2.2 are available. Furthermore it is possible to define rights and standard permissions respectively for object classes.

5.3 Standard Settings

The **@enterprise** standard rights are listed in section 4.2.
The role *sys* has the rights *edit*, *execute*, *edit-acl*, *create*, and *admin* for all objects. The user *sysadm* has the role *sys*. Additionally, *sysadm* has the right *conf*.

All changes of master data can be performed from users with the *sys* role.
For changing the configuration, viewing the logfile, shut down the server, and some similar tasks the *conf* right is necessary.

5.4 For what you need which rights?

The tables 5.1 and 5.2 will give you an overview for what you need which rights:

5.5 Example

This section contains an example for using the right-system.
Problem: The user John Smith should get the permission to administrate users of the organizational unit "Service". He should be allowed to edit the user attributes and the role-assignments.
For editing users he receives the right *edit* for objects of the organizational unit "Service". The admin right is needed to go to the administration.
For editing the role-assignments, we define the role edit-roles: With this role every role-assignment except for the role sys can be edited.

User A wants ...	**Necessary Right (Id)**	**Apply ...**
to create a new object	create	on all objects or the objectclass
to edit or delete an object	edit	on the object, the objectclass, all objects or the OU (if the object is assigned to an OE). *True for all objects apart from OUs.*
to edit or delete an OU	dept_edit	on all objects, the objectclass *Organizational Units* or a defined OU
configurate	conf	on all objects
to work in the administration	admin	on all objects
to view the log-file	admin	on all objects
to enter, edit or delete a permission	edit_acl AND execute	on the object, the objectclass or all objects (edit_acl); on the right (execute)
to enter, edit or delete a role assignment	edit AND execute	on user (edit); on the role (execute)
to execute a function	execute	on all objects, the objectclass *Application* or a defined application
to abort a process	proc_inst	on all objects or all OUs or the OU, the process is started in
to change the agent in the history	proc_inst OR set_agent	on all objects or all OUs or the OU, the process is started in
to view process history, list of documents and notes and all process forms and versions	view_procinst	on one process or all
to archive processes	conf	on all objects
to search for process instances	view AND proc_inst	on all objects or all OUs or the OU, the process is started in
to create statistics	stat	on all objects
to execute stored queries	execute	on all objects, the objectclass *Stored Queries* or a defined query
to delete master data	delete	on all objects, the objectclass
delete a standalone form	delete	on the form class
see changes of forms in process instance pi	proc_inst OR set_agent	on the OU/process definition, where pi is running
see changes of forms in process instance pi	view OR user A is agent of the stepinstance	on the OU only, where pi is running

Table 5.1: For what you need which right?

5.6 Permissions and Substitutions

The behavior of the rights system in context for substitutions is worth considering. The implementation follows the following two basic rules:

1. If a user takes a substitution he should not loose rights.
2. After taking a substitution the user should not have more permissions than both users together.

User A wants ...	**Necessary right (Id)**	**Apply ...**
to create an object	create AND edit	on the objectclass and edit on the folder
to edit an object, edit the metadata or replace a document	edit	on the object
to delete an object, delete a folder with content	edit	on the object, the folder and if the object is a folder on the whole content
to view an object	view	on the object
to move an object	edit	on the source-folder and the destination-folder
to copy an object	edit AND view	on the destination-folder (edit); on the object or the if the object is a folder on the whole content (view)
to rename an object	edit	on the object
to change the permissions on the object (access)	edit_acl	on the object
to create a version	edit	on the object
to view a version	view	on the object
to delete a version	edit	on the object
to view the properties	no right necessary	
to delete a form	delete AND edit	on form class (*delete*), on the folder which contains the form (*edit*)
to delete a subform	delete AND edit	on form class (*delete*), on the parent/main form (*edit*)

Table 5.2: For what you need which right in the DMS?

Right	Target Object	Scope	Apply in organizational unit	Positive
Personal Permissions				
Edit Objects	Organizational Units : Service	Objects of OU	local	
Administration		all Objects		
Permissions through role **Rollenzuordnungen bearbeiten in OE Service in Organizational Unit Service**				
Execute Objects	Roles	Object class		
Execute Objects	Roles : sys	Object		

Figure 5.3: **Example: Permissions**

The evaluation algorithm for permissions works as follows:

- Step 1: Evaluate the set of permissions without consideration of substitutions.
- Step 2: For all substituted users: Compute the set of all positive permissions (not "denies") for the substituted user in the substituted roles.

  Subtract all negative permissions of this user, regardless whether the right belongs to a substituted role or not. Add the resulting set to the result.

6 Workflow Modelling

In the following chapter we describe the object classes necessary to define processes. In principle, the definition of a process is the answer to the following question:
WHO does WHAT WHEN with WHAT?

- WHO: Who is responsible for the processing of a workflow? The agents must be defined for every single activity in a workflow. It is usually defined using roles.
- WHAT: What is done in the workflow? The work is decomposed in activities, which are done by one agent. The description of the tasks answer the WHAT question.
- WHEN: If you know which activities have to be done and who performs these activities, the order of execution must be defined. Often it is a simple sequence but can have a complex structure containing loops, branches, and parallelism.
- WITH WHAT: For performing the activities some informations are necessary. It must be defined, which activity needs which information and what new information is produced in an activity.

  We use forms to structure the information and describe the information exchange between the activities.

The definition of workflows contains the following objects:

- **Applications:** Applications group processes belonging together.
- **Tasks** elementary activities in processes.
- **Functions** are representations of interactive Java-methods used for execution of activities.
- **Forms** contain the local data of a process.
- **Processes** describe the structure of a business process.
- **Interfaces** allow the start of process instances by submitting an HTML form.

6.1 Applications

Applications group processes belonging together. All workflow elements belong to an application.

The object-details of application contain the following tabs:

- General
- History
- Access
- Properties

6.1.1 Tab: General

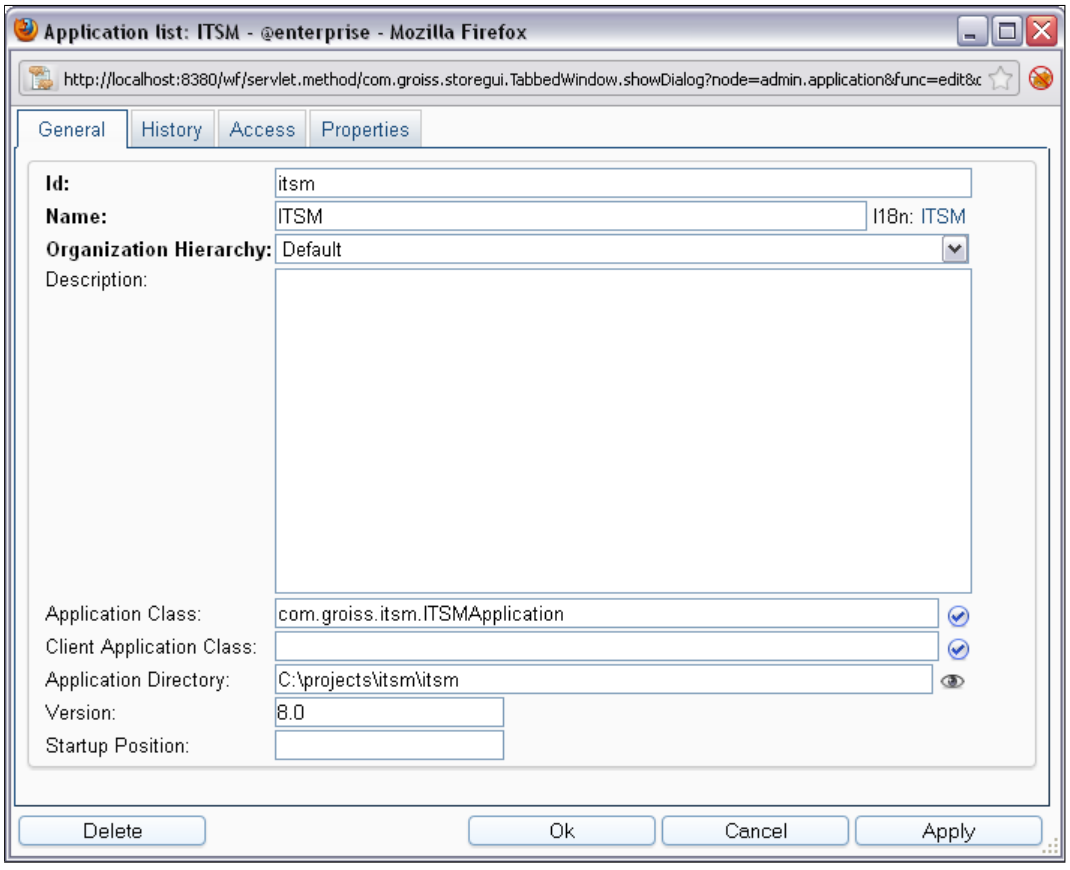


Figure 6.1: **Object-Details: Applications**

You can edit the following attributes (required fields are bold):

- **Id:** Unique identifier of the application.

- **Name:** Name of the application. By activating the I18n-link beside this field, the translations (if defined in tab *Properties*) of this key are displayed and can be edited directly by changing the values and activating the button *Save*. The changes are stored in the resource file of this application (see section 6.9).
- **Organization Hierarchy:** The hierarchy used for resolving hierarchic roles and rights.
- Description: Free text.
- Application Class: A class, which implements the interface com.groiss.wf.ApplicationAdapter can be specified. See the API documentation and the Programming Guide for details.
- Client Application Class: Analogous to the Application class, for usage on the Java Client.
- Application Directory: where the application is installed. Activate the icon *View Configuration* to display the content of the appropriate configuration file.
- Version: Version of the application. It could be helpful in case of an upgrade to differ applications of an older version of @enterprise.
- Startup Position: Applications are loaded due to this position in ascending order. It could be necessary, if application A2 has references in application A1 and application A1 has to be loaded before A2.
- Button *Upgrade*: This button is visible only, if a newer version of the application has been found on the file system. By activating this button all defined upgrade-actions will be performed. For further information about upgrading applications, please take a look in the API of @enterprise (*ApplicationAdapter.getVersion()* and *ApplicationAdapter.upgrade()*).

6.1.2 Tab: Properties

In this tab it is possible to define properties for this application. You can edit following attributes:

- Resource Strings: Enter a path to a resource-bundle (*.xls- and/or *.properties-files) which is used by this application, e.g. *com.groiss.itsm.resource.Strings*. If no resource-bundle exists on file-system, the resource editor is able to create a new one (see chapter 6.9). Further information about resource-bundles are available in *Application Development Guide* - chapter *Internationalization of Applications*.
- Application-Parameter: Here you can define parameters, which are used by this application. These parameters are stored in a XML-file (*properties.xml*) and are accessible by the *Configuration* of @enterprise (see section 10).
- User-Parameter: In this area you can define parameters for users, who use this application. These parameters are also stored in a XML-file (*properties.xml*) and are accessible by the *Settings* of each user (see *User Manual* - section *The group Extras*).

This tab does not provide the creation of all HTML elements (e.g. textfields, radio-buttons, etc.). For this purpose you have to create or adapt the file *properties.xml* of the corresponding application (see *Application Development Guide*).

Hint: The functions of this tab are available only, if an *Application Directory* has been specified.

6.2 Tasks

Tasks are the elementary activities in processes. The can appear in different processes of the same application and on different positions in one process.

The object-details of tasks contain the following tabs:

- General
- Escalation
- Functions
- History
- Access
- Referenced By

6.2.1 Tab: General

You can edit the following attributes (required fields are bold):

- **Id:** Unique identifier of the task.
- **Name:** The name of the task. This name is shown in the task column in the worklist. By activating the I18n-link beside this field, the translations (if defined in application mask - tab *Properties*) of this key are displayed and can be edited directly by changing the values and activating the button *Save*. The changes are stored in the resource file of this application (see section 6.9).
- **Version:** Version number of the task, a positive integer.
- **Application:** Application, the task belongs to.
- Description: Free text, visible in the worklist via the link to the task details. It can contain a short help text or instructions to the task.
- Preprocessing: Enter the name of a Java-method. The method is called, before the task is put into the worklist of a user. Furthermore it is possible to enter a GROOVY script in this field - enter *groovy:* <groovy-script> in this field. See the Programming Guide for details of such methods.

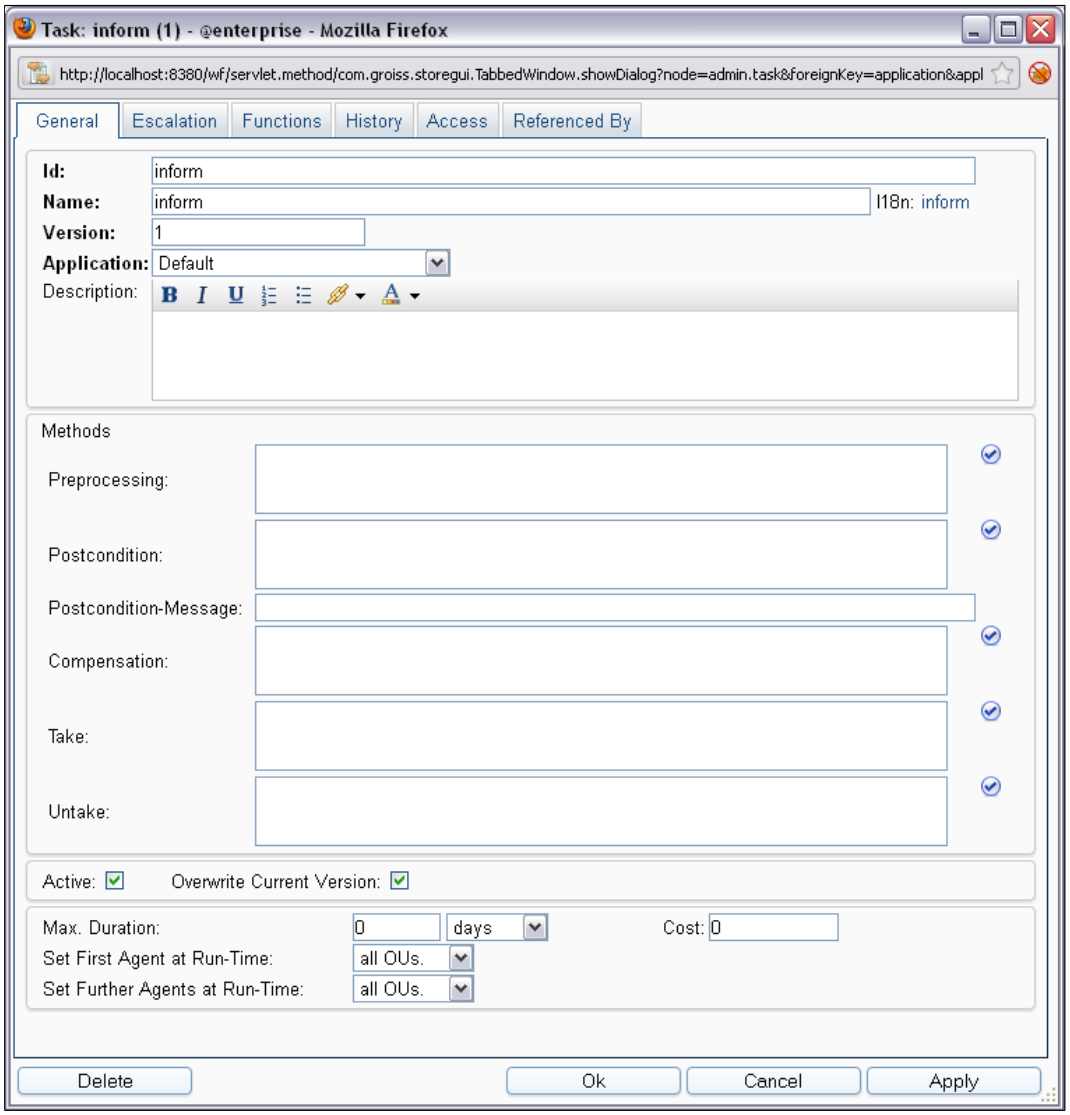


Figure 6.2: **Object-Details: Tasks**

- Postcondition: The condition which is entered here is checked at run-time when a user finishes the task (sends it to the next agent). This condition could be also a GROOVY script or XPath condition. If the condition is not true, finishing the task is not possible. The syntax of such conditions is described in section 7.1.5. When using Xpath conditions the prefix *xpath:* is needed analog to Groovy.

- Postcondition-Message: In this field you can insert the text of an error message, which will be shown when the postcondition evaluates to false.

- Compensation: This method or GROOVY script is executed when the activity is passed when going back to an earlier step in the process. It can be used to reestablish a consistent state.

- Take: This method or GROOVY script is called when the task is taken (from the role-worklist to the personal worklist).
- Untake: This method or GROOVY script is called when the task is given back to the role worklist.
- Max. Duration: The maximum duration of the task (in days, hours, or minutes).
- Cost: The costs of the task. This field is not used from **@enterprise** , but can be used in some statistics.
- Set First Agent at Run-Time: specifies, whether the agent can be set at run-time.
- Set Further Agents at Run-Time: specifies, whether further agents can be specified at run-time.

  Hint: The last two attributes have the value ranges "none" "within Dept.", "all Depts.". That means, no agents can be set at run-time, only agents belonging to the same organizational unit can be set, or no restrictions apply.
- Active: Indicates, whether the task is active. Further information about (de)activation of objects can be found under 2.2.1.
- Overwrite Current Version: On each update of a task, a new version is generated. This can be suppressed, when the checkbox "Overwrite Current Version" is checked.

Hint: If you delete one task the assigned escalations and functions are deleted also.

6.2.2 Tab: Escalations

With the help of escalations it is possible to react on timeouts during the execution of tasks. It is possible to define four different types of actions which determine what should happen in the case of a timeout. The system timer "Escalations" of **@enterprise** is responsible for checking the timeouts. If this timer is not running the system does not check if timeouts occur or not! Please notice that every escalation object will be executed one time only!

This tab shows all already defined escalation-objects. You can edit them or add new one by activating the functions in the toolbar.

You can edit the following attributes (required fields are bold):

- **Escalation Type:** Here you can select between 3 escalation types:
  1. Activity due time: This escalation type is set as default and will be triggered in the *Worklist*, *Role-Worklist*, *Suspension List* or *Role Suspension List*. For each task the *Maximum Duration* in the tab *Common* can be set. This escalation type will be fired, if this value was transcended.

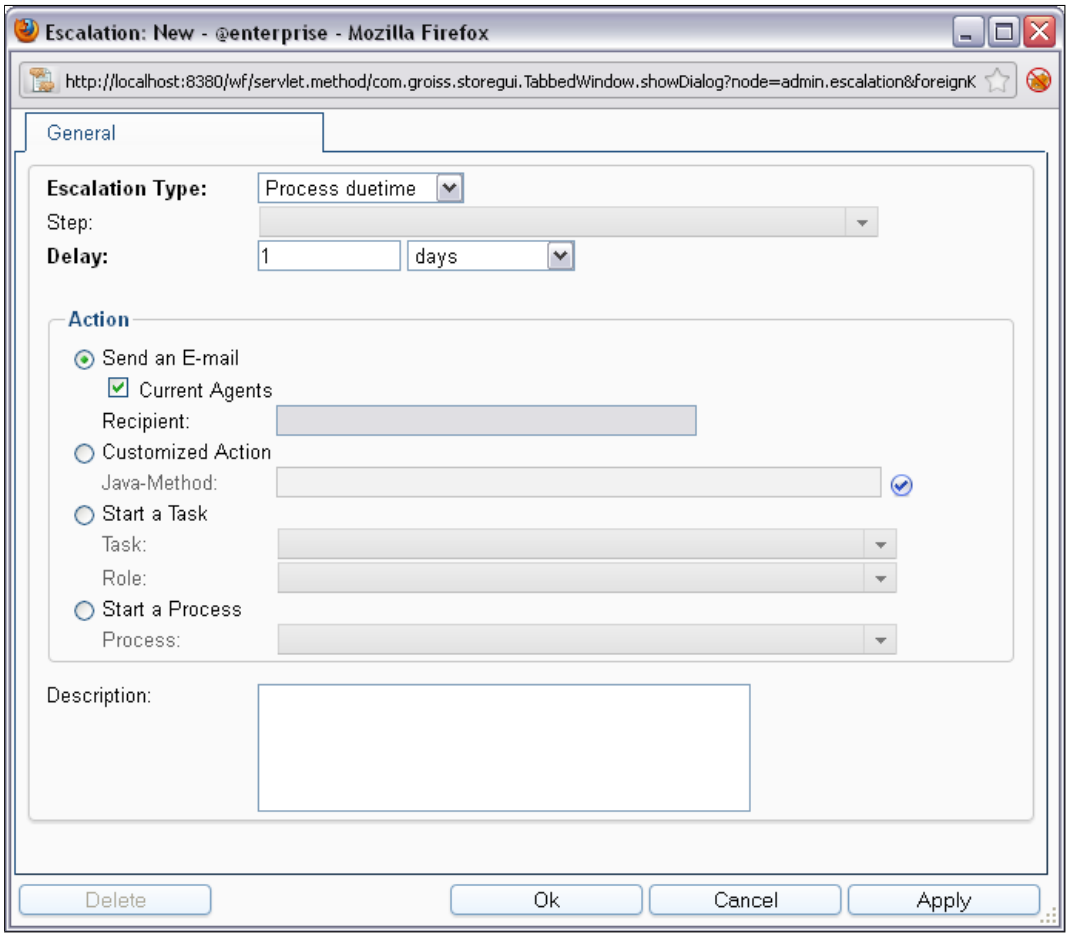


Figure 6.3: **Object-Details: Escalations**

2. Activity idle time: This kind of escalation will be triggered, when the task remains for a while (*Delay*) in the Role–Worklist. This type works in the Role–Worklist **only**. It is possible to enter a negative delay, but it makes no sense in this case!
3. Activity unseen: This escalation type will be triggered, when the current task is unseen in worklist or role-worklist.

- **Step:** Steps are not selectable in task escalations, only in process escalations.
- **Delay:** The period of time going by after the timer "Escalations" has noticed a time-out (in hours or days). This value can be negative to react early enough on a deadline. You can select between *hours*, *days* and *Working days*. Non-working days are Saturday and Sunday. It is also possible to specify additional non-working days under *Configuration* → *Calendar* (see chapter 10).
  Example:
  If 2 working days (48h) are entered, today is Thursday at 4pm and the process has the due date at the following Monday at 4pm, the escalation must be triggered (Assump-

tion: Only Saturday and Sunday are non-working days).

- **Action:** In **@enterprise** four kinds of actions are distinguished:
  - Send an Email: An e–mail is send to the recipient entered in the field "Recipient". If the option *Current Agents* is selected, an e–mail to the agents of the current task will be sent (if a valid e–mail address is entered on user detail-mask). If the current agent of the task is a role, all users which have the role (in this organizational unit) will be informed per e–mail.

    In order to function properly a valid mail server has to be entered into the field "SMTP Host" in the section "Communication" of the server configuration (see Installation Guide of **@enterprise**). As sender of the mail appears either the default value "enterprise@hostname" whereas the host name is the host name of the **@enterprise**–server. If you don't want to use the default sender enter the desired sender into the field "Mail sender" which can be found beneath the field "SMTP Host".
    Example: enterprise@lima.groiss.com
  - Customized Action: A Java–method which will be started at timeout. The package name has to be specified too. See the example in the Application Development Guide.
    Example: com.groiss.DemoClass.demoMethod
  - Start a Task: Starts the selected task. If you entered a value into the field "Role" the task will be started for this role. If this field remains empty the task is started for the user who caused the timeout. The new task belongs to the same process as the task which timeout caused the escalation action.
  - Start a Process: Starts the selected process as subprocess of the current process.
- Description: Free text.

6.2.3 Tab: Functions

In this tab you can define relations of the task to functions (see Fig. 6.4). General informations about functions can be found in section 6.3.

You can add all functions of the list **Available Functions** to the task. The functions of the list **Selected Functions** are already assigned to the task. To add a function select a function of the list *Available Functions* and activate the button **>**. To remove a function select a function of the list *Selected Functions* and click the button **<**.

Your changes are saved after activating the button **OK**, **Apply** or when changing the tab.

6.2.4 Supplement of forms

Forms are typically editable by the current users of a process step corresponding to the form visibilities. The most simplest corrections (e.g. setting another value in a read-only field) needs to go-back to the appropriate agent/step.

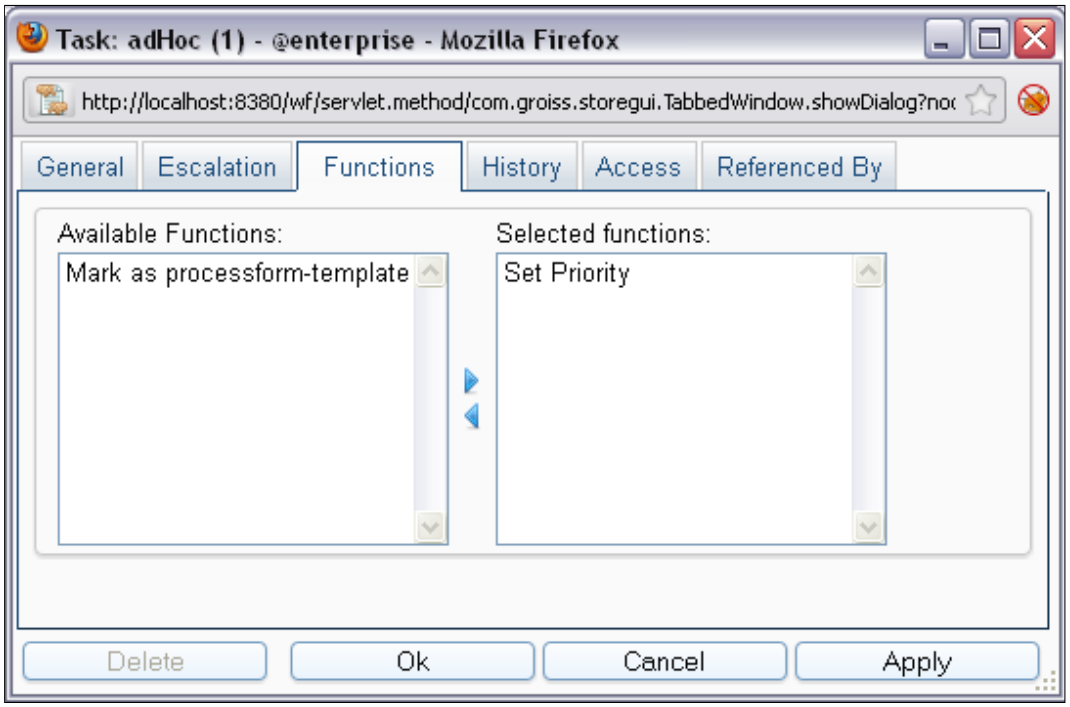


Figure 6.4: **Object-Details: Functions**

With the aid of the predefined *Supplement-Task* the handling can be simplified. This task will be assigned to the process definition *Process editor: Process → Tasks*. Now forms can be assigned to this task and also form-field visibilities can be defined.
Users with rigth *proc_inst* or *set_agent* are able to edit forms via the process history.
The process history contains an icon *Supplement* which allows to start a supplement task for the current process instance. This task will displayed in the worklist of the current user who is able to change the form and finish the task. The changes are displayed in the process history.
If the user ist not the current agent of the process instance (but contains one of the rights above), then the creation of supplement task (and also finishing) can be triggered by changing and saving the form directly via the process history.

6.3 Functions

Task–Functions (or Functions) are representations of interactive Java-methods used for execution of activities. Links to the functions appear in the worklist when working on a task.
The object-details of task-functions contain the following tabs:

- General
- History
- Access
- Referenced By

6.3.1 Tab: General

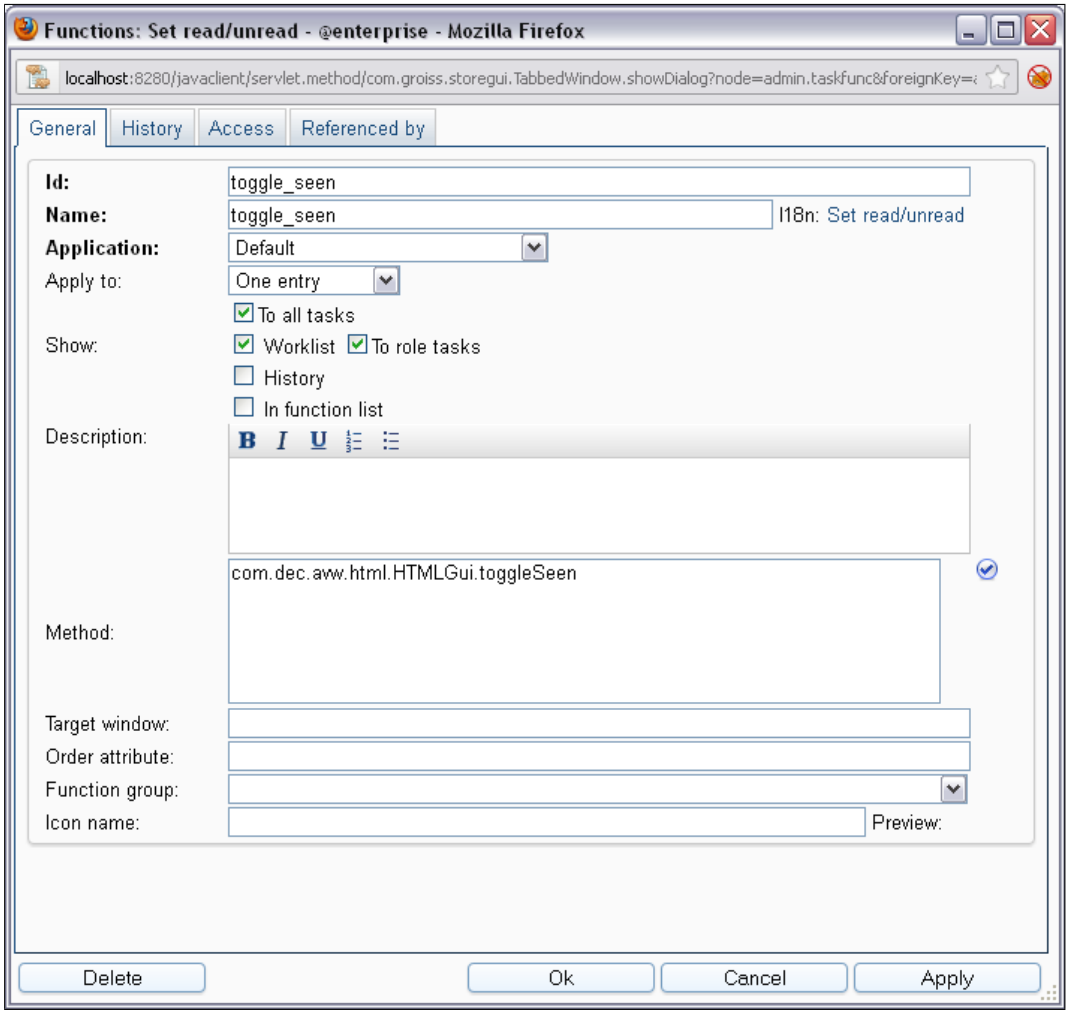


Figure 6.5: **Object-Details: Task-Functions**

You can edit the following attributes (required fields are bold):

- **Id:** Unique identifier of the function.
- **Name:** Name of the function. By activating the I18n-link beside this field, the translations (if defined in application mask - tab *Properties*) of this key are displayed and can be edited directly by changing the values and activating the button *Save*. The changes are stored in the resource file of this application (see section 6.9).
- **Application:** Application, the function belongs to.
- Apply to: Here you can select the type, if the function is a global or local function.
  - no item: The function is a global one, i.e. no entry must be selected (e.g. worklist entry)

  - one item: The function is a local one, i.e. only one entry must be selected
  - multiple items: Analog to *one item*, but more than one entry can be selected

- Show:
  - To all tasks: The function is automatically assigned to all tasks of the application.
  - Worklist: The function appears in the function menu of (role-)worklist.
  - To Role Tasks: The function is applicable also in the process form of role-worklist.
  - History: The function appears (corresponding to its type) in the process history.
  - To function list: The function can not be assigned to a task, for example an administration or search function. This task function appears in the main frame, when the link *Functions* will be activated in the navigation tree.
- Description: Free text.
- **Method:** The signature of the Java–method implementing the function. Parameters can be added by adding a ? and the parameterlist.
  Example: com.groiss.DemoClass.demoMethod?param1=val1&param2=val2

  Furthermore it is possible to enter a GROOVY script in this field - enter *groovy:* <groovy-script> in this field.
- Target-Window: The content of this field contains the name of the window or the frame where the output of the function will be placed. If the field is empty, the output is sent to the frame where the worklist resides. If you enter another name, the output is sent into a separate window with this name. In addition to this name you can add several parameters like width, height etc. by adding a semicolon and write the parameters like the javascript method *window.open* syntax. If you specify the target "_top" the output will be shown in the current browser window.
- Order Attribute: Enter an attribute as order attribute.
- Function Group: Select a self defined function group here.
- **Icon Name:** Define a icon for the function. Enter a absolute path name or a relative path name in the classpath.

Hint: Note, that a user must have the right "execute", to execute a function.

6.3.2 Standard Functions

@enterprise contains the following predefined functions:

- **toggle_seen – Set Read/unread:** Mark an worklist item as read/unread.
  Apply to: One entry, **Show:** assign to all Tasks, Worklist, to Role Tasks

- **from_clipboard – Insert from Clipboard:** Add the content of the clipboard to the process documents.

  Apply to: One entry, **Show:** assign to all Tasks, Worklist

- **into_clipboard – Into Clipboard:** Copy the process instance into the clipboard.

  Apply to: One entry, **Show:** assign to all Tasks, Worklist

- **make_copy – Copy to ...:** Send a copy of the worklist entry to another user in read-only mode.

  Apply to: One entry, **Show:** assign to all Tasks, Worklist

- **attach_note – Process Note:** Add a private or public note to the process instance.

  Apply to: One entry, **Show:** assign to all Tasks, Worklist

- **note_global – Process Note:** Same as note_all, but applicable when the task is not in the worklist

  Apply to: No entry, **Show:** Worklist, in Function List

- **set_duedate – Set Duedate:** Set the due-date of the process or the current activity.

  Apply to: One entry, **Show:** Worklist

- **addRelation – Add Relation:** Add a relation between two processes.

  Apply to: One entry, **Show:** Worklist

- **setPriority – Set Priority:** Set the priority of a process instance.

  Apply to: One entry, **Show:** assign to all Tasks, Worklist

  Further informations belonging to this function can be found in the programming guide of **@enterprise**.

Further information to this functions can be found in the **@enterprise** Users Guide.

6.4 Forms

Forms contain the local data of a process. In the user interface they are represented as HTML forms. Besides the functions described in chapter 2.1.2 the following functions can be found in the toolbar:

- Create new formtype
- Edit formtype
- Replace HTML
- View
- Create View

If the form classes of a form cannot be loaded, they will be shown as inactive table entries. The process for creating and editing a form is shown in figure 6.6.

The object-details of forms contain the following tabs:

- General
- Java-Class
- Database-Table
- Rights
- Standard Permissions
- History
- Access
- Preview
- Folder Settings
- Referenced By

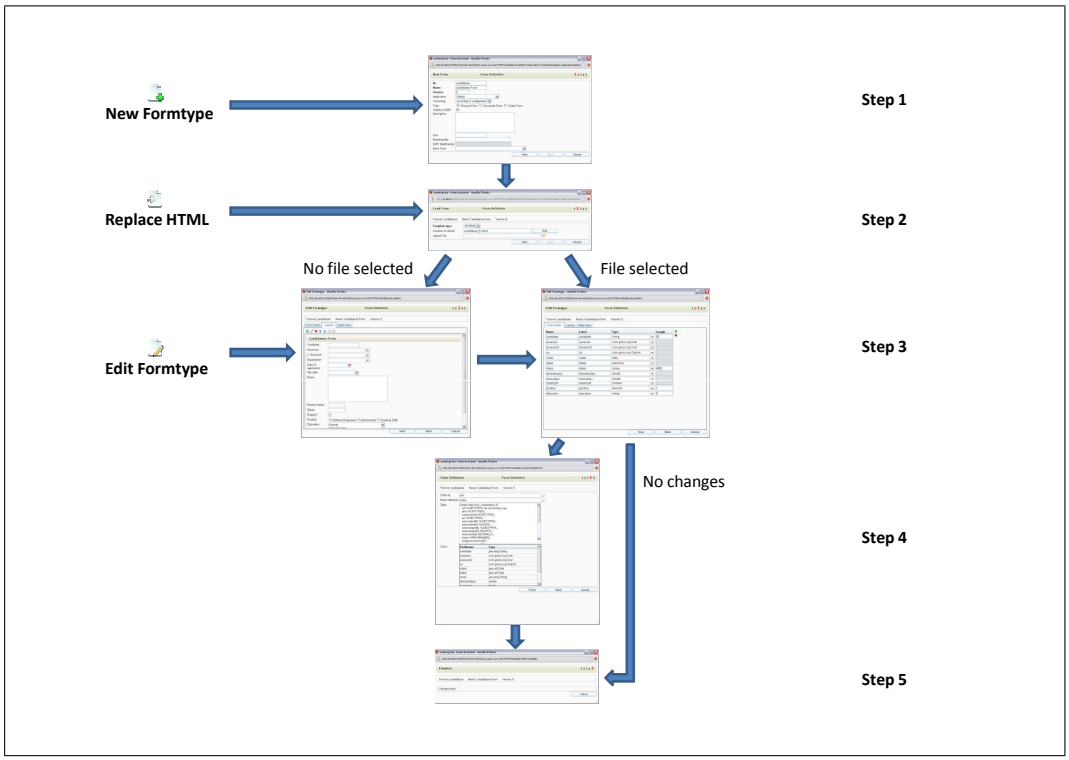


Figure 6.6: **Process for creating and editing a form**

6.4.1 Create new formtype

By clicking this function the form-wizard for creating and editing formtypes opens.

