schedulix!Web

User Guide
Release 2.11

Dieter Stubler   Ronald Jeninga

January 2, 2024

Copyright © 2024 independIT GmbH

Legal notice

Contents

List of Tables

List of Figures

14

1 General

1.1 Overview

This chapter will introduce you to the schedulix!Web user interface. General actions and procedures that are not just to be found in a particular dialog are explained and illustrated here in more detail.

The schedulix!Web user interface is a browser-based dialog system. There is no need to install an additional client software.

To start the schedulix!Web interface you have to open a web browser window (e.g. Microsoft Internet Explorer). Please enter the URL you have been given to get to the schedulix!Web interface. We recommend saving this URL in your browser as a bookmark so you can start the schedulix!Web interface with one click.
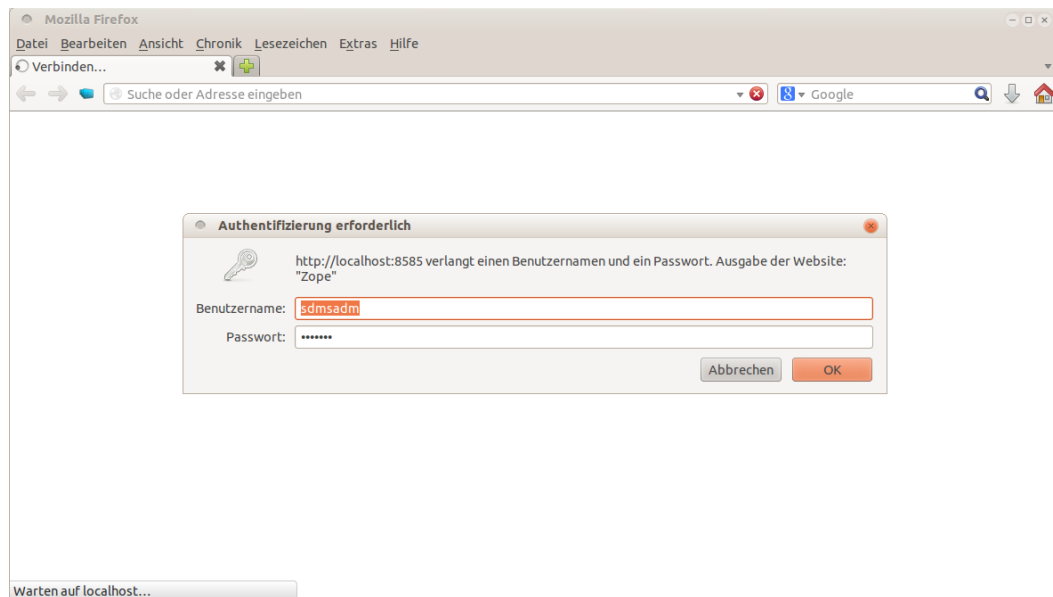
Figure 1.1: Login window

1.2 Login

After entering the URL, a login window opens for verifying your password. All users have to identify themselves with a user name and password to work with schedulix.

The input mask for the password prompt is shown in Figure 1.1, whereby the exact appearance may vary slightly depending on the browser.

Here you need to enter your personal user name and matching password and confirm them by clicking *OK*. Your login details can be obtained from your system administrator.
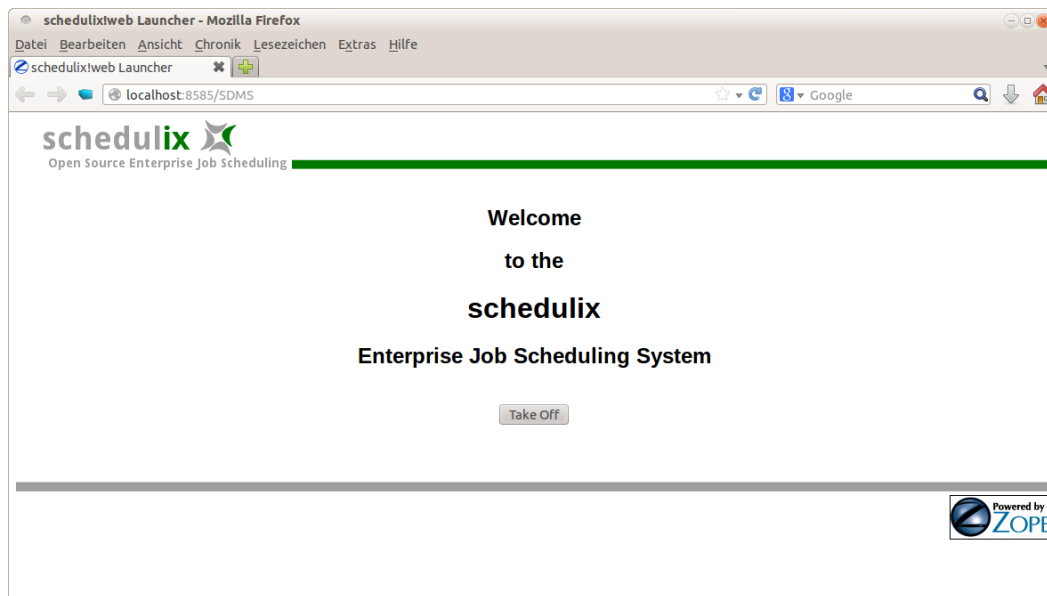
Figure 1.2: schedulixLauncher

After you have successfully logged in, the schedulixLauncher opens. Now open the main menu by clicking the *Take Off* button.

The launcher may be closed automatically depending on your browser. Your browser's security settings may prevent this or trigger a warning message.

After clicking the *Take Off* button appears the main menu called the *Main Desktop*.

1.3 Main Desktop

The *Main Desktop* is the main window of the schedulix!Web user interface. You can launch all the function dialogs in schedulix directly from this window. From the *Connection* options box in the header you can select which server connection you want to use for the next function dialog that is to be opened.
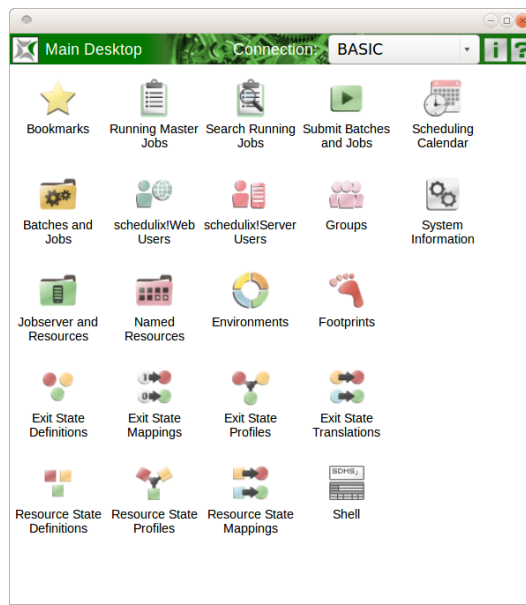
Figure 1.3: Main Desktop

The *Main Desktop* is shown in Figure 1.3.
The dialog boxes for maintaining or monitoring different objects can opened by clicking the icons shown below:

Icon	Description
	The *Bookmarks* dialog is used to create, edit and delete views of currently running jobs.
	The *Running Master Jobs* dialog gives a clear view of the current Master Jobs.
	The *Search Running Jobs* dialog is used to search for running jobs according to specific criteria.
	The *Submit Jobs* dialog is used for starting workflows.
	The *Batches and Jobs* dialog is used to create and manage folders, batches, jobs and milestones.
Continues on next page	

Icon	Description
	The *schedulix!Web Users* dialog is used for the administration of schedulix!Web users.
	The *schedulixServer Users* dialog is used for the administration of schedulixServer users.
	The *Groups* dialog is used for the administration of schedulixuser groups.
	The *Jobservers and Resources* dialog is used for the administration of jobservers, scopes and associated resources.
	The *Named Resources* dialog is used for creating, modifying and deleting Named Resources.
	The *Environments* dialog is used to define runtime environments for jobs.
	The *Footprint* dialog is used for the administration of footprints (resource profiles) in the system.
	The *Exit State Definition* dialog is used for defining and administering Exit State names.
	The *Exit State Mappings* dialog is used for the administration of translations between return values from the system processes and logical Exit States.
	The *Exit State Profile* dialog is used for the administration of profiles, i.e. a summary and prioritisation of Exit States.
	The *Exit State Translation* dialog is used for the administration of translations of Exit States to Exit States.
	The *Resource State Definition* dialog is used for defining and administering Resource State Names.
	The *Resource State Profile* dialog is used for the administration of profiles, i.e. a summaries of Resource States.
	The *Resource State Translation* dialog is used for the administration of translations of Exit States after Resource States have been changed.
	The *Calendar* dialog shows an overview of the pending submits.
	The *SysInfo* dialog shows information about the configuration and current state of the system.
	Shell allows the execution of API Commands.

Table 1.1: Description of each icon in the main menu

1.4 Layout of the object window

When a dialog box is called from the *Main Desktop*, it is opened in a new window. Each window is independent of other dialog windows and any number of dialogs can be open at the same time. A standard dialog looks like this:
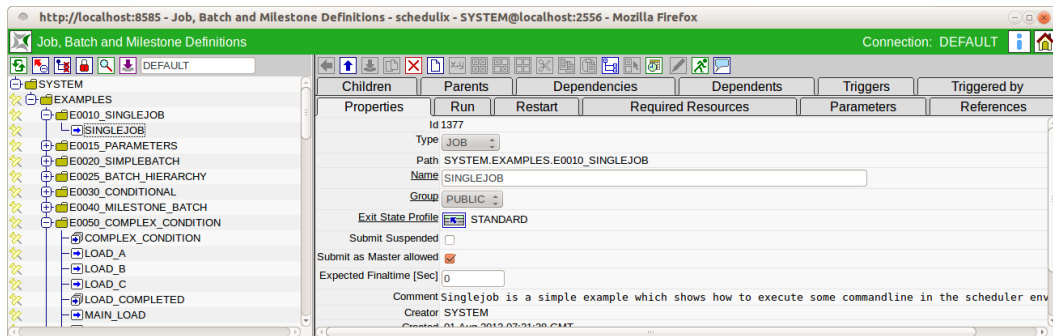
Figure 1.4: Example of a standard dialog

1.4.1 Title bar

Each dialog has a title bar at the top. The title bar contains the elements described below.

independIT icon/hourglass

The independIT icon is located in the upper left corner of the window. Clicking this icon opens a new window with the website of independIT GmbH. When you perform an action, the new or modified data has to be sent to the schedulix server or data has to be downloaded from it. This activity is indicated by the hourglass icon. No further actions can be performed in the window for as long as the hourglass icon is active. If you do try to perform an action, this will trigger an error message as shown in figure 1.5 In this case, the box has to be confirmed by clicking *OK* and you have to wait until all the data has been transferred to or from the server. You can continue entering data as soon as the hourglass disappears and the IndependIT icon is displayed again.

Name of the dialoge box
The name of the active dialog box is shown next to the independIT icon.