Step 1

In the first step of the wizard you define the main attributes of the new form. The meaning of the fields is described in chapter 6.4.6. If you want to create a view onto an existing form, select a form in the select list *Base Form*. Go to the next step by activating the button **Next**.

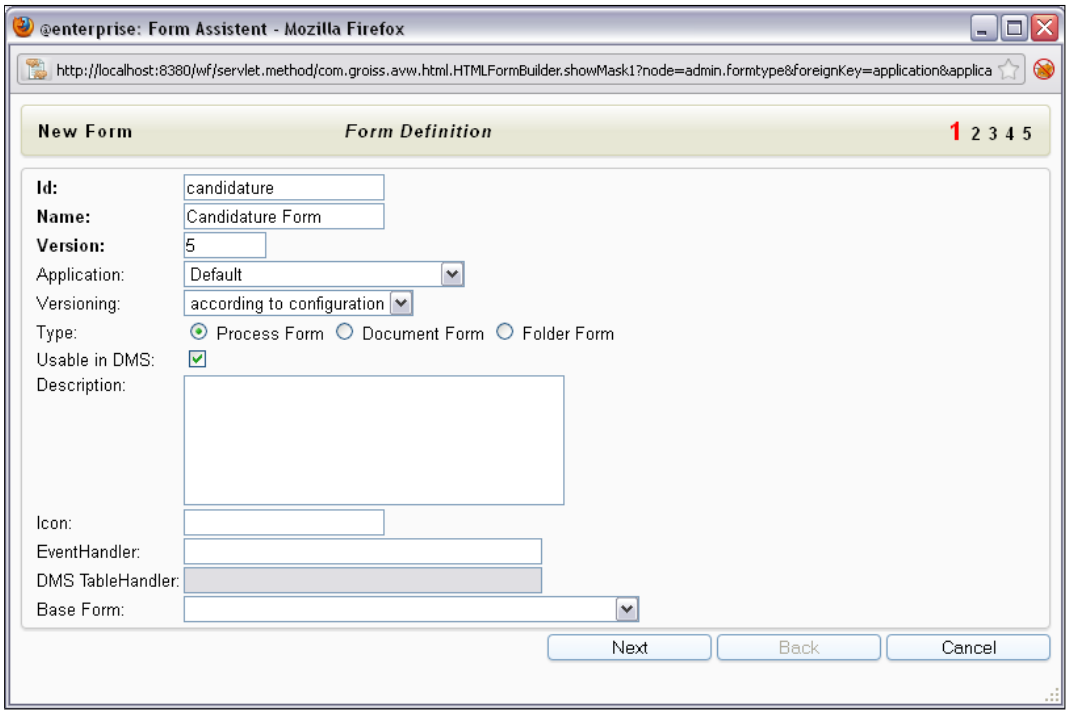


Figure 6.7: **Form-Wizard: Step 1**

Step 2

In this step you can choose the html-form which is the base of your form.
You can use HTML-forms , XHTML-forms or XFORMS. XHTML is a reformulation of HTML in XML, therefore XHTML files must be in correct XML syntax. If you do not want to use a file for creating a form, select the *Template Type* XFORMS and click on the button *Next* and the form-wizard will be opened (see section 6.4.1). If a path is entered in step 2, the template file on file-system can be adapted with button *Edit*, if activated (see Installation- and Configuration Guide), i.e. the HTML-code can be adapted.
If you supply an HTML file, the file is parsed and the result is stored in the database. The source file is no longer needed.
If using an XHTML-file, the file must be in the classpath of the server, for example in the classes directory. The file will be parsed too, to continue the form definition, but the source

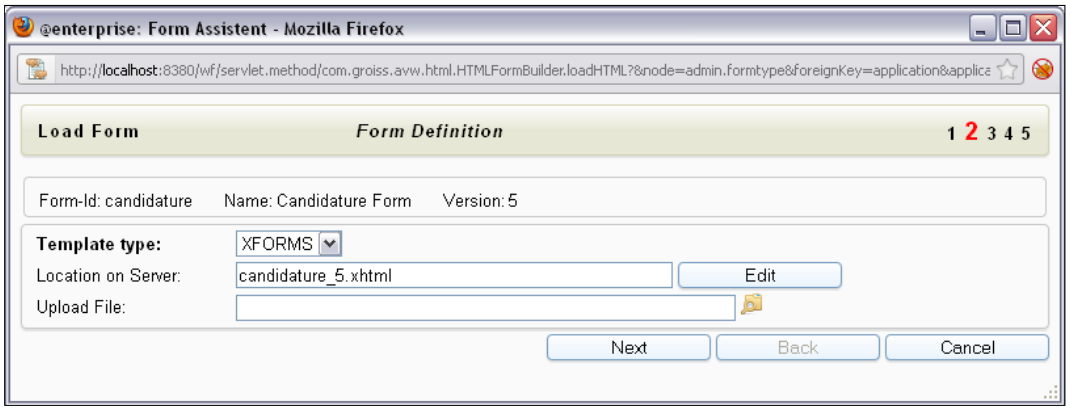


Figure 6.8: **Form-Wizard: Step 2**

file is used at run-time to show the form. The advantage of XHTML forms is, that changes in the source file immediately take effect when showing a form.

Step 3

If a file has been entered into the field *File*, the file gets parsed and is loaded onto the server. The result is shown in this step. A form field is defined by a *Name*, *Label*, *Type* and *Length*. If you have not supplied a file, the form-wizard will be available in tab *Layout* at this step (see section 6.4.1).

Information about the tab *Form fields* of step 3:

- Name: The name for the field in the database.
- Label: Specifies the string used as header for this column, when the contents are shown in a table. If you do not want to show this column (field) in the table of a superordinate form, keep this header empty.
- Type: The database type for the field. Note that the type information is used for creating a Java class and a database table. The default value for this field is read from the HTML–file. There it can be defined with the help of the attribute "dbtype" of the "input"–tag of the form field. Table 6.4.1 shows to which Java type the entered dbtype will be converted at the creation time of the Java class for the form. The restrictions of the database, for example length of varchar fields, have to be considered. The datatype char is used for strings with fixed length, If you store the String 'sw' in a field with type char(3) it will contain 'sw ' (a trailing space), the comparison with 'sw' will fail.
- Length: The length of the field in the database.

After defining all form fields you can change to the next step by clicking the button **Next**.

Hint: XForms allow to define fields with *binary* DB types. This field should be filled via API only.

dbtype	DB	***Java–Type***
VARCHAR	VARCHAR	String
CHAR	CHAR	String
LONG	LONG (Oracle < 8.1) CLOB (Oracle > 8.1) Text (SQL–Server)	String
DECIMAL	DECIMAL	long[a] double[b]
DATE	DATE (Oracle) DATETIME (SQL–Server)	date[c]
DATETIME	DATETIME	date[d]
NONE[e]		
Class Name[f]	DECIMAL(20) — oid[g] VARCHAR(200) — Class Name[h]	Class Name

[a] If an integer has been entered for the attribute *Length* (e.g. 3).

[b] If a real number has been entered for the attribute *Length* (e.g. 3,4).

[c] Only the date is shown.

[d] The date and the time are shown.

[e] The field does neither appear in the Java class nor in the database. It is used to store extensions in forms.

[f] This class has to implement the interface *com.groiss.store.Persistent* or it has to be a subclass of a class which implements this interface.

[g] The oid is kept in the form. The result of the method "toString()" of the corresponding object is shown.

[h] optional, if a class has been specified, which implements an interface, an abstract class or the interface *com.groiss.store.HasSubclasses*.

Table 6.1: **dbtype of a HTML–form–field and its representation in the database and in the corresponding Java class respectively**

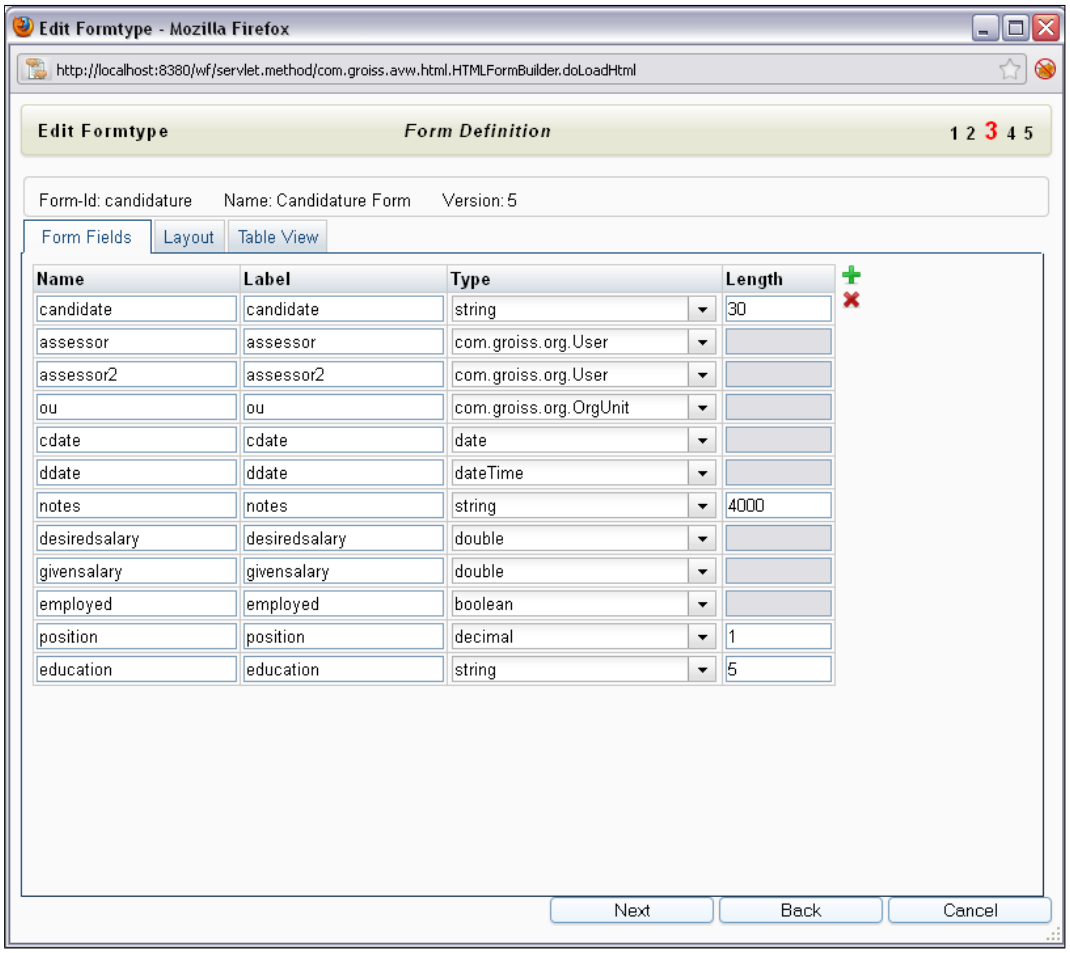


Figure 6.9: **Form-Wizard: Step 3**

Form-Wizard: Tab *Layout*

The form-wizard is very comfortable for creating and editing forms (see Fig. 6.10). The form-wizard is available only, if the *Template Type* XFORMS is selected in Step 2.

Hint: Forms created in **@enterprise** 7.0 are XHTML-forms. If you want to adapt them with form-wizard of **@enterprise** 8.0, you have to convert it to *Template Type* XFORMS by selecting the form and activating the toolbar-function *Edit Formtype*. If you do not convert the form, you will not be able to edit it with form-wizard.

In the menu bar under the header (Form–Id, Name, Version) the standard functions are provided for processing the formfields. Furthermore you can change the order of a field by clicking on the arrow-buttons. An other mentionable menu point is Properties, where you can assign titles to the form. The toolbar-function *Create new layout* generates a standard-layout with the existing form fields.

For the creation of forms the form-wizard offers following elements:

- **Text:** Simple text without an input option.
- **Line:** Horizontal Line.
- **Form Fields:** Contains all form fields, which will be described in the following.
- **Table:** It is possible to embedding subforms by entering a *Classname* and an *Id*. If you have already created a subform, you can select it by clicking on the symbol beside *Classname*. You can label the table by entering a text in the inputfield *Label*.

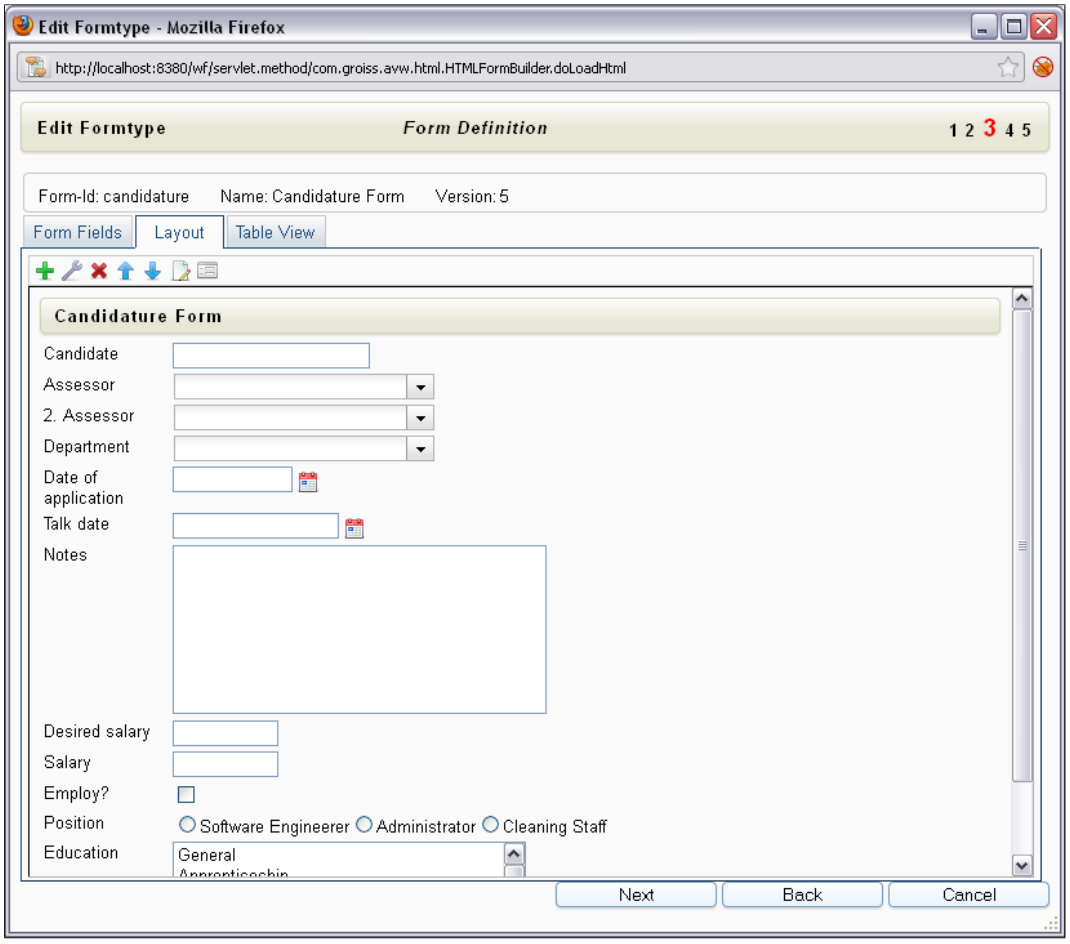


Figure 6.10: **Form-Wizard: Step 3**

The *Form Fields* consist of several elements, whereas the same following properties are available in all form fields:

- **Label:** Free selectable identifier. By activating the I18n-link beside this field, the translations of this key are displayed and can be edited directly by changing the values and activating the button *Save*. The changes are stored in the resource file of this application (see section 6.9). The link is visible only when the checkbox *Localize* is activated and a resource has been entered in application mask - tab *Properties*.

- **Localize:** Activate this checkbox to localize the field label.
- **CSS-Class:** CSS-class for style sheets.
- **Database Field:** Unique identifier, which identifies the field - has been created in this step (see 6.4.1)

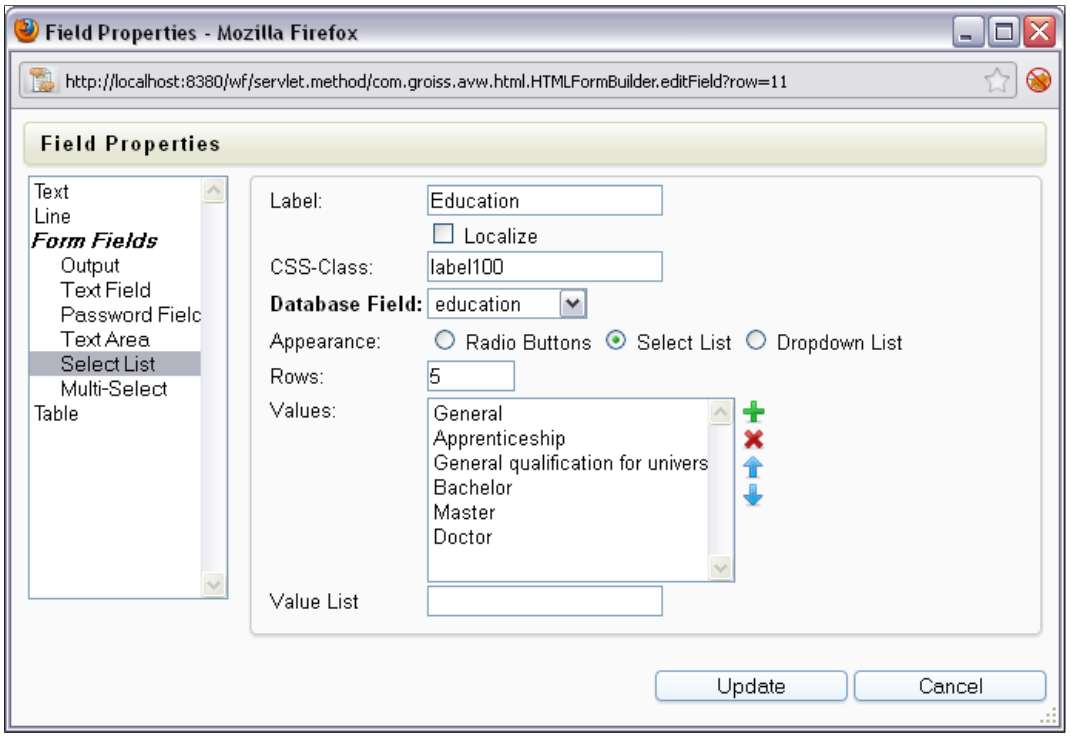


Figure 6.11: **Form-Wizard: Field Properties**

Each form field has further specific properties, which will be explained in the following:

- **Output:** Only output-field, where no data can be entered.
- **Text Field** and **Password Field**: Input-field, where data can be entered. The field *Columns* is for the size of the textfield.
- **Text Area:** The fields *Columns* and *Rows* are for the height and width of the text area.
- **Select List:** You can select between *Radio-Buttons*, *Dropdown-List* and *Select-List*. You have to add several values to the list *Values*, which are shown as selectable options, or you enter the ID of a *Valuelist*. A valuelist can be created in the DMS (form *Valuelist*, which exists of an ID and the values. If no integer value is entered in the field *rows*, a dropdown list will be shown as default, otherwise a select list.
- **Multi-Select:** Analog to *Select List*, but you can choose between *Checkboxes* and *Select-List* only.

Information about the tab *Table View* of step 3:

This tab allows to set the columns, which are visible in subtable, if the form is embedded as subform. The list *Available Columns* contains all available formfields. By selecting an entry and activating the *Add*-button, the field will be added to the list *Table Columns*. All fields of this list are shown as subtable-columns in the (superior) form. By activating the *Sort*-buttons beside the list *Table Columns*, the order of the columns can be changed.

After defining all settings you can change to the next step by clicking the button **Next**.

Step 4

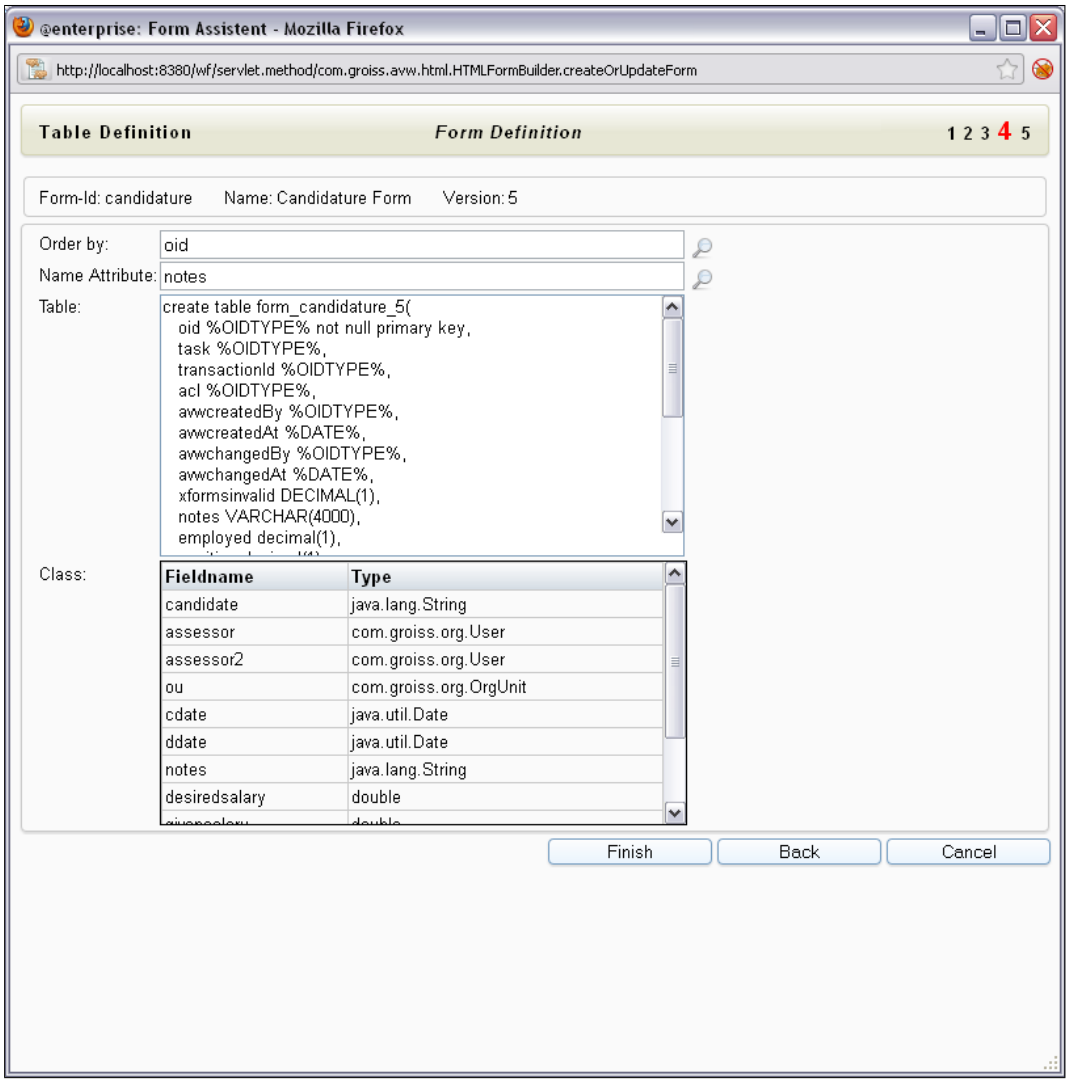


Figure 6.12: **Form-Wizard: Step 4**

This step shows you the following information:

- Ordered by: This attribute is used to define the order of the entries in the table which represents a subform of a form.
- Name attribute: The content of this field is shown as form name, for example in selection windows or in a DMS folder. Furthermore a regular expression can be entered, e.g. {formfield} (display_text), {formfield2}. The curly brackets are necessary to show values of formfields, i.e. the previous example could generate following output: Joe (firstname), Russel. *Jose* and *Russel* are values of the entered formfields, *firstname* is a free defined text (= display_text).
- Table: The database statement for creating the table which is used to store the content of the new form.
- Class: The fields and types of the Java class which represents the new form in **@enterprise**.

By clicking the button **Finish** the table and Java class are created and the new form is available in **@enterprise**. The Java class is **always** stored in *forms* directory of **@enterprise** (e.g. com.dec.avw.appl.myform_1). The form is also stored in *forms* directory, but with an exception: if the form is created in a self-defined application where the application directory is set (see chapter 6.1), the form is stored in the *classes* directory of the application directory.

Step 5

In the last step of the wizard you can see if any problems occurred while creating the form. If no errors occurred the form loading process is finished.

6.4.2 Edit Table

By clicking the function **Edit Table** the form-wizard opens. This is the same wizard as described above, but starts at step 3.

6.4.3 Replace HTML

The function **Replace HTML** in the detail mask of a form allows to change the HTML text of the form. The form can be already in use. You just have to load the new HTML file. It is not possible to change the database types of an existing field by executing this function!

If the new form contains no new fields or fewer field respectively, the form-wizard in step 3 is shown. By clicking the button *Next* the replacement of the HTML–form is finished. The Java class and the database table will not be changed.

If the new form contains some new fields the form-wizard in step 3 is shown. By clicking the button *Finish* the replacement of the HTML–form is finished. The Java class and the database table are also adapted.
If your form points to a XHTML file, you have to use this function only if you want to add fields to the form.

6.4.4 Create View

In **@enterprise** it is possible (via the extended search) to search for form field contents independent of the form versions. Therefore it is necessary to create a database view over all version of the form. This database view contains all form field which exist in all form versions.

By clicking this function a HTML–page is shown which contains the following informations (required fields are bold):

1. **Create View: *formid*:** The SQL–statement which will be used to create the database view. By clicking the button "Create View" the view is created.
2. **Replace Existing View: *formid*:** The SQL–statement which will be used to replace the current database view by a new one. By clicking the button "Replace Existing View" the old view is replaced by the new one.

Hint: Depending upon whether there exists already a database view or not, you can either use the function *Create View* or the function *Replace existing View*.

6.4.5 View

After activating this function the object-details of the selected form opens and the tab *Preview* is active.

6.4.6 Tab: General

You can edit the following attributes (required fields are bold):

- **Id:** Unique identifier of the form.
- **Name**: Name of the form. By activating the I18n-link beside this field, the translations (if defined in application mask - tab *Properties*) of this key are displayed and can be edited directly by changing the values and activating the button *Save*. The changes are stored in the resource file of this application (see section 6.9).
- **Version:** Version number of the form, a positive integer.
- **Application:** The form belongs to this application.
- **Type:** With the help of this attribute you can define in which context the form should be used. There are three different possible usages:
  - Process Form: The form is attached to a process and can be used at corresponding tasks.
  - Document Form: This form is used to describe the meta data of documents within the DMS.
  - Folder Form: This form is used to describe the meta data of folders within the DMS.

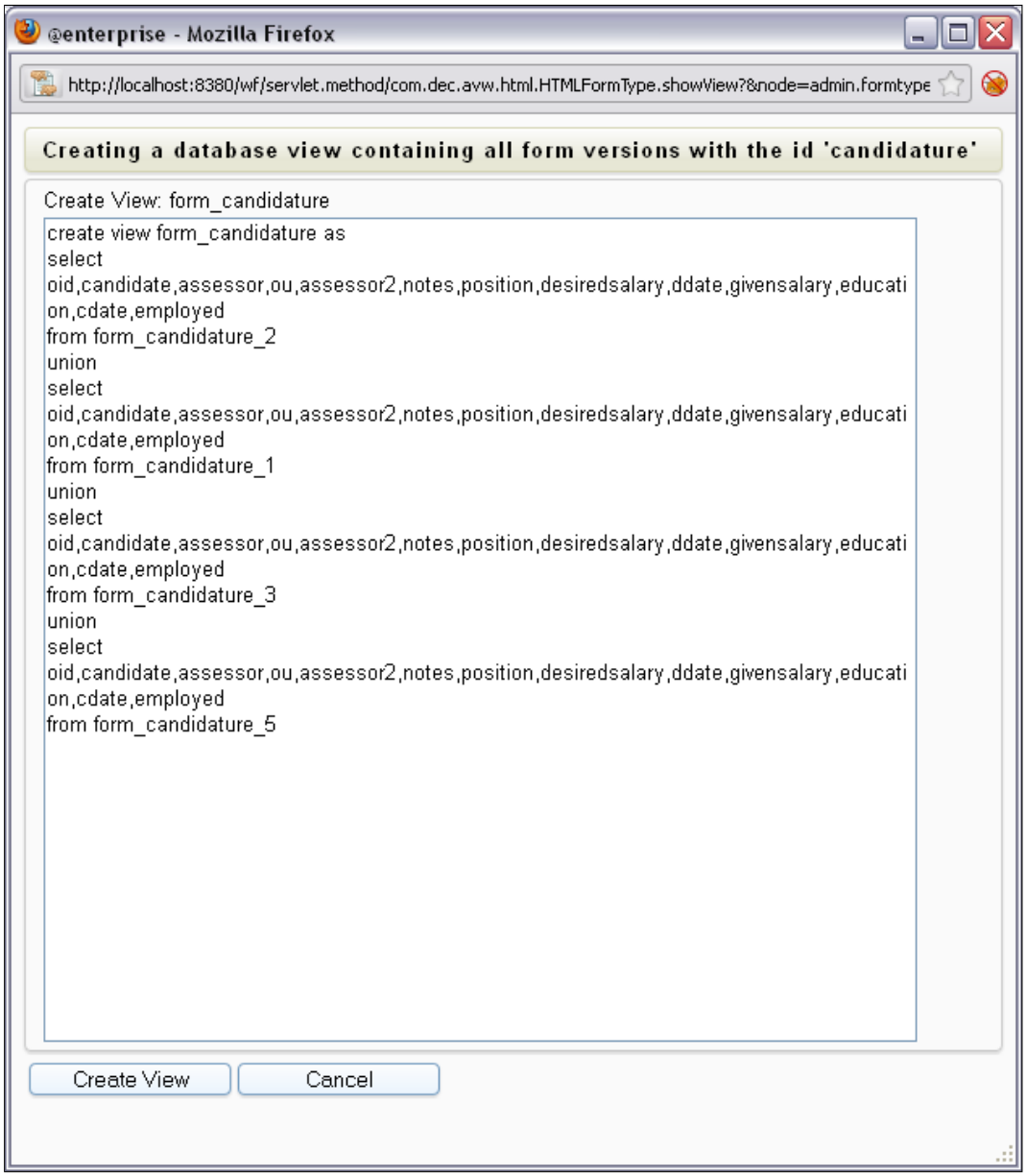


Figure 6.13: **Function: Create View**

- **Mask type:** Here you can define one of the following formats:
  - HTML
  - XHTML
  - XFORMS
- Form Description: Free Text.
- Active: see chapter 2.2.1

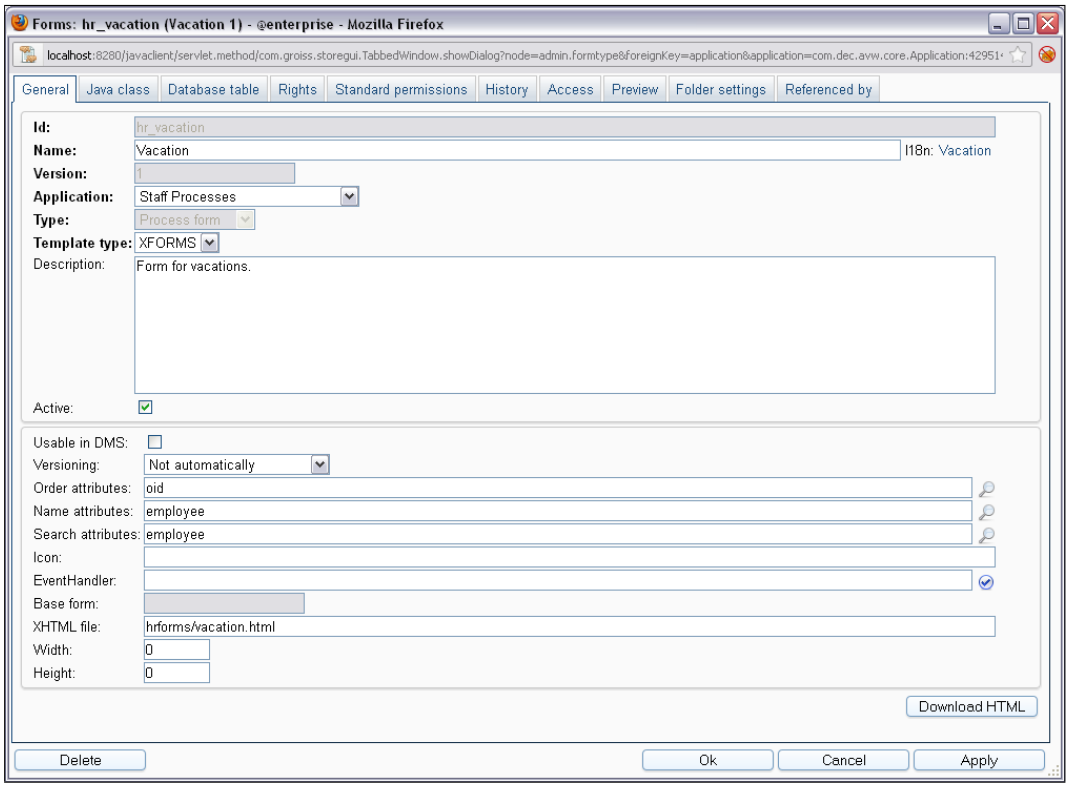


Figure 6.14: **Object-Details: General**

- Usable in DMS: If this checkbox is checked the form can be used within the DMS.
- **Versioning:** Here it is possible to define when a version of the form should be created. This setting is only relevant for forms which can be used within the DMS. Furthermore this setting takes effect only if the form is not used as a process form, because in this case a new version is created automatically every time when the function "finish" is carried out. There are four possibilities to configure the versioning:
  - according to configuration: the versioning happens as defined in the system configuration under section "DMS" (see the installation guide of @**enterprise**).
  - not automatically: in this case the user has to create a version manually with the function "Make Version".
  - on agent change: the versioning happens every time when the agent of the form has changed.
  - on every change: the versioning happens every time when the form has been changed.
- Order attributes: This attribute is used to define the order of the entries in the table which represents a subform of a form.
- Name attributes: The content of this field is shown as form name, for example in selection windows or in a DMS folder. Furthermore a regular expression can be en-

tered, e.g. {formfield} (display_text), {formfield2}. The curly brackets are necessary to show values of formfields, i.e. the previous example could generate following output: Joe (firstname), Russel. *Jose* and *Russel* are values of the entered formfields, *firstname* is a free defined text (= display_text).

- Search attributes: Here you can define the attributes which are used for quick search. It is possible to define the quick search function for each DMS folder (see section 6.4.12). If no search attributes are entered, the name of the DMS object is used by default.
- Icon: The icon representing the form. Specify the path in the class path.
- EventHandler: A Java–class implementing the interface "com.groiss.dms.FormEventHandler" or "com.groiss.dms.XHTMLFormEventHandler. So the application programmer has the possibility to react on several events (onDelete, onInsert, onShow and onUpdate) which are triggered during the manipulation of the form.
- Base Form: The current form can be derived from a base form, whereas this field contains the name of the base form.
- XHTML-File: A reference to the XHTML-Page in the Classpath.
- Width and Height: Specifies the size of the page.

By clicking the button **Download HTML** you can store the HTML–form of the current form to your local file system. There you can edit it and afterwards you can upload it to the system via the function *Replace HTML* (see 6.4.3).

6.4.7 Tab: Java-Class

This tab shows the fields and types of the Java class which represents the new form in **@enterprise** .
The button *Re-generate Java classes* creates/regenerates a new Java class of this form (regenerating is also possible without existing java class) and stores it in the *forms* directory of your **@enterprise** installation.

6.4.8 Tab: Database-Table

This tab shows the database statement which has been used to create the table of the form.

6.4.9 Tab: Rights

see chapter 5

6.4.10 Tab: Standard Permissions

see chapter 5

6.4.11 Tab: Preview

This tab displays the HTML-view of the form.

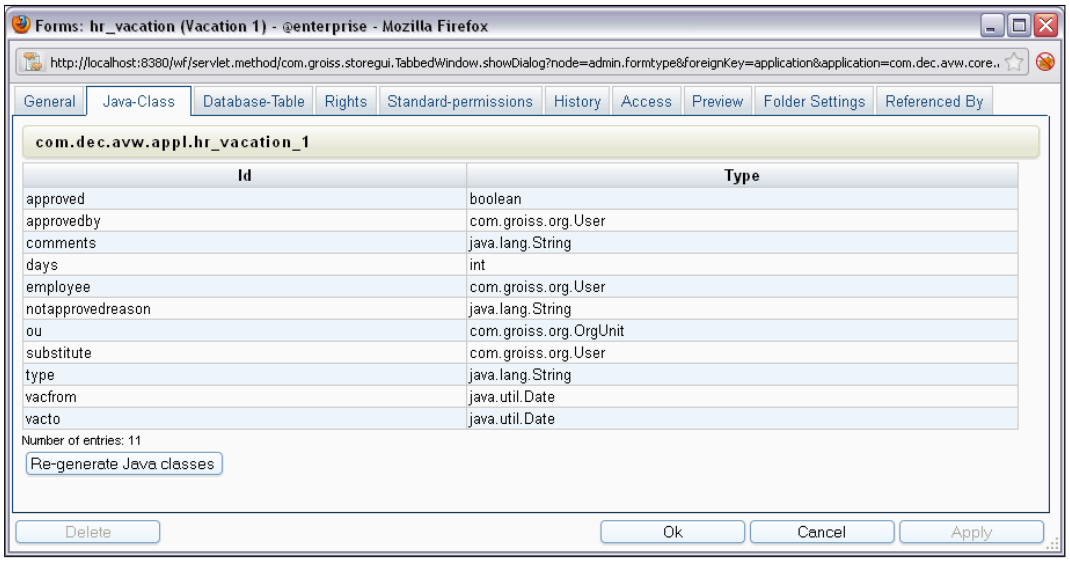


Figure 6.15: **Object-Details: Java-Class**

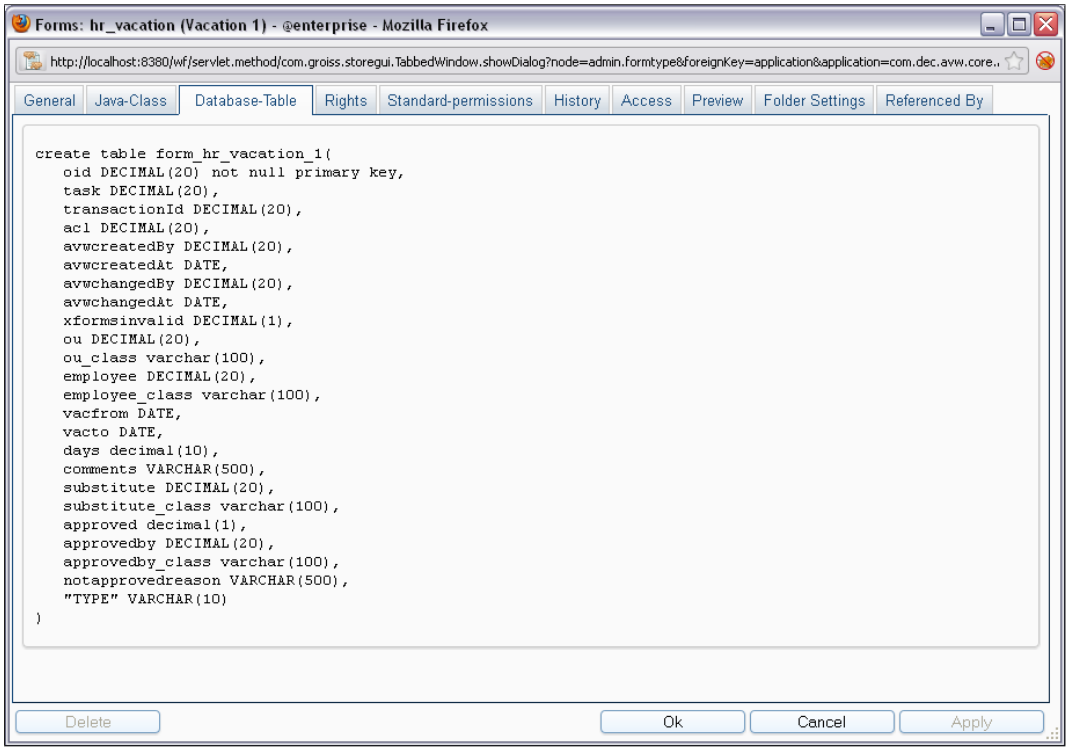


Figure 6.16: **Object-Details: Database-Table**

6.4.12 Tab: Folder Settings

If the form is a folder, you can modify the design, how the folder content is displayed, in this tab (see figure 6.17). It is possible to

- Add columns, edit and delete them and change the order
- Add functions, delete them and change the order
- Add forms, delete them and set their allowance (*allowed* or *denied*)

Further information about can be found in the @enterprise *User manual*. The changes of this page are used for all folder instances of this formtype.

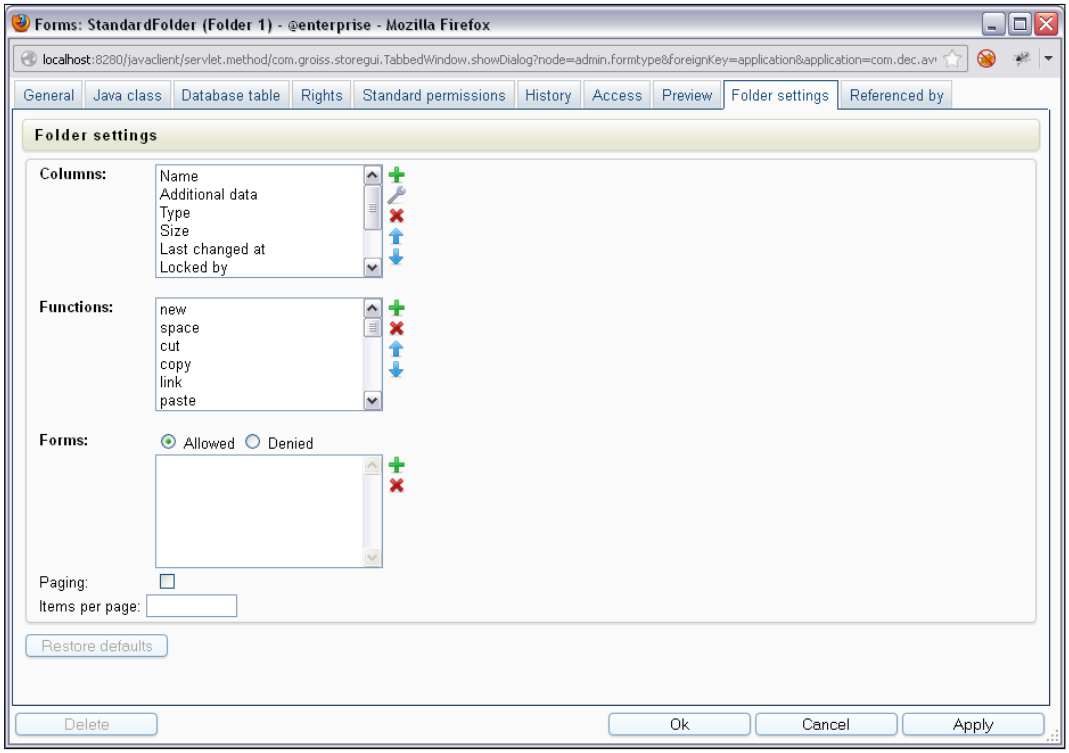


Figure 6.17: **Form-Details: Folder Settings**

6.5 Processes

Processes describe the structure of business processes. Besides the functions described in chapter 2.1.2 the following functions can be found in the toolbar:

- Create new process with the process-editor
- Edit a process with the process-editor
- Load WDL / XWDL

- Process overview

The object-details of application contain the following tabs:

- General
- Source
- Graphical Representation
- Components
- Visibility of Forms
- Escalation
- Functions
- History
- Access
- Folder Settings
- Referenced By

6.5.1 Create new process with the process editor

After activating this function, the process-editor starts. You can create a new process. Detailed information about the process-editor can be found in chapter 7.2.

6.5.2 Edit a process with the process editor

By clicking this function the process editor is started with the selected process as argument. With the help of the process editor you can edit the process graphically.

6.5.3 Load WDL / XWDL

With this function you can load a process from a WDL or XWDL script file. A window is shown where you may enter the following information (see Fig. 6.18):

- First chose, if you want to load a WDL file. Enter (or select) the name of the file containing the process specification.
- The checkboxes *Generate Tasks* and *Generate Roles* allow you to specify whether tasks and/or roles unknown to the system should be generated automatically.
- Click "Compile" to load the script file and save the process in the database.

After the compilation the system informs you whether the operation has been successfully or whether errors have occurred.

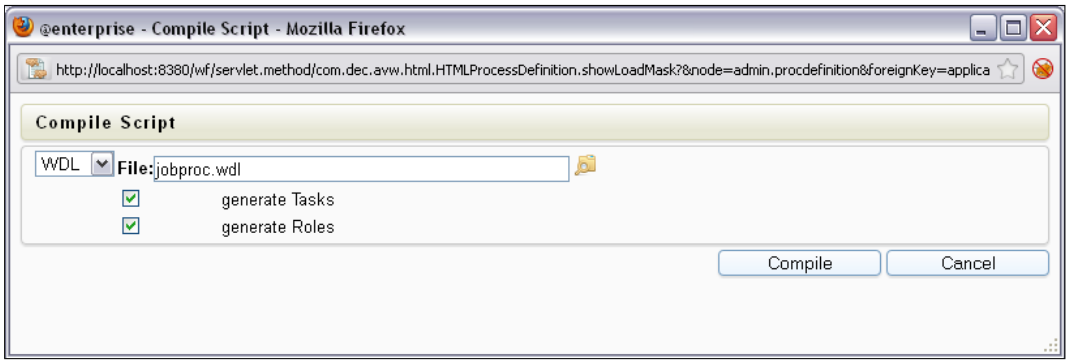


Figure 6.18: **Load Script**

6.5.4 Process overview

This function shows an overview of all used components of the process (see Fig. 6.19). Each component is divided in blocks which can be hidden or displayed. This mask allows to manipulate the referenced components (e.g. tasks, roles, etc.) by activating the corresponding links. Furthermore it is possible to generate a PDF.

6.5.5 Tab: General

You can edit the following attributes (required fields are bold):

- **Id:** Unique identifier of the process.
- **Name:** Name of the process. By activating the I18n-link beside this field, the translations (if defined in application mask - tab *Properties*) of this key are displayed and can be edited directly by changing the values and activating the button *Save*. The changes are stored in the resource file of this application (see section 6.9).
- **Version:** Version number of the process, a positive integer.
- **Application:** The application, where the process is running.
- Subject: A form field, which content is used as subject of the process instance at runtime. @enterprise offers the possibility to enter a pattern (regular expression).
- Instance Id: Here you can define an Instance Id which identifies the started process instance uniquely. It is also possible to enter a pattern (regular expression) in following format:

  { letter* "{" ("n" | "nn" | "ny" | "nny" | "y" | "yy" | "ou") "}" letter* }*

  Explanation:

  - n.....next number
  - nn....next number for this process
  - ny....next number per year

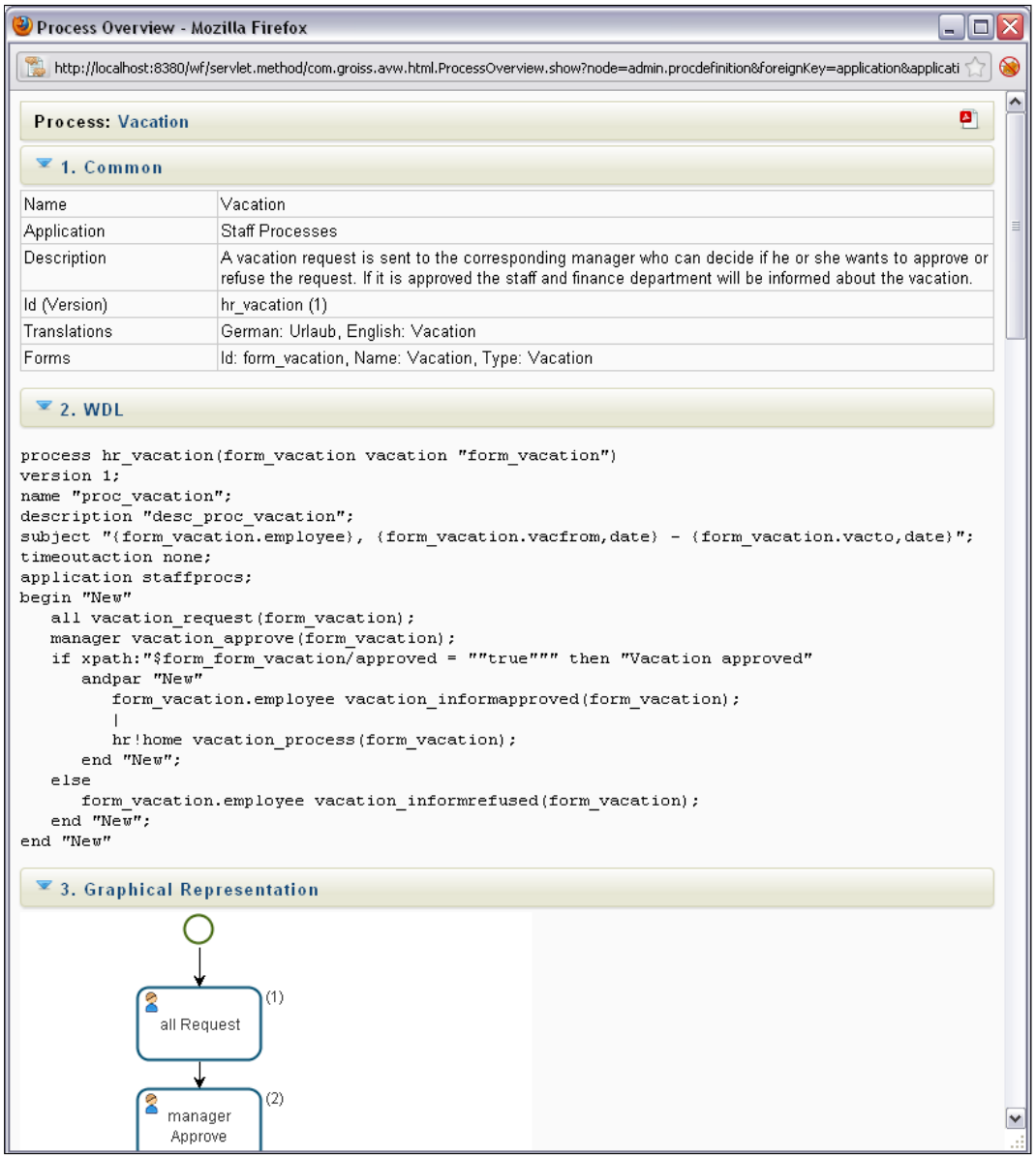


Figure 6.19: **Process overview**

- nny...next number for this process for this year
- y.....year with last 2 digits only
- yy....year
- ou....organizational unit

Hint: If the Instance Id contains spaces and the parameter *webdav.show.subject* (see Installation- and Configuration Guide - chapter *Parameters without GUI*) is set to 1 in configuration file *avw.conf*, attached dms-documents of the process instance cannot

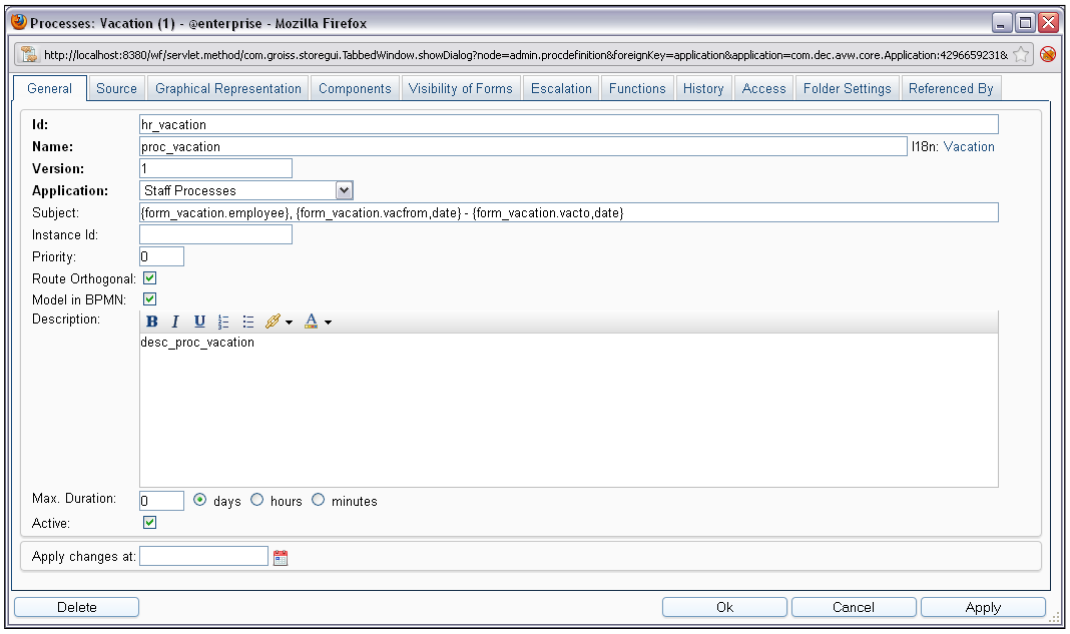


Figure 6.20: **Object-Details: General**

be opened and following error occurs: Error 1002: The document could not be found!

- Priority: The priority of the process.
- Route Orthogonal: If this Checkbox is active, the edges of the process are routed orthogonally in the process editor.
- Model in BPMN: If this Checkbox is active, the process is opened in BPM-Notation. If a process painted in WD-Notation is opened in process editor and this checkbox will be activated, the user will be asked, if he wants to change the notation. If the user confirms, the painted process will be transformed to BPM-Notation. In this case the buffer for functions Undo/Redo and Copy/Cut/Paste in process editor will be cleared! More details can be found in chapter 7.
- Description: Free text.
- Max. Duration: Maximum running time of the process, specified in days, hours or minutes.
- Active: see chapter 2.2.1
- Apply changes at: see section 2.2.1

6.5.6 Tab: Source

In this tab the WDL-Script of the selected process is shown (see Fig. 6.21).

By clicking the button **view XWDL** the XWDL–Definition of the process is shown in a new window. Activating the button **download XWDL** you can download the XWDL–Definition of the process to your local file system.

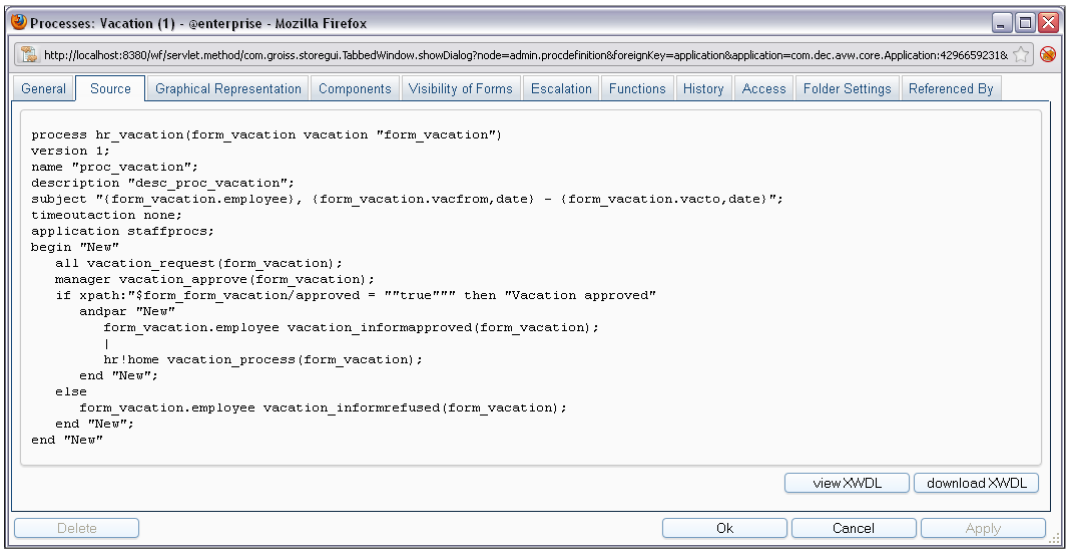


Figure 6.21: **Object-Details: WDL View**

6.5.7 Tab: Graphical Representation

This tab shows the graphical representation of the process like in the process editor.

6.5.8 Tab: Components

This tab lists the version of the tasks and forms used in the process (see Fig. 6.22). Furthermore roles, subprocesses, webservice operations and imported files by the web service are shown. By activating a link (e.g. task) within the tab a new detail-window of the object is opened.

6.5.9 Tab: Visibility of Forms

The tab *Forms* gives you an overview about all forms, which are assigned to the process. For each form a further tab is displayed, where the visibilities are listed of the individual tasks. In this overview a task appears only, if a form was assigned at the process definition.

If you want to change the visibility of a form field in a task, activate the link of the appropriate task. The HTML–page *Formfield Modes* will be shown.

Information of the HTML–page *Formfield Modes*:

- Form Type: The listed formfields under *Form Field* refer to this form.

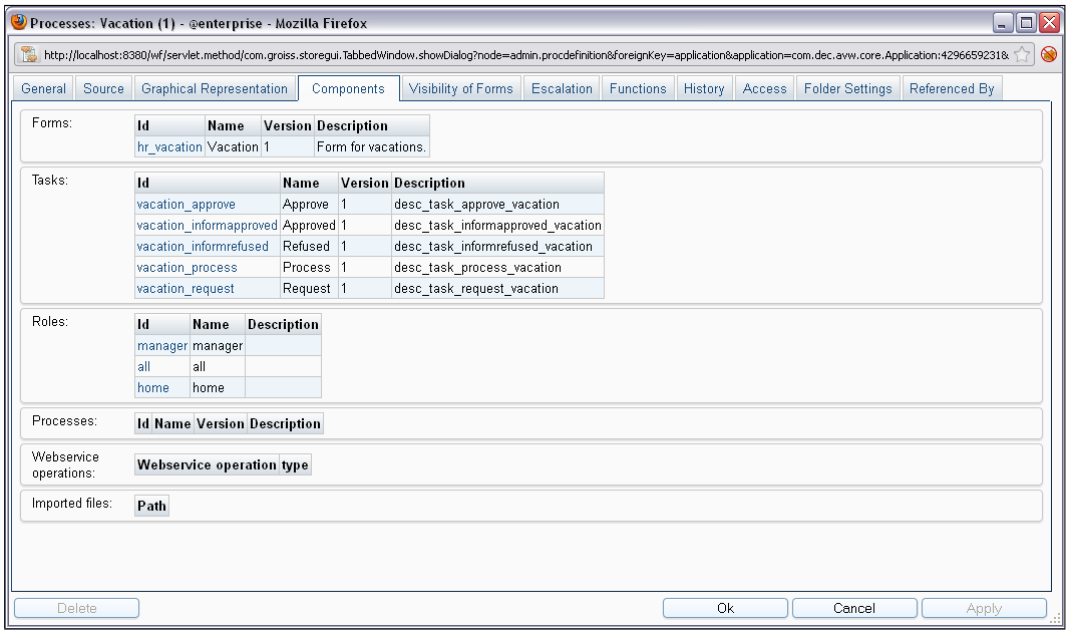


Figure 6.22: **Object-Details: Components**

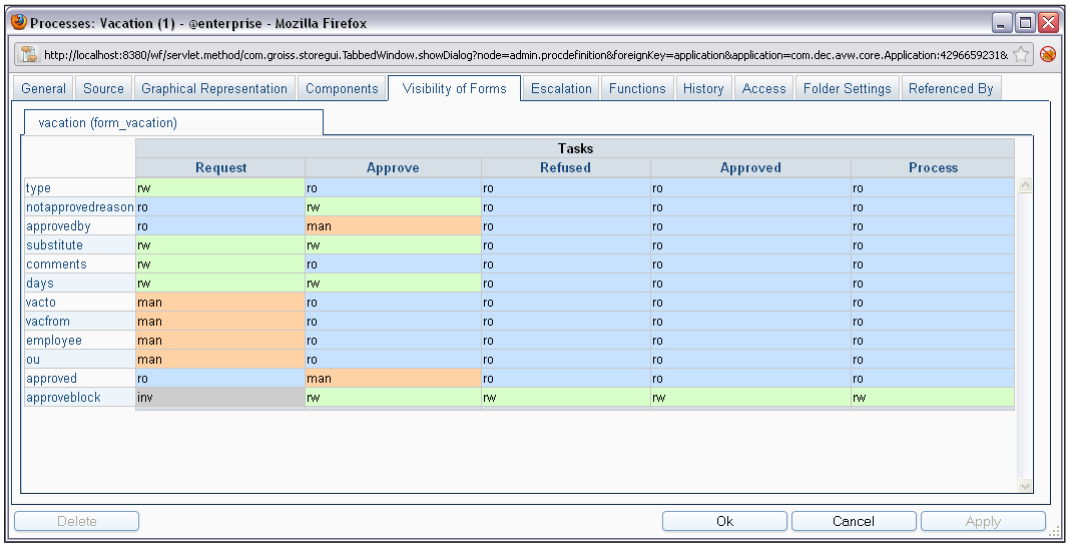


Figure 6.23: **Object-Details: Process forms**

- Task: The form is assigned to this task.
- Process: The previous mentioned tasks is assigned to this process.
- Take Visibilities from..: The visibility of an other task in this process can be taken, if there are differences between the tasks with reference to the visibility of forms.
- Form Field: Name of the form field, whose visibility should be specified.

- Invisible: If this radio–button is activated, the form field will not be shown.
- Readonly:
  1. Disabled: If this radio–button is activated, the form field can not be changed and it will dye grey.
  2. Text: If this radio-button is activated, the form field can not be changed, but it will not dye grey. This option does not work with XForms, because it is not supported!
- Writeable:
  1. Optional: If this radio-button is activated, the form field can be changed.
  2. Mandatory: If this radio-button is activated, the form field is changeable and must be filled.

If a subform is existing, whose visibility should be set, the information on the HTML–page *Formfield Modes* looks as follows:

- *Form Type*, *Task*, *Process* and *Take Visibilities from* are the same.
- Table: Name of the subform, whose visibility should be set.
- Invisible: If this radio-button is activated, the table of the subform will not be shown.
- Readonly: If this radio–button is activated, the table of the subform will be shown, but cannot be changed.
- Optional: If this radio–button is activated, the table of the subform will be shown and can be changed by the button *New Table Entry*.
- No Insert/Delete: If this radio–button is activated, no further entries can be added or deleted.

Additional you have the possibility to set the visibility of a form field for all tasks. You have to click on the link of the adequate form field and the HTML-page *Formfield Modes* will be shown. This site is analogue to the HTML–page for form fields, but the visibilities will be set for tasks and not for form fields.
By clicking the button **Preview** a new window will be opened, where the form with the made settings will be shown.
By clicking the button **Ok** your changes, which belong to the visibilities of the form fields, are saved.
By clicking the button **Close** your changes are not saved and the current HTML–page will be closed.

6.5.10 Tab: Escalations

With the help of escalations it is possible to react on timeouts during the execution of processes. It is possible to define four different types of actions which determine what should happen in the case of a timeout. The system timer "TimeoutTimer" of enterprise is responsible for checking the timeouts. If this timer is not running the system does not check if timeouts occur or not!

This tab shows all already defined escalation-objects. You can edit them or add new one by activating the functions in the toolbar.

You can edit the following attributes (required fields are bold):

- **Escalation Type:** Here you can select between following escalation types:
  1. Process due time: This escalation relates to the due date of a process, which was entered at the process-start.
  2. Activity duedates: Analog to *Activity due time* of tasks, but this escalation is used by every task in the process (see section 6.2.2).
  3. Batch unfinished: This escalation relates to unfinished batch-steps within this process. The step is specified via the steplabel.
  4. Sync unfinished: Analog to *Batch unfinished*, but for Sync-step.
  5. Receive unfinished: This escalation type relates to unfinished Receive-steps within this process. The step is specified via the steplabel.
- **Step:** If more steps are using the same task in process definition, you can define one step which will be triggered. Steps are selectable only, if escalation type is *Batch unfinished*, *Sync unfinished* or *Receive unfinished*.
- **Delay:** see section 6.2.2
- **Action:** See section 6.2.2. For action *Send an E–mail* to current agents all agents of active tasks will be collected (excepting tasks of subprocesses) and informed. The definition of action *Start a task* in process escalations is different to task escalations, because role **and** task must be entered in process escalations.
- Description: see section 6.2.2

Note: The process–escalation is an improved form of the timeout–handling of the process–editor. If you have created a new process, you should use this form.

6.5.11 Tab: Functions

This mask is analog to 6.2.3, but the functions are used in whole process instead of single tasks.

6.5.12 Tab: Folder settings

This tab offers the possibility to adapt the DMS folder settings for this process. This function is equal to the folder-form settings in section 6.4.12. It is possible to

- Add columns, edit and delete them and change the order
- Add functions, delete them and change the order
- Add forms, delete them and set their allowance (*allowed* or *denied*)

6.6 Interfaces

Interfaces allow the start of process instances by submitting an HTML form. The URL looks like following:

http://host:port/wf/servlet.method/
com.groiss.storegui.FormWrapper.showProcessStartForm?formclass=<formclass>

Perform the following steps to create such an interface:

1. Define a process *p* which uses a form of type *ftyp* as input form (i.e. the form is specified in the argument list of the process).
2. Add an entry in the interface list, which specifies that process *p* will be started when a form *ftyp* is submitted. The organizational unit of the created process instance is the third information piece necessary.
3. Test the interface with the link "Interface Forms" in the administration task list. Click on the link for the form, fill in the form and start the process by clicking the submit button.

The object-details of interfaces contain the following tabs:

- General
- History
- Access

6.6.1 Tab: General

In the interface mask you can edit the following attributes (required fields are bold):

- **Form:** Select a form type.
- **Process:** Select the process you want to start.
- **Organizational Unit**: OU where the process instance will be started.

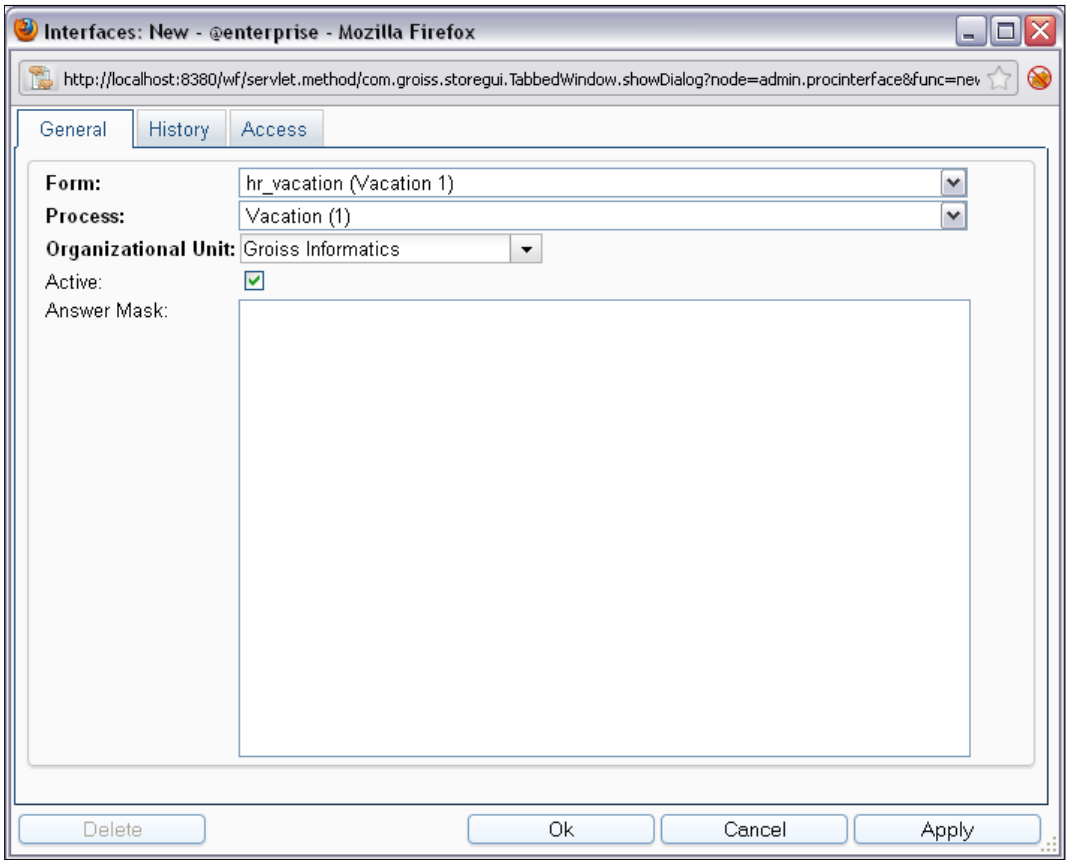


Figure 6.24: **Object-Details: Interfaces**

- Answer Mask: Contains the HTML text, which is returned to the user, who started the process. The fields %pid% and %process% will be replaced by the process id and process name.
- Active: Indicates whether the interface is active. Further information about (de)activation of objects can be found under 2.2.1.

The following string is an example for such a answer text:

<h3> Thank you for your order!</h3> Your order number is %pid%.

6.7 Function Group

Function Groups allows the grouping of task–functions. A function group consists of an unique *Id* and an arbitrary *Name* and has to be assigned to an application. By activating the I18n-link beside the name-field, the translations (if defined in application mask - tab *Properties*) of this key are displayed and can be edited directly by changing the values and activating the button *Save*. The changes are stored in the resource file of this application (see section 6.9). Optionally an order attribute can be entered for sorting the function groups. If

several task–functions belongs to different groups, they will be displayed as groups in the worklist under *Functions*. A function group can be deleted only, when it is not assigned to a *Task-Function* or a *Stored Query*.

6.8 GUI Configuration

With GUI configuration it is possible to define masks for users (worklist, dms, etc.) and the appropriate rights via the tab *Assignments* to users or rights. The elements of the mask are stored as XML files in *classes* directory of the current **@enterprise** installation or in *classes* directory of the application directory. With **@enterprise** it is possible to

- create new masks
- edit and delete available masks
- copy available masks

6.8.1 Tab: GUI Configuration

This tab allows to create and adapt masks for users. New masks need an *Id* and optionally a *Name*. Furthermore a *Description* can be entered. Selecting an *Application* is mandatory and also setting an *URL* or selecting the radio-button *Tree* for creating a XML-tree (see fig. 6.25). The *Id* is the filename of the XML, which is stored in *classes* directory.

The toolbar for creating a XML-tree offers following possibilities:

- New: Add a new node for example worklist, dms, functions, etc. (see section 6.8.1)
- Edit: Adapt an existing nodes; double-click on the element result in the same function
- Delete: Remove existing nodes and their subnodes
- Up: Selected node is moved one position upwards on the same level
- Down: Selected node is moved one position downwards on the same level
- Right: Selected node is moved one level downwards
- Left: Selected node is moved one level upwards
- Properties: In this window it is possible to set diverse properties for the current mask (see section 6.8.1)
- Preview: Displays the adapted mask like the users would it see.

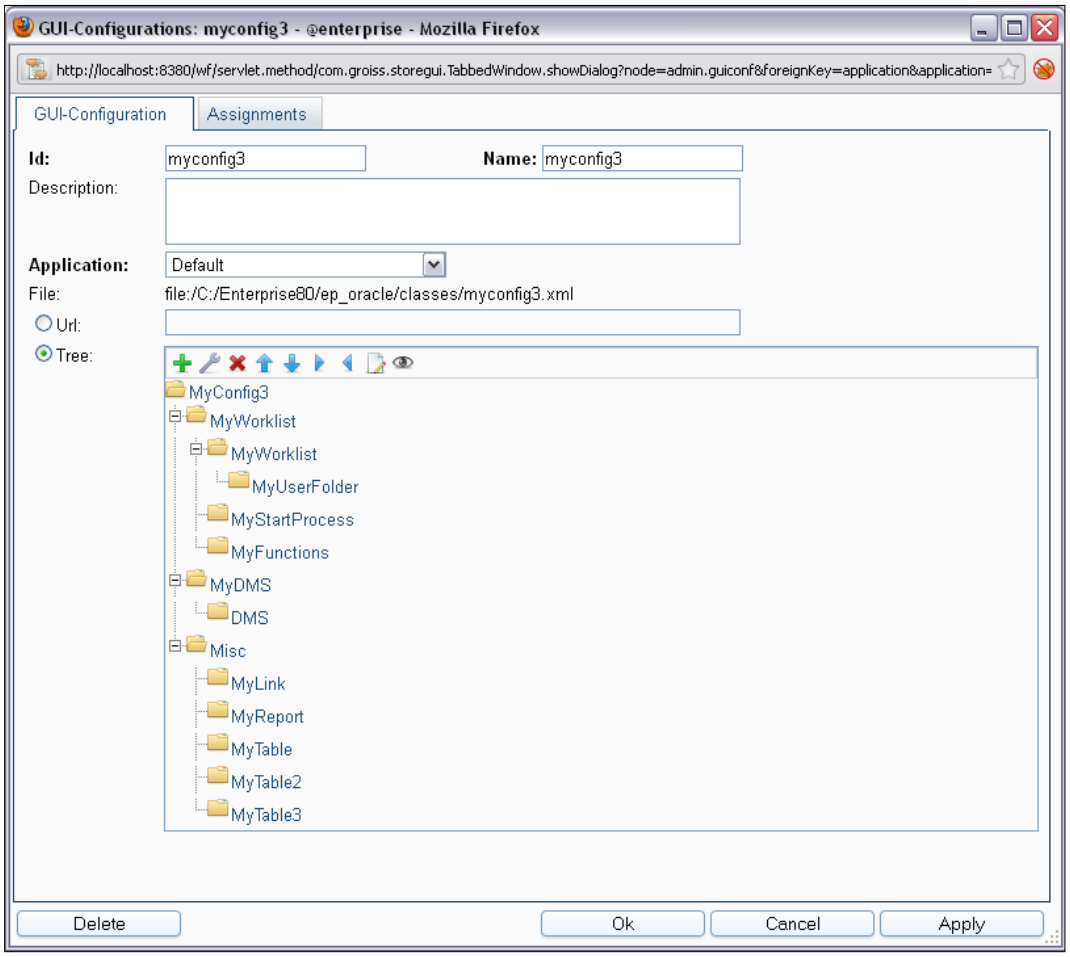


Figure 6.25: **Tab: GUI Configuration**

Node properties

By choosing the function *New* or *Edit* a new window is opened, where a new node can be added or an existing node can be edited (see fig. 6.26). Nodes always contain a *Label*. If the checkbox *Localize* is activated and the entered string is found in the default- or a specified-resourcebundle, the name will be translated (name must not contains @). By activating the I18n-link beside this field, the translations (if defined in application mask - tab *Properties*) of this key are displayed and can be edited directly by changing the values and activating the button *Save*. The changes are stored in the resource file of this application (see section 6.9). If checkbox *XML* is activated, HTML tags can be used in *Name* field, i.e. MyNode will be displayed as bold text. If this checkbox is not activated, the html code (if entered) will be displayed. Further static elements are the checkbox *Clickable* and the selectlist *Access*. If the checkbox *Clickable* is activated, the node and its subnodes will be displayed as tree. The selectlist *Access* allows to set roles. Users, who has this roles, are permitted to access this node, i.e. the node is visible and/or selectable on the mask.
Furthermore there are fix elements for each node:

- Id: Unique identification of the node. If no Id is entered, **@enterprise** will assign an Id automatically.
- CSS-Class: Possibility to enter a css-class.
- Default: If this option is activated, the page of this node is loaded in the right frame when the frameset is initially loaded (after login). This option should be assigned to one node in navigation tree only!