About

Version and system information can be displayed by clicking this button.
The following information is shown in this window:
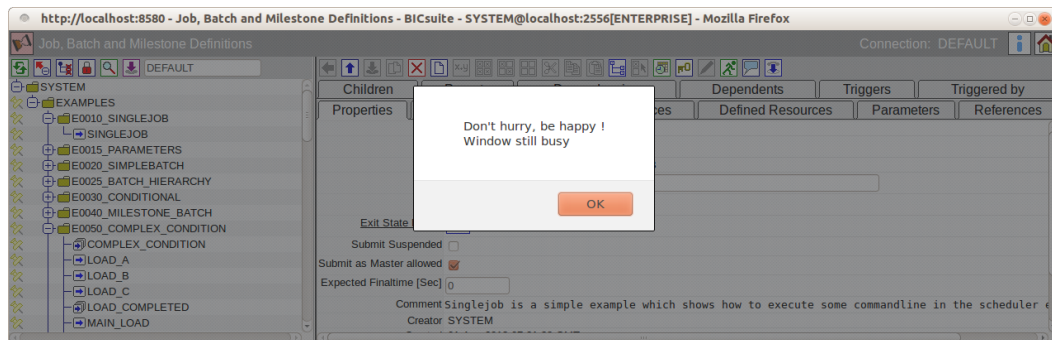
Figure 1.5: Busy message
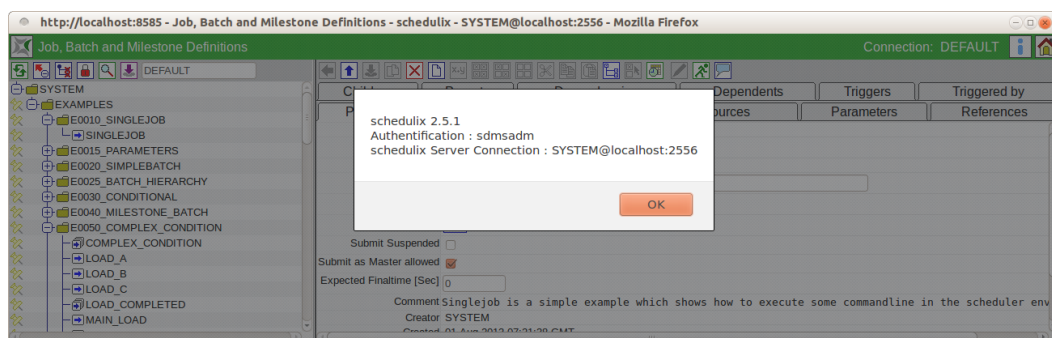
Figure 1.6: About window

schedulix!Web n.n.n: Current version of the schedulix!Web user interface.
Authentication: Name of the user who is currently logged in.
schedulix Server Connection: Connection data for the schedulix server.

Help

The *Help* icon is only visible on the *Main Desktop* instead of the *Home* icon. Clicking the *Help* icon takes you straight to the online documentation.

Home

The *Home* icon is used to call the *Main Desktop*. If the *Main Desktop* window has been minimised or it is in the background, it is restored and brought into focus. If it has been closed, clicking the icon opens a new *Main Desktop* window.

1.4.2 Navigator

The left pane of a dialog window is the Navigator. It shows the objects that have already been defined. These are displayed either in a hierarchy (if the type of objects can be arranged hierarchically, for example named resources) or as a simple list if there is no hierarchical order. The hierarchical view is a tree structure.
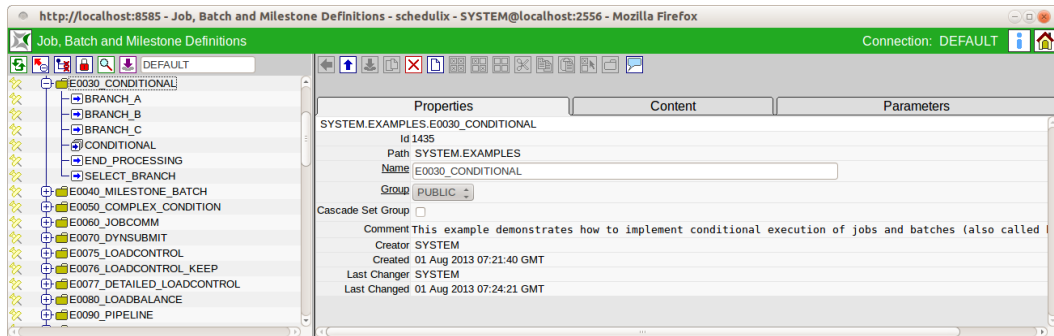
Figure 1.7: Navigator with a hierarchical view

If an object displayed in the hierarchical view is a container that can contain other objects, this is indicated by a folder icon like e.g.

These are different objects (folders, scopes, named resources, etc.) depending on their colour and how they are displayed. If a container holds sub-objects as well, a ⊕ or ⊖ is displayed. By clicking this icon, the contents of the container can be displayed ⊕ or hidden ⊖.

1.4.2.1 Standard buttons

Clicking the buttons above the Navigator as described below controls the window view and whether contents are displayed or hidden:

 Refresh

The *Refresh* button is used to refresh the list of objects in the navigation. After clicking this button, all the available objects are read out from the server again.

 Collapse All

With this button, an expanded hierarchy (i.e. a hierarchy in which the sub-folders are displayed and the parent folder has a ⊖) is completely closed again. This means that only the objects at the top level of the hierarchy are visible.

Expand All

This button is used to fully open a closed or partially open hierarchy. This means that all the top level containers are scanned for sub-containers or sub-objects and these are displayed. The folder icon is preceded by the ⊟ character. If the sub-containers hold further sub-objects, these are also displayed until all the objects throughout the entire hierarchy are shown on the screen. With a large hierarchy containing a large number of objects and sub-objects, this can take a relatively long time. For this reason, the button is not available for every hierarchical object type.

Show Leaves, **Hide Leaves**

These two buttons allow non-container objects to be displayed and hidden in hierarchies in the Navigator. Only one button is alternately always visible.

Show Locked, **Hide Locked**

These two buttons allow the names of objects without read privileges to be displayed and hidden. Only one of these two buttons is alternately always visible.

Save View

The *Save View* button is used to save the current state of the hierarchy in the Navigator.

Search

This button activates the search mask in the navigation. This enables you to very easily find an object with a known name even in a complex hierarchy. Clicking the *Search* button opens the search mask:
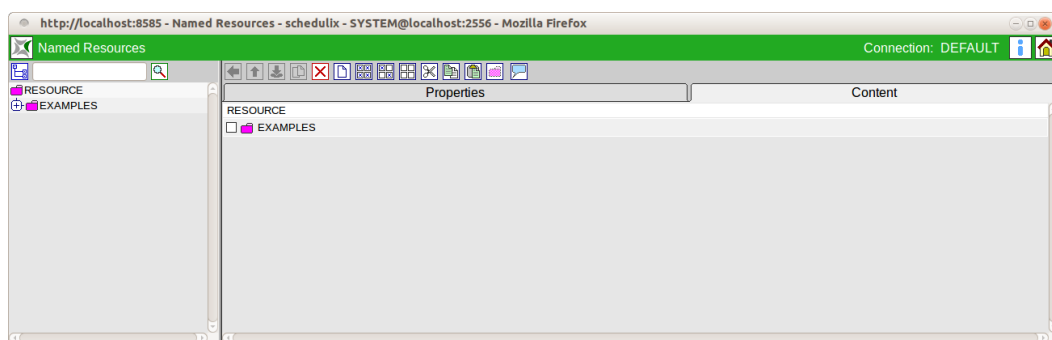
Figure 1.8: Searching in the Navigator

Start Find

This starts the search. In the example shown here, a search is being run in the entire hierarchy for *Named Resources* that contain the search string.

Hits are shown as a list in the Navigator. Hierarchies are not taken into consideration. The result could look like this:
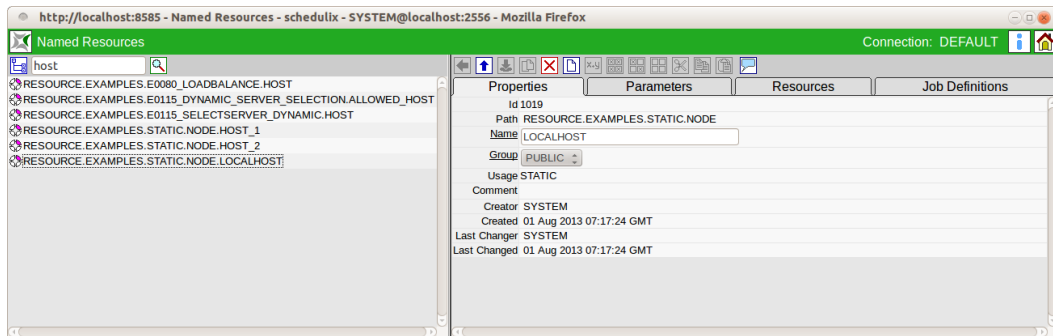
Figure 1.9: Search results

Here we were searching for the term "host", and three Named Resources were found that contain this term. The search is not case-sensitive with regard to the search term.

Exit Find

Clicking this button closes the search mask and the standard button bar is displayed above the Navigator again.

1.4.3 Editor

The Editor is the dialog window for modifying or creating objects. When an object is selected in the Navigator, its data is displayed in the Editor and can be modified here. A selected object can be saved, deleted or duplicated using the buttons at the top of the Editor. A new object can be created by clicking the *New* button.

The data in the Editor is either displayed in full in one window or it is logically grouped in different tabs.

The number of tab sheets in a dialog varies depending upon the type of the object selected in the navigation. During an action (modifying or creating an object), you can switch between the single tabs as often as you need to and in any order. You do not need to save your changes in between because all the data (including the data in hidden tabs) is kept in the memory. Only when you click the *Save* button is the data in all the tabs sent to the schedulix server and saved there. If any changes have been undone by clicking the *Cancel* button, all the modified data (including in any hidden tabs) is discarded.
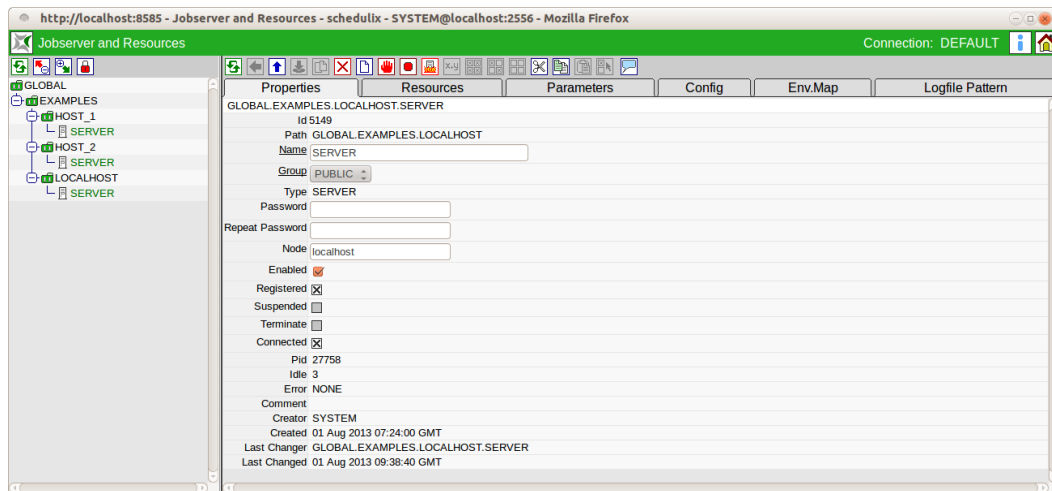
Figure 1.10: Editor with tabs

A tab sheet is selected simply by clicking the respective tab on which the name of the tab sheet is shown. The data from the selected tab is then displayed in the mask and the tab is graphically moved into focus and highlighted in a light colour. The inactive tabs are dark in colour.

The Editor button bar shows different enabled and disabled buttons depending upon which tab is open.

1.4.3.1 Standard buttons

The following standard buttons are shown in many dialog boxes and are used in an identical or comparable way in each dialog.

 Cancel

This button is used to undo actions. All changes that have not yet been saved can be revoked. This means that all changes made since the data was loaded or last saved are discarded.

The *Cancel* button is only enabled if a change has been made. Changes are recognised only when you leave the first field in which a change was made. If there is only one field in the mask, you have to press the keyboard tabulator key to accept your changes. This then enables the *Cancel* button.

To prevent entered data from being lost, a confirmation query is displayed after you have clicked the button. If you really want to discard your changes, you have to confirm this query by clicking *OK*. Otherwise, you can click *Abort* to return the previous dialog and preserve the changes you have made.