You can select between following node types:

- Text: Free text, which is displayed in the navigation tree; could be used for making groups.
- Link: Creates a link the navigation tree. Following attributes can be set:
  - Target Window: The content of this field contains the name of the window or the frame where the output of the function will be placed. If the field is empty, the output is sent to the frame where the worklist resides. If you enter another name, the output is sent into a separate window with this name.
  - URl: This field offers the possibility to enter an URL oder to select a @enterprise-link by activating the search-icon beside the field.
- Start process: With this node you add a link to the list of all startable @enterprise-processes. Further properties like at node *Link*. Furthermore a *Worklist Id* can be entered (e.g. standard.wl), which is the *wlid* in XML. If such an Id is set, the worklist with the corresponding Id is shown after process start.
- Function List: By adding this node global functions of one or more applications can be displayed.
- Function: This node allows to add several functions to the navigation tree. The added function does not know the context of the right frame, so global functions should be used (apply to no entry).
- Report: Here you can set a stored query, which will be executed by activating this link in navigation tree (analog to 8.3)
- Worklist: Defines the worklist with following attributes:
  - **Worklist-Adapter:** Enter your own worklist adapter here. For further information about worklist adapter please take a look into the *Application Development Guide* - chapter *Customizing the Worklist*.
  - **Type:** Set the worklist-type (e.g. worklist, role-worklist, suspension, role-suspension, etc.). It is possible to select more types simultaneously for a node.
  - **Columns:** The columns, which are displayed in the table. You can add, edit and delete columns and change their order. Columns contains an *Id* and a *Name*, which can be translated by activating the checkbox *Localize*, or an *Icon*. If the checkbox *Visible?* is activated, the column is displayed at the first call, otherwise you can add it by using the column picker. The *Id* can contain the

option, which tab should be opened when activating the link in the worklist, e.g. *process-form0* means, that the column *Process* is displayed with a link to the first form in the tab view (default is a link to process history).

- **Functions:** Here you can add available *Task Functions* or set *Action Keys* (e.g. new, cut, copy, etc.). It is also possible to change the order of the functions.
- **No documents/notes:** Activating this checkbox avoids selection of documents and notes of processes when worklist is displayed. Activate this checkbox only, if no documents or notes are used in processes and the performance of the worklist table should be optimized!
- **No userfolder filter:** Activating this checkbox avoids filtering by userfolder contents when worklist is displayed. Activate this checkbox only, if no userfolders are used and the performance of the worklist table should be optimized!

- User Folder: Definition of a user folder in the navigation tree, which is a placeholder. Attributes are analog to node type *Worklist*, but without the possibility to set a *Type*. This node can be added to navigation tree once only!

- DMS: This node allows to create and adapt a DMS-folder. You can set following attributes:

  - **Columns:** Analog to node type *Worklist*
  - **Functions:** Analog to node type *Worklist*
  - **Forms:** In this list you can define which form types are allowed or denied for this dms-folder. If the list is empty and the radio-button *Denied* is activated, all available form types of @enterprise are available for the users. Selecting a radio-button option is valid for the whole list only.
  - **Paging:** If this checkbox is activated, the paging mechanism of **@enterprise** for DMS tables is used (analog to worklist-paging).
  - **Items per page:** This defines the maximum number of entries in this DMS (folder) tables when paging is enabled.

  This node can be added to navigation tree once only like node *User Folder*!

- Table: With this node a link to a table can be created. This table can be a form-class (selectable via icon). Following attributes can be adapted:

  - **Classname:** A form-class can be selected here
  - **Columns:** Analog to node type *Worklist*
  - **Functions:** Analog to node type *Worklist*
  - Detail Window Properties: Here you can add several parameters like width, height etc. by adding a semicolon and write the parameters like the javascript method *window.open* syntax.

Note: Only form-classes created with wizard can be used correctly!

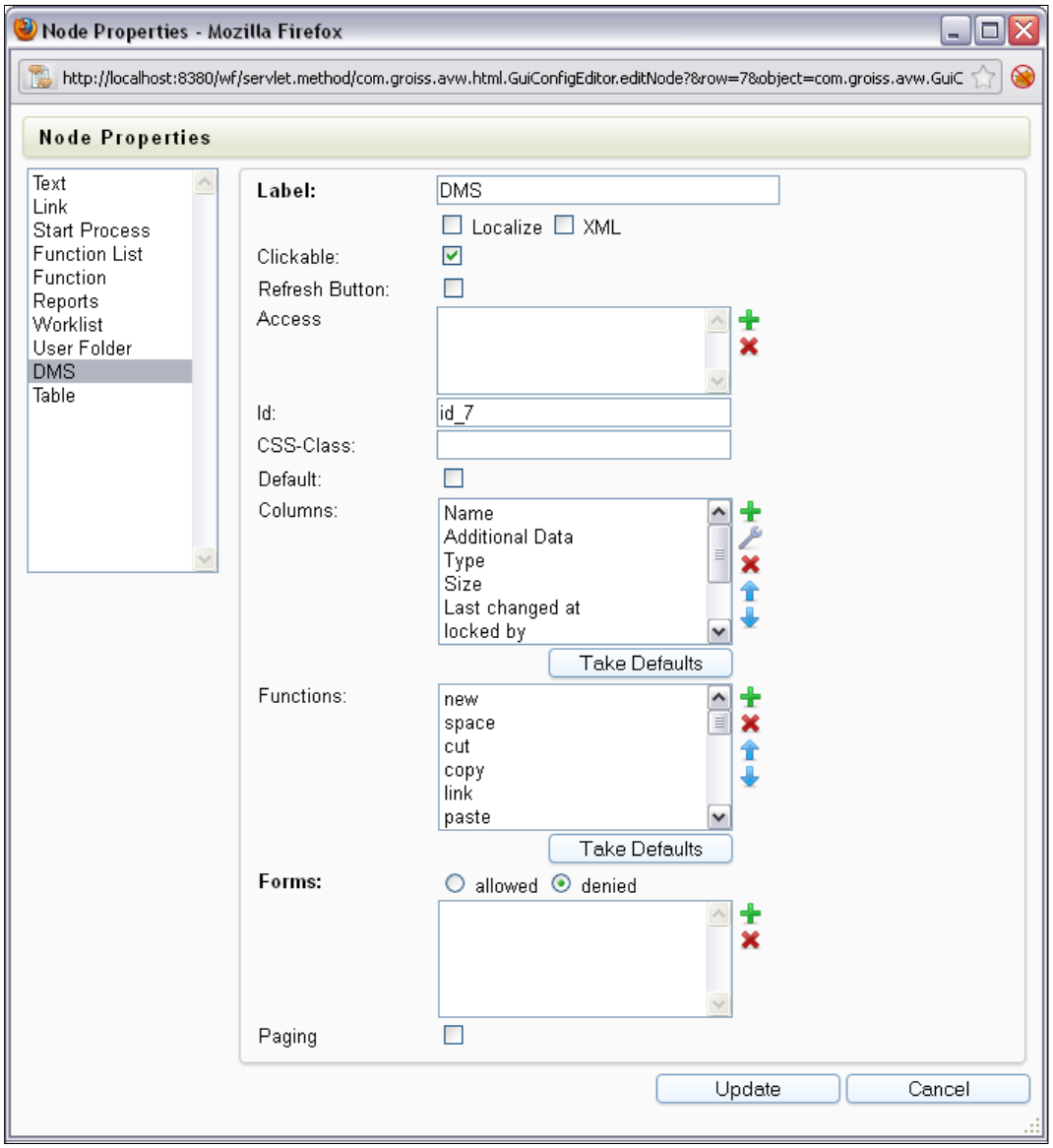


Figure 6.26: **Node properties**

Properties

With this function it is possible to define mask-specific properties. Following attributes are available:

- HTML-Mask: Equal to attribute *framepage* in XML-file *standard.xml*.
- HTML-Mask (right-to-left): Analog to *HTML-Mask*, but for attribute *framepageRTL*.
- Use buttons for first tree level: All tree-elements of first level are represented by buttons (default behavior of @enterprise). Do not use this function, if external links (e.g. link to another website) are used - @enterprise-links are possible!

- Show Folder icon: A folder icon is displayed at each element. Checkbox *Clickable* must be activated.
- Clickable: Representation as tree (expandable); cannot be used with function *Use buttons for first tree level* simultaneously!
- Refresh Interval (sec.): Timeintervall in seconds until site will be refreshed.
- Tree Rendering Method: A method can be entered here, which manipulates the tree rendering. This method must consist of a return value *Page* and a parameter *com.groiss.ds.Pair*, e.g. public Page createTree(Pair pair). This method is called without entering the parameter, e.g. <package>.<class>.createTree. An example of such a method can be found in *Application Development Guide* in chapter Configuring the Worklist Client.

6.8.2 Tab: Assignments

When using different client configurations you can now specify which user and/or role uses this configuration. The scope is either a user or a role, if more than one record matches, the one with the higher preference is chosen.

Following the description of the detail mask:

- **Agent:** The tree or URL is set for this agent. You can select between a *User* or a *Role* (with *Organizational Unit*).
- **Preference:** It is possible to assign more than one tree or URL to an agent. For this purpose you can set a preference whereas the settings with the highest preference is used at the login.

6.9 Resource Editor

This section describes the usage of the **@enterprise** Resource Editor. This tool allows to view the Resource Bundles of **@enterprise** and adapt the resource files (Strings) of installed applications. The resource editor is active only, if a resource has been entered in detail mask of the application (see section 6.1 - *Tab: Properties*). The application *Default* does not need these entries, because the standard **@enterprise** resources are always displayed (in readonly-mode). The standard **@enterprise** resources can be enhanced by a new language by activating the toolbar-function *New column* (see section 6.9.1)

Hint: The resource editor creates/adapts a csv-file (Strings.xls) and property-files when storing the changes (depending on the entered path on detail mask of the application). Resources can be adapted only, if a csv-file and/or property-files exist on the file-system. For further information about resource files please refer to **@enterprise** *Application Development Guide*.

Activate the link *Resources* to get a spread sheet of the application resource data (Strings). If this link is activated in application *Default*, a new page will be displayed where you

can select between *Strings* and *Errors* which are the standard **@enterprise** resources. The toolbar functions are explained in section 6.9.1. Following columns are available in the spread sheet (see figure 6.27):

- **LN:** Symbolizes the row number.
- **Key:** This column contains all keys of the resource file which should be translated. Existing keys cannot be changed in this view.
- **Language columns:** A set of columns is displayed whereas each column represents a language. Select the appropriate cell to edit the value. The fist language column is the default language (= *Strings.properties*).

The behaviour of the table (sorting, column picker, etc.) is equal to the standard **@enterprise** behaviour described in section 2.1.

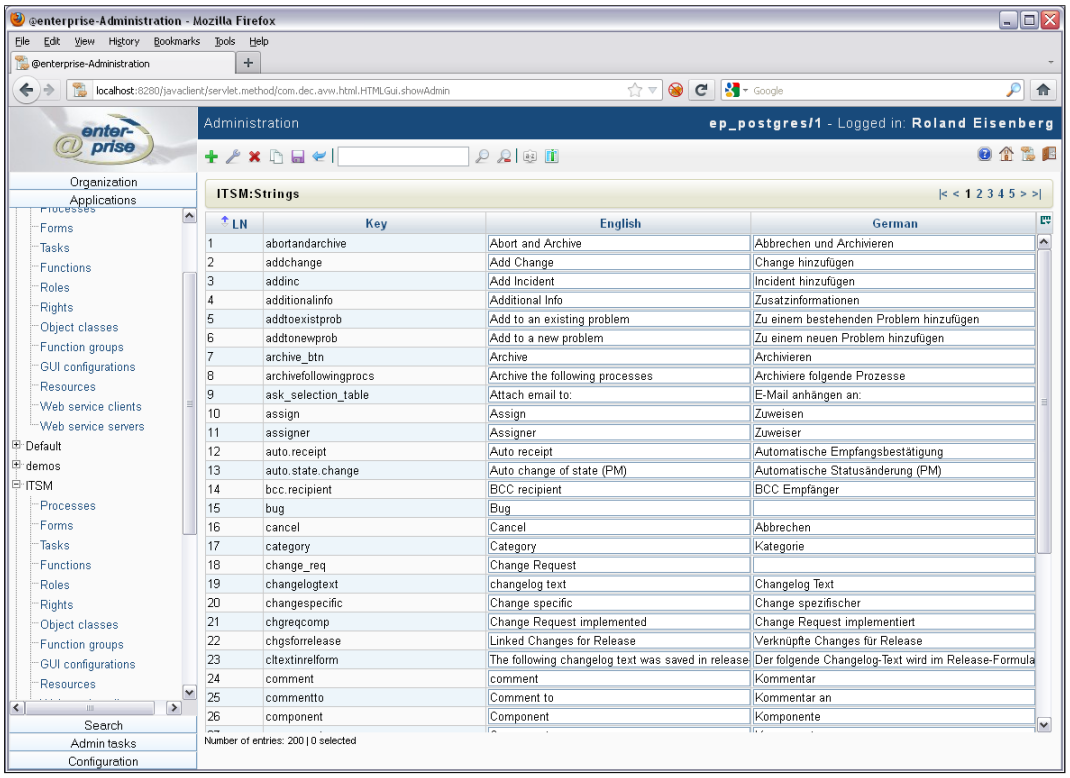


Figure 6.27: **Resource Editor spreadsheet**

6.9.1 Toolbar functions

This section describes the several functions for adapting the resource file. If the resource could not be adapted (e.g. if resource is within a jar-file), most of this functions are not allowed to execute. The toolbar contains following functions:

- **New line:** This function adds a new row to the spread sheet. If no row is activated, the first available empty row on last page will be activated or, if no empty row is available, a new page with an empty row will be created. If a row is selected and this function is activated, the new row will be inserted at a position depending on the sorted column.

- **Edit line:** Select a row and activate this function to get an overview of the selected key and its translations in a popup-window. This overview allows to adapt the translation strings and to step to the next or previous row. Activating the button *Apply* leads in refreshing the spread sheet (= changes are stored temporarily). The changes will be persistent when the toolbar-function *Save* will be activated (= changes are stored in resource files).

- **Delete line:** Activating this function leads in removing the selected row from table.

- **Copy line:** The selected row is copied when this function is activated. The copied row is inserted at a position depending on the sorted column.

- **Save:** If this function is activated, all changes of the current spread sheet are saved to the appropriate csv- and property-files. The csv- and property-files are stored in the same directory which has been defined on application mask (see section 6.1). If the files are read from a jar-file and this function is activated, only new columns (= new languages) will be stored in the classes-directory of the appropriate application.

  Hint: The created csv-file is encoded in *UTF-16LE* after activating this function.

- **Discard changes:** If this function is activated, all not saved changes are discarded (removed from session).

- **Shortsearch:** Enter a string into the textfield and activate this function to get a restricted result. This search works analog to the standard short search of **@enterprise**.

- **All entries:** Activate this function to display all entries of the table (spread sheet).

- **Sort table for resource files:** If the temporary sorting order of the spread sheet should be used for the csv- and property-files, this function should be activated. The changes are persistent only when activating the function *Save* in toolbar.

- **New column:** A popup-window will be opened where a new column can be added by selecting a value of the dropdown-list and activate the button *Create*. The default languages are displayed, if no further languages has been configured in **@enterprise** *Configuration* → *Localization* → *List of Locales*. This function can be executed always even though existing resources are read from a JAR-file.

6.9.2 Converting csv-files

If csv-files are used, they must be encoded with character set UTF-16LE. Following function is available to convert from Cp1252 to UTF-16LE:

http://'host':'port'/'context-root'/servlet.method/
com.groiss.reseditor.ResourceEditorService.convertXLS?resource=<reurl>

The parameter *resource* must be the URL to the appropriate csv-file, e.g. for application *myappl* Strings:

http://'host':'port'/'context-root'/servlet.method/
com.groiss.reseditor.ResourceEditorService.
convertXLS?resource=com/dec/myappl/resource/Strings.xls

6.10 Web Services

This chapter describes the creation of web service server/client objects which can be used for one of the web server nodes in process editor (see section 7.2.15. The requirement for creating server/client objects are existing WSDL-files in the classpath of **@enterprise**.

Please notice that WSDL files must correspond to the WS-I Basci Profile 1.1 (http://www.ws-i.org/profiles/basicprofile-1.1.html).

If you want to offer web service which are not corresponding to the WS-I Basic Profil (e.g. RPC web services, ...), these services can be added/activated via the function *Admin-Tasks* → *Communication* → *Web services* → *Local services* (see chapter 9.5.5). This kind of web services cannot be used (automatically) in processes.

Webservice Server/Client objects can be defined in every application.

6.10.1 Webservice clients

With client objects it is possible to define which web service with its parameters (IN-/OUT-parameter) is called. OUT-parameter are submitted to web service and IN-parameter are received from web service. Client objects are applicable for submitting data to an other server for processing.

The object-details contain following tabs:

- General
- Callable Operations
- History

Tab: General

You can edit the following attributes (required fields are bold):

- **Id:** A free assignable ID of the web service client object. The ID must be unique per application.

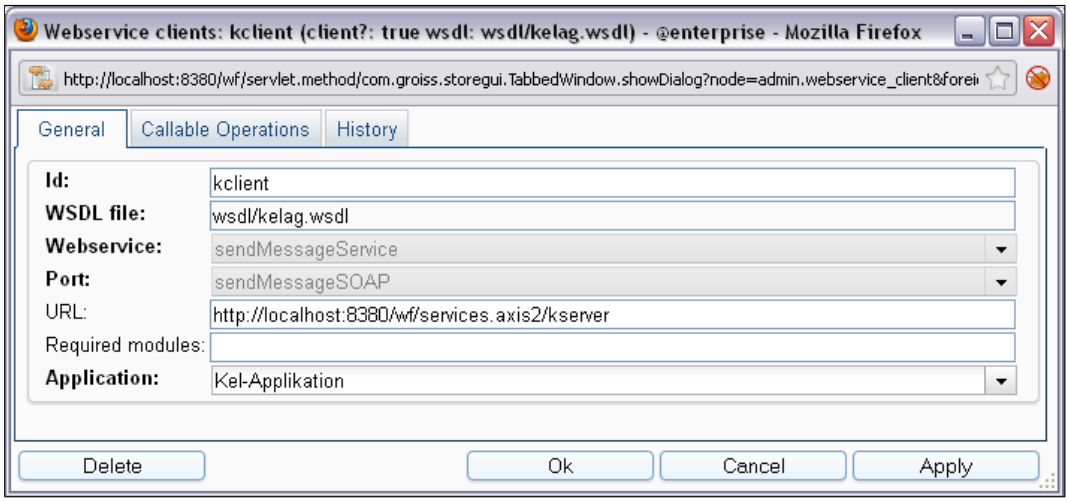


Figure 6.28: **Tab: General**

- **WSDL file:** The path to a WSDL file in **@enterprise** classpath.
- **Webservice:** A selection of web services is offered depending on the definition in WSDL-file. Selection is available only, if a correct path to a WSDL file has been entered.
- **Port:** Depending on the select *Webservice* a appropriate port can be selected which was defined in the WSDL file.
- URL: The URL of the web service which should be called. If nothing is entered, the URL defined in the WSDL file is used.
- Required modules: If needed, a comma separated list of AXIS2 modules can be entered, e.g. rahas,rampart,scripting
- **Application:** The application where the client object should be stored.

After storing the information on tab *General* an *Operations* object should be created in tab *Callable Operations*. This object allows to define IN-/OUT-parameter.

Tab: Callable Operations

In this tab a table of all operations of the current client object is displayed. This table contains the default toolbar functions and the function *Execute webservice operation* which allows to test the selected operation object with its OUT-parameter.

Activate the toolbar function *New* to create a new operation object. A new dialog will be opened where you can select an *Operation* which has been defined in WSDL file (see figure 6.29). Afterwards a XML should be created by using the function *Generate XML* which is stored in field *XML*.

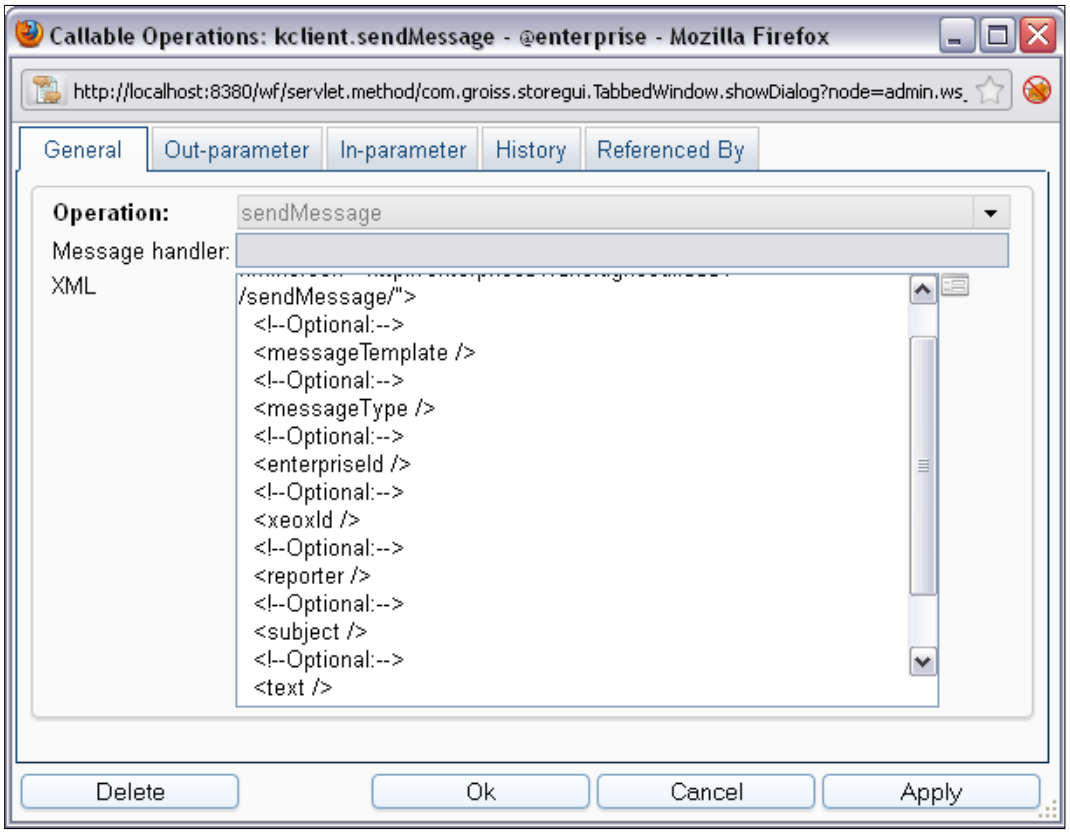


Figure 6.29: **The *Operation*-Objekt**

After successful creation of an *Operation* object, IN- and OUT-parameter can be defined. For web service client objects OUT-parameter are parameter which should be submitted for processing. IN-parameter are parameter which are received form web service (e.g. status notification about processing). A parameter is defined by an *Id*, a *Name* and a *Path (XPath)* which are required fields. Prefixes, which are defined in root-element of the WSDL file, can used as namespace-prefix in XPath expression . Parameter can be created manually by activating the toolbar function *New* or automatically by activating the toolbar function *Generate Parameters*. It is not possible to create duplicates (identified by ID)!

6.10.2 Webservice server

With server objects it is possible to provide web services at the server. Other systems are able to call these services.

The *Webservice server* dialog is analog to object *Webservice clients*:

- General: Contains the same attributes as *Webservice clients*, but no URL can be entered.
- Callable Operations: Analog to *Webservice clients*, but the toolbar function *Execute webservice operation* is not available. IN-parameter are parameter which are

received for processing and OUT-parameter are parameter which should be submitted (e.g. status notification about processing). Optionally a *Message handler* can be entered which has to implement the interface *com.groiss.ws.server.MessageHandler*. If a handler is entered, this operation cannot be used in a process definition.

- History: Analog to *Webservice clients*.

7 Process Definition

In this chapter we describe the definition of processes. **@enterprise** provides two ways for defining processes.

1. graphical definition using the process editor,
2. definition of the process as a script in the Workflow Definition Language (WDL).

Both options have the same expressiveness - you can define a process with the process editor, save it as a WDL-script, edit the script, load it again, and make additional changes in the process editor[1].
In the next section we describe the script language WDL, afterwards the handling of the process editor is shown.

7.1 WDL

In the following we describe the syntax and semantics of the language elements of WDL. The language has resemblance to a structured programming language and allows the definition of workflow processes. Each WDL script consists of a process header, a declaration section, and a statement section. Example:

```
process jobproc()
name "jobproc";
description "simple process";
version 1;
subject f.subj;
forms f Jobform;
begin
  <label_order> all order(f);
  loop
    f.recipient a_task(f);
    exit when (f.finished = 1);
  end;
  label_order:user inform(f);
end;
```

[1] Graphical layout and annotations are not preserved across notations.

The process definition starts with the keyword process, followed by the process id and a list of arguments. The declaration section contains a set of keyword-value pairs, for example version 1;.
The statement section begins with the keyword begin and ends with end. In between the structure of the workflow, containing task calls, subprocesses, system steps and control structures is described.

7.1.1 Lexical Conventions

In WDL the following lexical rules apply:

- Ids

  Ids are identifiers for tasks, roles, users, and similar entities. The following conventions apply:

  Ids start with a letter or $ or / or \. After the first character more of these characters plus digits can follow. The length of an id must not exceed 80 characters.

- Strings

  Strings are character sequences enclosed in double quotes. A double quote within a string is denoted as two consecutive double quotes.

  Example: "This is a string." "This is a string with two ""double quotes""."

- Comments

  All characters between "/*" and "*/" are ignored. Comments can span lines. [2]

- Case-Sensitivity

  WDL is case-sensitive, this means "If" is not equal to "if". All keywords use lower case characters.

- Keywords

  The WDL keywords are listed in table 7.1. A keyword enclosed in single quotes is no longer interpreted as a keyword, but as an id.

7.1.2 Process header

Syntax:

```
processdef =
   "process" id "(" [formdecl{ "," formdecl}] ")"
   { pdeclaration ";" }
   "begin" [nodename]
      statseq
   "end" [nodename] .
```

Description:

[2] Comments are ignored when loading the WDL script, therefore they are not visible in the system.

abort	adhocTasks	and	andpar	application
autofinish	baseform	batch	begin	branch
call	choice	corr	correlation	current_tx
days	description	do	else	elsif
end	exception	exit	for	forms
gobackonerror	goto	hours	if	in
instanceid	invoke	loop	maxtime	minutes
name	new_tx	newthread	none	not
null	or	orpar	out	owner
parallel	priority	process	raiseEvent	receive
registerForEvent	repeat	reply	skipable	start
startfunction	startnow	subject	success	sync
system	then	timeoutaction	timeouttask	unregister
until	version	when	while	

Table 7.1: Keywords in WDL

- Id: id (internal name) of the process.
- Parameter list: Forms which are parameters of the process. These are used when the described process is called as subprocess from another process. The forms are passed by reference, this means the form data are not copied.
- pdeclaration: declarations, see below.
- statseq: sequence of statements.

7.1.3 Declaration part

In the declaration part some general information about the process is specified.
Syntax:

```
pdeclaration =
   "name" string
 | "description" string
 | "version" number
 | "subject" ( formfield | expressionstring )
 | "maxtime" number ("days" | "hours" | "minutes" )
 | "timeoutaction" ("none" | "abort" )
 | "timeouttask" taskstmt
 | "forms" formdecl { "," formdecl}
 | "application" application
 | "instanceid" string
 | "priority" number
 | "adhocTasks"  adhoctask { "," adhoctask }
.
```

Description of the declarations:

- name: Name of the process, is displayed in the end user interface.
- description: free text
- version: Integer, declares the version of the process
- subject: specifies the content of the subject column in the worklist. Can be a single formfield designation (formid.fieldid), or an expression referencing several formfields. More information can be found in section 6.5.
- maxtime: intended maximum running time of the process, specified in days, hours, or minutes.
- timeoutaction: Reaction in case of timeout, two possibilities:
  - none: no reaction
  - abort: Process termination
- timeouttask: definition of an action, which is performed in case of timeout.[3]
- forms: declaration of forms as process local data containers. The definition of a local form is:

  formid formtype ["baseform" baseformid] ["formname"]

  - formid: is the id of the local form in this process
  - formtype: is the id of a formtype defined in the system
  - formname: is the local display name of a form in this process (optional)
  - baseform: if the declared form is a view-form, the base form must be specified here

  Example: forms rg bill, ls item_list, rgsum shortbill baseform rg;
- application: id of the application the process belongs to.
- instanceid: Id which identifies the started process instance uniquely. It is also possible to enter a pattern which allows to specify a numbering scheme. More information can be found in section 6.5.
- priority: The priority of the process.
- adhocTasks: Can be used to declare additional tasks which may be instantiated programmatically during the execution of the process. They provide a means to define form and field visibilities. Syntactically, they are task statements (see below) without a declared agent list, since the agents will be specified at run time. Each adhoctask is defined as:

  taskid "(" [formlist] ")" [nodename]

All declarations, except the name, version and application are optional.

[3] The specification of a timeoutaction and a timeouttask is deprecated. Escalations are a much better way to formulate events along the process execution timeline with corresponding actions.

7.1.4 Basic Statements

The statement section is the central part of the process specification, it is enclosed between the keywords begin and end. It contains at least one statement. Statements are terminated with a semicolon.[4]
Syntax:

```
statseq = { [ "<" labelid ">" ] statement [ nodename ] ";" }.


statement =
 (
    batchstmt
  | branchstmt
  | choice
  | exitstmt
  | gotostmt
  | ifstmt
  | invokestmt
  | loopstmt
  | par
  | parforstmt
  | raiseEvent
  | receivestmt
  | registerForEvent
  | repeatstmt
  | replystmt
  | subproccall
  | sync
  | systemstmt
  | taskstmt
  | unregister
  | whilestmt )
.
```

labelid: An id of this step within the process definition. Must be unique and can be used as exact reference to this step.
nodename: A string used as the display name for the statement (and the corresponding node in the process editor). Does not need to be unique.
e.g. <ordertask> all order(form) "place the order";
In the following we describe the different statements:

Manual Tasks Specifications

Manual tasks are denoted as:

[4] The first statement should be a task statement, because the agent definition of the first statement is used to determine the agents who may start the process. For example, if a process starts with a loop, the process cannot be started with the normal user interface because it will not appear in the list of startable processes of any user. However, such processes can be started using the API or can be used as subprocesses.

Syntax:

```
taskstmt =
  ( "none" | agentlist ) taskid "(" [ formlist ] ")" [ "skipable" ].

agentlist =  agent { , agent }.
```

Description:
agentlist: There are several possibilities to define the agents of a task:

- The agent can be a **user**, specified as the id of an **user**. Should be used only in special cases, because the process definition should usually stay independent of specific users.
- The agent can be a **role**, the id of the role is specified. Each user who has the role is a potential agent of the task. The task will appear in the role-worklist of these users.
- Additional to the role an **organizational unit** can be specified. The notation is:

  org_unitid "!" roleid. Example: marketing!sek.

  The organizational unit of the current task is changed to the given OU. The organizational unit of the overall process does not change.
- **Agent of a previous step**: The agent of this task is the last agent of another task. The other task is referenced via its labelid according to the syntax: labelid ":user"

  Example:

  ordertask:user sek task1();
- **Agent from a form field**: The agent is taken at run-time from the content of a field in a process form. the content is either a role id, a user id, a role id together with an organizational unit id, or the specification of an agent of a previous step.
- **Empty agent**, Syntax: none.

  At run-time the agent must be set either programmatically or manually by the agent of the previous step.
- **Java-Method:** Name of a Java Method which returns either a role id, a user id, a role id together with an organizational unit id, or the specification of an agent of a previous step or a user or role object.
- **Sequence of agents**: Can be formed by a comma separated list of agent definitions in the variants stated above. The task is routed to the agents of the list in a sequential manner[5].

Note that the agent of the process definition can also be overwritten at run-time by a preprocessing method of the task.
taskid: The id of a task defined in the application. If you specify an id which is not the id of an already defined task, you can use the option "Generate Tasks" when loading the

[5]Preprocessing is executed once before the first agent, postconditions are executed once after the last agent

process. The task is then generated with the id from the process definition (and the same name, all other fields empty).
formlist: Comma separated list of formids. Forms, which have been defined either in the argument list of the process or were declared as local forms.[6]
skipable: If a taskstmt has the empty agent ("none") and is marked as skipable and no agent is set at runtime, the corresponding task is simply omitted. The task would be instantiated only if an agent has been set via a preprocesing method.

Subprocess Call:

A process can be called as part of the execution of another process. This allows to design processes in a reusable and modular manner or to build layers of abstraction to provide a proper level of detail.
Syntax:

```
subproccall =
  "call" subprocid "(" [formlist] ")".
```

The call statement instantiates one process of the definition denoted by subprocid as part of the current process execution. Execution is synchronous, the called process will get control and when it ends, the control recommences in the calling process after the call statement.
Forms can be passed along the call. The formlist is a comma separated list of form ids. The forms are passed *by reference*, no data is copied. The formids of the call refer to form variables in the calling process (actual parameters) and must match the forms declared in the parameter list of the called process.

System Step

A system step is used to execute a Java method without any manual intervention. The name of the method is specified after the keyword system and followed by a comma separated list of string literals which is enclosed in parentheses. Since such methods are executed synchronous, they should be rather short in terms of execution time.
Syntax:

```
systemstmt =
   "system" methodname "(" [ string { "," string } ] ")".
```

Note, that you must specify the full-qualified method name including the package name.
Example: `system com.groiss.demo.Step.exec("p1","p2");`

Batch Steps

Like system steps, batch steps are also executed automatically by the engine. The main difference is that batch steps are called asynchronously and can have an arbitrary long execution time. A handler class must be specified to be able to react to events during this

[6] At run-time, the icons for those forms will appear in the worklist for instances of this task. The form content is visible and editable in this task. See section 6.2 for a description how to restrict the rights to view and edit forms in a task.

asynchronous execution. Detailed information concerning batch jobs can be found in the Application Programming Guide and in the API-documentation.
Syntax:

```
batchstmt =
  "batch" batchAdapterClassName "(" [ paramstring ] ")
    { "startnow" | "newthread" | "autofinish" | "gobackonerror" }.
```

Note, that you must specify the full-qualified class name, including the package name. Example: batch com.groiss.demo.DemoBatchAdapter("param").

7.1.5 Control Structures

The flow of control in a process is defined using the control structures of WDL. All the usual control structures like sequence, alternative execution and repeated execution are provided along with the crucial ability to specify parallel execution.

Sequence

Sequential execution of statements is specified by simply listing the statements one after another.

Example:
Execute first the task insert_order() from role sec. After this activity is finished, the activity survey should be performed by a member of the role clerk. After this, in the organizational unit production the task manufacture should be performed by users in the role worker.

```
...
sec insert_order(order);
clerk survey(order);
production!worker manufacture(order);
...
```

Conditions

Conditions are used in WDL in the following control structures:

- Alternatives: if, choice
- Loops: while, repeat, loops - exit when
- Postconditions in tasks

Comparisons of form values and literals and boolean Java methods can be combined in the usual manner via logical operators to form complex conditions. Additionally, WDL-conditions can be defined in Groovy and via XPath-Conditions. For more information about Groovy and XPath-Conditions see the *Application Development Guide*.
Syntax:

```
cond = expr1 { "or" expr1 } .

expr1 = expr2 { "and" expr2 }.

expr2 = [ "not" ] expr3.

expr3 =
     "(" cond ")"
   | methodcall
   | booleanformfield
   | formfield relop (number | string | formfield | "null").

relop = ( "=" | "<>" | "<=" | ">=" | "<" | ">" ).

formfield = formid "." fieldid.
```

Examples:

- f.recipient = null
- f.ordervalue > 100000
- com.groiss.Check.isAvailable("f.amount") and f.class > 3
- (f.recipient <> null or f.value > 10000) and f.class = 4
- groovy: form_f.subject == "Book"
- xpath:$form_f/subform[@id='1']/form/status = 'ok'

Java methods should have 0 to n literal string parameters and a return value of type boolean. See the @enterprise Programming Guide for details on writing such Java methods.

If: system evaluated alternatives

if and elsif constructs allow the conditional execution of process parts. Syntax:

```
ifstmt =
   "if" cond
   "then" [nodename] statseq
   { "elsif" cond "then" [nodename] statseq }
   ["else" statseq ]
  "end".
```

Description:

- cond: A condition as defined above.
- statseq: a statement or a sequence of statements.

Example:

```
if order.amount <= 2000 then "small orders"
   clerk write_confirmation()
elsif order.amount <= 5000 and order.class = 4 then "medium orders"
   manager approve()
elsif ...
   ...
else
   ...
end
```

Choice: mixed automatic and manually evaluated alternatives

Choice statements allow the user to choose the process path from a predetermined but run time dependent set of available paths.
Syntax:

```
choice =
   "choice" [ nodename ]
     { branchname [ "," cond ] ":" statseq }
   "end".
```

Description:
Each path has a name (denoted with branchname), where an arbitrary string can be given, and an optional condition. The engine first checks the conditions of all potential branches, only the branches where no condition is specified or the condition evaluates to true are presented to the user for the final selection. When no conditions are given, the selection is done purely manual.

Example:

```
choice "manual selection"
   "order now", f.sum < 5000:
      sec order(f);
   "check again":
      clerk check(f);
   "archive":
      system Archive.insert();
end;
```

While: repeated execution

Syntax:

```
whilestmt =
   "while" cond "do" [ nodename ]
     statseq
   "end"
```

.

Description:
The statements in the loop body (between "do" and "end") are executed over and over again, as long as the condition evaluates to true. Since the condition is evaluated before the body of the loop, the body may never be executed zero or more times.
Example:

```
while f.proved = 0 do
   sec correct(f);
end;
```

Repeat: accepting repeated execution

Syntax:

```
repeatstmt =
   "repeat" [ nodename ]
      statseq
   "until" cond.
```

Description:
The statement sequence in the body is executed repeatedly until the condition evaluates to true. Since the condition is at the end of the statement block the statements are executed at least once.
Example:

```
repeat
   clerk insert_data(order);
   call check_data(order);
until order.data_ok = 1;
```

Loop - exit when : generalized repeated execution

Syntax:

```
loopstmt =
   "loop" [ nodename ]
      [ statseq1 ]
      "exit" "when" cond;
      [ statseq2 ]
   "end".
```

Description:
The statements in statseq1 are executed. The condition of the "exit when" is evaluated. If this result of the evaluation is false, the statements of statseq2 are executed and the loop is executed again. If the evaluation result is true, the loop terminates without further execution of statseq2.

Andpar and Orpar: parallel execution

Parallel execution of process paths can significantly reduce the overall processing time. The two control structures andpar and orpar allow the definition of a predetermined number of parallel execution paths.
Syntax:

```
par =
  ( "andpar" | "orpar" ) [ nodename ]
        statseq
    { "|" statseq }
  "end" [ "do" parmethod ].
```

Description: The parallel branches are separated by the bar "|". When the par is reached, all parallel branches are instantiated simultaneously. Continuation depends on the kind of parallelism:

- andpar: Process is continued, when all parallel branches are finished.
- orpar: Process is continued, when one parallel branch is finished.

The parmethod is described down below.
Example for andpar:
For the handling of a complicated business case the consultation of three assessors is necessary. After they make their assessment, a final judgment can be performed.

```
  ...
  andpar
     assessor1 make_assessment(s_form1);
   | assessor2 make_assessment(s_form2);
   | assessor3 make_assessment(s_form3);
  end
  s_ou!manager judge(s_form1, s_form2, s_form3)
...
```

Example for orpar:
When calculating the route for a business trip two route planners are consulted. However, the result of one of them is sufficient for going on in the process:

```
...
clerk insert_tripdates(flyform);
orpar
    clerk check_routeplanner1(flyform);
 |  clerk check_routeplanner2(driveform);
end
...
```

The (*paramethod*) can be specified at the end of an andpar and is used to implement generalized forms of parallelism.
When a fixed number of branches (n of m) have to be finished, the method

com.groiss.wf.SystemAction.join(n,action)

can be used. Both parameters are strings:

- n: the number branches that must be finished, in order for the whole par construct to be finished
- action: contains the value *none* or *cancel*. Active branches will be aborted by setting the value *cancel*.

Example for an andpar with n of m finished branches:
For the handling of a simpler business case, the consultation of two assessments out of three are necessary:

```
...
andpar
   assessor1 make_assessment(s_form1);
 | assessor2 make_assessment(s_form2);
 | assessor3 make_assessment(s_form3);
end do com.groiss.wf.SystemAction.join("2","cancel");
...
```

If overall completion of parallelism can not be defined by completion of a fixed number of branches, but is rather computed at run time, an arbitrary Java method can be called. More about that can be found in the *Application Development Guide*.

Parallel For: runtime determined number of parallel branches

The parallel for statement can be used to split the process execution into a number of identical parallel paths where the number is determined at runtime.
Syntax:

```
parforstmt =
   "parallel" "for" (localformid "in" formid."subformtableid | iteratorclass)
   "do" [ nodename ]
      statseq
   "end" [ "do" parmethod ].
```

With the help of this control sequence it is possible to either generate a parallel branch for each row of one of the subform tables of a main table or to generate parallel branches according to an iterator.
Description:

- localformid: new local variable referring to the corresponding sub form within the parallel branch.
- formid: id of the mainform.
- subformtableid: id of the sub form table as defined in the tablefield for the formtype of the mainform.

- iteratorclass: name of a class which implements the interface com.groiss.wf.ParForIterator.

The end node of a parallel for can take an optional parmethod in order to implement specific end conditions: When a fixed number of branches (n of m) have to be finished, the method

com.groiss.wf.SystemAction.parforJoin(n,action)

can be used. Both parameters are strings:

- n: the number branches that must be finished, in order for the whole par construct to be finished
- action: contains the value *none* or *cancel*. Active branches will be aborted by setting the value *cancel*.

If overall completion of parallelism can not be defined by completion of a fixed number of branches, an arbitrary Java method can be called. More about that along with examples of parfor constructs with subforms and with iterators can be found in the *Application Development Guide*.

Branch Statement:

The branch statement allows one to split the process execution into a main path and into an ancillary flow (the branch).
Syntax:

```
branchstmt =
  "branch" [nodename]
    statseq
  "end".
```

Statements in the branch execute in parallel to the statements in the main flow. Termination of either one does not terminate the other one, so branches may outlive the main execution path of the process.
Example:

```
begin
  ..
  clerk enter(f);
  supervisor check(f);
  branch "hold in evidence"
     recordkeeper inform(f);
     ...
  end;
  worker build(f);
  ...
end
```

Goto Statement:

Gotos allow to deviate from the structured flow of control and to jump to other parts of the process specification.
Syntax:

```
gotostmt =
   "goto" labelid.
```

The flow of control resumes at the statement denoted by the labelid.
Example:

```
<entry> clerk enter(f);
supervisor check(f);
if (f.quality <> "OK") then /* denotes exceptional case */
   goto entry;
end;
worker build(f);
```

In this script the goto statement causes that the tasks enter and check to be repeated when the quality is not acceptable. Note that a repeat until would usually be a better formulation of the flow, but the designer might have chosen the goto explicitly to distinguish the exceptional flow from the usual execution sequence.
When used excessively or with poor judgment, gotos can severely harm the readability of a process description and make it almost unmaintainable. If at all, use them with care and only in well founded cases.

7.1.6 Event Mechanism

The event mechanism allows to signal process progress to (other) process instances which expressed interest in such an event. On arrival of such an event a handler can be called or the execution of a stalled process can be continued. Detailed information about events can be found in the *Application Development Guide*.
Syntax:

An event can be raised with:

```
raiseEvent =
   "raiseEvent" "(" eventname "," "current_tx" ["," form] ")".
```

Events can be waited for with:

```
sync =
   "sync" "(" eventname "," eventhandler ["," form] ")".
```

Registration of a handler for an event is done via::

```
registerForEvent =
   "registerForEvent" "(" eventname "," eventhandler ["," form] ")".
```

Handlers can be unregistered with:

```
unregister =
  "unregister" "(" eventname ")".
```

Description:

- eventname: the name of the event.
- current_tx: the event handler should be carried out in the same transaction (no other value possible).
- form: either a form or a form field which serves as the context object.
- eventhandler: a Java class implementing the interface com.groiss.event.EventHandler.

7.1.7 Web services

The WDL provides elements to incorporate Web Services into process descriptions in a straightforward manner. Web services can be called via invoke, process execution can be stalled with receive until a web service is called by an external entity, or a reply can be send as an answer to a webservice invocation issued earlier.
Web services nodes must reference the service operation to be used and provide a mapping between the message elements and the process data containers (the forms). Web services and operations are defined via the admin interface in @enterprise manually or on the basis of a WSDL file.
Details can be found in the *Application Development Guide*.
Syntax:

Incoming Message (RECEIVE):

```
receivestmt =
  "receive" [ "start" "process" ] operationspec "(" [incorrparams] ")" ["end"].
```

Reply Message (REPLY):

```
replystmt =
  "reply" operationspec "(" [outparams] ")".
```

Outgoing Message (INVOKE):

```
invokestmt =
  "invoke" [address "."] operationspec  "(" [inoutparams] ")"
     ["success" statseq ]
     ["exception" statseq]
  ["end"].
```

Common statement-parts, which are used by the webservice-nodes are:

```
operationspec =
  serviceid "." operationid.
```

```
incorrparams =
  incorrparam {"," incorrparam}.

incorrparam =
  inparam | corrparam.

corrparam =
  ("corr" | "correlation") xpath "=" messagecomp.

inoutparams =
  inoutparam {"," inoutparam}.

inoutparam =
  inparam | outparam.

inparam =
  ["in"] xpath "=" messagecomp.

outparams =
  {"," outparam}.

outparam =
  ["out"] messagecomp "=" xpath.
```

Short description of the syntactical elements:

- serviceid: The ID of the web service.
- operationid: The operation-ID of the web service.
- address: The URL of the web service.
- messagecomp: The ID of the (IN/OUT)-parameter of the message.
- xpath XPath expression denoting the form element to map.
- statseq: Sequence of statements.

Example for webservice nodes:

```
...
invoke mywebservice.SendMessage(
    MessageTemplate="form_ticket/messageTemplate",
    MessageType="$form_ticket/messageType", enterpriseid="$pi/id",
    xeoxid="$form_ticket/xeoxId", reporter="$form_ticket/reporter",
    'subject'="$form_ticket/subject", """0"""=SendMessageResult)
  exception
   administrator inform(ticket);
end;
...
```

```
 receive kserver.sendMessage(
   corr "$pi/id"=enterpriseId,
   "$form_ticket/messageTemplate"=messageTemplate,
   "$form_ticket/messageType"=messageType,
   "$form_ticket/enterpriseId"=enterpriseId,
   "$form_ticket/xeoxId"=xeoxId,
   "$form_ticket/reporter"=reporter,
   "$form_ticket/subject"='subject',
   "$form_ticket/text"=text,
   "$form_ticket/analyst"=analyst
 ) waitforincomingmessage;
...
 reply kserver.sendMessage('out'="""0""");
...
```

7.2 The Process Editor

The @enterprise process editor provides you an easy way to define workflows. The process editor supports the notations *BPMN* (Business Process Modeling Notation - see figure 7.1) and *WDN* (Worklist Definition Notation - see figure 7.2).

To start the process editor, go to the system administration, select the application where you want to define the process and click on the link "Processes":

- Click on the toolbar icon "Create" to create a new process with the editor. The editor is opened in BPM-Notation by default (can be changed in process editor settings).
- If you want to edit a process, select it in the list and click "Edit". The process editor will start and show the selected process definition. The representation of notation depends on the set notation during creation of the process.

7.2.1 The Process Editor Window

The main window of the process editor has the following sections:

- **Title bar:** In the title bar you see the name of the process you edit.
- **Menu bar**: The menu bar contains the following menus:
  - Process
  - Edit
  - View
  - Help
  - Symbol Bar
- **Drawing area**: In this area you see the graphical workflow definition.
- **Function list**: The function list shows the function buttons for editing the process definition.

Hint: To avoid problems with popup-blocker, we recommend to turn it off!

7.2.2 The Functions of the Menu Bar

The *Process* Menu

- **New:** With this function you discard the current contents of the process editor and start editing a new process.
- **Open:** With this function you can open existing processes for modification.

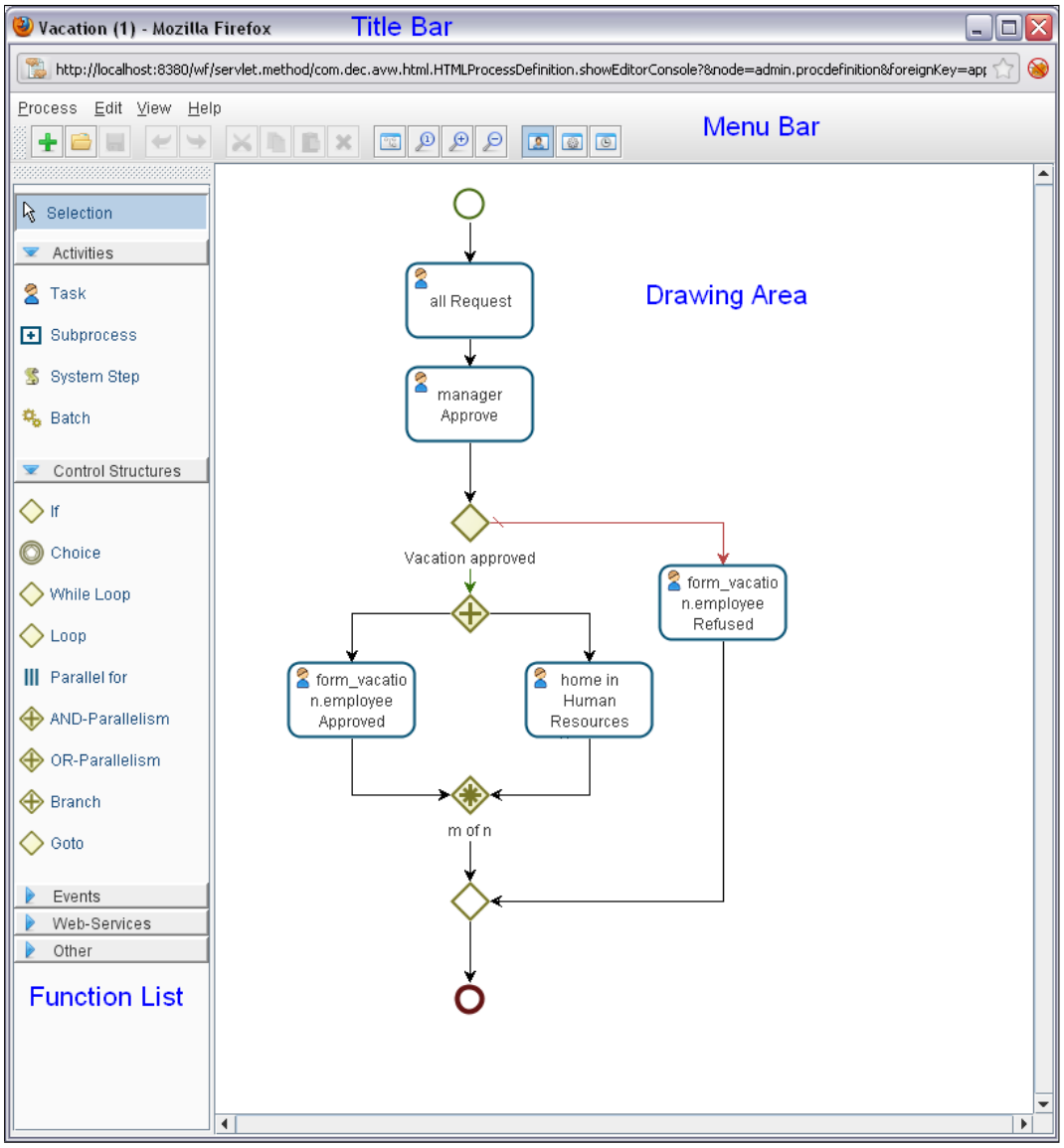


Figure 7.1: **Process Editor in BPM-Notation**

- **Save:** You save the changes. This means the process is stored in the server's database. The system informs you, whether the operation was successful. If steps are not specified sufficiently (e.g. no task is assigned to an activity), the process will be saved and set on inactive. Then the process has to be enabled manually, if you want to use it (see chapter 6.5).

  Note: Saving a process is possible when at least the name and the id has been set (see function *Properties*).

- **Save as...:** Save the current process under a new name. A dialog window for specifying name and id will appear (Fig. 7.3).

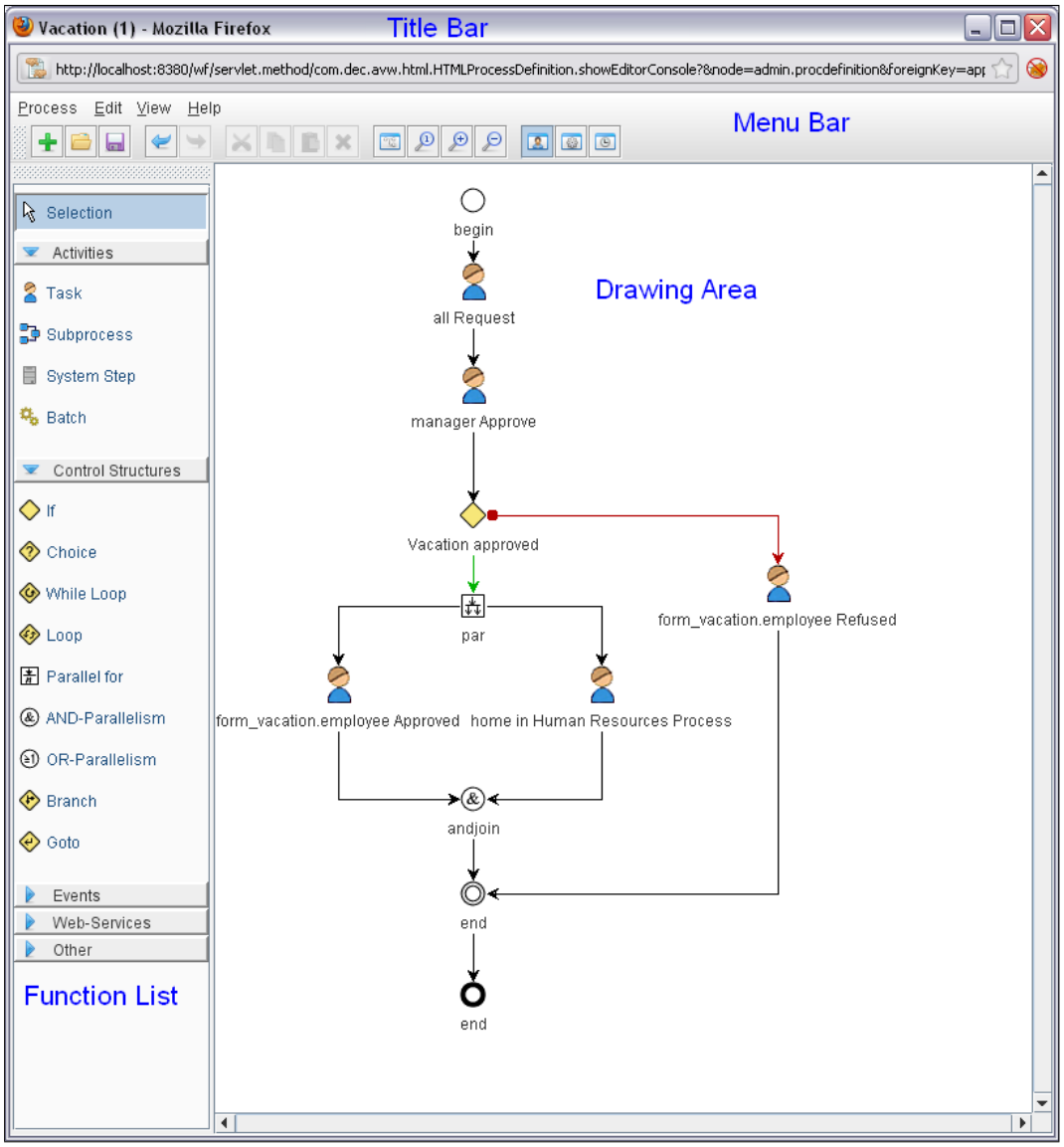


Figure 7.2: **Process Editor in WD-Notation**

- **Page Setup:** The page format dialog appears and allows the setting of paper format properties.
- **Print:** Print the process with the format properties defined in the "Page Format" menu.
- **Properties:** This function opens the process-properties (see section 7.2.3).
- **Tasks:** With the help of the task mask you specify those task which can be assigned to a recipient of a task while you are changing the agent of a task.
- **Timeout-Task:** The reaction to process timeouts is defined here (see section 7.2.5).

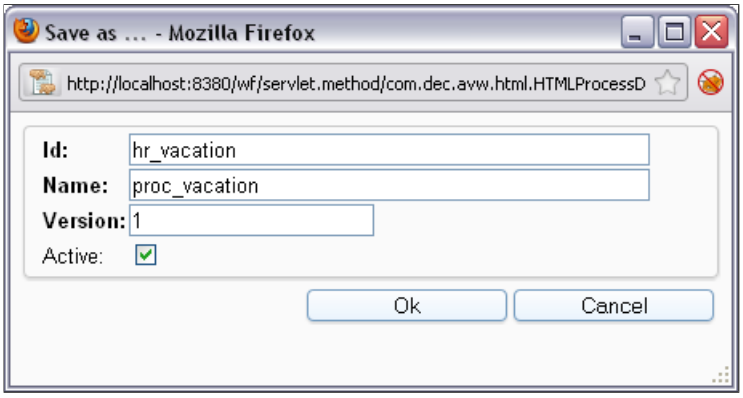


Figure 7.3: **Function "Save as"**

- **Exit:** With this function you leave the process editor. If you have unsaved changes a dialog appears which allows discarding the changes or saving them.

The *Edit* Menu

- **Undo:** With this function the last n steps can be undone in the drawing area.
- **Redo:** This function is analog to *Undo*.
- **Cut:** With this function it is possible to cut elements from a place in the process and paste it to an other place in the process. Click on the elements first and then select this function. If you cut elements, you can paste it one time only. All settings will be kept for the cut elements.
- **Copy:** With this function it is possible to copy elements from a place in the process and paste it to an other place in the process. Click on the elements first and then select this function. If you copy elements, you can paste it more than one time, but not all settings will be kept (e.g. visibility of forms).
- **Paste:** This function pastes previous cut or copied elements at the selected place. Select this function first and then click on the desired place to insert the element. The element in the clipboard is displayed beside the mouse cursor until you have pasted the element, selected another function or you have pressed the key *Escape*.
- **Delete:** This functions allows to delete individual elements. If a node (e.g. Loop) contains further elements, a popup windows appears and asks you, if you really want to delete. Click on the element first and then select this function.
- **Activity properties:** This function opens the detail-view of this task, where you can add actors and forms.
- **Task properties:** This function opens the task-properties for this activity (see chapter 6.2).

- **Time Management:** If you have activated this function for a step or batch, you can select a previous created duration statistic or you have the possibility to enter values. Furthermore an overview of the step related histograms can be displayed (see section 7.2.7).
- **Annotate:** This function allows to annotate each node in process editor (node must be selected first). The annotation will be linked with the selected node. Perform a double-click on the textfield to add a text and then confirm with *Return*.
- **Exception Handling:** This function is available for node *Outgoing Message* (INVOKE) only. It allows to add (and remove) an exception flow to this node which will be executed, if the invoke-function fails (e.g. server does not reply).
- **Additional Edge:** With this function additional edges can be added to *Choice*, *AND-Parallelism* and *OR-Parallelism*. Select one of this object first and the choose this menu point.
- **Select All:** All elements in the drawing area are selected by this function.
- **Invert Selections** This function selects all elements in the drawing area, which are not selected before.

The *View* Menu

- **Mini Map:** By activating this function a popup-window will be opened, where an overview about the drawing area will be shown. So it is possible to keep a better overview. By moving the red square, the position in the drawing area will be changed. If you want to zoom in or out in the mini map, you have to use the following explained zoom–function (changes are also shown in the drawing area).
- **Zoom:** This function contains following 3 subfunctions:
  1. *Normal viewing*: The drawing area is shown in the size, which is given at the start of the process-editor.
  2. *Zoom In*: The shown area will be enlarged.
  3. *Zoom Out*: The shown area will be reduced.
- **Align:** With this function the elements of the drawing area can be aligned.
- **Show End-Node:** This function marks the end-node of the selected element.
- **Route Automatically:** You have the possibility to remove edges, when the function *Route Orthogonal* is activated. Select the edge and move it in the desired direction. For automatically routing of the edge, select the edge and then this function.
- **Settings:** With this function you can set following properties:
  - Snap to Grid: The elements and edges will be aligned by the grid.
  - Grid Style: You have the choice between 4 possible grid styles:
    1. *Invisible:* Switch grid off.

2. *Dot Grid:* Display of dots.
3. *Cross Grid:* Display of crosses.
4. *Line Grid:* Display of lines.

– Rulers: At the top and left margin of the drawing area a ruler can be displayed. You can select between:

1. *none:* No rulers are shown.
2. *in centimeter:* The measurement of the rulers is centimeter.
3. *in inch:* The measurement of the rulers is inch.

– Show Page Borders: This function shows margins in the drawing area.

– Model new processes in BPMN: If this checkbox is active, new processes will be created in BPM-Notation.

– Route New Processes Orthogonal: If this function is active, the edges of new created processes will be displayed orthogonally.

– Hide Control Edges: This function allows to hide light grey dotted edges in process editor, e.g. the control edge of a GOTO node.

– Hide goto-Help: If this function is activated, the help–window will not appear when you insert a goto.

– Printer-Zoom: You can define the print-zoom of the process here.

– Applet Look-and-Feel: Specify the Look-and-Feel for the process editor:

1. *according to configuration:* This schema will be used, which was set under *Administration* → *Configuration* → *Localization* (see *Installation Guide* - Chapter 3).
2. *metal*
3. *windows*

The *Help* Menu

- **Help:** The help–page of @enterprise will be shown (see section 2).
- **About @enterprise Process Editor:** Shows you information about the process editor and the used libraries.

The symbol bar

You can reach the most used functions in a faster way than using the previous described menus:

- **New:** see section 7.2.2
- **Open:** see section 7.2.2
- **Save:** see section 7.2.2
- **Undo:** see section 7.2.2
- **Redo:** see section 7.2.2

- **Cut:** see section 7.2.2
- **Copy:** see section 7.2.2
- **Paste:** see section 7.2.2
- **Delete:** see section 7.2.2
- **Mini Map:** see section 7.2.2
- **Normal viewing:** see section 7.2.2
- **Zoom In:** see section 7.2.2
- **Zoom Out:** see section 7.2.2
- **Show activity properties on node double-click:** If this function is activated and you make a double-click on a node, the *Activity Properties* will be displayed.
- **Show task properties on node double-click:** If this function is activated and you make a double-click on a node, the *Task Properties* will be displayed.
- **Show time management on node double-click:** If this function is activated and you make a double-click on a node, the *Time Management* will be displayed.

The context menu

The context menu is a fast and comfortable form of handling in the drawing area. By clicking the right mouse button on an element in the drawing area the menu will be opened. The context menu includes some components of the menu bar:

- **Cut:** see section 7.2.2
- **Copy:** see section 7.2.2
- **Delete:** see section 7.2.2
- **Activity Properties:** see section 7.2.2
- **Task Properties:** see section 7.2.2
- **Time Management:** see section 7.2.2
- **Annotate:** see section 7.2.2
- **Exception Handling add:** see section 7.2.2
- **Additional Edge:** see section 7.2.2

Hint: If you want to work faster with the process editor, you can use *Shortcuts* or *Mnemonics*. The particular shortcut of a function is displayed beside the function.

7.2.3 Process Properties

On the process–properties mask you have the possibility to set properties relating to the process. The tabs are described here:

- **Common:** Analog to 6.5, but the field *Apply changes at* is not available.
- **Forms:** Here you can set the forms for the process. With a click on the button *Add* a new window appears (see figure 7.4), where you can select a form type and define some information about the usage of the form in the process. The window *Add Form* contains following information:
  - Id: You have to give the form a local id in the process.
  - Name: Here you can enter a name for the form (optional).
  - Type: Select one of the listed types for the process form. You can add additional forms by clicking on the button *New* beside the list (see section 6.4).
  - Mode: Here you can specify the purpose of the adding form.
    local: An instance of the form is created when the process is started (a process instance is created). This is the default.
    inout: The form is handed over from another process. This means that the currently edited process is used as subprocess.
  - Baseform for View: Select here the baseform for the view. The type of the form currently defined and the base form must be compatible, i.e. the form must be a view to the baseform.

  The buttons *Ok* and *Cancel* work in the usual manner.
  Use the button *Remove* to remove a form from the process. Use the button *Edit* to change the *Mode* or *Id* of the process-form. If you change an existing form-id, the id's will be replaced automatically in objects like *activity*. In structures like *If* where a condition field exists, the id must be changed manually, otherwise you will be informed when saving the process. If you use the button *Remove* or *Edit*, a form must be selected first.
- **Source:** Analog to 6.5.
- **Components:** Analog to 6.5.
- **Visibility of Forms:** Analog to 6.5.
- **Escalation:** Analog to 6.5.
- **History:** Analog to 6.5.
- **Access:** Analog to 6.5.
- **Referenced By:** Analog to 6.5.

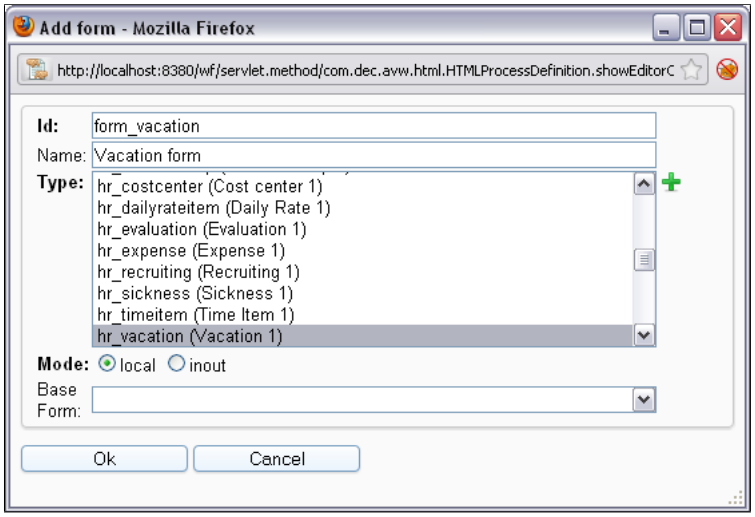


Figure 7.4: **Add form to process**

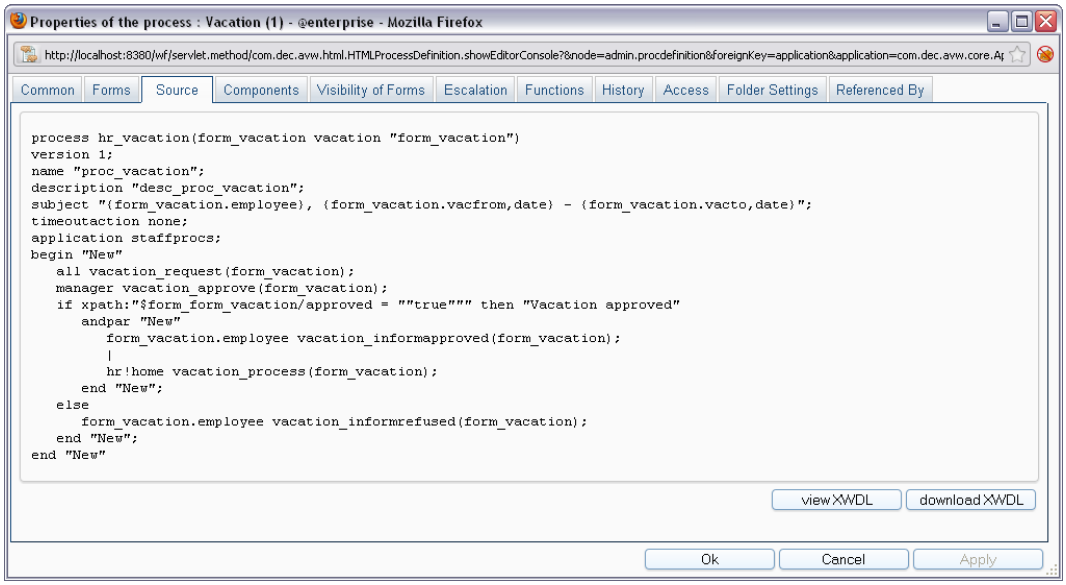


Figure 7.5: **WDL-Source**

7.2.4 Tasks

With the help of the task mask (see figure 7.6) you specify those task which can be assigned to a recipient of a task while you are changing the agent of a task. This function is not activated for the worklist by default. For this purpose add the *action key* adHoc in the gui-configuration at the node type *Worklist* → *Functions*. More details can be found in section 6.8.1.

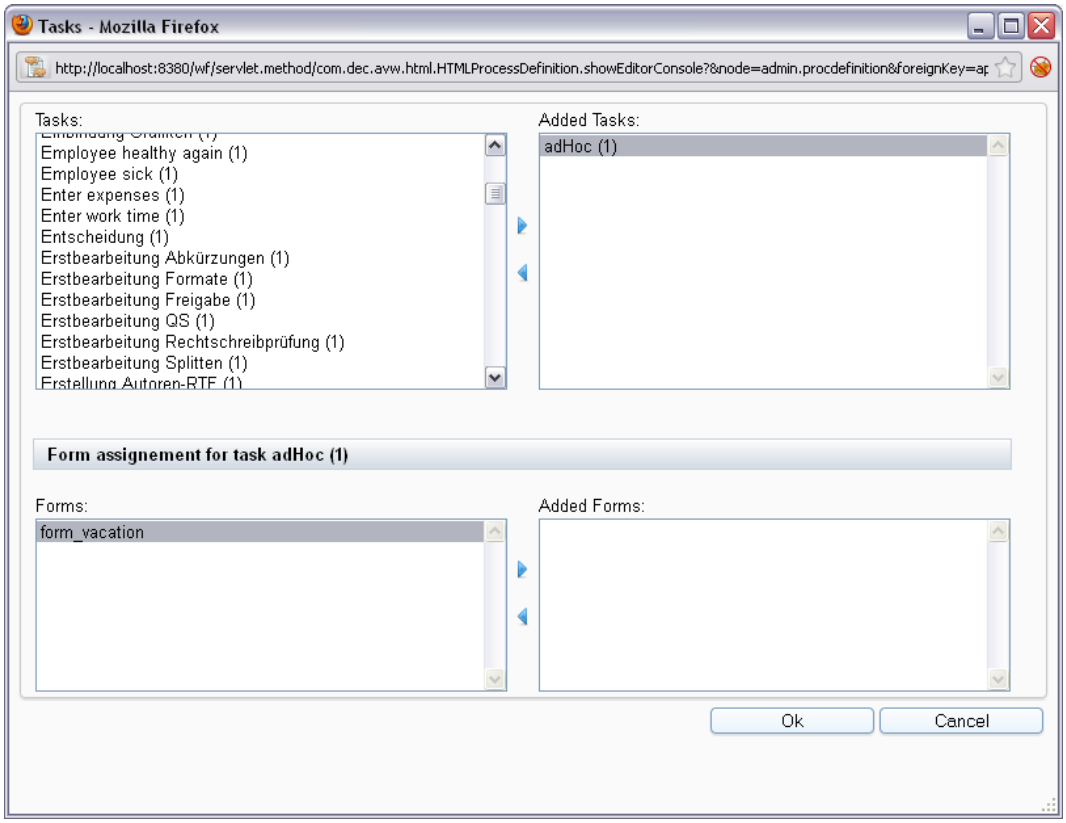


Figure 7.6: **Tasks**

Add Task

The following steps are necessary:

1. Select the menu item "Process → Tasks". The dialog of figure 7.6 is shown.
2. Select a task of the list "Tasks".
3. Click the "Add Tasks"–button. Now the added task appears in the list "Added Tasks".
4. If you want to assign a form to a task then do the following:
   (a) Select a task of the list "Added Tasks".
   (b) Select a form of the list "Forms".
   (c) Click the "Add Form"–button. Now the added form appears in the list "Added Forms".
5. Click the button "OK". Now your entries are stored in the database and the dialog is closed.

Delete Task

The following steps are necessary:

1. Select a task of the list "Added Tasks".
2. Click the button "Delete" right beside the list "Added Tasks". If you want to delete more than one task repeat the steps 1 and 2 as often as required.
3. Click the button "OK".

Delete an assigned Form from a Task

The following steps are necessary:

1. Select the task of the list "Added Tasks" to which the form you want to delete has been assigned. Now you can see the form in the list "Added Forms".
2. Select the form you want to delete of the list "Added Forms".
3. Click the button "Delete" right beside the list "Added Forms". If you want to delete more than one form repeat the steps 1 to 3 as often as required.
4. Click the button "OK".

7.2.5 Timeout Task

With the menu entry "timeout handling" you can define an activity which is started, when a process timeout occurs, i.e. the process is still running after the time span defined in max. duration (see chapter 6.5).

The following information must be provided for the definition of a timeout activity:

1. On selection of the menu item *Timeout Task* the dialog shown in figure 7.7 appears.
2. Specify the activity by inserting a task id or using the task selection window.
3. Choose an agent, which should receive the timeout activity in his worklist.
4. Add the forms necessary for the timeout activity.
5. Confirm your choices with the button *OK* or cancel the operation with the *Cancel* button.