Up

Clicking the *Up* button takes you up one level in the hierarchy.

Save

A change that has been made is saved with this button. The *Save* button is only enabled if data has actually been changed. Changes are only recognised once you have left the changed input field.

Clone

This button is used to create a new object which possesses all the properties of the currently selected object. To enable the button, you have to enter a new name for the object. When you click the button, a new object is created with this name and displayed in the navigation window.

Important: If you click *Save* instead of the *Clone* button, only the name is changed, i.e. a new object is not created.

Drop

This button is used to delete an entire object. A confirmation query is displayed before the object is displayed.

If you really want to delete the object, confirm the query *OK*. If you want to cancel the operation, press the *Abort* button.

The object is deleted from the server after the action has been confirmed. There is now no way to undo the deletion. If the deletion was unintentional, you will have to create the deleted object again.

In the lists, you can also delete selections. The procedure here is as follows. First, you have to select the rows you want to delete either individually or by clicking the *Selection* button. Now click *Drop*. All the selected entries are deleted after you have confirmed the prompt.

New

This button is used to create a new object. If different objects can be created in a dialog, first you have to select the type for the new object. The input fields and tabs for the corresponding object type are then displayed. If a selection is not necessary, the Editor opens immediately after you click the button. If an Editor window still contains any unsaved changes, the button is disabled to prevent the changed values from being accidentally deleted.

Show Folder Path, **Hide Folder Path**

These buttons are switches. When you click the *Show Folder Path* button, the scopes, submitted entities, scheduling entities and folders are displayed together with the

complete parent hierarchy. The levels are each separated by a full stop. The view looks this after you have clicked the button:
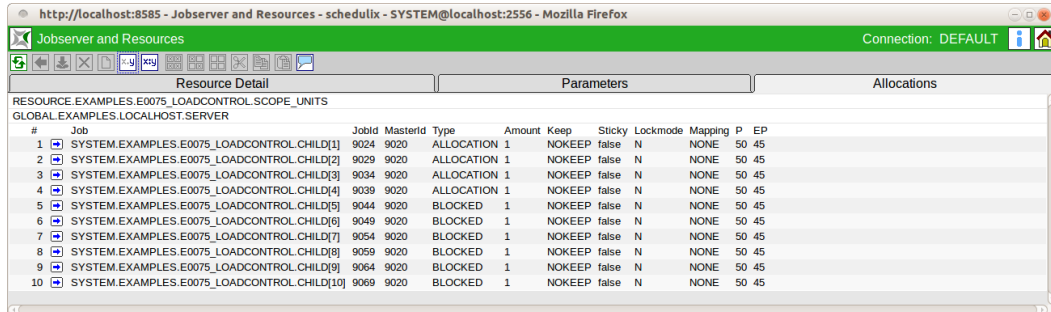
Figure 1.11: Show Folder Path

When you click *Hide Folder Path*, the previously visible parent folders are not displayed. The window looks like this:
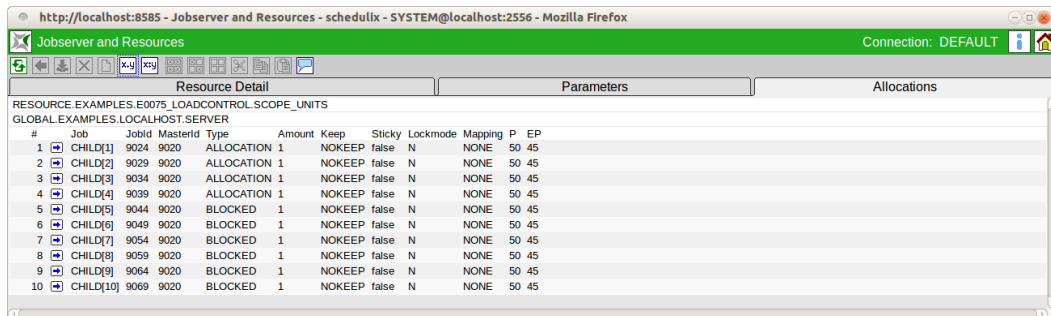
Figure 1.12: Hide Folder Path

x:y **Show Hierarchy Path**, **x:y** **Hide Hierarchy Path**

These buttons are switches. Clicking the *Show Hierarchy Path* button displays the task with the complete superordinate parent-child hierarchy. If a task has a parent, this is shown in the name and so on until the top level has been reached. The levels are each separated by a colon (Figure 1.13).

When you click *Hide Hierarchy Path*, the previously visible hierarchy is no longer displayed. The view looks this after you have clicked the button (Figure 1.14):

12
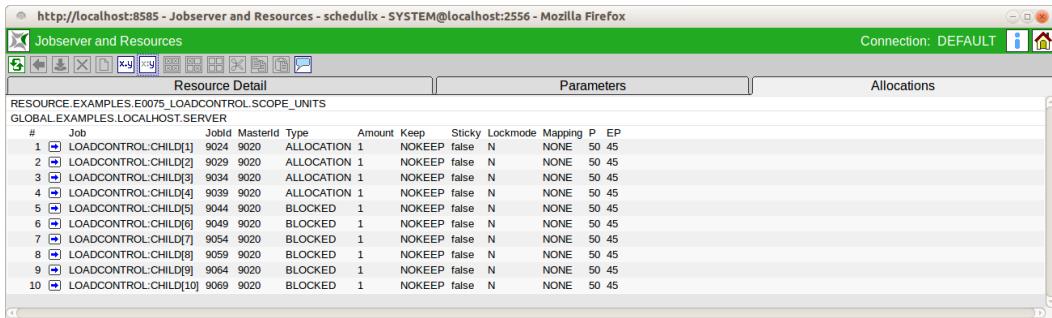
Figure 1.13: Show Hierarchy Path
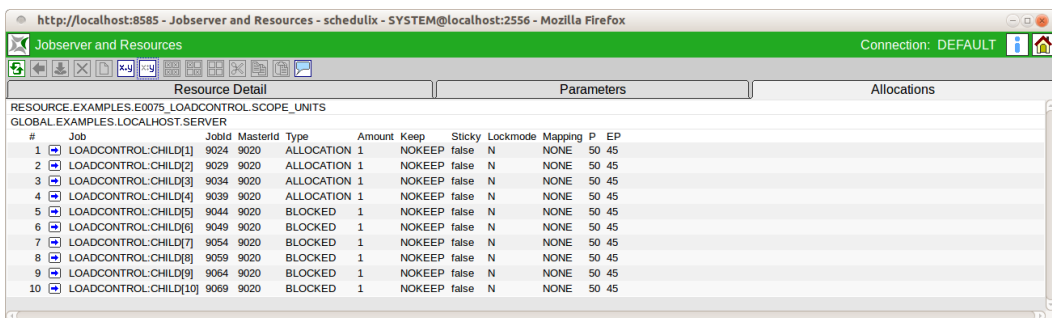
Figure 1.14: Hide Hierarchy Path

Select All

This button is for selecting all the objects. This means that all the objects currently displayed in the Navigator or the Editor window (depending on the button bar where the button is) are selected. All the objects are then marked with a cross ☒ in front of their name.

Once all the objects have been selected, they can then be cut, copied or deleted.

Toggle Selection

This button is for reversing the current selection. This means that all the selected objects are deselected and all the objects that have not yet been selected are selected. The selected objects can now be deleted, cut or copied.

This behaviour is illustrated in the following two screenshots.

Your selection is reversed when you click the Toggle Selection button.
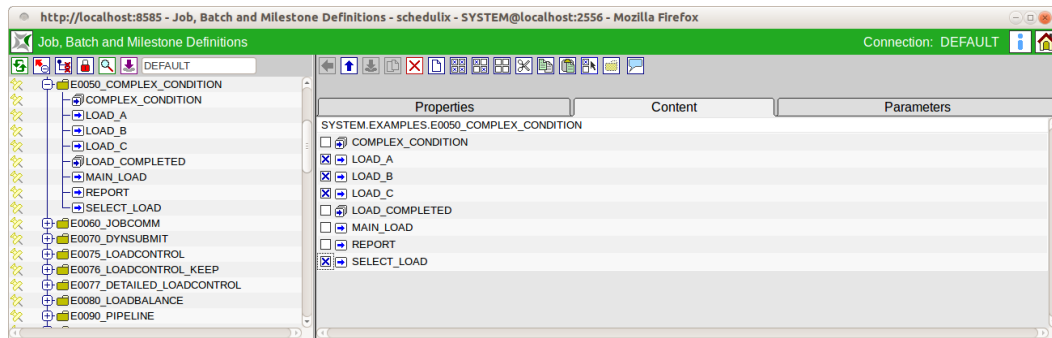
Figure 1.15: Reverse Selection; Initial State
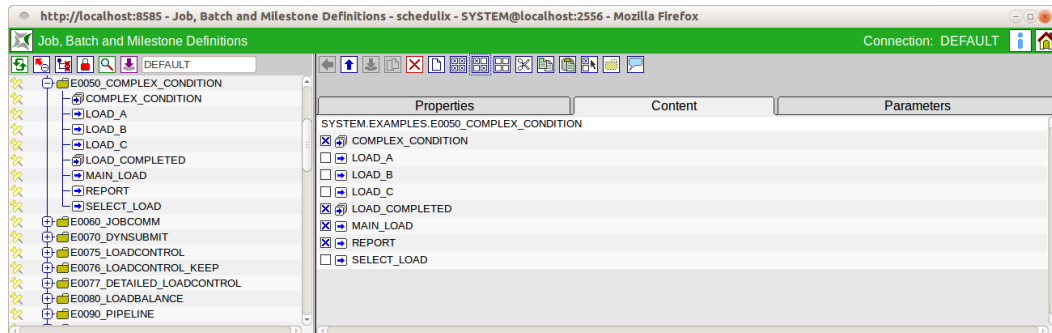
Figure 1.16: Reverse Selection; Result

Deselect All

This button reverses your selection and the markings on the selected entries are removed. This allows you to undo a selection that you have made.

Cut

This button is used to cut out objects out of a list or a tree. The objects are copied to the schedulix clipboard. The objects are only marked for cutting, but they disappear from the current view. They can be inserted somewhere else by clicking the *Paste* button.

Copy

With the *Copy* button, objects can be copied to the schedulix clipboard for inserting them somewhere else by clicking *Paste*.

Paste

This button is used to paste previously selected and suitable objects into a list. The objects have to be selected beforehand in another dialog or pop-up window.

Select

With this button you can open a window where you can select objects that are suitable for a list and copy them to the clipboard by clicking the *Copy* button. The objects in the clipboard can then be inserted into the list by clicking the *Paste* button.

Time Scheduling

This button opens the Time Scheduling dialog window. It is only displayed if you have selected a Master Submittable Object in the navigation window.

Grants

This button opens the dialog for administering the access privileges.

Edit

The *Edit* button activates an edit mode. What data is edited depends upon the context.

1.5 Selecting values

1.5.1 Selecting values from drop-down lists

With some input fields it is necessary to make a selection from a range of default values. In these cases, a *drop-down* box is displayed containing a list of all the possible values.

A *drop-down* box looks like this:
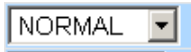
NORMAL ▾ A single value is selected from the drop-down list by expanding it by clicking the button next to the box or using the keyboard shortcut ALT-DOWN ARROW.

The expanded drop-down list can look like this:

NORMAL ▾
NORMAL
CASCADE Clicking the required value places it into the field. Alternatively, you can select the values using the DOWN ARROW or UP ARROW keys.