Hint: This form of timeout-handling is available for downward compatible reasons only. Use for new created processes either the process–escalations (see section 6.5.10) for the whole process or task–escalations (see section 6.2.2) for single tasks. The process–escalations are available under *Process* → *Properties* → *Escalation*. The task–escalations can be found by selecting an activity and clicking on *Edit* → *Task properties*.

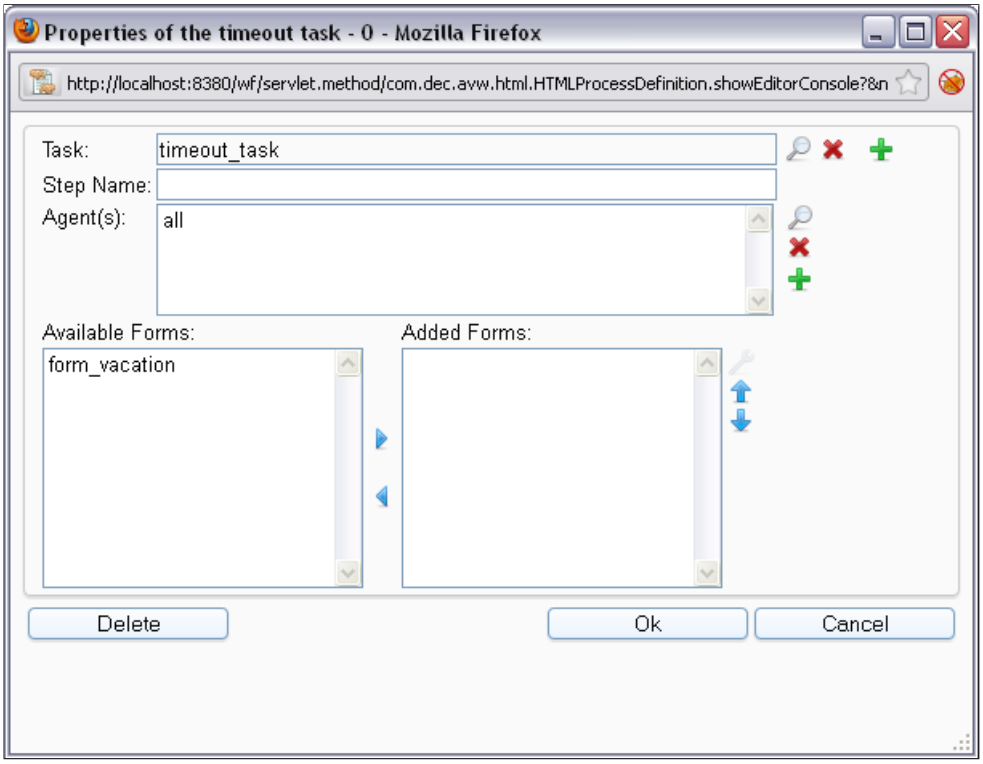


Figure 7.7: **Timeout–Mask**

7.2.6 Properties of an Activity

You can edit the properties of an activity, when you perform a double–click on the node (if function *Show activity properties on node double-click* is activated only) or click with the right mouse–button on the node an select in the context menu *Activity Properties* - the property window will appear (see Fig. 7.8).

- **Activity:** Specify the activity by inserting a task id or using the task selection window.
- Step name: Specify the name of the node which can be localized, if the value starts with @@@ and ends with @@, e.g. @@@myname@@.
- Label: Must be unique within the process and has the same syntactical conditions as a @enterprise-id.
- Icon: The icon which will represent the activity in the drawing area of the process editor. If no icon is specified here the default icon of **@enterprise** is used. An icon is handled like a resource in **@enterprise**, i.e. the icon is part of the classpath. Example: Path *lang/default/images/pred/nodes/event_register.png* shows the icon for node *Register-Event*.
- **Agent(s):** Add an agent by clicking the "+" button besides the agent list. The agent selection window appears. You have several possibilities to define an agent, the tabs on the window let you choose between them:

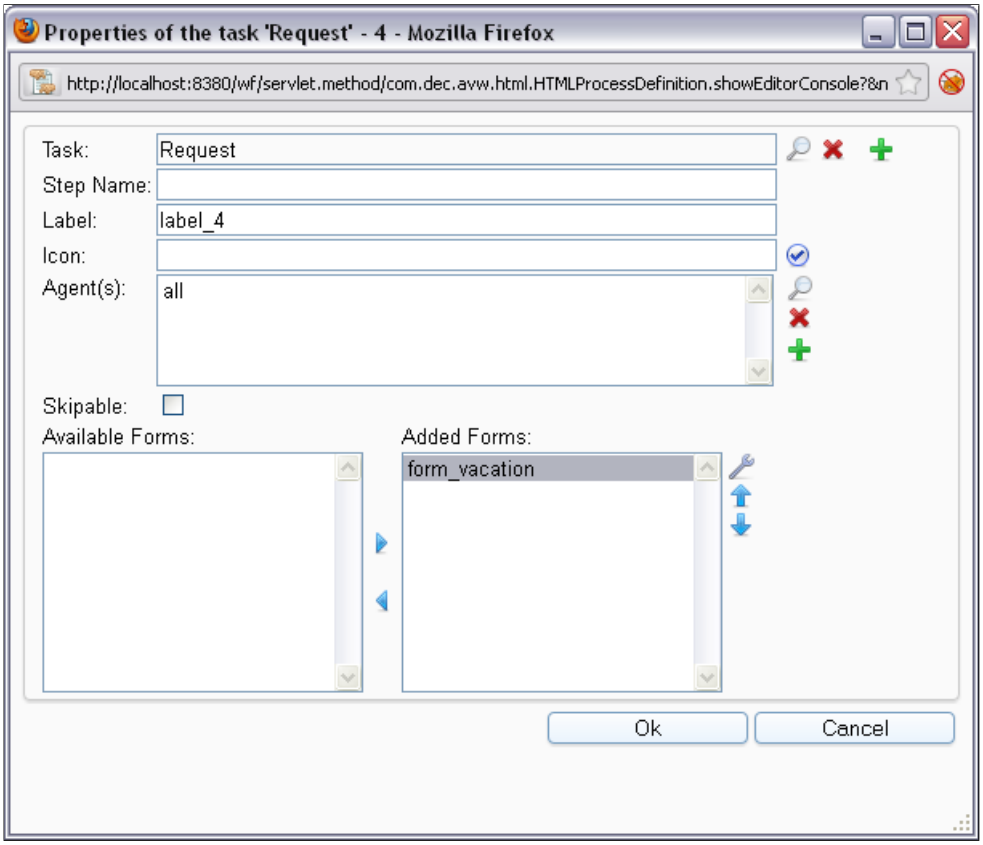


Figure 7.8: **Properties of an Activity**

- *User:* Select a user in the list and click apply. The user id appears in the agent line on the bottom of the window.
- *Role:* Select a role and click apply.
- *Task:* Select a task in the list. The agent will be set at run-time to the last agent of the selected task. Note, that you can only select a task which is performed *before* the current task.
- *Form field:* Select a form and then a field in the form. The agent is taken at run-time from the content from a field in a process form. The content must be either a role id, an user id, a role id together with an organizational unit id, or an agent of a previous step. See the WDL description for the syntax of the agents.
- *Organizational Unit:* Org. units can be combined with role and user. At run-time, the organizational unit of the current task will be set to the given OU. The organizational unit of the overall process will not change.
- *Method:* Define a JAVA method (no Groovy script). Return value must be an *Agent* or a *String* in WDL syntax.

To remove an agent, select it in the list and click the "x"-button right or the list.

- **Skipable:** When the checkbox is activated,the task is skipable, this means when no agent is set at build-time and run-time, the task is skipped.
- **Available Forms:** Add and remove process forms to/from the activity. To add a process form, select the form in the list an click on the arrow button. To remove it, select it in the "Added form" list and click the "x"-button. You can set the visibilities of a form by selecting an entry in the list of *Added Forms* and click on the Edit-icon beside this list (analog to *process*). The order of process forms can be changed by using the buttons beside the list, i.e. the form at the top of this list is displayed as leftmost tab in worklist.

7.2.7 Time Management

The time management allows to see *Duration* and *Result* about the time graph, which was created in the chapter 9.1.15. Every interactive task (activity or batch) should contain associated time graph information (TimeNode). If not, you can enter the values manually (or select a statistic from the drop–downlist) or let mining task generate it for you by activating the link *Create*. The tab *Result* contains an overview of the step related histograms. For each step and iteration there are 5 types of time histograms:

1. **Duration histogram (blue):** Shows, how long it takes to complete this step.
2. **Earliest possible start (green):** Shows, when this task can be started in best case. This histogram is generated, if the parameter *Create Start Histograms* under *Configuration* $\rightarrow$ *Time Management* has been set at graph generation.
3. **Earliest possible end (yellow):** Shows, when this task can be completed in best case.
4. **Latest allowed start (orange):** Contains time information about last allowed time (from process completeness perspective). Negative values are calculated from process deadline.
5. **Latest allowed end (magenta):** Contains time information about latest allowed end of this step (from process completeness perspective).

Positive values are given from process start, negative values from process deadline.

Each histogram can be presented in following variants:

- Normal view
- Accumulated view

The accumulated view makes it easy for user to find out such integrated values like: how long will be the duration of Y% of steps, or how much percent will be done at time X (see figure 7.9).

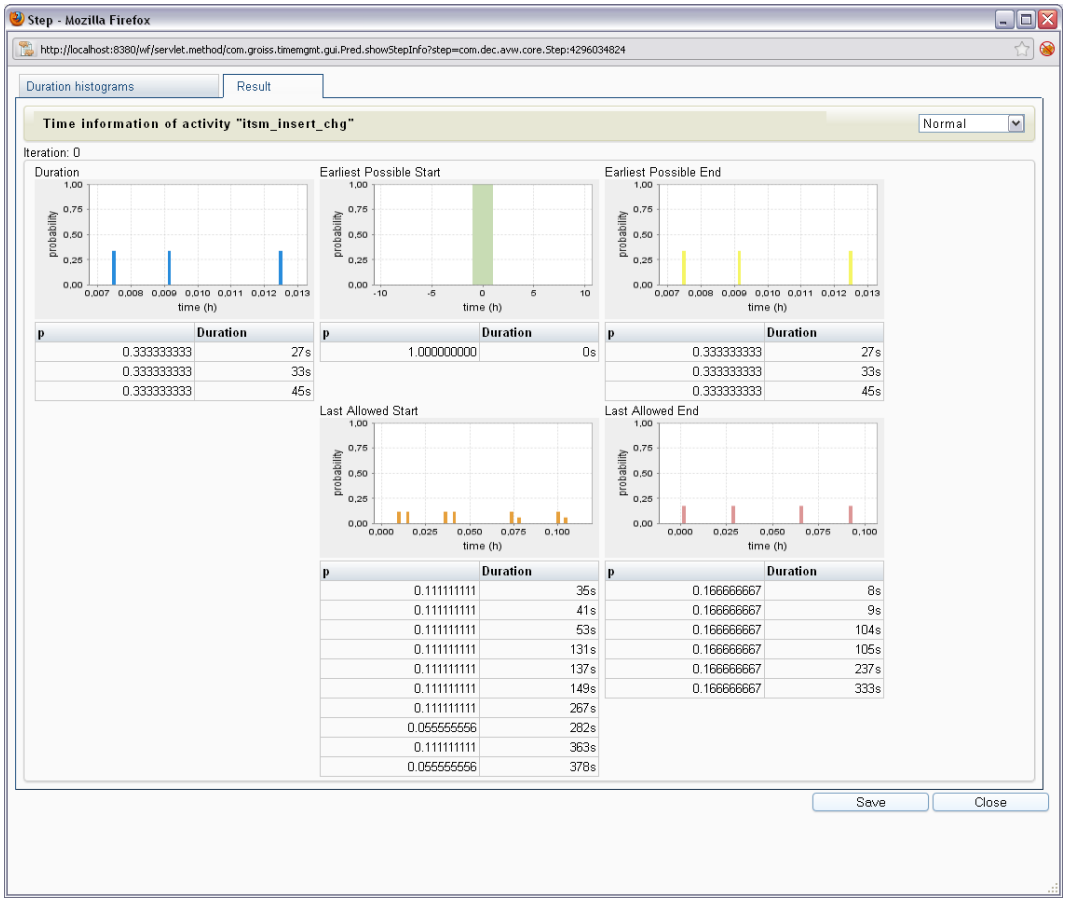


Figure 7.9: **Result of time graph**

7.2.8 The Function List

The function list contains the functions for the graphical modelling of processes.
After selection of a function you can perform the action in the drawing area of the process editor window. If the orthogonal routing is activated, the nodes can be moved only vertically or horizontally by pressing the *Shift-Key* and moving with the mouse.

- **Selection:** In this mode you can move and edit the objects in the drawing area.
- **Task:** This function allows the insertion of new activities. After selection of this function you can drop an activity on en edge in the process graph by simply clicking on this edge. A new activity will appear. On a double–click on the activity a property window for setting the activity properties appears.
- **Subprocess:** Subprocesses can be inserted in the same way as above.
- **System Step:** System steps can be created and the method to be called can be specified. Enter the fully qualified name of a Java method which should be executed in the step.

- **Batch Step:** Batch steps can be inserted, the name of a Java class (the batch adapter) can be specified. The class provides a callback interface for events during the life cycle of a batch step. For details, please consult the Application Programming Guide and the API documentation.

- **If:** The if control structure consists of two nodes, an if node and a corresponding end node. These two nodes are connected with two edges, a green and a red one. The green edge is the path followed when the condition of the "if"-node evaluates to true, the red edge is the path followed when the condition evaluates to false.

  A double–click on the "if"-node opens a window where you can edit the condition.

  If you click in the if-mode on the red edge you add an additional if-node without a corresponding end node. This control structure corresponds to the if-elsif control structure:

  ```
  if condition 1 then
    action 1
  elsif condition 2 then
    action 2
  elsif condition 3 then
    action 3
  else
    action 4
  end
  ```

  Note: Use the WDL-Script window to see how the graphical definition corresponds to the WDL script.

- **Choice:** Every choice branch has a name and an optional condition. At run-time the engine first checks the conditions of all branches, only the branches where no condition is specified or the condition evaluates to true are shown for selection.

  Insert the choice in the usual way. You see a black arrow, whereas the black arrow is a possible choice branch, where you can add activities. If you want to add alternatives, select the choice and activate *Additional path* in the menu *Edit* or click with the right mouse button on the choice and select in the context menu *Additional path*.

- **While Loop:** With this control structure you create a condition node, where the green edge goes in a loop back to this node, the red edge goes to the original following node. Activities dropped onto the green edge are the loop body. Process execution goes through the body until the condition of the while node becomes false.

- **Loop:** The loop control structure consists of two nodes, the loop node, and the exit node. The exit node is a conditional node, so two edges leave this node: The red one goes back to the loop node, the green one goes to the original follower.

- **Parallel for:** The parallel for control structure consists of two nodes in WD-notation, the parfor node and the end node. In BMP-notation this control structure is represented as BMPN-subprocess. If you click within the *Parallel for* border, but not on an element, the whole frame will be selected. In this case e.g. you can move the

whole *Parallel for*-structure or delete it. A double–click opens the same dialog as double–clicking on *Parallel for* start-node.
See the section about "Parallel For" in the WDL chapter of this book for an example of an parfor.

- **AND-Parallelism:** With this control structure you can create parallel process execution paths. Between the nodes "par" and "andjoin" several paths can be created. To add alternatives, select "par" and activate *Additional path* in the menu *Edit* or click with the right mouse button on "par" and select in the context menu *Additional path*. See the section about parallelism in the WDL chapter of this book for an example of an andpar.

- **OR-Parallelism:** Works like the AND-Parallelism above, the only difference is at run-time: The process execution continues after one parallel path has been finished.

- **Branch:** The branch allows to add an additional path which is processed independently by the main process flow. For example the main process flow is finished, but the branch can be processed furthermore.

- **Goto:** Use the goto function to jump to an arbitrary node in the process structure. For inserting a goto do the following: Activate the goto function by clicking it in the function list. Click on the edge where the goto should start. Then take the arrowhead of the drawn through line and put it by pressed left mouse button to the destination node and leave the left mouse button. The dashed line from the goto shows the original way of the process. If the drawn through line shows on an activity, the label of this activity is shown in the detail view of the goto. In the detail view of a goto you can set the *Target Label*. If the drawn through line shows on an element in the process editor and you change the label in the goto, the changes will be accepted in the target node.

  Hint: In BPM-Notation the drawn through line cannot cross the borders of a *Parallel for*.

  Be careful when using gotos! Jumping out of and-parallelism can cause strange effects.

- **Events:** This event control structures consists of a single node which stands for a special action in the context of events. The event control structures are *Raise-Event*, *Sync-Event*, *Register-Event* and *Unregister-Event*. See the section about "Events" in the WDL chapter of this book for an example of an event.

- **Web services:** In this area following nodes can be selected:

  – Outgoing Message (INVOKE): If this node is selected, the defined web service is called during run-time and the appropriate data will be submitted. If this action fails and an *Exception Handling* has been defined, the exception flow will be performed (see definition of exception handling in section 7.2.2).

  - Incoming Message (RECEIVE): If this node is selected, it will be waited for data of the (previous called) web service. If data are received, they will be processed according to the definition.
  - Reply Message (REPLY): If this node is selected, a reply message will be send when node *Incoming Message* has been processed successfully.

  The properties of each node are described in section 7.2.15.

- **Annotation:** If you have selected this function, you can add a textfield at any place in the drawing area. Perform a double-click on the textfield to add a text and then confirm with *Return*.

7.2.9 Conditions for Ifs, Choice, Loops

Perform a double–click on the node, the property window will open. See section 7.1.5 for the syntax of conditions.

7.2.10 Properties for System Steps

Perform a double–click on the node, the property window will open. Insert the full-qualified method name, including the optional parameters in the input field of this window.

7.2.11 Properties for Batch Steps

Perform a double–click on the node, the property window will open. Insert the full-qualified class name of the BatchAdapter, including the optional parameter in the input field of this window. The execution of the batch steps can be modified using the checkboxes. Details can be found in the Applications Programming Guide and in the API documentation.

7.2.12 Properties of a Subprocess

Perform a double–click on a subprocess and a property window opens, where you can select the process and the forms handed over to the subprocess.

- **Process:** Specify the process by inserting a process id or using the process selection window.
- Node name: Self defined name for this node which replaces the default name. Can be localized, if the value starts with @@@ and ends with @@, e.g. @@@myname@@.
- Icon: Path for displaying an alternative icon which is a resource in **@enterprise** classpath.
- **Available Forms:** Add and remove process forms to/from the subprocess. To add a process form, select the form in the list an click on the arrow button. To remove it, select it in the "Added form" list and click the "x"-button.

7.2.13 Properties of an Event

There are four following events:

- Raise: raise an event.
- Synchronize (Sync): stop process execution and wait for an event.
- Register: register for a certain event.
- Unregister: unregister for a certain event.

Perform a double–click on an event and a property window opens. where you can edit the following properties of the event:

- **Event Name:** the event name.
- **Event Handler:** a Java–class implementing the interface "com.groiss.event.EventHandler".
- Context: the context object.
- Step Name: The name for this node which can be localized, if the value starts with @@@ and ends with @@, e.g. @@@myname@@. If nothing is entered, the default step name is used.
- Label: Must be unique within the process and has the same syntactical conditions as a @enterprise-id. The label is relevant for process escalations of type *Sync unfinished* (see section 6.5.10).

By clicking the button "Ok" your entries are stored and the current dialog will be closed.

By clicking the button "Cancel" your entries are discarded and the current dialog will be closed.

7.2.14 Properties of a Parallel For

Perform a double–click on the "parfor"–node and a property window opens, where you can edit the following properties of the parfor statement:

- for each Subform in: If this radio button is checked the parallel for statement is executed for the sub form entries of a form, like it is described in the WDL sub section (see case one of Parallel For under 7.1). Select the appropriate subform (Mainform.Subform-Id) from the dropdown-list.
  - Form Id within the Loop: The id of the selected subform within the parallel for construct.
  - Form Name within the Loop: The name of the subform within the parallel for construct.
- Iterator: If this radio button is checked the parallel for statement is executed for the specified class, like it is described in the WDL sub section (see section Parallel For under 7.1)

- Label: Must be unique within the process and has the same syntactical conditions as a @enterprise-id.

Furthermore it is possible to define a method in end-node of parfor (see section *Parallel For* in chapter 7.1).

7.2.15 Properties of Web service nodes

Select a web service-node and perform a double–click on the node to open the appropriate property window. For each node you can define a *Step name* which can be localized, if the value starts with @@@ and ends with @@, e.g. @@@myname@@. If nothing is entered, the default step name is used. Furthermore a *Label* can be defined which must be unique within the process and has the same syntactical conditions as a @enterprise-id. The label in node *Incoming Message* is relevant for process escalations of type *Receive unfinished* (see section 6.5.10).
Following properties can be defined for node *Outgoing Message*:

- **Webservice operation:** Select an existing web service client operation which was created previously (for this application). See section 6.10.1 for more details.
- **Address:** Select an address to call the web service. You can choose between reading the address from WSDL-file, enter a XPath-expression or enter an URL.

  Example for XPath-expression: Read from configuration parameter of application *myappl*:

  ```
  string($configuration_myappl/property[@name='webservice.address'])
  ```

- **Out-parameter:** Here you can enter a list of parameter as XPath-expression which should be submitted. The parameters are defined in WSDL-file and has been defined during the creation of the web service client.
- **In-parameter:** Analog to Out-parameter, but for data which should be read from web service.

Following properties can be defined for node *Incoming Message*:

- **Webservice operation:** Select an existing web service server operation which was created previously (for this application). See section 6.10.2 for more details.
- **Start process:** If this checkbox is activated and this node is the first step in the process flow, a new process instance will be started. If this checkbox is no activated in this case, no instance can be created. If this node is not the first step in process, this checkbox must not be enabled!
- **In-parameter:** Here you can enter a list of parameter as XPath-expression which should be read. The parameters are defined in WSDL-file and has been defined during the creation of the web service server.

- **Correlation parameter:** Here you can enter a list of parameter as XPath-expression which has been defined during creation of the web service server. A mapping can be defined to assign automatically an *Incoming Message* to a process instance.

Following properties can be defined for node *Reply Message*:

- **Webservice operation:** Select an existing web service server operation which was created previously (for this application). See section 6.10.2 for more details.
- **Out-parameter:** Here you can enter a list of parameter as XPath-expression which should be submitted. The parameters are defined in WSDL-file and has been defined during the creation of the web service server.

By clicking the button "Ok" your entries are stored and the current dialog will be closed.
By clicking the button "Cancel" your entries are discarded and the current dialog will be closed.

For more information about the wdl-syntax please take a look in section 7.1.

8 The Search of @enterprise

8.1 Standard Search

Find process instances, see the User Manual for a description of this function.

8.2 Document Search

For finding documents, see the User Manual for a description of this function.

8.3 Extended Search and Stored Queries

This function offers extended functionality for finding process instances. Here you can also find statistics, which have been predefined by an administrator.Read the Monitoring Manual for details.

9 Administration tasks

9.1 Server

9.1.1 Server monitor

The server monitor provides an overview about the following server activities:

- the database connections.
- the HTTP–threads. How many threads and database connections are currently in use.
- the user sessions (HTTP–sessions).
- the memory usage.
- the number and kind of the errors occurred so far (log file content).
- date and time when the server has been started.
- the number of server requests since the server has been started last time (both RMI and HTTP).
- the number of carried out database statements.
- the content of log files.
- the system properties and Thread-Dumps
- additional server informations like version, configuration, licence etc.
- the number of active process steps (tasks) in the whole system (worklist-cache must be started or activated)

User-Sessions

With the help of the administration function *User Sessions* it is possible to get information about the logged in users and when they was logged in.

When a user is logged in, the number of the logged in users will be checked with the licence (Concurrent-User). If the login is possible, an user-session will be generated. This user-session is as long as valid, until the user activates the *Logout* button. If no logout happens,

the user-session is valid for 24 hours and will be finished automatically. Only user-session which are inactive less than 4 hours, will be checked with the licence.

You can display the user-sessions as user list or in form of a histogram. Further you can set the time horizon for better display.

User List: For displaying the user list, you have to take further restrictions.

- **Logged in users:** All user-sessions which are active.
- **All users:** All user-sessions, also inactive sessions.
- **User:** All user-sessions of the selected user.

With option **Last access (minutes)** you can define (depending on selected *User list* option) which users should be displayed where last access was done x minutes before.
You will find following information in the result table:

- **User:** Contains the first and last name of the user.
- **Client IP:** The IP-adress of the user.
- **Date of Initialization:** The initialization-date of the user-session (login date of the user).
- **last access:** The date, when the user was active in the system. By activating the link (only visible when a thread is running) a new window opens, where you can see details about this activity (inter alias HTTP–sessions).
- **Date of Logout:** You can see the date of logout or the link *Logout*, when a thread is running. This link kills the session and the user will be logged out (can be killed clusterwide).

Histogram:

- **hour:** The time interval starts with 0 minutes..
- **day:** The time interval starts at 12pm.
- **week:** The week starts with the start-day of the time horizon at 12pm.

9.1.2 Server Control

Here you can control the server with the following functions:

- **Shutdown server:** Shut down the **@enterprise** Server.

- **Restart Server:** Restarts the **@enterprise** Server.
- **Show patch files:** Checks the local system, if a patch is needed and displays all possible changes.

  Further information regarding the patch-mechanism can be found in the installation manual.

- **Execute Database Upgrade:** Use this link to check whether your software installation is consistent with the database (the version numbers are compared).

- **Reload Database Connections:** All database connections (of the current node) will be marked as "old". Before assigning a connection to a transaction the "age" will be checked first, then old connections will be closed and new ones opened.

- **Initialize Log File:** When activating this function the current log file will be closed and a new log file will be created. Use this function if you want to record some events in a log file.

- **Reload Configuration:** This function allows to search for changes in file*avw.conf* and *appl.prop* of each application (not configuration parameter definition in XML file!) which has not been changed via GUI and load the changed values into the Configuration object. After loading the method *reconfigure()* is called for each service (and each application where application class implements the interface *com.groiss.component.Service*). More information can be found in the *Application Development Guide* - chapter *The Configuration File*.

- **All passwords have to be changed on next login**

- **Show reporting schema:** Displays the reporting schema (XML file) in a seperate window.

- **Reparse Reporting schema**

9.1.3 Events

In the section *Events* you can search for all recorded system events of @enterprise:

- **From** and **To:** Here you can set time restrictions
- **Type:** The event type *startup* or *shutdown* can be selected
- **Search:** This function searches all recored system events, depending on the search-options. If no option was set, all recorded events will be displayed in a table.
- **Delete all:** With this function you can delete all recorded events which are displayed in the table.

9.1.4 Worklist-Cache

In the cache administration the functions **Update**, **Refresh** and **Check Cache** (each of them described below) are available.

The engine constructs the worklists via heavily cross-linked in-memory structures. Database operations are hardly ever invoked.

The worklist cache can be in one of three states:

- **Switched Off:** In this state, the cache is completely switched off. It does not consume any memory, the worklists are still constructed via the previous DB-intensive approach.
- **Started:** This state is an intermediate state in which the in memory cache structures are build and maintained, but where the worklist operations do not actually use the cache. This mode of operation might be useful to compare the worklists with and without cache.
- **Activated:** This is the full operations mode of the worklist cache component. All suitable worklist construction operations use the memory cache structures[1].

The initial state of the worklist cache can be configured via the server configuration in section tuning.

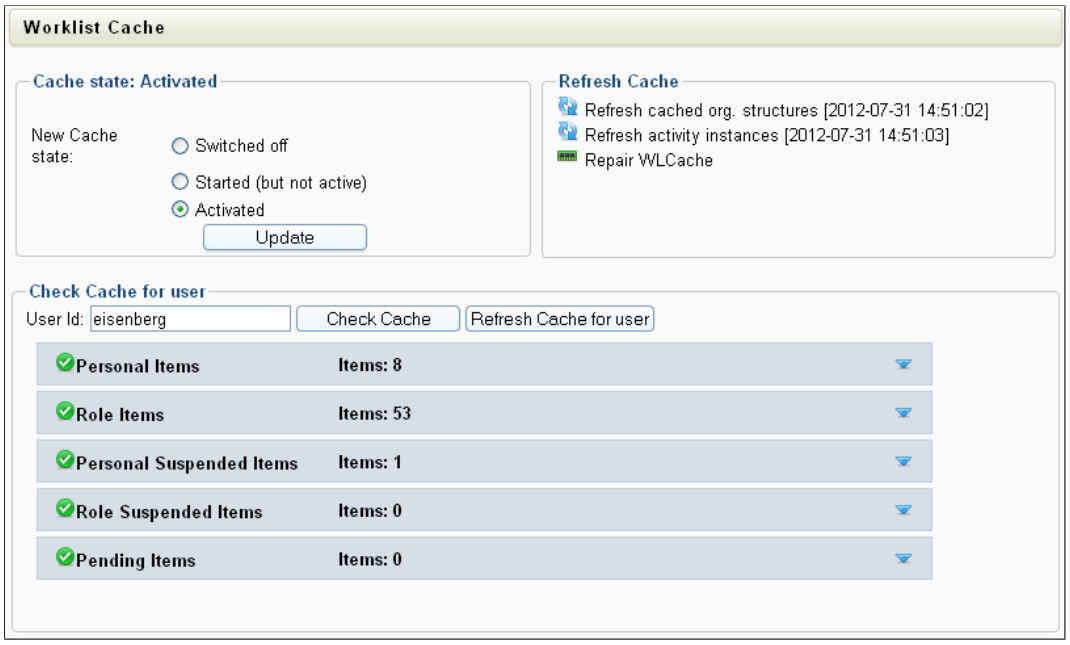


Figure 9.1: **Worklist-Cache**

Update

With the help of this function you can change the state of the cache to one of the three states described above. The state transition from off to started or to activated is relatively expensive in terms of time. All other state transitions (in particular between started and activated and vice versa) are quite fast.

[1] A notable exception are worklist methods which make use of SQL-WHERE-clauses via the API. Such calls will consult the database in each case, independent from the particular state of the worklist cache.

Refresh Cache

A refresh of cache structures is needed in the following cases: new applications, new departments, changes in the department-tree, new roles.
For this purpose following functions are available which can be executed manually:

- *Refresh cached Org. Structures:* With the help of this function you can refresh the organizational structures of the cache.
- *Refresh Activity Instances:* This function refreshes the Workitems. This would be also accomplished by switching the worklist cache off and then on again.
- *Repair WLCache:* If there have been inconsistencies in the worklist cache of one Node N due to "stop the world" garbage collection pauses, they can be repaired with this function. By entering a time interval (e.g. start and end of the GC-Phase of the inconsistent node) and by selecting another node M, node N gets information from node M which step instances changes during the interval. Nod N uses this information to update its internal state to the current data base state of those step instances.

For each function the timstamp of last execution is displayed within the brackets.

When a user logs in, his current roles and substitutions are accounted for. So changes in the assigned roles of the user (or the users he substitutes) are reflected after the login.
Changes in the substitutions are accounted for immediately after the changes (without the need for the substitute to log in again). This is the case for manual changes of substitutions as well as for changes made by the CurrentSubstitutesTask because the period of substitution starts or ends. Please note that the CurrentSubstitutesTask must be set to active in the Timer administration.
During a refresh of the cache structures, some of the structures are instantiated twice (the old and the new version). So additional memory usage during cache refresh and after it should be expected until garbage collection kicks in.
Two methods are available to refresh cache structures:

- com.dec.avw.wlcache.WLCache.getInstance().refresh() This takes into account all organizational changes. Corresponds to the link "Refresh" in Administrative Tasks / Cache Administration. Use e.g. after importing a batch of users programatically.
- com.dec.avw.wlcache.WLCache.getInstance().refreshUser(User u) This function considers changed roles and changed substitutions for one user. It does not take into account new applications, departments, depttrees, roles, ...

Check Cache

With the help of this function you can check the cache consistency for a certain user. The system compares the contents of the users personal worklist, role worklist, suspension list, role suspension list and pending items list according to the worklist–cache to the contents of the corresponding lists according to the database state.

If no discrepancies could be detected, the lists will show an icon in the form of a green tick. In case of discrepancies, the affected lists are marked with a red cross and the offending items are displayed. The administrator can then fix such discrepancies by clicking the provided Update-links or using the button *Refresh Cache for User* which refreshes the whole cache for the current user. The changes are reflected immediately, that is the worklist cache is updated with the latest state of the stepinstance in the database. Display refresh must be triggered explicitly by the administrator.

9.1.5 Class Path

For the convenience of application programmers we support the reloading of classes. A precondition for doing this, is to distinct between system classes and application classes (and resources). System classes are loaded by the system class loader and can not be unloaded. They reside in the lib directory of the installation.

The application classes are in the lib and classes directories of the applications. The form classes generated by the system are in the forms directory. These classes are loaded from the application class loader and are reloadable.

To enable class reloading check the checkbox in the configuration (parameter group tuning). To check your classpath use the Classpath link in the administration.

Show shadowed classed

This method lists all resources, which name is found more than once in the class path or in subfolders of the class path entries. resources, which are found more than once, are shown in the following syntax.

relative path of resource Number of found resource with this relative path
Absolute Path of the resource used by the system
Absolute Path of shadowed (unused) files
...

9.1.6 Timer

The timer triggers time-controlled events. It is used for some system tasks but also open for application timers. If you click the "Timer"-Link you see the list of timers already defined. You can add entries or change the properties of existing entries in the usual manner. Furthermore the toolbar-function *Execute* is available which allows to execute the selected timer.

The object-details of timer contain the following tabs:

- General
- Access
- History

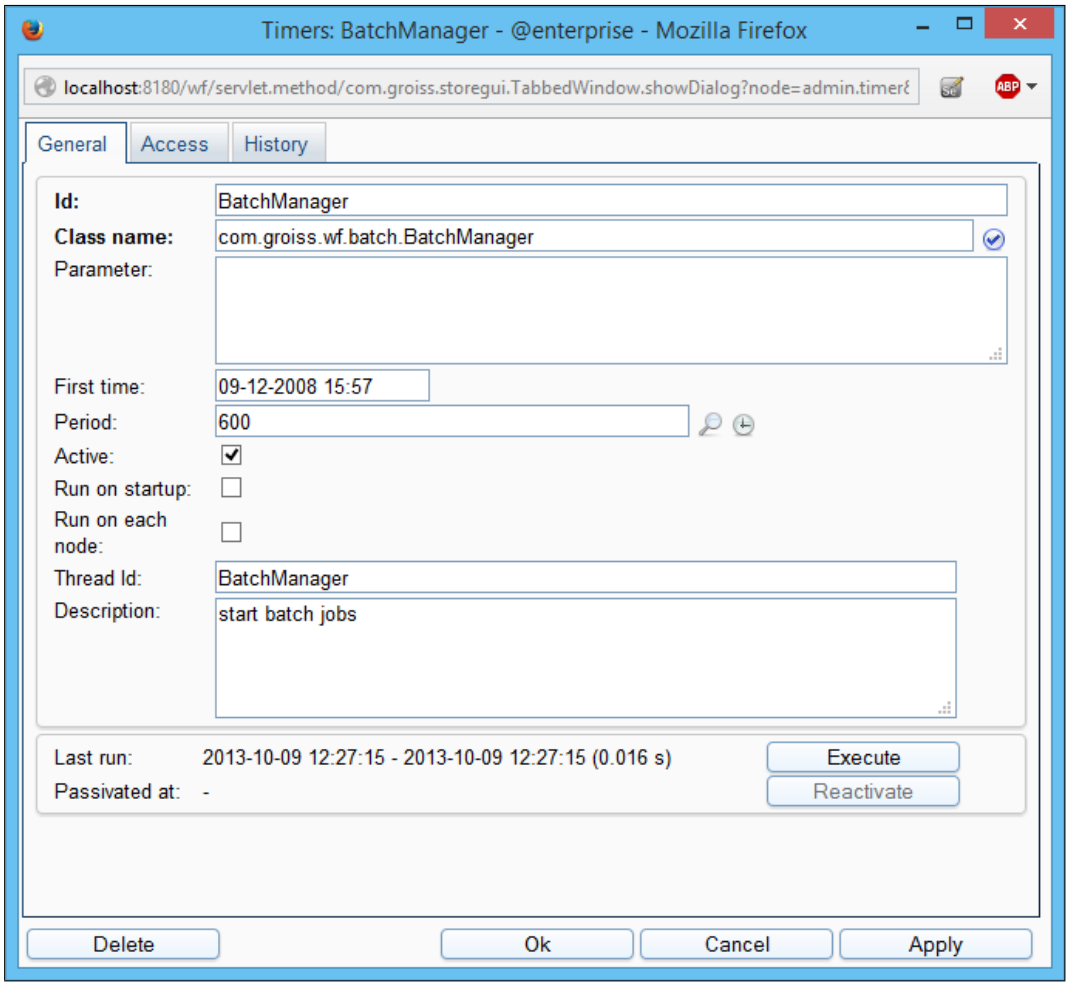


Figure 9.2: **Tab: General (Timer)**

Overview of Standard–Timers

Standard-timers are:

- **ArchiveTimer:** Archives finished processes. For more information see the *Installation Guide*.
- **BatchManager:** Starts and finishes batch jobs. Only needed, when batch job steps are used in process definitions.
- **CalendarReminder:** Checks, if there are any calendar entries which specified reminder time is reached and sends mail notifications for those entries. Keep it switched on, if the DMS is used.
- **CleanUpDMS:** Deletes empty directories in the checkout area and also deletes ACLs, which where DMS-object specific, but are now unreachable. Keep it switched on, if the DMS is used.