1.5.2 Choosing values using the Select button

If the list of available values is only read from the server at runtime you have to use the *Select* button. In the next screenshot, a Resource State Profile can be chosen by clicking *Select*:
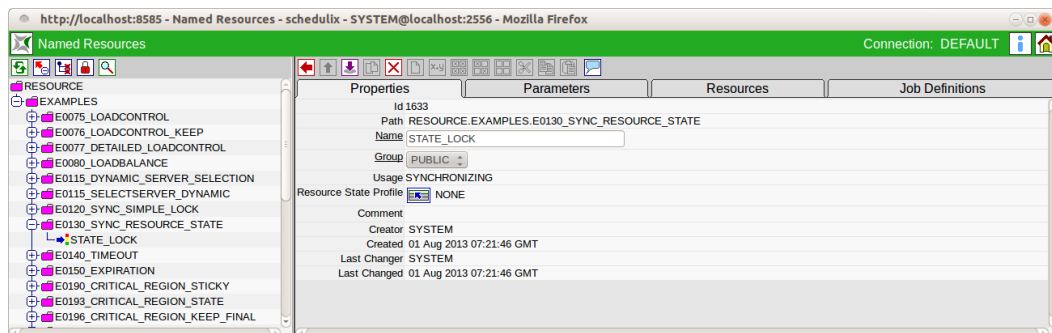
Figure 1.17: Choosing a Resource State Profile; initial state

When you click the *Select* button,

a list of possible objects is displayed in the Editor.

Clicking one of these objects places the element into the field. You can jump back from the selection mask to the editing window without making a selection by clicking the *Cancel* button.
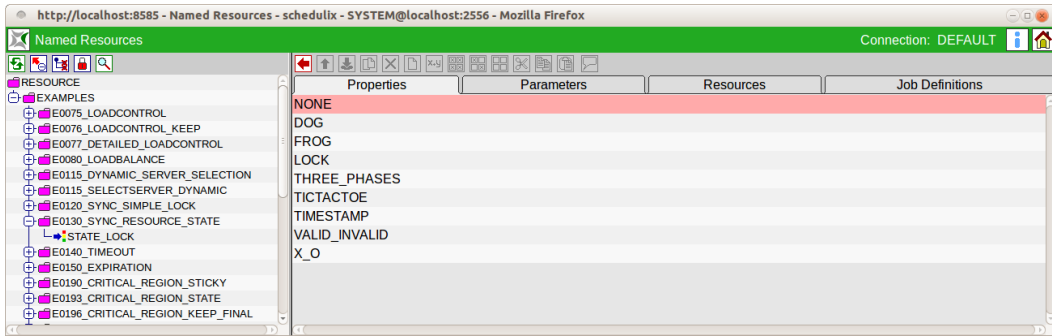
Figure 1.18: Choosing a Resource State Profile; selection list
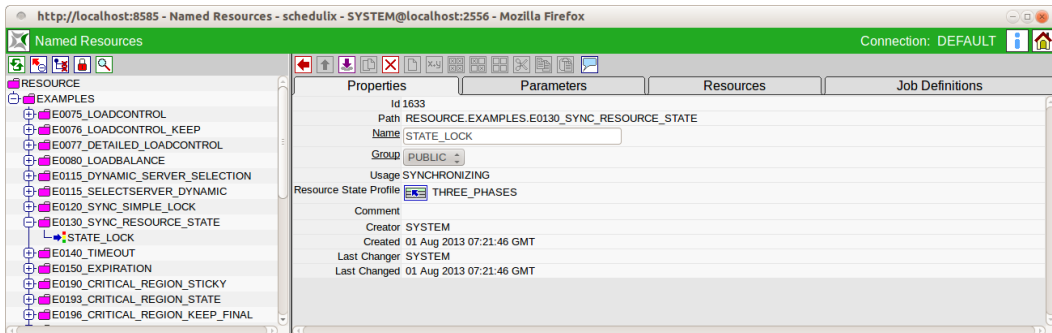
Figure 1.19: Choosing a Resource State Profile; result

1.6 Handling standard lists

In many dialog windows it is possible to not only enter values into the fields, but you can also create lists with links to additional objects. This way, for example, a list of Resource States belonging to a Resource State Profile is administered in that profile. Here is an example:
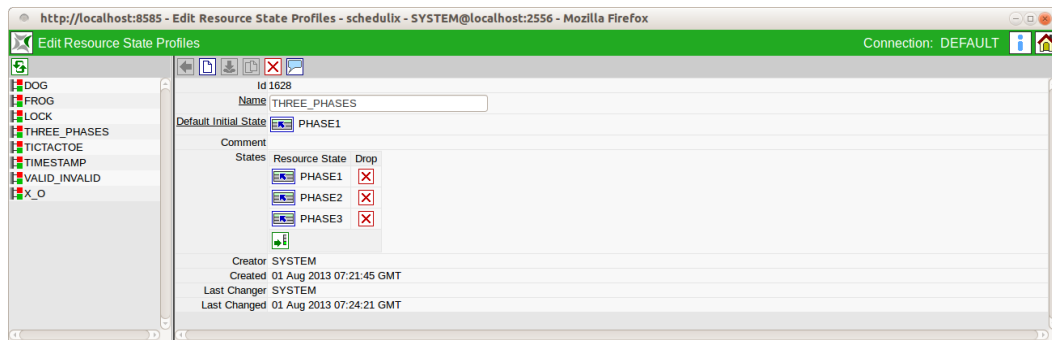
Figure 1.20: Example of how standard lists are handled

Here, to the standard fields *Name* and *Default Initial State* we want to append a list of states.
The following actions can then be performed in such a list.

1.6.1 Adding a line

Add

A new line is added to the open list by clicking the *Add* button. An empty line appears after you have clicked the button.
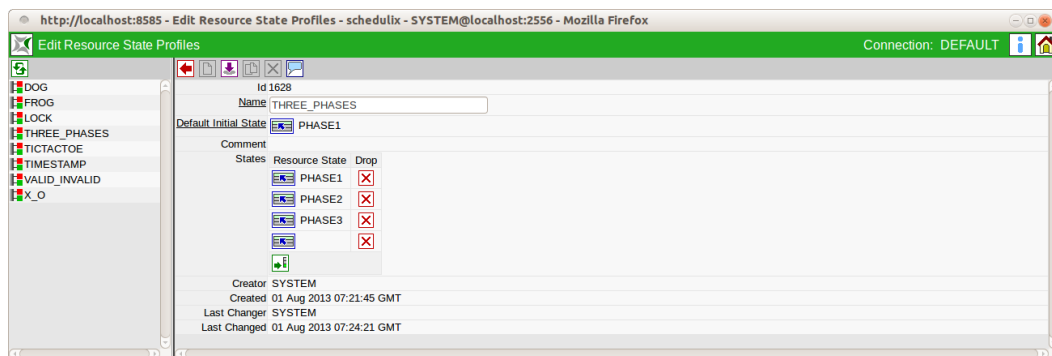
Figure 1.21: Handling lists; adding a line

In our example, the dialog looks like in figure 1.21.

A value can then be chosen using the Select button.
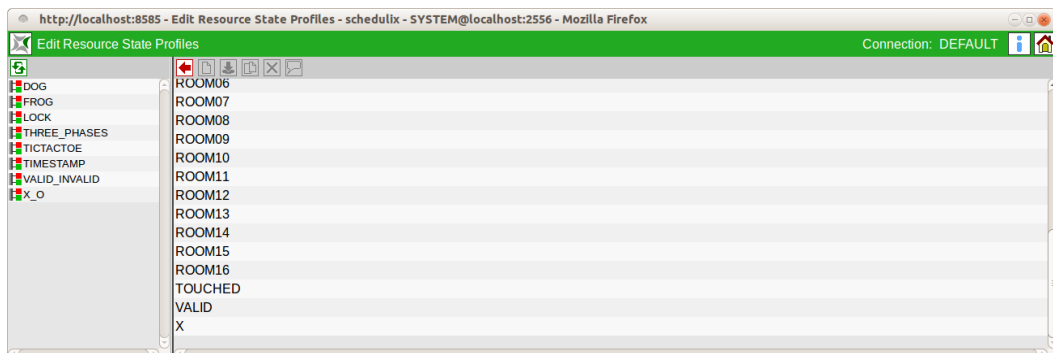
Figure 1.22: Handling lists; choosing a value

In this example, the value "Valid" was chosen.
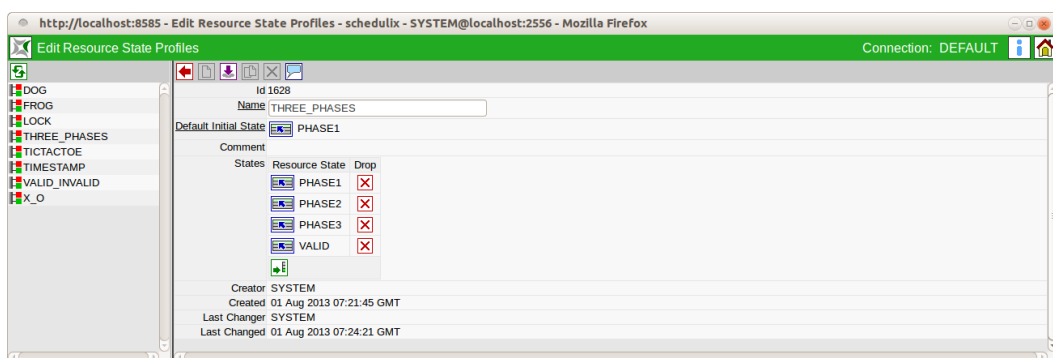
Figure 1.23: Handling lists; result

You now have to save the change with the *Save* button.

You can also modify an existing line by choosing a dependent object using the *Select* button. In this case, the link to the old object is deleted and a new link is created to the selected object.

1.6.2 Deleting a line

Drop (delete)

When you click the *Drop* button, this removes a line and the link between the object loaded in the Editor and the selected object in the line is deleted after you have confirmed the prompt. In our example, clicking the *Drop* button will delete the line "PHASE1".
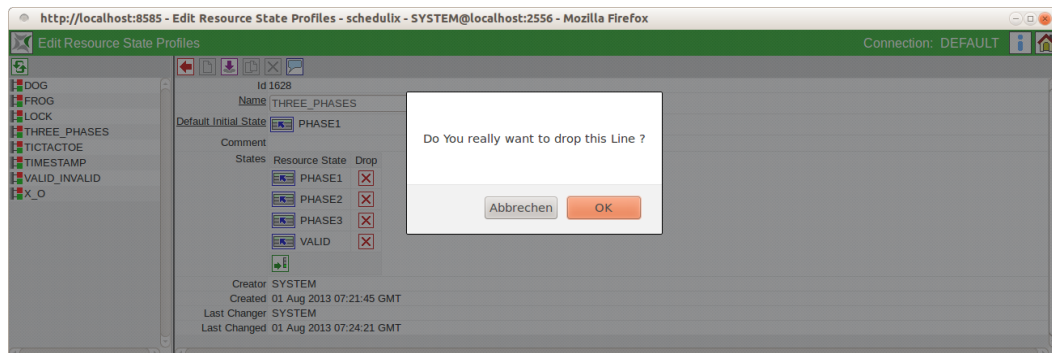
Figure 1.24: Handling lists; deleting a line

The line with "PHASE1" in it no longer exists once you have clicked *OK*. The object in the line (i.e. the Resource State "PHASE1") still exists, but it is no longer allocated to this profile.

The change now has to be saved on the server by clicking the *Save* button.

1.6.3 Changing line values

In many dialogs, not only does a line hold additional properties from other objects, but additional fields also have to be maintained for this link. These are shown in the list as a field string after the linked object. In the Footprint dialog, for example, not only does a *Named Resource* have to be selected in a line, but the other fields *Amount* and *Keep* need to be maintained as well.
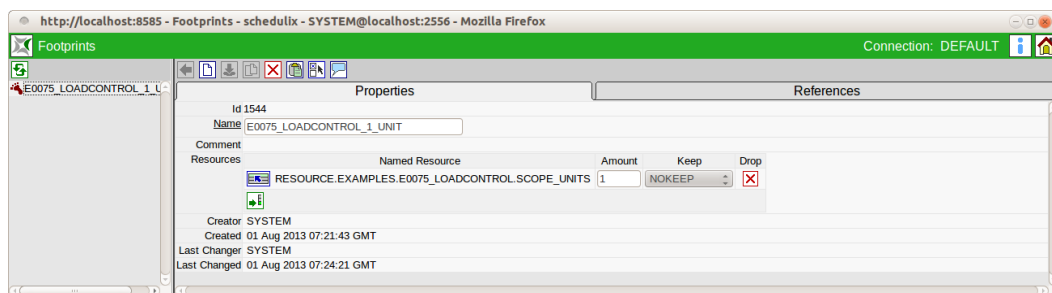
Figure 1.25: Handling lists; editing

This is done analog to how standard fields are edited.

The changes now have to be saved by clicking the *Save* button.

1.7 Copy & paste and the clipboard

The schedulix!Web user interface features an intelligent copy & paste functionality and an object-sensitive clipboard. This allows objects to be selected, cut, inserted and deleted between different dialog boxes or in the same dialog. So long as the action has not been performed, the data is located in the clipboard of the schedulix!Web interface system and can be used by other suitable dialogs.

The term *object-sensitive* describes the logic routine executed when inserting objects from the clipboard. Before objects are inserted from the clipboard, the system checks whether they are of the same type as the required objects in the dialog into which they are to be inserted. It is therefore not possible to insert an object of type A if the recipient dialog is expecting an object of type B. If there are no objects of type B in the clipboard, the insert operation cannot be performed.

This mechanism is not to be confused with the standard clipboard system used by Windows operating systems. The schedulix clipboard functions totally independently of the Windows clipboard and vice versa. The data stored in the schedulix clipboard is available exclusively to schedulix dialogs and cannot be used or read out by other Windows applications.

How the clipboard is used is demonstrated in two examples in the next sections.

1.7.1 Moving objects in the hierarchy

In a folder hierarchy, you can use the clipboard to move objects to another level in the hierarchy.

Example:

The navigation window for the *Batches and Jobs* dialog shows the following hierarchy:
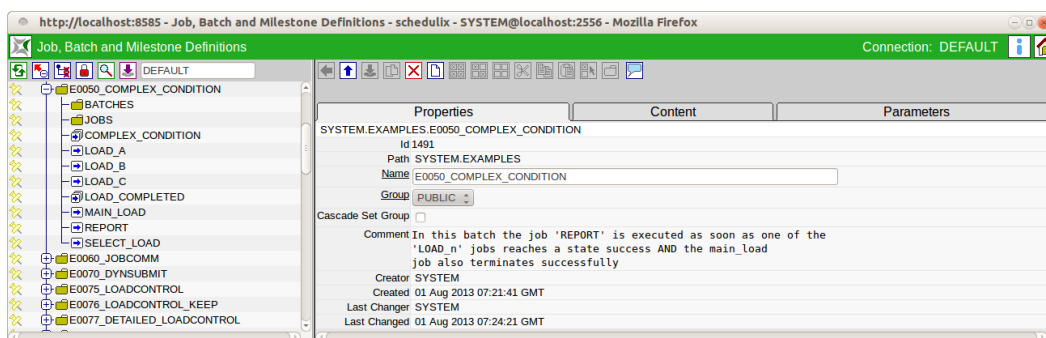
Figure 1.26: Moving objects; initial state

Below the COMPLEX_CONDITION folder there are various jobs and batches. This folder needs to be reorganised. The batch "COMPLEX_CONDITION" is to be

moved to the BATCHES folder, and then the jobs "LOAD_A", "LOAD_B" and "LOAD_C" are to be moved to the "JOBS" folder.
This can be done as follows using the clipboard:

1. Select the objects to be moved

   First you select the batch COMPLEX_CONDITION in the Navigator. The list of all the objects in the COMPLEX_CONDITION folder are now displayed in the CONTENT tab in the Editor.
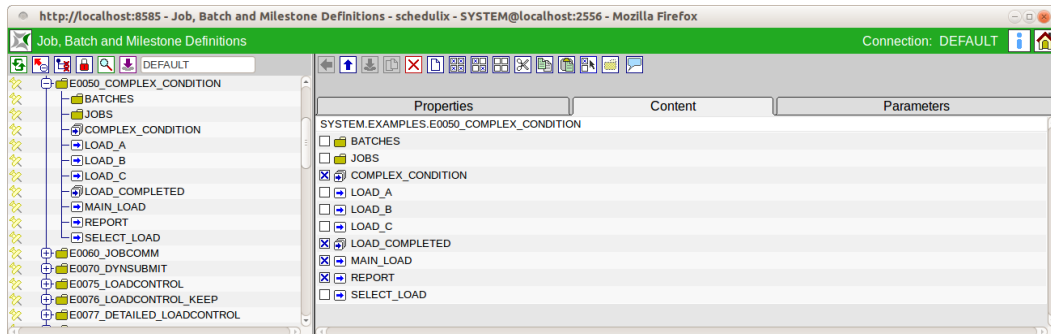
Figure 1.27: Moving objects; Content tab in the Editor

   The batch is now marked by clicking the check box in front of the name and the icon.
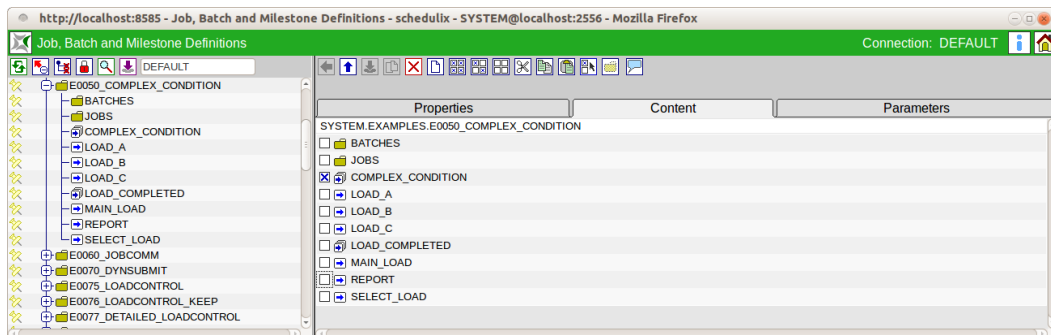
Figure 1.28: Moving objects; selection

2. Cutting out the selected objects
   Using the *Cut* button, the selected batch is cut out and disappears from the screen view. The object has now been saved to the clipboard. The dialog looks like this after the batch has been cut out:

   The important to remember here is that although the object that has been cut out is no longer visible, it is still in the folder. So if you close the window now,
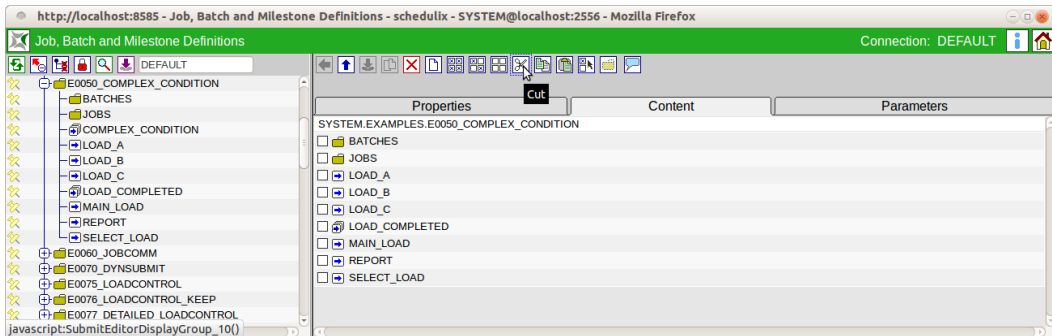
Figure 1.29: Moving objects; cut

nothing has happened. It is only actually moved during the subsequent paste operation.

3. Selecting the target folder

   We now need to choose the target dialog and target folder for the move operation. This can be done simply by clicking the respective folder in the navigation pane of the target dialog. As the clipboard functions between different windows, the target folder can also be located in another dialog.

   In our example, we have chosen the "BATCHES" folder and the "Content" tab opens.
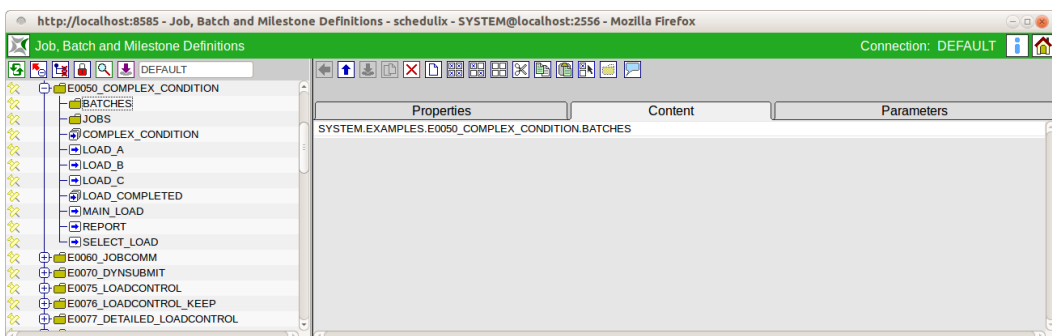
Figure 1.30: Moving objects; choosing the target folder

4. Inserting the selected objects

   To complete the move, the object is inserted from the clipboard. This is done using the *Paste* button. Pasting the object now triggers the move action on the server. This action has now been completed and cannot be undone. The result looks like this:
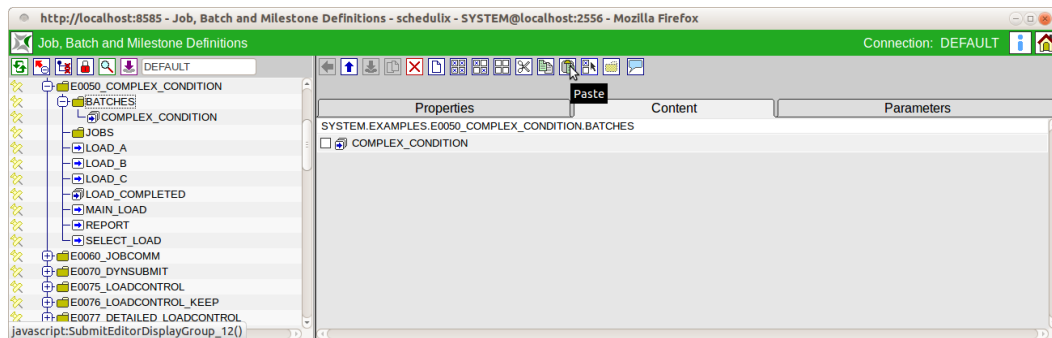
Figure 1.31: Moving objects; paste

Performing the action with the aforementioned jobs and the JOBS folder produces the following hierarchy:
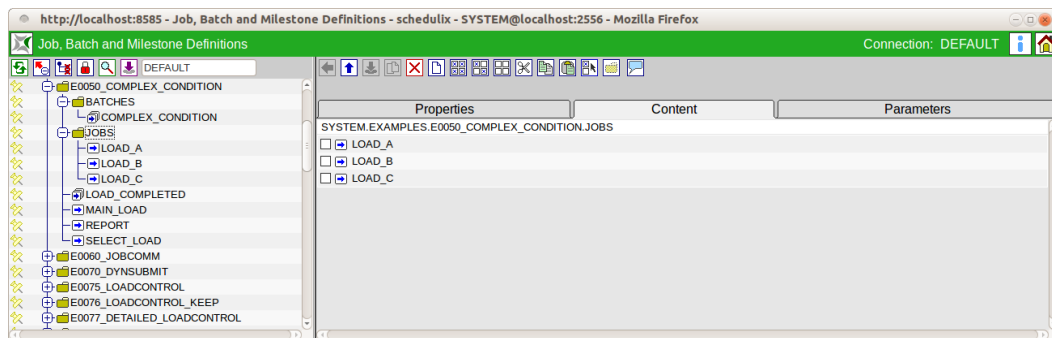
Figure 1.32: Moving objects; result

24

1.7.2 Linking objects

In some dialogs, objects are linked to other objects or lists of objects. An example of this is the definition of dependencies. The clipboard can also be used to fill such a list.