- **ClusterCheck:** Checks whether other nodes are running and reassigns cluster timer. This timer is only needed, when using the @enterprise cluster. For more information about @enterprise cluster and related times see the *Installation Guide*.
- **CurrentSubstitutes:** Checks, if some substitution specifications have to be enabled or disabled due to the time periods specified at those substitution. Needed when user or role substitutions are used.
- **DeferredUpdate:** On each run this timer takes a look, if there are any deferred updates of master data for which the time to execute has been reached. And if so, those updates will be performed by this timer. Keep it switched on.
- **DeleteUserSessions:** This timer deletes user sessions which are expired. The parameter *avw.keep.user.sessions* defines the duration (in days), how long a user session should be active.
- **Escalations:** This timer checks on each run, if there are any timeout tasks to start or any escalations to fire. For detailed information see section 6.2.2 or 6.5.10. Needed when any escalations or timeouts are used in the process definitions.
- **Expiration:** In BPEL it is possible to define an expire date for an activity (not a task). If this expire date is reached, the activity will be finished automatically even when this timer runs.
- **HeartBeat:** Informs the cluster that this node is alive. This timer is only needed when using the @enterprise cluster. For more information about @enterprise cluster and related times see the *Installation Guide*.
- **IndexRefresh:** Refresh the full-text search index in ORACLE. This timer is only needed, if you use full-text search under ORACLE.
- **LDAPDirSyncTask:** Synchronizes with LDAP Directory Servers. For detailed information see section 9.5.2. Needed when periodic LDAP synchronization is configured.
- **Log:** This timer will remove all log entries (excepting the current log entry/the last change) which are older as specified in *Configuration* → *Localization* → *Keep object changes (days)*. An alternative way is to use the field *Parameter* of the timer which overrules the configuration parameter. If the timer parameter is a positive integer D, then all log entries older than D days will be removed. If the timer parameter is a property string, the retention period can be specified for individual classes. The property string consists of elements of the form classname=Dn (separated by line break). If zero is used as Dn, then the classes log entries will not be removed. If * is given as a classname, the corresponding Dn parameter applies to log entries for all classes not explicitly mentioned in the property (all other classes). e.g.

  ```
  *=30
  com.groiss.org.User=0
  com.groiss.org.OrgUnit=1000
  com.groiss.org.Role=1000
  com.groiss.org.UserRole=365
  ```

- **MailGetter:** Download mails and perform the configured actions. For detailed information see section 9.5.1. Only needed, if any mailbox contents should be processed automatically.
- **SeenObjectCleaner:** Removes all see-information which is not needed anymore. The seen-information is used to indicate, if a work item is new (=unseen) or not. Keep it switched on.
- **Suspension:** This timer will investigate all suspended work items, if it is time to see those items again in the various worklists (i.e. it performs a time triggered automatic 'see again'). Keep it switched on.
- **WfXMLTask:** Sends WfXML messages from outgoing buffer and gets messages from passive partners. This timer is only needed, if WfXML is used. For detailed information see section 9.5.4.

Tab: General

You can edit the following attributes (required fields are bold):

- **Id:** Short name of the entry.
- **Classname:** Name of the class which contains the timer action. The class should implement the interface com.groiss.timer.TimerTask
- **Parameter:** A String parameter for the "run" function.
- **First Run:** The time of the first run.
- **Period:** Interval in seconds or in form of cron-pattern (see section 9.1.6).
- **Active:** Only when checked, the timer task is performed.
- **Run on Startup:** When checked, this timer task is started on startup.
- **Run on every Node:** The timer is running locally on every node of the cluster.
- **Thread Id:** If you specify a non-empty string, the string is used as thread identifier. All timers with this string as thread id are executed in the same thread. Default is, that all timers are executed in one thread.
- **Description:** Free text.
- **Last Run:** Shows, when the timer had its last run (start- and end-time) and the duration (in seconds).
- **passivated at:** Time, when no connection to the server existed and the timer was set temporarily inactive.

Activating the button **Execute**, executes the actual timer immediately.

Activating the button **Reactivate**, releases the timer of the passive status.

By clicking on this symbol a popup-window will be opened, where you can enter the period in seconds or in form of cron pattern (see section 9.1.6).

If you click on this symbol, a popup-window will be opened, where the next five invocations are shown.

Cron-Pattern

The cron-pattern comes from the UNIX-world and is used for tasks, which should be executed automatically in recurring intervals.
@enterprise uses this pattern to start timers as desired. **@enterprise** adheres to the V7-standard of cron.

A row consists of five defined columns. These columns contain the time data (minutes, hours, days, months, weekdays), whereas the columns are separated by spaces. The entries for the time data are shown in the following table:

Minutes	0-59 and * for all minutes
Hours	0-23 and * for all hours
Days	1-31 and * for every day
Months	1-12 and * for every month
Weekdays	0-7 and * for every weekday (0 and 7 for Sunday)

Furthermore cron offers following advanced functions:

- A comma , allows more time data
- A hyphen - specifies a period
- A slash / divides into a time range

Examples:

- Every day at 9h and 15h the timer will be executed: 0 9,15 * * *
- On the 15th of every month at 09:50h the timer will be executed: 50 9 15 * *
- The timer will be executed every Saturday at 00:00h: 0 0 * * 6
- The timer will be executed every 30 minutes: */30 * * * *
- Every day from 8h to 20h the timer will be executed every 20 min: */20 8-20 * * *

For further information about cron, please take a look at *http://en.wikipedia.org*

9.1.7 Object History

View the history of the objects in the database. You can see who has changed which objects and view older versions of objects.

9.1.8 Interface Forms

You can view the list of the interface forms you have defined, see chapter 6.6. Upon click on a form you can fill it and start the associated process.

9.1.9 Pending Changes

In the administration task list you find the entry *Pending Changes* showing a list of objects having queued changes. You can also withdraw the changes in this list.

9.1.10 Event Registrations

In the administration you can view the list of registrations and you can add and remove registrations. Processes waiting in a sync can be finished manually from the process history. The following informations are displayed in the event table:

- Registrant: The id of the process registered for the event.
- Event Name: The name of the event for which the registration took place.
- Context: The context object for the event.
- Event Handler: The Java–class handling the event.

9.1.11 Manage certificates

To use SSL you need a certificate, which is stored in a keystore. First, you have to set the SSL Parameter (e.g.: keystore path, port) in the Configuration and ensure that the com.groiss.ssl.SSLHttpd is in the list of services (see *Installation and Configuration Guide - chapter Configuration*). To communicate in a secure manner @enterprise server needs a certificate which proofs the integrity of the server's public key. You may generate a self signed certificate, which covers the needs for internal communication, or request an official certificate from a certification authority (CA). Anyway the server needs a RSA key pair (public and private key), which can be generated by clicking the button "Generate self signed certificate". @enterprise generates the key pair by using the keytool, which automatically generates a self signed certificate to this key pair.

Generate selfsigned certificate

To generate a key pair, the following parameters have to be specified.

- **Alias name**: The alias name is so to say the id of the specific entry in the keystore.
- **Country**: a two-letter country code, e.g., "US"
- **Company Name**: The official name of the company.
- **Organizational Unit**: the specific department
- **E-Mail**: E-Mail address of the administrator.
- **Hostname**: The hostname of the server

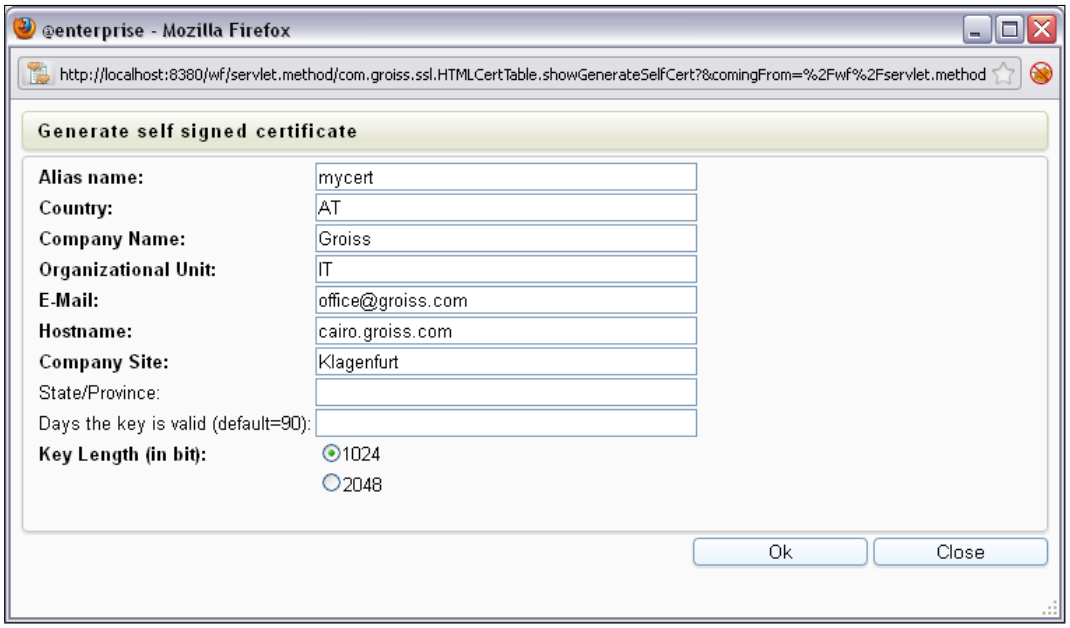


Figure 9.3: **Generate selfsigned certificate**

- **Company Site**: The city where the specific department of the company is located
- **State/Province**: The state of the company site.
- **Days the key is valid**: The key and the assigned certificates may expire. The validation time of the entry can be specified in days. The default value is 90 days.
- **Key Length (in bit)**: Can be chosen: 512, 1024 or 2048 bit of length.

Create Certification Request

Choose the entry of the keystore, which you want to use to create a CR. You can download the CR by double-clicking it or by choose it and click the "create Certification Request" button. If you have created the CR you can request a certification at a certification authority (CA). How to do this can be found in the documentation of your CA.
You can get some example certificates at "www.secude.com/trustfactory/" or "www.trustcenter.de".

Import Certificate

hen the CA sends the requested certificate, you need to import it into the keystore. To do so, click on the button “import certificates” and specify the following parameters.

- **Alias name**: Ensure that the alias name is the same as the alias of the key pair for which the request was generated.
- **Encoding of the certificate**
  According to the encoding of the received certificate, there are 2 different ways to import.

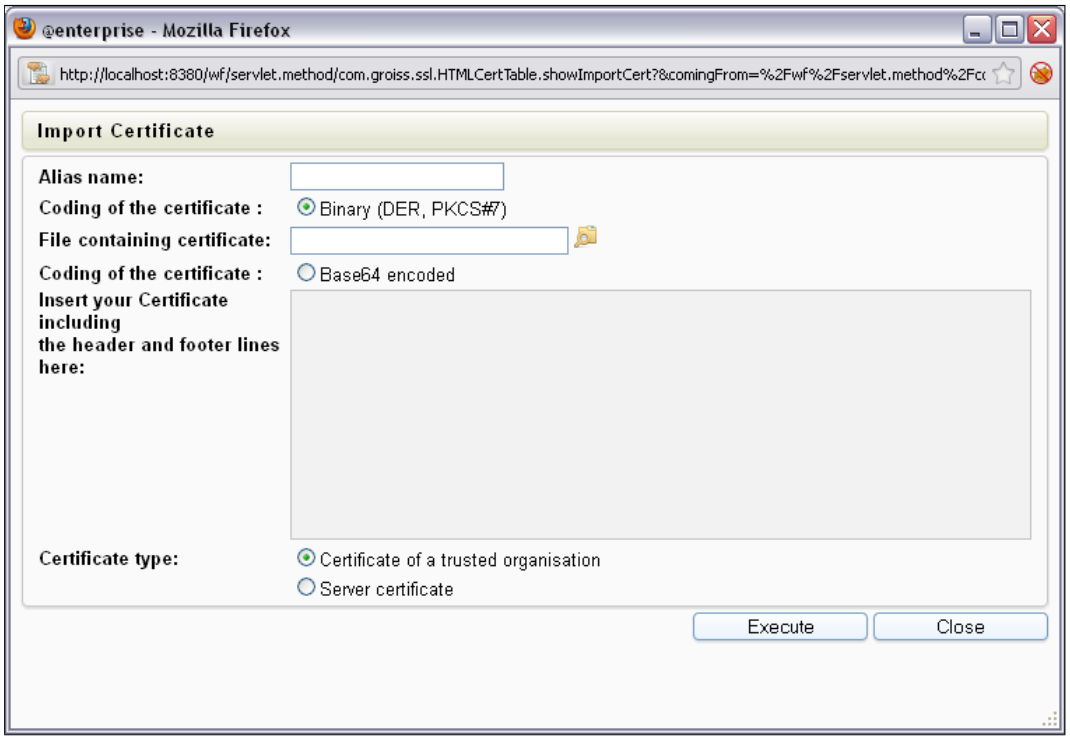


Figure 9.4: **Import Certificate**

  - Binary (DER, PKCS#7): in this case you have to specify the file, which holds the certificate.
  - Base64 encoded: just copy the certificate including header and footer lines in to the textarea

- **Certificate type**: The certificate to import can be either the certificate of the server or the certificate of a trusted organization (also called trust anchors). A trust anchor is the root of a certificate chain an is needed, if the "require client certificate" option is selected. The server accepts only client certificates, which are signed by a certification authority, which certificate is stored in the keystore as a trust anchor. If the client can not provide a certificate which is signed by one of the trustanchors in the keystore, the connection will be refused.

If any entry of the keystore is not needed any longer, you can delete the entry by clicking on the delete button.
Note: After any modification of the keystore the server needs to be restarted.

9.1.12 Running Nodes Monitor

A cluster is a set of **@enterprise** engines which share a common database schema and which are configured identically. The aim of this configuration is to provide enhanced availability and scalability. Further informations on clusters can be found in the Installation

manual.

Informations about the cluster architecture of **@enterprise** can be found in the installation guide. The attributes of a cluster and node respectively are described there also.

9.1.13 Full-Text Search

The status of the full-text search may be administrated in the system configuration. There you can activate or deactivate the full-text search and if you are activating it, you may initialize it afterwards. Initialization must be done if you want to use full-text search for all documents and forms which were last amended while the full-text search was not active. The full-text search will be available for all forms and documents created or changed after the specified date (or for all if no date is specified).

9.1.14 Query Tool

A simple interface for executing SQL-queries has been implemented. You find it in the Administration Tasks, group Server. Because direct database access may be an enormous security risk, the functionality is only available when the following two conditions met:

- The configuration parameter database.direct.access has value 1. There is no user interface to set this parameter, you must directly edit it in the configuration file.
- The user must have the execute right on all objects (every user having the sys role has this). Substitutions are not considered here.

9.1.15 Duration statistics

The duration statistics contains a list of all collected time data. By activating the toolbarfunction *New* you can collect time data for the timemanagement (see figure 9.5). The purpose of mining is to extract duration statistic information from process history. This duration statistic will be used later to generate time management run-time structures (time graph).
If you want to collect time data, you have to specify a process type, activate the checkbox *Perform Process Mining* and optionally specify a timer interval when the process instance was started. Additionally a name should be entered, so that this statistic can be found in the *Duration statistics* table. The mining gathers the duration for each task in this process and determinate branching information. After finishing of mining task, you will get a page with status information.

You can get details by selecting an entry in table *Duration statistics*, or delete some of them, that are no longer needed. The detail view of an entry contains some common information in tab *Common*, shows *Duration histograms* for each task in a process and also *Branching probabilities* (see figure 9.6). The tab *Time graph* contains a visualized representation of the collected data by activating a link (mask is the same as described in section 7.2.7). For each duration statistic it is possible to define an own implementation class which has to extend the adapter-class *com.groiss.timemgmt.DefaultTimeManagementImpl*. More details can be found in the **@enterprise API**.

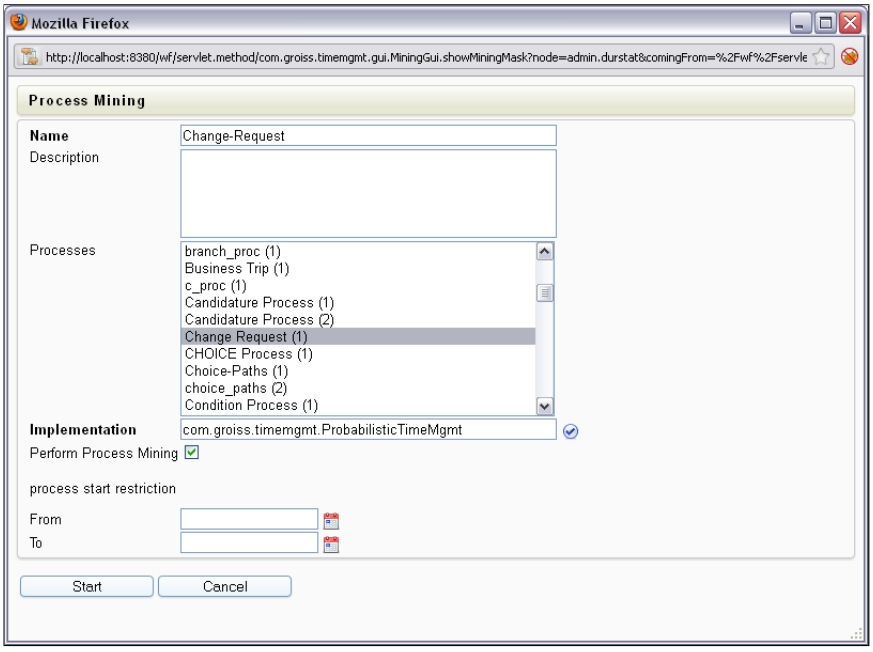


Figure 9.5: **Process Mining**

Hint: Be note that in case of an own implementation class no timegraph will be generated and the function *Regenerate the graph* is not active anymore!
The tab *Duration histograms* contains the probabilities for each task in the process. The tab *Branching probabilities* contains the probability for each branch, how often this branch was run through. In this case the process has an IF–construct, which only has one activity in the TRUE–branch (green line in the process editor). The probability of this branch is 0, because the activity was never called. The ELSE–branch (red line in the process editor) has the probability 1.

Duration statistic from mining can be used at run-time after generation of a time graph. This task is separated from mining task, because each process type is able to use only one set of time data (duration statistic). Thus you can make unlimited number of mining snapshots and decide which one you will use for run-time later.
This appropriate function is available in the detail view of duration statistic. You have to transfer the *Process Definition* from *inactive* to *active*. After that you will be asked, if you want to create a timegraph. After finishing the generation, the status logs of the generation process will be displayed (see figure 9.7). If you want to save the result of the timegraph generation, please activate the button *Save*. For regenerating a timegraph for the selected process of the list of active processes, please activate the button *Regenerate the graph*.

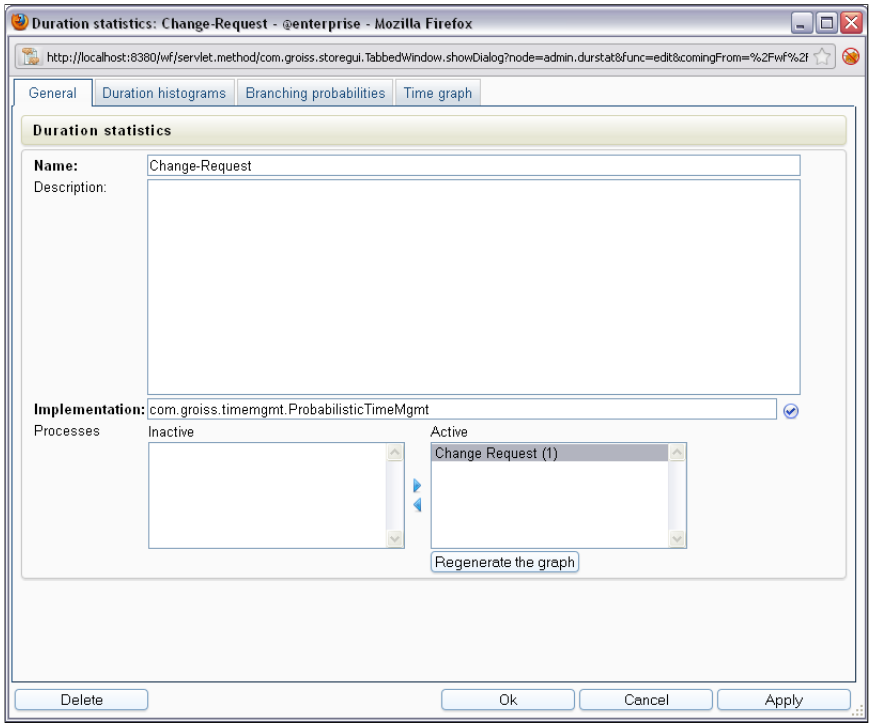


Figure 9.6: **Duration statistics**

9.2 User

9.2.1 Disable/Enable Login

This function disables the login to the server, only the users, which have the right *Configuration*, may login in this mode. Other users receive the message you provide in the message area.
There are following information availabe:

- Logins enabled: If this radio–button is activated, there are no restrictions at the login.
- Logins disabled: If this radio–button is activated, only system administrators can login after server restart. At server restart upgrade(s) of application(s) will be performed (if defined). For further information about upgrading applications, please take a look in the API of @enterprise (*ApplicationAdapter.getVersion()* and *ApplicationAdapter.upgrade()*).
- Message: If the login is disabled, you can enter a message here, which will be shown, when a user logs into the system.

9.2.2 Permission Test

With the help of this function you are able to detect if a certain permission has been assigned to a certain user. The informations of the corresponding HTML–page are described in

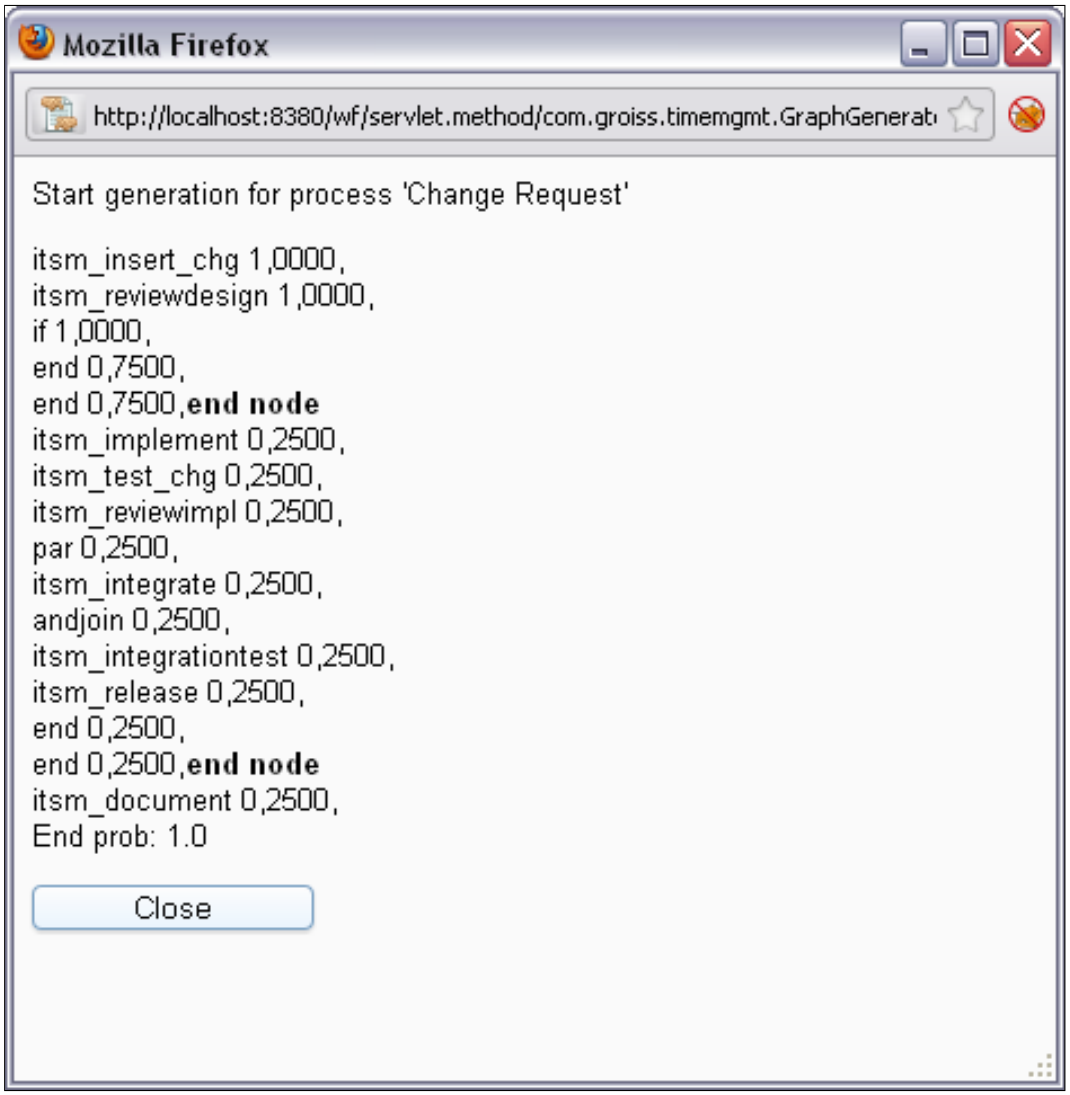


Figure 9.7: **Status of generation process**

detail in chapter 5. By clicking the button "Test" the system checks wheter the user has the permission or not and the result is displayed.

9.2.3 Expired passwords

If the password policy defines, when passwords are expired, the administrator can check, which users have expired passwords.

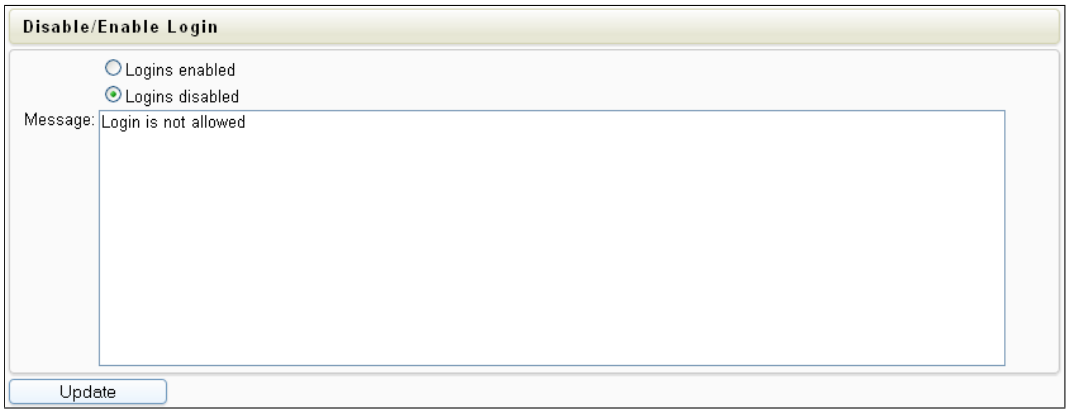


Figure 9.8: **Disalbe/Enable Login**

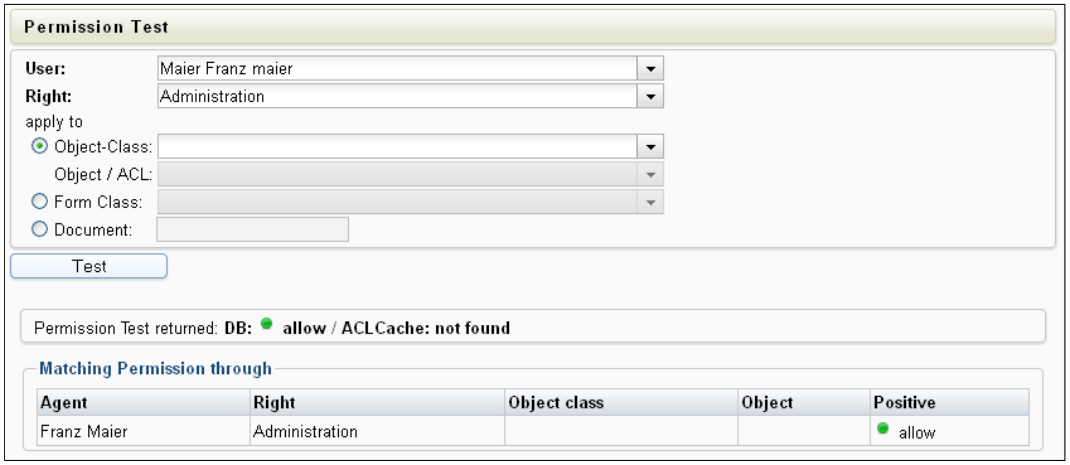


Figure 9.9: **Permission Test**

9.3 Import/Export

9.3.1 Import/Export in XML Format

The import/export functions allow you to export data (master data and runtime data) from one **@enterprise** installation and import it to another. The data are exported to a file in XML format.

Export

You can export different types of data. XML Export shows you a list of all exportable data types and lets you choose from options depending on the chosen data type. Figure 9.11 shows the available export types. You can export only one type of data at a time. If the selected type has additional options to choose from, an option section will become visible (like for organizational units, as you can see in the figure). The first element of the export

6 users with expired passwords found!

Id	Name
arb	Andreas Reichenberger'
bush	Bush
danet	Danet Admin
mobi	Mobi
user_with_all_rights	Benutzer mit allen Rechten
user_without_rights	Benutzer ohne Rechte

Figure 9.10: **Users with expired Passwords**

screen is the "Export Description" text area - you can use this to optionally add a description to the export file. If you import the export file later, the description text will be displayed.

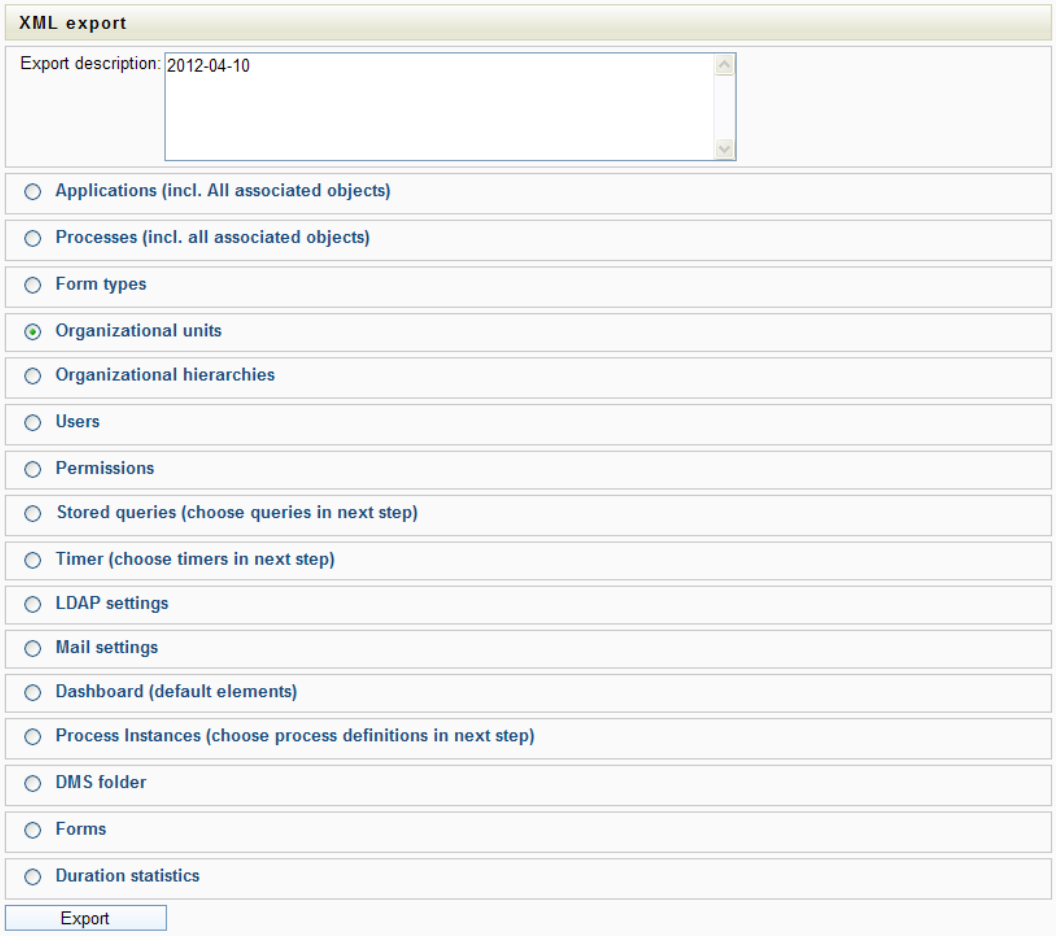


Figure 9.11: **Export in XML–Format**

@enterprise can export the following data:

Applications Export one complete application with all process definitions and other master data defined in it. This includes all objects that are defined in the applications'

processes (see *processes* below), plus data defined in the application: rights, object classes, task functions, tasks, form types and roles. Furthermore rights (defined for roles), process interfaces, and default URLs for roles are included. Rights for roles will be imported only if the target object exists on the import system.

Processes Export one process plus tasks, steps, form types, and roles used in the selected process definition, process interfaces and rights (e.g., rights on a form type). Rights will be imported only if the required agents and departments already exist on the import system.

Formtypes Export of form types. All forms, which was created in @enterprise, can be exported. Before importing forms, the form templates and other references must be available on the target system, i.e. the appropriate application with their form types must be available. An important point is the export and import of forms which have references to other forms (e.g. a customer-form contains a reference to a country-form). To ensure the correct references on target system, you have the possibility to select all participated forms and export them in **one** form-cluster. If the participated forms are exported separately (i.e. each form will be exported in an other form-cluster), all referenced form types are exported automatically.

Organizational Units Export all organizational units.

Organizational hierarchies Export of all organizational hierarchies and their organizational units.

Users Export all users. Optionally you can include roles and rights defined for these users, and user settings as well as the users' dashboard elements.

Mind: user settings can contain a link to a home page. This link will not be modified by the import/export of **@enterprise** - thus, if it contains OIDs of specific objects (e.g. applications, etc.), the link will most likely not work any more after importing it to another system. The same restriction also applies to dashboard elements, which can contain arbitrary OIDs, too.

Permissions Export all permission lists (ACLs) of @enterprise.

Stored Queries Export stored queries - you can choose one or more queries in a second step. Optionally you can include access rights defined for the exported stored queries. Referenced objects (such as process definitions, tasks, forms, etc.) will not be included in a stored queries export. Stored queries will be imported only, if these required objects already exist on the import system.

Timer Export one or more timers. If you select to export timers, you can choose the desired timers in step two.

LDAP Settings Export all LDAP entries. This exports the LDAP entries defined in *Communication → LDAP*.

Mail Settings Export all Mailboxes defined in *Communication → Mailboxes*.

Dashboard (default elements) Export the default dashboard elements. User dashboard elements will not be included in this export (they can be exported directly with the users). Default dashboard elements are the elements that an administrator saved as default.

Process Instances Export process instances (runtime data) of one or more process definitions. This includes all step instances, form instances, adhoc steps, and so on. Rights on exported objects can optionally be included. You can restrict the exported process instances by defining a start restriction (only export process instances that have been started between two definable dates). The target process definitions can be selected in a second step.

Master data (like process definitions, users, roles, etc.) are not included in a process instance export. Process instances are only imported on a target system if the required master data already exists. Thus, on the target system you should first ensure that the required master data exists and afterwards import process instances.

DMS Folder Export a folder of the DMS with its content (runtime data). This includes documents, forms, notes, web links and subfolders (recursive). Links to other DMS objects cannot be exported and will be ignored. Access rights defined on the exported objects can be included optionally. Agents (users or roles) and departments that occur in such right definitions will not be exported. The rights will be imported only if the required agents and departments exist on the target import system.

You can export the Common folder or a specific user's folder (or one of their subfolders). If you want to export a user's folder, first select the user and then the folder.

Forms Export of form instances. All forms (instances), which was created in @enterprise, can be exported. Before importing form instances, the form classes and other referenced objects (e.g. process definition where form instance is process form) must be available on the target system, i.e. the appropriate application with their form types must be available.

Duration statistics Export of a duration statistic entry. All duration statistics which was created in @enterprise, can be exported. Before importing duration statistics the referenced objects must be available on the target system, i.e. the appropriate process definition must be available. On the target system the time graph must be regenerated, if needed (see section 9.1.15).

The server writes the XML file to its temporary directory. After an export file has been completely written, the browser will ask you if you want to download the file.

Import

Importing a XML file is done in three steps.

1. First you upload the XML file to the server.
2. The browser displays information about the XML file's content.
3. The content of the XML file will be imported and you will see information about the imported elements in the browser.

The import will be aborted and an error message will be displayed, if an error occurs. Imported objects are already stored in @enterprise!

If export-files of earlier versions of @enterprise (e.g. @ep7.0) should be imported, the user will be informed about the older export-file and has to select an application for the default objects. This selection is necessary for assigning application-objects (e.g. processes) to the right application. This selection will be ignored in some cases, e.g. if the email-settings are imported.