To illustrate this and demonstrate the procedure for such an action, we will define the dependencies for a job.

Here you can see the job "END_PROCESSING", which does not yet have any dependencies:
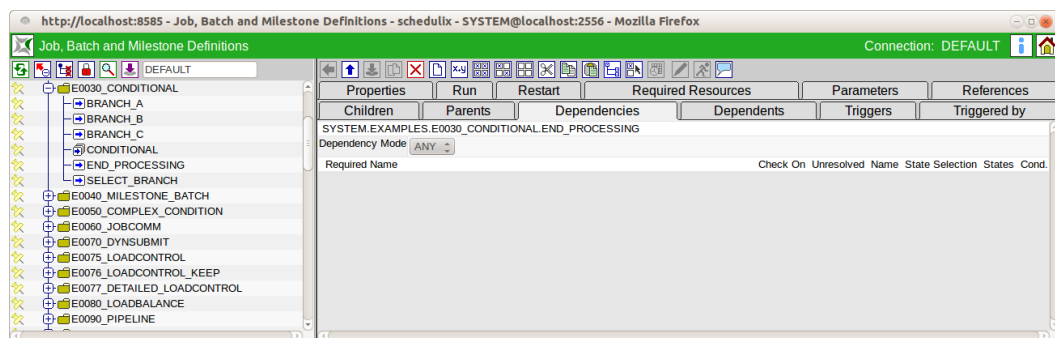
Figure 1.33: Linking objects; initial situation

Additional jobs have to be added at this point to make the execution of this job dependent upon other processing steps. These can be marked beforehand in other dialogs and copied to the clipboard. We now take the following steps:

1. Selecting the objects

   Jobs can be marked in the "Content" tab of a folder, for example. In the next dialog, we have selected the three processing steps A, B and C.
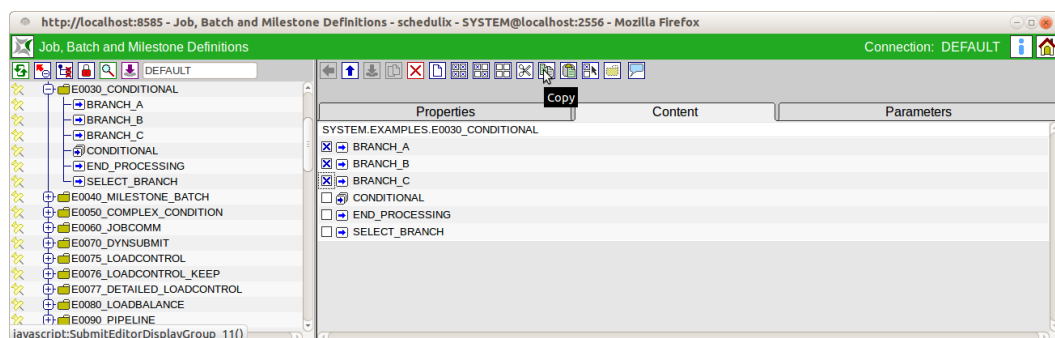
Figure 1.34: Linking objects; selection

2. Copying the objects to the clipboard

Clicking the *Copy* button copies the selected jobs to the clipboard.

3. Inserting the objects

   In the Dependency Editor window of the "END_PROCESSING" job, we now click the *Paste* button and the cached jobs are added to the list. The list now looks like this:
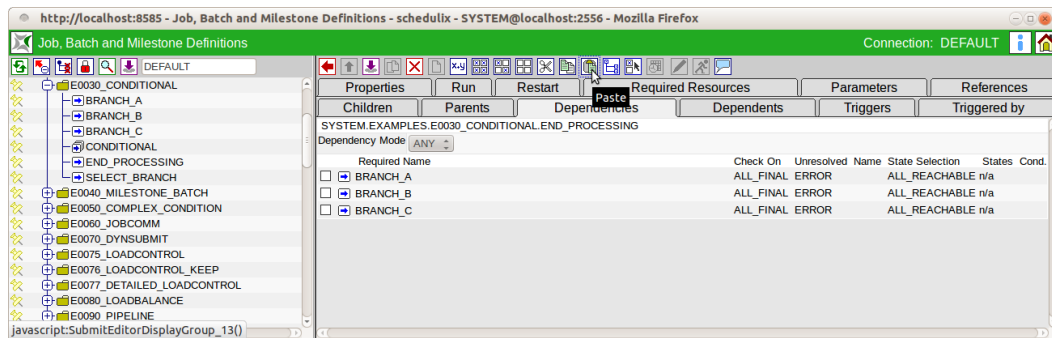
Figure 1.35: Linking objects; paste

4. Saving the change

   In contrast to the action for moving objects, which does not have to be additionally saved, copy actions in an editor window do need to be saved by clicking the *Save* button.

   Important: The cached data remains in the clipboard. If the selected jobs are to be added to other dialogs, you just have to choose the appropriate target and then click the *Paste* button.

1.8 Graphic Legend

Different graphics are used in this documentation to illustration correlations.

1.8.1 How schedulix objects are represented

The following symbols and their meanings are used in the graphics for schedulix objects:
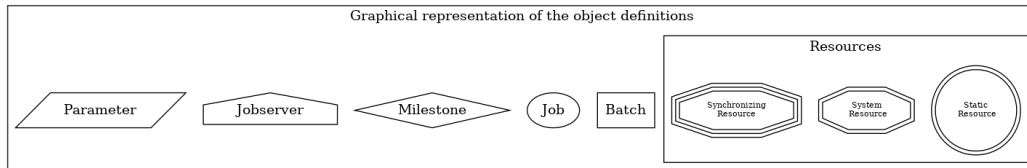
Figure 1.36: Legend for the graphical representation of schedulix objects

1.8.2 Representation of parent-child relationships in workflows

Relationships between parents and children are represented in the graphics as follows:
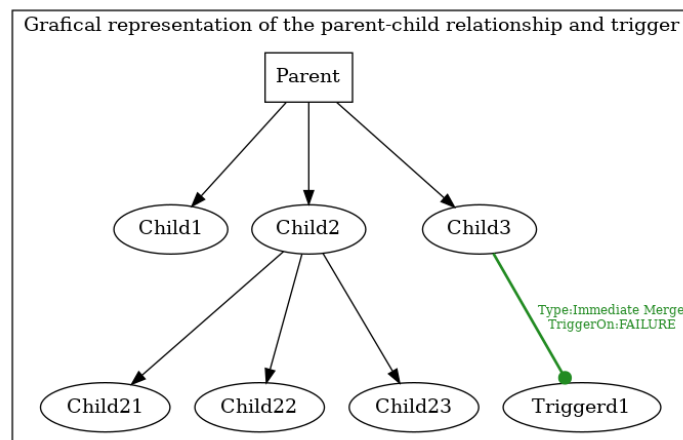
Figure 1.37: Legend for the graphical representation of parent-child relationships

The family tree is usually read from top to bottom from the highest ranking parent down to the respective children. The arrow always points from the parent to the children.

More details about parent-child relationships between tasks can be found in Chapter Batches and Jobs.

1.8.3 Representation of dependencies between Scheduling Entities

Dependencies between individual Scheduling Entities are represented in the graphics as follows:
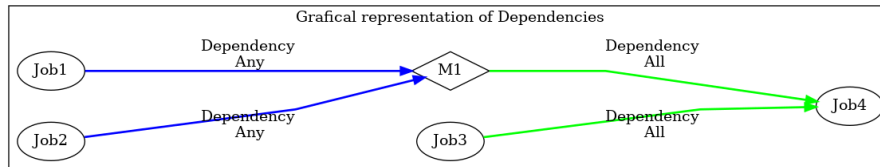
Figure 1.38: Legend for the graphical representation of dependency relationships

Dependencies are displayed from right to left. This means that the first job is shown on the far left, then comes the next one time-wise, etc.

The arrows always point from the required Scheduling Entity to the dependent Scheduling Entity. This thus shows the timeline for the Scheduling Entities. For example, A→B→C means that A is run first, and is followed by B first and then C. A is the required entity of B and B is the required entity of C. B is accordingly dependent upon A. C is dependent upon B (and so implicitly also dependent upon A).

The colour of the arrows corresponds to the interlinking of multiple dependencies. If the colour is green, all the required tasks are linked with "AND" (the dependency ALL).

If the colour is blue, all the required tasks are linked with "OR' (the dependency ANY).

If a task is only dependent upon precisely one other task, the arrow is green since in this case the links AND and OR are equivalent.

More details about dependency relationships between Scheduling Entities can be found in Chapter Batches and Jobs.

1.9 Comments

Comments can be saved for most objects.

Comment

Clicking the *Comment* button opens the *Edit Comments* mask. Here you can enter any text comments relating to the object. Text comments consist of a tag and a comment text. On display tags will be rendered as headlines. If the tag is left empty, the comment text will be concatenated with the preceeding text comments comment text. Apart from entering the comment, no other action is performed on the object.

As an alternative to text comments, a link can be set to an intranet page instead. To do this, select "URL" as the type and enter the URL in the comment box. The links are also activated in the schedulix!Web interface. This allows any number of complex documents to be linked with the schedulix objects.
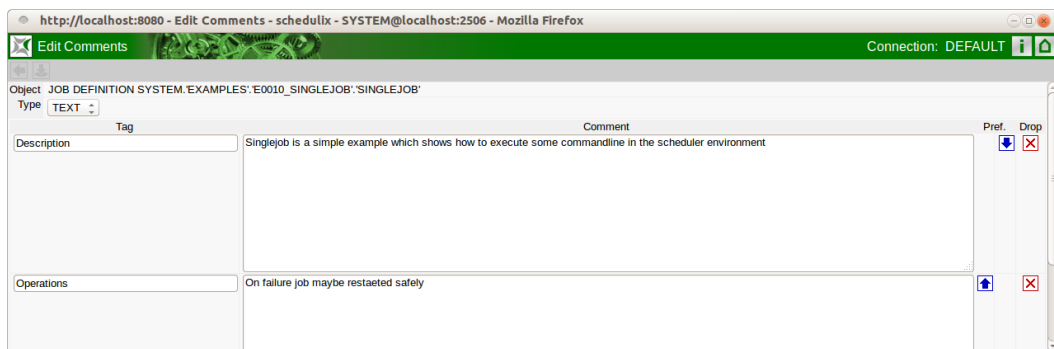
Figure 1.39: Recording comments

1.10 Standard fields

The following fields are defined for all objects:

Creator   Name of the user who created the object.

Created   Date and time at which this object was created.

Last Changer   Name of the last user who changed the object.

Last Changed   Date and time at which this object was last changed.

2 Exit State Definitions

2.1 View
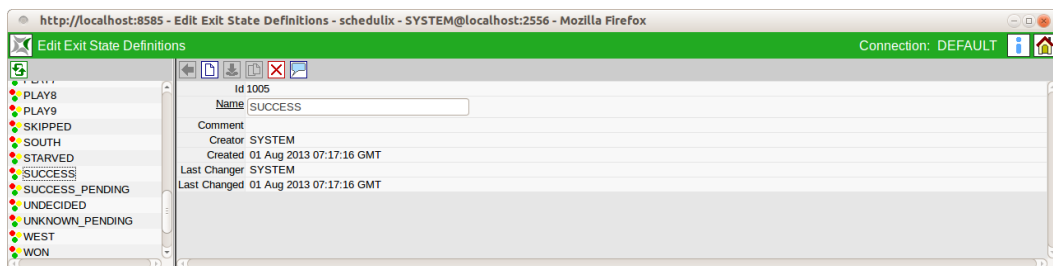
Figure 2.1: Exit State Definitions

2.2 Concept

2.2.1 In short

The Exit State Definitions dialog is used to create logical names for exit states of a Submitted Entity.

2.2.2 Detailed description

Exit State Definitions are logical names for exit states of a Submitted Entity. These logical names do not have any other properties.

2.3 Editor

The properties of Exit State Definitions are maintained in this tab.
Exit State Definitions can only be edited by users who belong to the "ADMIN" group.
All the input fields are "read only" for all other users.

The fields have the following meanings:

Id   System-wide unique number for identifying the object.

Name   Unique name of the Exit State Definition. This name can be freely chosen.

3 Exit State Mappings

3.1 View
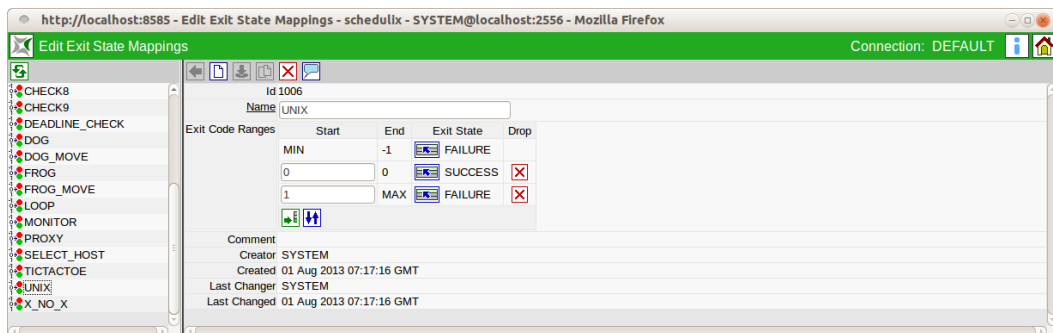
Figure 3.1: Exit State Mappings

3.2 Concept

3.2.1 In short

Exit State Mappings are used for translating numerical Exit Codes into logical Exit States.

3.2.2 Detailed description

Exit State Mappings are used to assign numerical value ranges of exit codes to logical Exit States. Each program that is run returns a numerical value to the calling program when it has finished. On UNIX systems, a value of 0 usually means that the program has been completed without any errors (SUCCESS). Values unequal to 0 indicate an error (FAILURE). However, deviating values with other meanings are also possible. For a better understanding of this logical meaning within schedulix Exit State Mappings can be used to assign value ranges to a logical name. The values range from -2^{31} to $2^{31} - 1$.

3.3 Editor

Names are assigned to numerical value ranges in this editor.
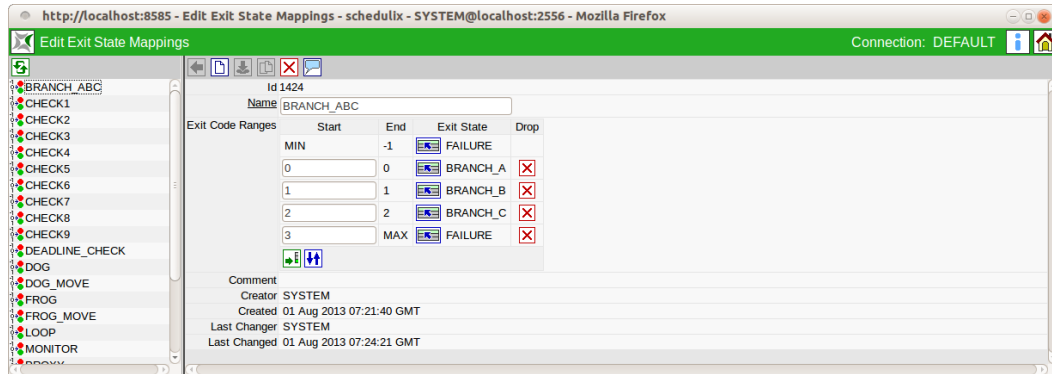
Figure 3.2: Exit State Mapping Editor

Exit State Mappings can only be edited by users who belong to the "ADMIN" group. All the input fields are "read only" for all other users.
Descriptions of all the standard buttons can be found in Chapter 1.4.3.1 Standard buttons.
The buttons in the list shown above have the following meanings:

Insert new value range

This button adds another row to the table of Exit Code Ranges. The row is inserted at the end of the table and allows a new start value to be entered. This new end value is determined automatically when you save or sort the data.

Resort list

An unsorted list that is created by adding new intervals can be arranged in the correct order with this button.

Select Exit State

This button is for selecting a previously defined Exit State from a list.

Delete value range

This button deletes a range of values together with the Exit State from the list. The start value of the next interval is replaced by the start value of the deleted interval, i.e. the next interval is enlarged automatically.

The fields have the following meanings:

Id System-wide unique number for identifying the object.

Name Unique name of the Exit State Mapping. This name can be freely chosen.

Exit Code Ranges The assignment of the respective value range is shown in a table. An Exit State can be assigned to a range of values in each row. The first row contains the start value -2^{31}, while the last row has the end value $2^{31} - 1$.

4 Exit State Profiles

4.1 View
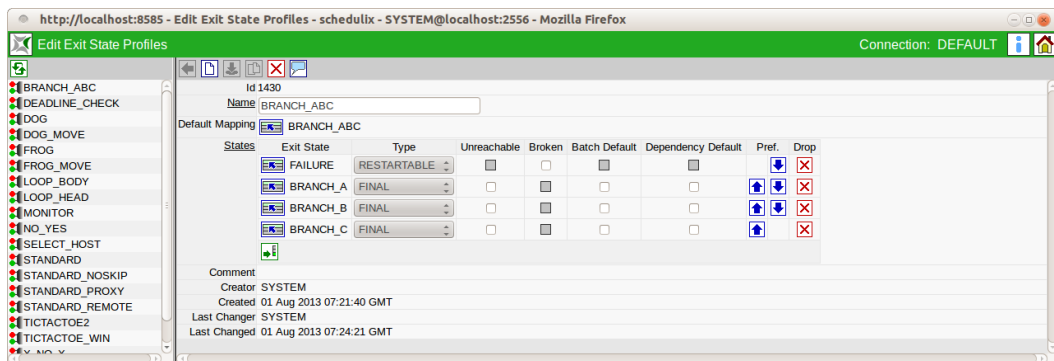
Figure 4.1: Exit State Profiles

4.2 Concept

4.2.1 In short

An Exit State Profile describes which Exit States can be achieved by a Submitted Entity. The Exit State Profile also defines whether a Submitted Entity can change its Exit State or if the Exit State is conclusive (FINAL).

4.2.2 Detailed description

Each Submitted Entity can finish with varying Exit States. The set of all valid Exit States for a Submitted Entity is described in an Exit State Profile. This description also defines the preferred Exit State that is crucial in a hierarchical workflow structure for determining the correct Exit State for a parent element. The Exit State with the highest preference is used.

4.3 Editor
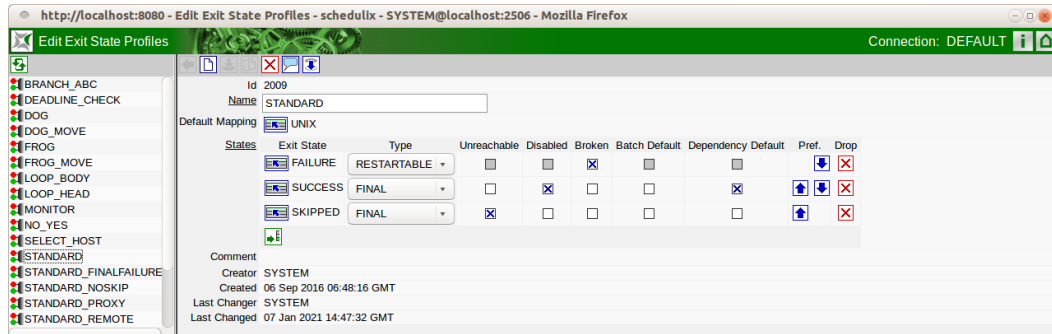
Figure 4.2: Exit State Profile Editor

In this editor, the Exit States belonging to a profile can be defined and the properties *Type*, *Unreachable*, *Broken* and *Preference* can be added to them.

Exit State Profiles can only be edited by users who belong to the "ADMIN" group. All the input fields are "read only" for all other users.

The fields shown above have the following meanings:

Name   The *Name* field is used for assigning a unique name to an Exit State Profile.

Default Mapping   The *Chooser* button is for selecting an Exit State Mapping from a list that is then used by default in conjunction with the Exit State Profiles provided that no other mapping has been defined in the job definition. An Exit State Mapping is used to translate numerical Exit Codes into Exit States.

The profile must contain an entry for each of these Exit States. However, not all Exit States in an Exit State Profile need to be accessible via the mapping.

Type   One of three types can be assigned to each Exit State: FINAL, RESTARTABLE and PENDING.

The Exit State can no longer be changed once a task has attained its Final State. The result is final. Dependencies between jobs and batches can only be based on such FINAL Exit States.

If a task is to be capable of being restarted after it has been run, for example because an error has occurred, the Exit State must be of the type RESTARTABLE.

PENDING describes a state type that allows the Final State to be set externally via the API. A task in a PENDING state can be neither restarted nor are any dependencies fulfilled by it.

An example of how PENDING states are used:

38

A job sends an e-mail to an employee and requests the release of a result. After sending the e-mail, the job stops with a PENDING state. The employee can now manually set the state to FINAL. Alternatively, a process evaluating the response e-mail can change the status. Dependent tasks can only be started once the state is FINAL.

Unreachable   A maximum of one Exit State in the list can be marked as being Unreachable. This Exit State is set when a job in a process can no longer be executed because the dependencies cannot be fulfilled.
The Unreachable State is not attained if one or more predecessors have been cancelled. Because the cancel operation is a manual intervention, the consequences of the intervention have to be handled manually as well.
The Unreachable State must be of the type FINAL.

Disabled   A maximum of one Exit State in the list can be marked as being Disabled. This Exit State is set when a job in a process is disabled and thus not executed. If Disabled has not been set, the FINAL state with the lowest priority preference is used which (if present) has not been set as being UNREACHABLE.

Broken   A maximum of one Exit State in the list can be marked as being Broken. This state is set when a job is put into an Error State due to a failure. Such an error occurs, for example, if the run program cannot be started. The Error State must be a RESTARTABLE State. The Broken flag allows a trigger to be used to automatically respond to such error situations.

Batch Default   A final Exit State in the list can be labelled as the Batch Default. This state is set when a batch or milestone with this profile has no children and therefore also has no defined Exit State. If a Batch Default has not been set, the state with the lowest priority preference is used which (if present) has not been set as being UNREACHABLE.

Dependency Default   One or more final Exit States can be labelled as the Dependency Default. If a dependency between batches,jobs and milestones is created by selecting the DEFAULT state, the Exit States of a profile that are marked as being the Dependency Default fulfil the dependency condition regarding the Exit State.

Preference   The *Preference* buttons are used to set the preference of the individual Exit States. The rows are shifted up or down by clicking the buttons. A higher position indicates a higher preference.
To determine the resultant Exit State of a Submitted Entity, the Exit State with the highest preference is selected from the list of Exit States of the children and its own

Exit State. In doing so, an Exit State is only taken into account if it is present in the Exit State Profile of the parent or if the state is translated into an Exit State of the parent using an Exit State Translation.

Example: A batch has three subordinate child jobs. The batch is to display an error when at least one of its children returns an error. This means that the FAILURE state must precede the SUCCESS state.

The SUCCESS state will usually have the lowest priority, warnings have a higher priority, while Error states have the highest priority.