Import/Exports Dependencies

If you want to copy data from one server to another server, it is necessary to perform the imports in the right order. The exports can be done in any order. Runtime data (process instances, DMS content) and stored queries, as well as access rights usually require master data to exist on the import system. If the data does not exist, the objects will not be imported. If you perform imports in the following order, everything should work fine:

1. Users (without rights)
2. Organizational units
3. Applications, processes, users (incl. rights), ACLs
4. Process instances, DMS folder, stored queries, timer, LDAP settings, Mail settings, dashboard elements

9.3.2 Archive Processes

This function deletes process instances in the **@enterprise** database. If an archiving class is installed (see the configuration group "Classes"), the archive method of this class is called with each process instance. This can be used to store some information about the process instance in an external storage.
For archiving process instances perform the following steps:

1. Select an application or one specific process type.
2. Specify the finish date. All process instances of the given type which have been finished before this date are archived.
3. If you want to delete also running process instances, check the according checkbox.
4. Archive the processes with the button "Archive".

9.3.3 Install Application

If you have a *.zip or *.jar file containing an application tree (see Application Development Guide of **@enterprise**) the application can be installed very easily. Enter the corresponding file name into the field "File Name". Afterwards enter the destination directory for the new application into the field "Destination Directory" and click the "Install" button. This will transfer the zipped application to the server, extract it, and install it.

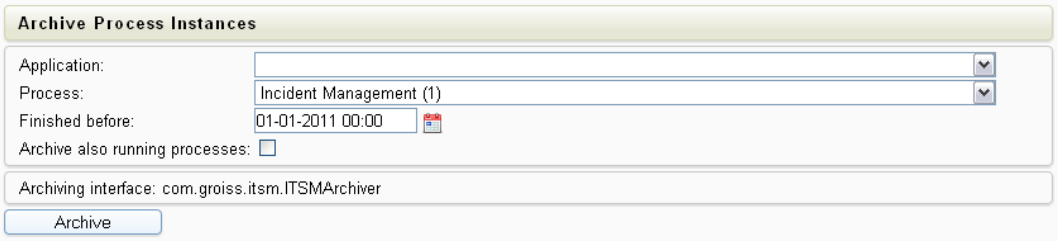


Figure 9.12: **Archive Process Instances**

9.3.4 File Import

The new file import component allows the specification of the structure of the import source and the target objects:

- *Import Definition:* An import definition file (*import.xml*) is necessary to use this function. This file must be stored in classes-folder of @enterprise or within an application-folder (see *Application Development Guide* - section *Organization of Files*). Following an example of an import definition:

```
<?xml version="1.0" encoding="iso-8859-15" standalone="yes"?>
<importDeclarations>

 <import name="resources">
 <targetClass>com.groiss.calendar.pers.Resource</targetClass>
 <columns>
 <column name="name"/>
 <column name="description"/>
 </columns>
 <keyField>name</keyField>
 <delimiter>;</delimiter>
 </import>

</importDeclarations>
```

  The keywords of the Import definition are described in section *Keywords of Import Definition*.

- *File:* Choose a source to upload a file:
  - Upload: If this function is selected, you have the possibility to enter a path.
  - local: Selecting this option allows to upload files, which are stored in @enterprise-folder (=root).
  - Classpath: This function allows to upload files, which are in classpath only.
  - According to Definition: The file, which is entered in the *Import Definition* (import.xml), is used.

- *Mode:* This dropdown-list offers following three upload-modes:
  - Parse File only: The file will be parsed only and no object are created in @enterprise.
  - Skip Database Operations: The file will be parsed and compared with existing objects (without database operations).
  - Import: The file will be uploaded and objects will be created in @enterprise (with database operations).
- *Load:* Activating this function loads the selected file.

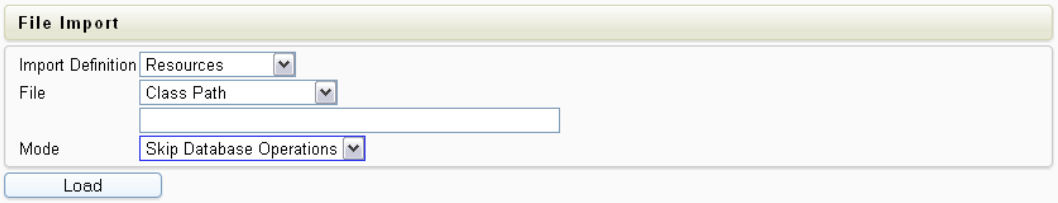


Figure 9.13: **File Import**

Keywords of Import Definition

- <import>: The import description which has the format <*import name="name"*>. Following attributes can be defined for this keyword:
  - ignoreHeader: If *true*, the first row is ignored.
  - useOrgData: If *true*, the OrgData-methods of @enterprise are used instead of Store-methods.
- <targetClass>: Symbolizes the import type (= target class).
- <targetCondition>: Restriction of *targetClass* elements. Only these elements are compared with the imported ones, not existing elements will be deleted.
- <keyField>: Field of target class, which contains the key (necessary for import).
- <importHandler>: If no *keyField* was set, a import handler must be entered which implements the interface *com.groiss.fileimport.ImportHandler*.
- <constants>: Contains a set of constants (<*constant name="name" value="val"*/>), which are added to the set of values of each row.
- <*extensionClass*>*:* Name of the class for additional data of master data objects (users, OUs)
- <delimiter>: Delimiter for fields, e.g. ;
- <escapeMode>: Exception handling, if a character occurs which has to be escaped. Backslash or Duplicate, e.g. special character is quoted: dÁrtangnon or d”artangnon

- <commentchar>: Rows are ignored which start with this character.
- <charset>: All valid Java charsets (default: StringUtil.getCharset());
- <file>: Path to file.
- <columns>: Contains a set of rows which will be imported:
  <column name="name" startcol="1" endcol="10" length="100" [format="dateformat"] [mapping="mappingName"] />
- <dateformats>: A set of dateformats can be entered:
  <dateformat name="name" timezone="timezone" locale="locale"/>
  Example:

  ```
  <dateformats>
  <dateformat name="date">ddMMyyyy</dateformat>
  </dateformats>
  ```

- <mappings>: Definition of mappings in format *<mappings name="name"> <mapping*><keys><key>M</key></keys><value>1</value></mapping> </mappings>*.
  Example:

  ```
  <mappings>
  <mapping name="lang">
  <keys><key>EN</key></keys>
  <value>en_US</value>
  </mapping>
  </mappings>
  ```

9.4 Reorganization

9.4.1 Change Role Assignments

Change the role assignments of a set of users from one organizational unit to another.
Select in the field "from old Department" the organizational unit, from which you want to move or copy role assignments to another organizational unit (see Fig. 9.14).
After clicking "Next" you can select which role assignment should be moved, copied or remain unchanged.

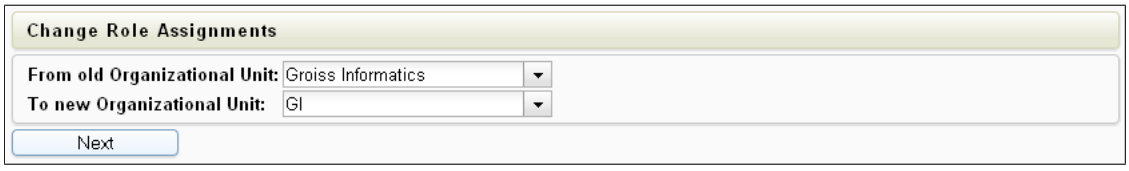


Figure 9.14: **Role Assignments (1)**

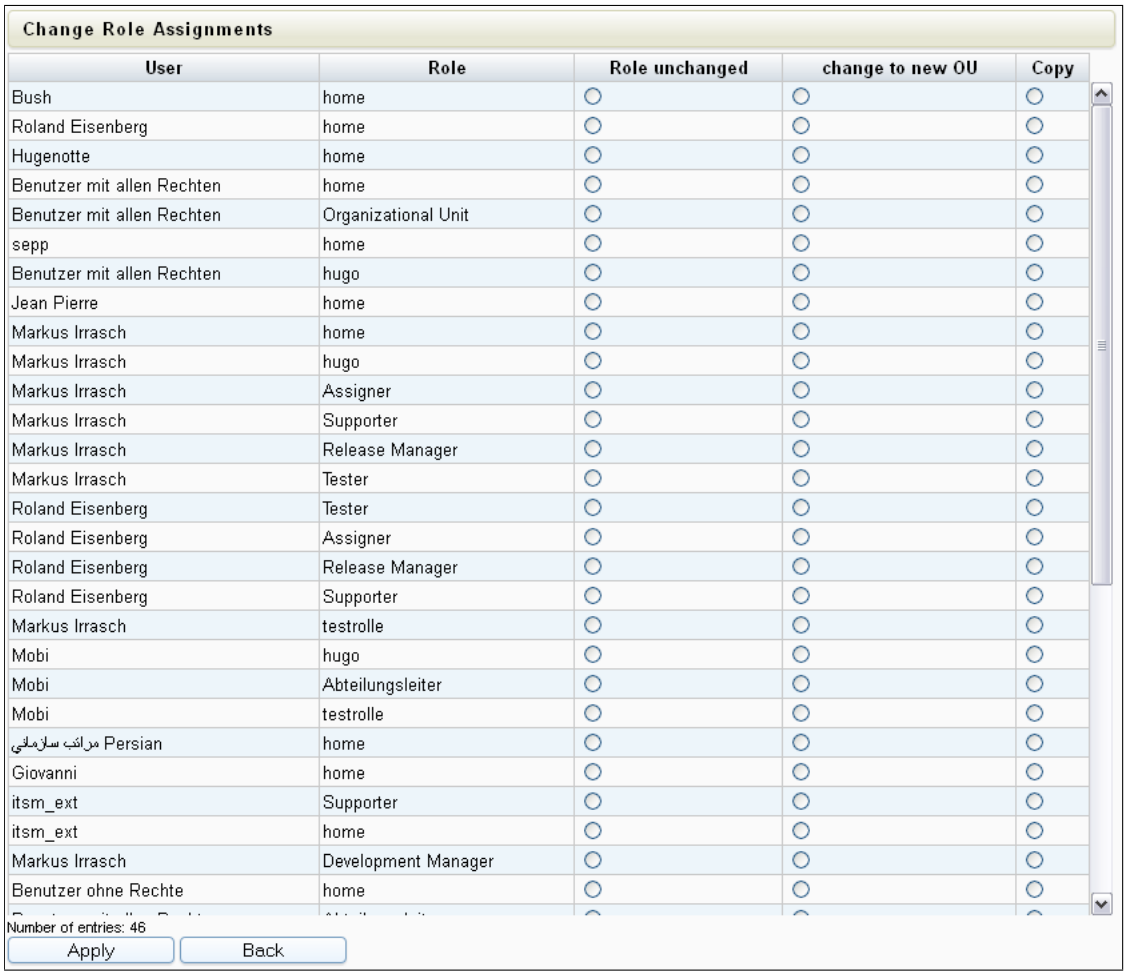


Figure 9.15: **Role Assignments (2)**

9.4.2 Analyze Process Instances

A list of process instances is shown, which have no valid agent. This case (no valid agent) can happen, when an organizational unit is deactivated, or role assignments are deleted.
Example: An existing process will be finished, but has no valid following agent. The process instance occurs in the table with problem status *Finish not completed*. Now you can click on the process instance id to open the detail-view and assign an agent who is able to finish the task. For this purpose you have to activate the link of the last active agent to assign the task to a new agent. The 3 question marks (???) in the process-history symbolizes that the instance has no following agent; should not changed in this case, otherwise a new process instance will be created.

9.4.3 OU History

Here you can capture the history of changes to organizational units manually. This might be interesting if you want to know which organizational unit emanated from another during

the process of changing your organizational structure.

9.5 Communication

9.5.1 Mailboxes

The system can handle incoming emails. Several mailboxes can be defined together with an action to perform for emails. Access to the mail-boxes is performed with the IMAP4 protocol.

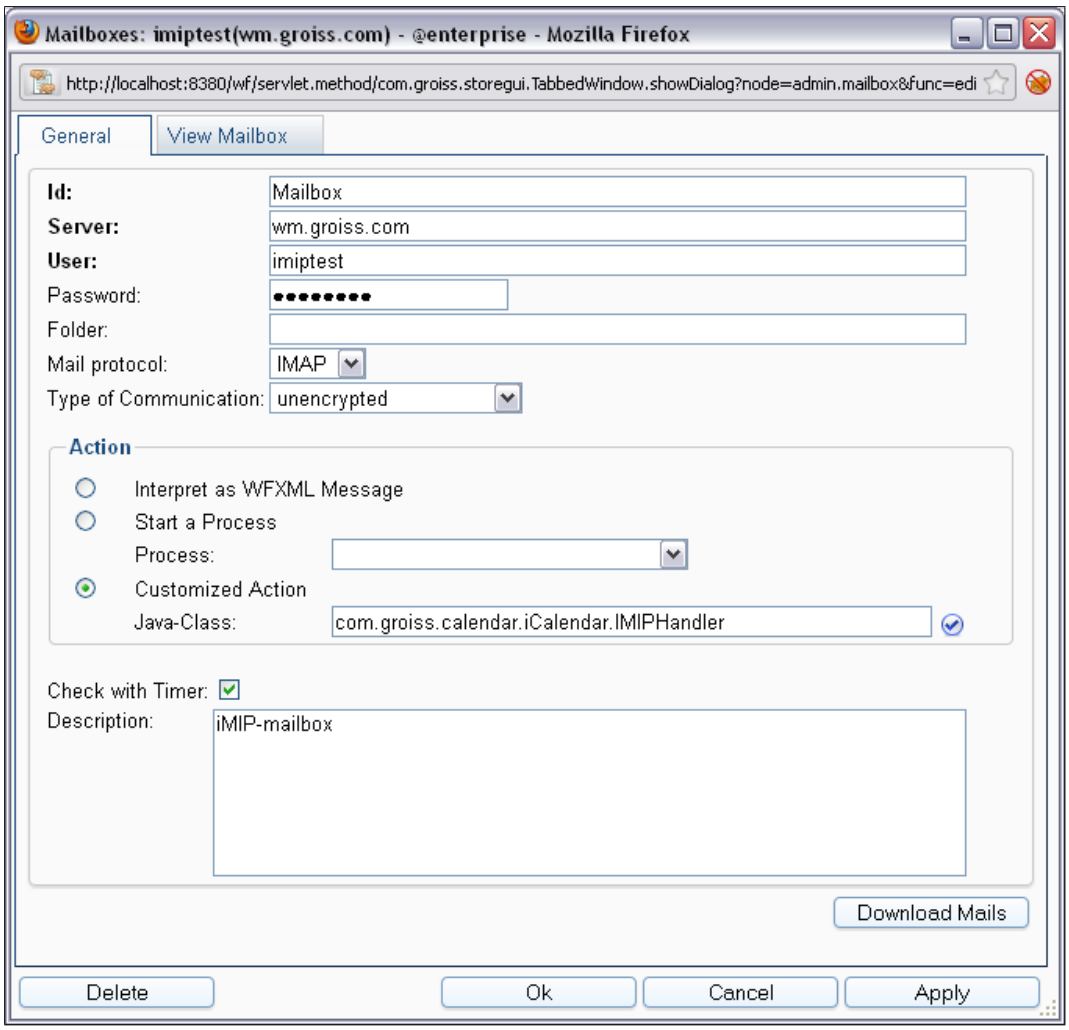


Figure 9.16: **Tab: General (Mailbox)**

Tab: General

You can edit the following attributes (required fields are bold):

- **Server:** Mail-Server,
- **User:** username for the mail-box,
- **Password:** password for accessing the mail-box,
- Protocol: This parameter specifies the protocol, which is used to get your mails from your mailserver. The possible options are IMAP4 and POP3.
- Type of communication: The level of security is set by this parameter. There are 3 possible options:
  - plain: Plain communication means, that the data is transmitted in plain text.
  - encrypted: In this case the data is SSL encrypted, but the certificate of the mail server will not be validated.
  - trusted: To communicate secure, the mail server has to authenticate itself to @enterprise. This is done by checking the certificate of the mail server. To add trusted server you have to import the certificate into the @enterprise keystore (chapter 9.1.11).
- **Action:** One of the following actions must be selected:
  - Interpret as WFXML Message
  - Start the selected process
  - Customized Action: Specify a Java class which implements the interface com.groiss.mail.MailHandler
- **Check with Timer:** The MailTimer reads the mail-box and performs the specified action.
- **View Mailbox:** View the contents of the mail-box.
- **Get Mails Now:** Performs the defined action on the contents of the mail-box.

9.5.2 LDAP

Here you can define LDAP (Lightweigth Directory Access Protocol) server entries. They can be used to synchronize @enterprise organizational data with existing directory services. We provide a predefined LDAP schema and a corresponding mapping mechanism. Customer specific synchronization semantics can be implemented as well. Details for such mappings can be found in the programming manual.

Tab: General

You can edit the following attributes (required fields are bold):

- **Name:** Name of the Server
- **Server:** Hostname of the LDAP Server
- Port: Port of the LDAP Server (port 389 is used as default)

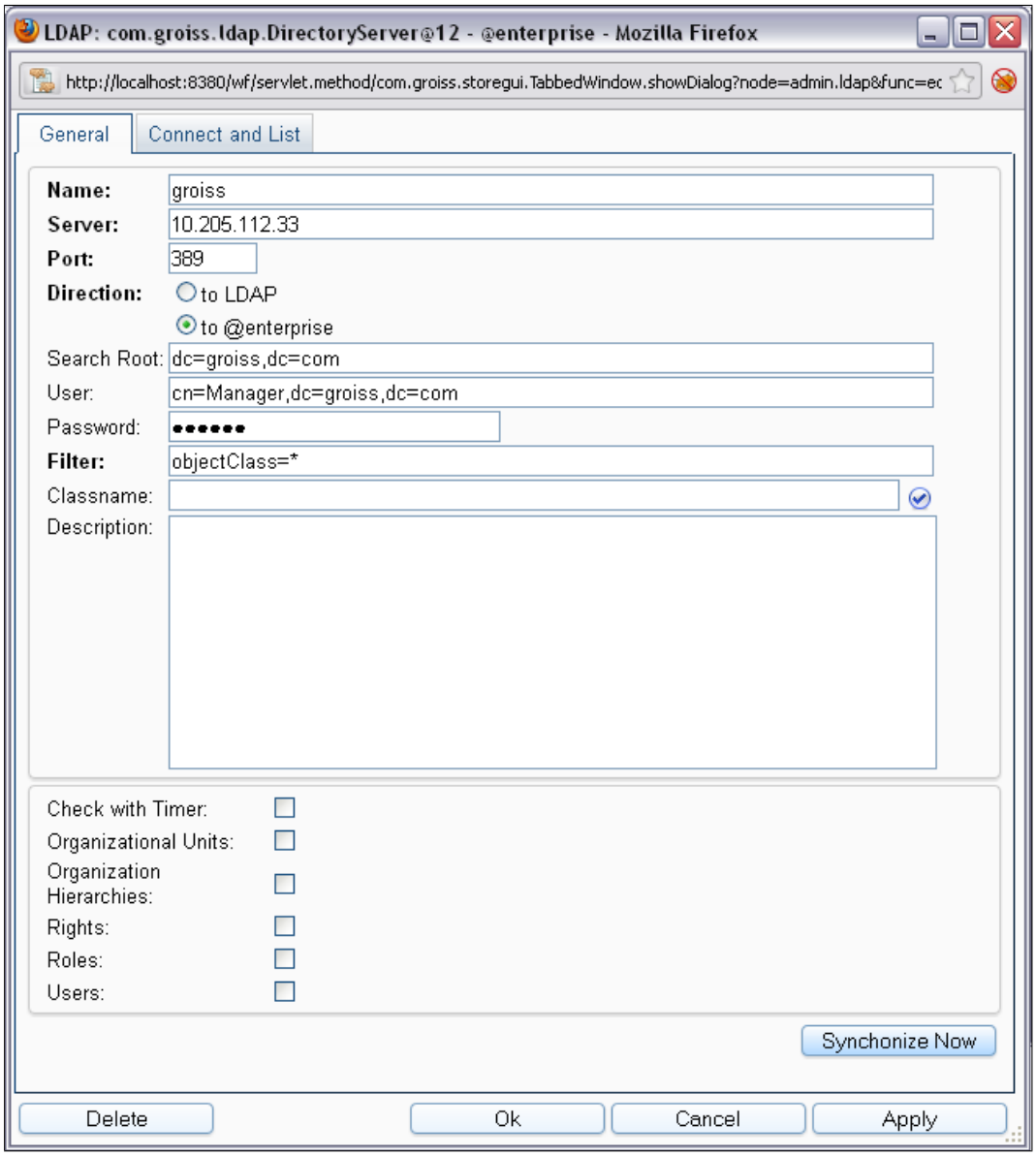


Figure 9.17: **Tab: LDAP**

- **Direction:** Direction of synchronization: either
  - to LDAP or
  - to @enterprise
- Serch Root: LDAP Root, e.g. *dc=my,dc=org*
- User: LDAP–Account, e.g. cn=admin,dc=my,dc=org
- Password: Password for the Account.
- **Filter:** LDAP Filter: allows to select just specific LDAP entries e.g.: (objectClass=*)

- Classname: by specifying a class which implements com.groiss.ldap.DirectorySyncer, one can realize proprietary schema mappings.
- Description: free text. In this text you can set the parameter *_pagesize*, if the result of read entries are too big (e.g. the search root is not deep enough). With this parameter the result will be read in paged way, depending on the number of entries per page, e.g. _pagesize=500.
- Check with Timer: if checked, the LDAPDirSyncTask-Timer executes the synchronization automatically.
- Organizational Units: if checked, Organizational Units are synchronized.
- Organization Hierarchy: if checked, Organization Hierarchies are synchronized.
- Rights: if checked, Rights are synchronized.
- Roles: if checked, Roles are synchronized.
- users: if checked, users are synchronized.

The synchronization can also be carried out by clicking the **Synchronize Now** button.

Tab: Connect and List

Through choosing this tab, one gets a listing of the contents of the LDAP Server.

9.5.3 Batch Jobs

With the help of this function it is possible to search after batch jobs. As search criteria the process id, the state of the batch job and/or the time-period where a batch job has been started, can be used.
After activating the button *Search* a result table with all batch jobs are displayed depending on the search criterias. By double-clicking on an entry the detail mask can be opened and subsequently edited. By activating the button *Abort and go back* the batch job will be aborted and returned to the last interactive task of the process.

More details about Batch Jobs can be found in the *Application Development Guide* in section *Batch Processing*.

9.5.4 WfXML

Wf-XML is a protocol for process engines that makes it easy to link engines together for interoperability. Wf-XML 2.0 is an updated version of this protocol, built on top of the Asynchronous Service Access Protocol (ASAP), which is in turn built on Simple Object Access Protocol (SOAP).
@enterprise contains an implementation of the standard. @enterprise can receive Wf-XML messages to start a process, get the current state of a process and change a process' state; and the system can also send all types of messages.

Detailed Information about this topic can be found in the *Application Development Guide* of **@enterprise**.

9.5.5 Web Services

The link *Local services* provides a table of all found web services in **@enterprise**. It is possible to add a new web service, delete and (un)deploy it. The creation of web service clients/server is possible per application where the appropriate functions are available.

Detailed information about this topic can be found in the *Application Development Guide* of **@enterprise**.

10 Configuration

This chapter describes the configuration of **@enterprise**–server. Further information about

- License
- HTTP–Server
- Database
- Directories
- Logging
- Classes
- Localization
- Communication
- Cluster
- Workflow
- DMS
- Search
- Tuning
- Security
- Password Policy
- Calendar
- Time Management
- Change Administrator Password
- Initialize Database Scheme

are available in the *Installation Guide - Chapter Configuration*.

11 Dashboard

In the system administration of **@enterprise** it is possible to create a dashboard, which an be aligned for the needs of the system administrator or user. After activating the link *Dashboard* under *Admin-Tasks* it can be clicked into the empty site, whereby a popup–window will be shown where following functions are available:

11.1 New

By activating the button *New* a new HTML–site will be shown, where you can add new windows to the dashboard. Several possibilities are provided:

- URL: Enter an URL of a HTML–site, which you would like to see in a window on your dashboard and confirm your inputs with *Return*.
- Stored Queries: By activating this link all stored requests will be shown in a window on your dashboard.
- Administration: By activating this link the links of the administration tasks will be shown in a window on your dashboard.
- Calendar: By activating this link a calendar will be shown in a window on your dashboard.
- Worklist Overview: By activating this link an overview about the number of worklist entries will be shown in a window on your dashboard.
- News: By activating this link news will be shown in a window on your dashboard. Therefor a folder with the name *News* under *Common* must be created in the DMS, where messages can be lodged (e.g. a note).
- Appointments: By activating this link the appointments of the present day will be shown in a window on your dashboard.

Note: Each window in a dashboard can be moved (like in Windows) to any place inside the dashboard and/or be changed in its size.

11.2 Open

By activating the button *Open* an existing dashboard profile can be loaded. First the profile must be stored with the function *Save as*. There are 2 kinds of dashboards: the personal and other dashboard(s). The personal dashboard list contains all dashboard, which are stored by the current user. Other dashboards has been stored by other users which have set the *share*-right for the current user (via tab *Access*).

11.3 Save

By activating the button *Save* the current dashboard will be saved. A new dialog will be opened with following attributes:

- **Name:** The unique name of the dashboard must be entered here.
- *Description:* Free text
- *Default:* Select between
  - Dashboard user/admin for <User>: Dashboard settings are stored for current user as default. There is a differentiation between *user* dashboard (created in worklist) and *admin* dashboard (created in administration).
  - Dashboard user/admin for all users: Dashboard settings are stored for all users as default. There is also a differentiation between *user* dashboard (created in worklist) and *admin* dashboard (created in administration). If a user has no dashboard, this dashboard will be displayed.
- *Owner:* This field is read-only and shows the owner of this dashboard.
- *Dashboard-Id:* This field is also read-only and shows, if the dashboard is/was created in worklist (user) or in administration (admin) of @enterprise.

It is also possible to define *share*-rights by using the tab *Access*. In this case other users are permitted to open this dashboard (see function *Open*).

Hint: The function *Save* is available for users, who are owner or has the role *SYS*.

11.4 Save as

After activating the button *Save as* a new window is opened (analog to function *Save*). This function allows to store the current dashboard under a new name.
If a user changes the dashboard by activating the buttons *New* and *Save as*, it will be his personal dashboard and the Default–Dashboard remains unchanged. The user has the possibility to open an existing (default) dashboard profile by using the function *Open*. The identification is made by the URL parameter *id*.

11.5 Delete

This function deletes the current dashboard settings (dashboard profile).

12 Administration Shell

This chapter describes the administration shell which allows to administrate **@enterprise** via a command line. It can be used to:

- Assemble administration actions as a script and execute it on several servers
- Synchronize changes between development system and production system
- Send a script to a system operator
- Document actions

12.1 Architecture and Invocation

The administration shell has a client and a server component. The server component is integrated into the @enterprise server. The client component is packaged in a separate jar file *adminshell.jar* in the *bin* directory of @enterprise. The client connects to the @enterprise server via plain HTTP or secure HTTPS. This can be configured on the configuration mask *Communication*. The administration shell must be activated via the hidden parameter *ep.adminshell.enable*. More details can be found in the *Installation- and Configuration-Guide*. Please note that the operating user needs the right *execute* on all objects for the connection to the server! Furthermore the user needs the corresponding rights for perform the server commands.

The admin-shell client can be invoked with the following call:

```
java -jar adminshell.jar url user [password]
  [-log logfile | -append logfile] [-passwdfile file] [-execute scriptfile]
```

Parameters:

- **url:** The URL of the server, e.g. *http://localhost:8380/wf/*. If no context-root is entered, *wf* will be used by default.
- **user:** The username of the operating user
- **password:** The password of the user (if existing). If you do not specify a password, you must use the option *-passwdfile* or you will be asked for the password at the login.

Options:

- **-log logfile:** The logfile defines a file where the admin-shell logs the interactions (on the client).
- **-append logfile:** Same as *-log* except that the logfile is appended to.
- **-passwdfile file:** The file contains the plain password for the given user in the first line without any preceding and trailing characters.
- **-execute scriptfile:** Executes the script in scriptfile.

12.2 Commands

Two groups of commands can be executed:

1. **Client commands** are executed on the client and define some behavior of the script client.
2. **Server commands** are executed on the server and contain the functions of the administration.

12.2.1 Client commands

Following client commands are available:

- **exit**: Exits the client.
- **help** or **?**: Print a command summary
- **log <file>**: Log commands to the given file
- **log off**: Commands are not logged anymore
- **append <file>**: Log commands to the given file. If the file already exists, commands are appended.
- **execute <file>**: Executes the given script file

Commands not in this list are sent to the server.

12.2.2 Server commands

The commands on the server are interpreted as Groovy expressions. Groovy is a script language based on Java. Comments have the same syntax as in Java (inline- and block-comments). Server commands are terminated by a line containing only the character . (dot) and will be logged in serverlog at loglevel 1 and higher.

The following variables are in the initial context (varname and instance of):

- **admin**: com.groiss.server.Admin

- **store**: com.groiss.store.Store
- **engine**: com.groiss.wf.WfEngine
- **dms**: com.groiss.dms.DMS
- **orgdata**: com.groiss.org.OrgData
- **config**: com.groiss.component.Configuration (the System Configuration)
- **user**: com.groiss.org.User (the current user)
- **session**: javax.servlet.http.HTTPSession (the HTTPSession)

They can be used as starting points for the execution of methods (see API for details). Every command is executed in its own transaction. After executing, a commit, if an error occurs, a rollback is performed.

If you want to use (own) variables for the script, you can define them with the command:

```
set(varname,value);
```

Retrieve the value of the variables with:

```
get(varname);
```

Own declared variables have the advantage to survive transactions, because they are written into the session.

12.3 Examples

12.3.1 Setting a configuration parameter

```
config.setProperty("database.connections",5);
config.store();
.
```

Alternative formulation with a variable:

```
set("connections",5);
.
config.setProperty("database.connections",get("connections"));
config.store();
.
```

12.3.2 Restart the server

```
admin.restartServer(); //restarts the server - no login necessary for current user
.
```

12.3.3 Add a role to or remove one from a user

```
u = orgdata.getById(com.groiss.org.User.class,'my_user'); //replace by existing user
role = orgdata.getById(com.groiss.org.Role.class,'sys'); //get SYS role
checkuserrole = store.get(com.groiss.org.UserRole.class,"role = ? AND userid = ?",
  role.getOid(), u.getOid()); //with prepared statements - new Object[] {args}
//If User has no sys-role, add it
if(checkuserrole == null) {
 userrole = orgdata.createUserRole();
 userrole.setRole(role);
 userrole.setUser(u);
 userrole.setActive(true);
 orgdata.insert(userrole);
}
//If User has sys-role, remove it
else {
 orgdata.delete(checkuserrole);
}
.
```

12.3.4 Set the interval of a timer

```
t =  orgdata.getById(com.groiss.timer.TimerEntry.class,'Suspension');
t.setPattern("360");
store.update(t);
.
```

12.3.5 Worklist handling

Check worklist of application *default* and finish expired tasks:

```
appl = orgdata.getById(com.groiss.org.Application.class, "default");
worklist = engine.getWorklist(appl,true);
for(com.groiss.wf.ActivityInstance ai:worklist) {
 duedate = ai.getDuedate();
 //if ai's duedate is expired, finish task
 if(duedate != null && duedate.getTime() < new java.util.Date().getTime()) {
  try {
   engine.finish(ai);
  }
  catch(ex) {/*Do nothing, but continue with finishing other ai's*/};
 }
}
.
```

12.3.6 Session handling

Check session and invalidate it, if lastAccessed is not in tolerance time. Log session information in server-log on level 2:

```
attrbnames = session.getAttributeNames();
invalidate = false;
log = " \n";
log = log + "Session-Parameter:\n";
for(String attrname:attrbnames) {
 attrvalue = session.getAttribute(attrname);
 if(attrname.equalsIgnoreCase("lastAccessed")) {
  if(attrvalue instanceof java.util.Date) {
   onehour = 60*60*1000; //tolerance time
   //invalid, if not in tolerance time
   if((attrvalue.getTime()+onehour) >= new java.util.Date().getTime() ||
      (attrvalue.getTime()-onehour) <= new java.util.Date().getTime()) {
    invalidate = true;
   }
  }
 }
 log = log + "Attribute-Name: " + attrname + "/Attribute-Value: " + attrvalue + "\n";
}
log = log + " ";
com.groiss.util.Settings.log(log,2); //write all session parameter to Server-Log on Level 2
if(invalidate == true) {
 session.invalidate();
}
.
```

13 Process–Cockpit

The Process–Cockpit gives an overview of the processes within the organization. It provides information about the definition and the instances of a process. The standard GUI has a link to the cockpit within the group *Extras*.

13.0.1 Configuration

The processes are shown in a hierarchy that must be defined in the document management system (DMS) of **@enterprise**. Create a structure of folders reflecting the organization of processes in your company, for example:

1. Operating processes
   (a) Manufacture
   (b) Marketing
      i. Manage sales plans

Define the path to your process structure in the configuration of **@enterprise** (see *Installation- and Configuration manual* - Chapter *Process Cockpit*).
The leaves of the tree should be process-cockpit folder-forms (other nodes can be ordinary folders).
The folder form provides following types (see figure 13.1):

- **Process with Definition:** Here you can select one process definition only. The checkox *Show Instances* allows to show/hide instance data in tab *Runtime*.
- **Process without Definition:** For processes where no unique **@enterprise** process is available. A name and a description can be entered. For this kind of processes the assignment of instances is done in following way:
  Processes with process forms exist which contain a field with name *area*. The valuation of this field are the nodes of the Process–Cockpit. The used processes area defined in the configuration with parameter *Common Processes* (see *Installation- and Configuration manual* - chapter *Process Cockpit*). An example is given in the *User Manual* in section *Details for processes*.
- **Process Group:** An intermediate node in the process trees which needs a name and a description. In this case you can add documents to the folder which are displayed as links in table *Documents* on process detail page.

Following fields are visible in cockpit form depending on the selected type:

- **Responsible:** The person who is responsible for the process or process group.
- **Available Reports:** A list of reports suitable for this process is displayed at the process detail page (e.g. in tab *Runtime*).
- **Directly executed Reports:** A list of reports which are executed when the runtime process detail page is shown (e.g. in tab *Runtime*).
- **Functions:** A list of functions which are shown on the process detail page (e.g. in tab *Runtime*).
- **Links:** A list of links consists of a URL and a text which are shown on the process detail page (e.g. in tab *Runtime*).

A link to the folder forms is displayed in the toolbar of the detail view of the Process–Cockpit where details of the shown process (or process group) can be configured.

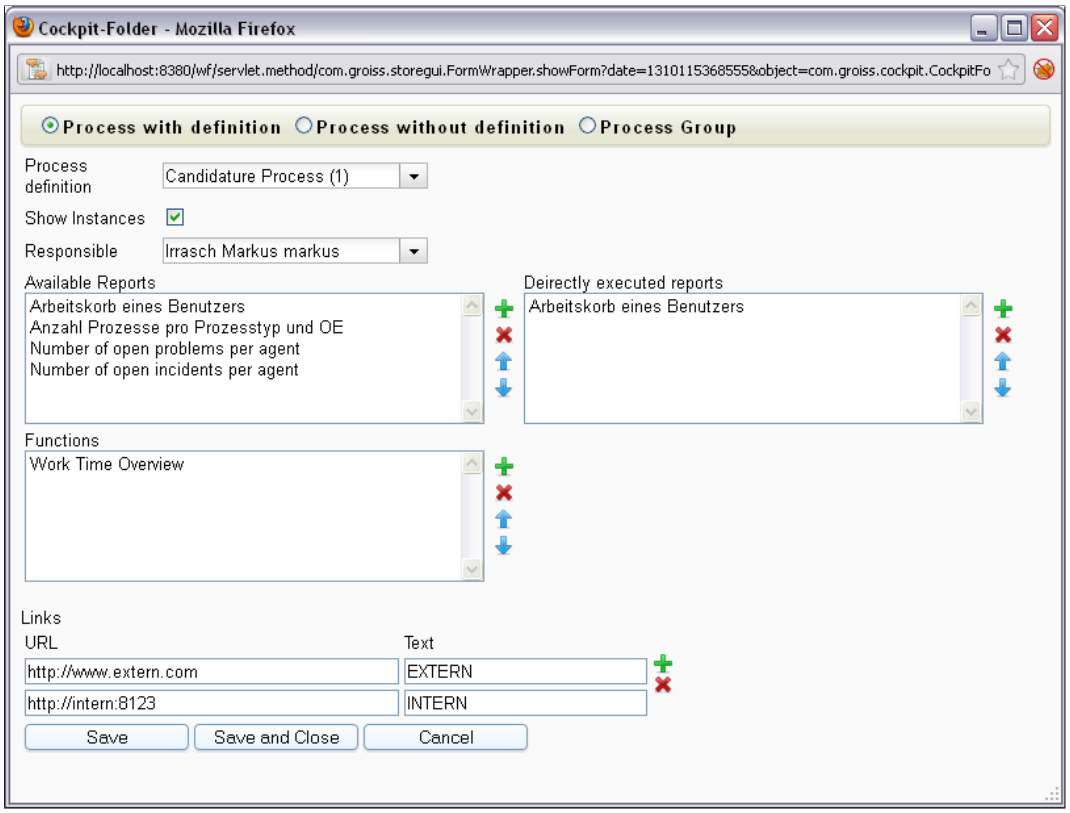


Figure 13.1: **Process–Cockpit Form**

13.0.2 Rights

Everyone who has the *edit* right on folders of the Process–Cockpit, can manipulate it. Everyone who has the *view* right on a folder, gets a link and the associated detail page within

the cockpit. The instance reports (Running, Finished, This week, This month, Deadline Violations) can be executed without right-check. For all other reports the right *execute* is needed.