If an empty batch (one without any children) becomes FINAL, then there is no Exit State from which the FINAL Exit State of the batch can be determined. In this case, the Exit State from the Exit State Profile with the lowest preference is used. This is usually the SUCCESS state. The system attempts to use a FINAL state which is not the Unreachable State if one actually exists.

5 Exit State Translations

5.1 View
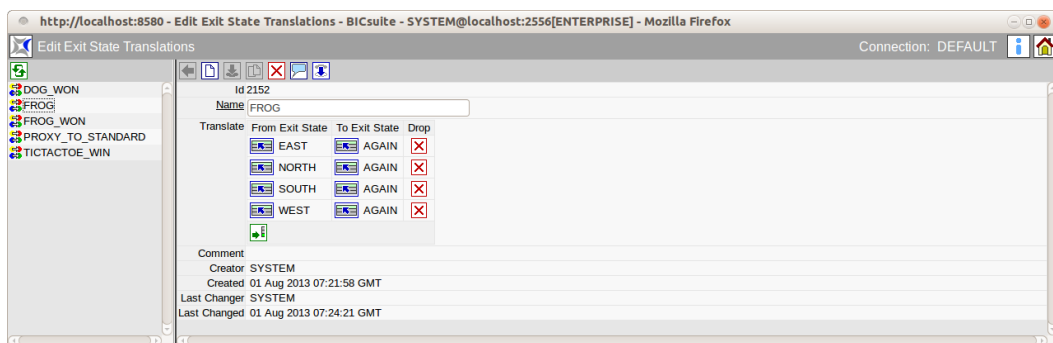
Figure 5.1: Exit State Translations

5.2 Concept

5.2.1 In short

Exit State Translations are required to translate the Exit State of a child into an Exit State of its parent.

5.2.2 Detailed description

Exit State Translations are used to translate a child's Exit States into Exit States of its parent. In the absence of a translation, the identity is used or the Exit State of the child is ignored by the parent. If the translation is available, all the Exit States have to be either implicitly or explicitly translated.

For example, if a child uses an Exit State Profile with the Exit States SUCCESS, WARNING, and FAILURE and the parent's Exit State Profile only knows SUCCESS and FAILURE, the final Exit States SUCCESS and WARNING can be translated into the parent's SUCCESS Exit State. The restartable FAILURE Exit State is translated into the parent's FAILURE Exit State.

5.3 Editor
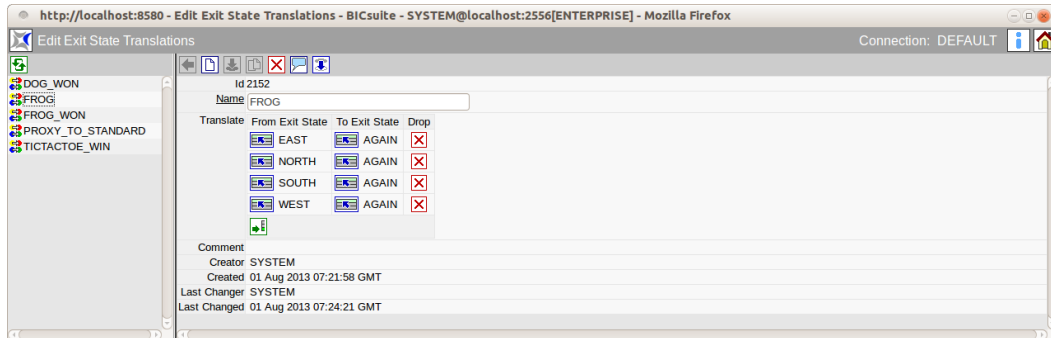
Figure 5.2: Exit State Translation Editor

Exit State Translations are defined and modified in this editor.
Exit State Translations can only be edited by users who belong to the "ADMIN" group or another group that has been granted the "manage exit state translation" privilege. All the input fields are "read only" for all other users.
The fields shown above have the following meanings:

Id The Id contains a system-wide unique number.

Name The *Name* field is used for assigning a unique name to an Exit State Translation.

Translate From/To Exit State Which state of a child is to be translated into which state of the parent is entered in each row.

6 Resource State Definitions

6.1 View
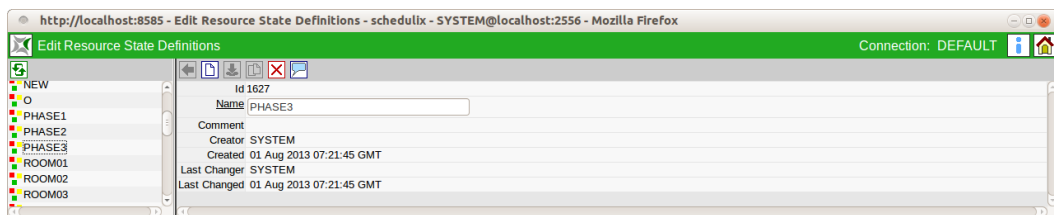
Figure 6.1: Resource State Definitions

6.2 Concept

6.2.1 In short

Resource State Definitions are used to create names for Resource States.

6.2.2 Detailed description

Resource State Definitions are logical names for Resource States. Each Synchronizing Resource can take on several states that can be set when a job has been completed.

6.3 Editor

The properties of Resource State Definitions are maintained in this tab.
Resource State Definitions can only be edited by users who belong to the "ADMIN" group. All the input fields are "read only" for all other users.
The fields have the following meanings:

Id System-wide unique number for identifying the object.

Name Unique name of the Resource State Definition. This name can be freely chosen.

7 Resource State Profiles

7.1 View
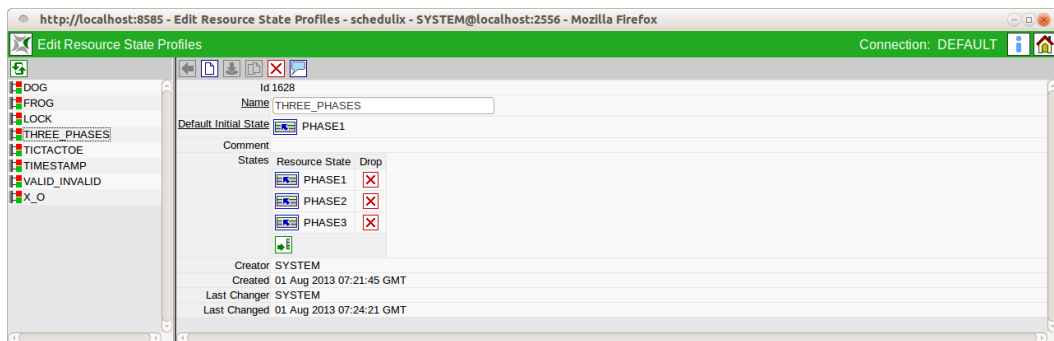
Figure 7.1: Resource State Profiles

7.2 Concept

7.2.1 In short

A Resource State Profile describes which states a resource can take on.

7.2.2 Detailed description

Each Synchronizing Resource can take on different Resource States. The set of all valid Resource States for a resource is described in a Resource State Profile. The initial state of a resource is also defined in this profile, whereby the state does not necessarily have to be present in the list of states.

7.3 Editor
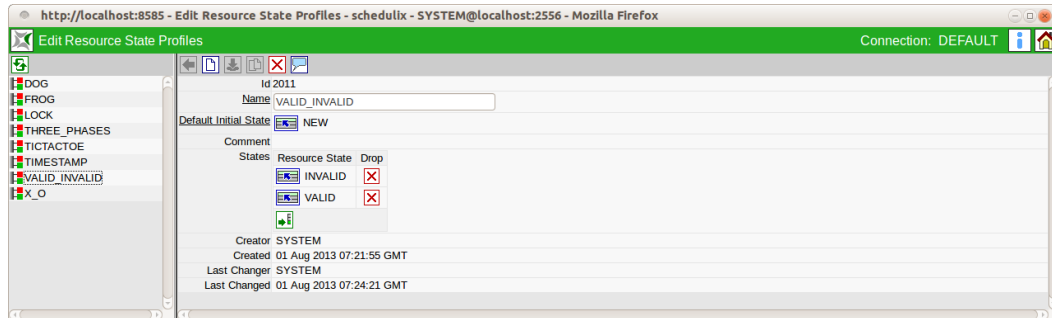
Figure 7.2: Resource State Profile Editor

The Resource States that belong to a Resource State Profile can be defined in this editor.

Resource State Profiles can only be edited by users who belong to the "ADMIN" group. All the input fields are "read only" for all other users.

The fields shown above have the following meanings:

Name   The *Name* field is used for assigning a unique name to the Resource State Profile.

Default Initial State   The *Default Initial State* field defines the initial state of the resource. This Resource State does not have to be present in the list of valid Resource States.

Resource State   The valid Resource States are shown in the *Resource State* column in the *States* table.

7.3.1 Example

A database table for a data mart is shown in the scheduling system as a Synchronizing Resource with a State Model.   The state chart looks like the diagram in Figure 7.3

The definition of a suitable Resource State Profile is shown in the screenshot above.
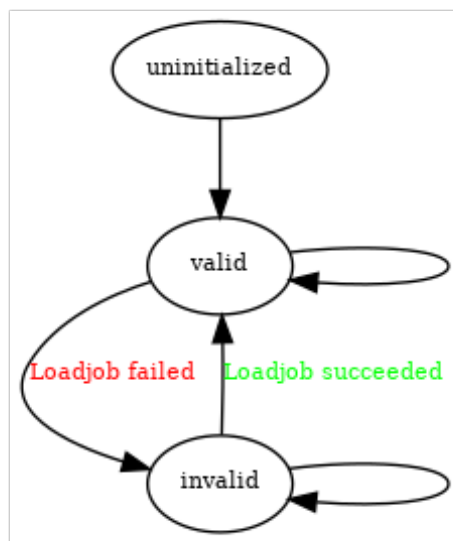
Figure 7.3: State chart for a resource

8 Resource State Mappings

8.1 View
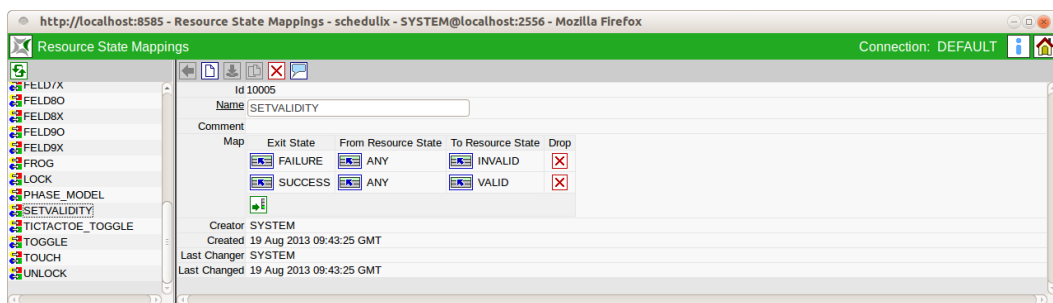
Figure 8.1: Resource State Mappings

8.2 Concept

8.2.1 In short

Resource State Mappings describe the state transition of a resource with certain Exit States.

8.2.2 Detailed description

Resources can change their Resource State after completion of a job depending upon the job's Exit State. Resource State Mappings describe these state transitions. Respectively one Exit State and one Resource State determine the new state of the resource.

For example, a "TABLE" resource can take on the state "VALID" or "INVALID". If the loading process belonging to the table is completed successfully (SUCCESS), the Resource State "VALID" is to be set. In the event of an error (FAILURE), the table is to be marked as being "INVALID". The corresponding definition of the Resource State Mapping would look like in Figure 8.2.

If no mapping exists for the combination of 'Exit State' and 'From Resource State', the status of the resource remains unchanged. This means that it is not necessary to define a mapping from SUCCESS/VALID to VALID.
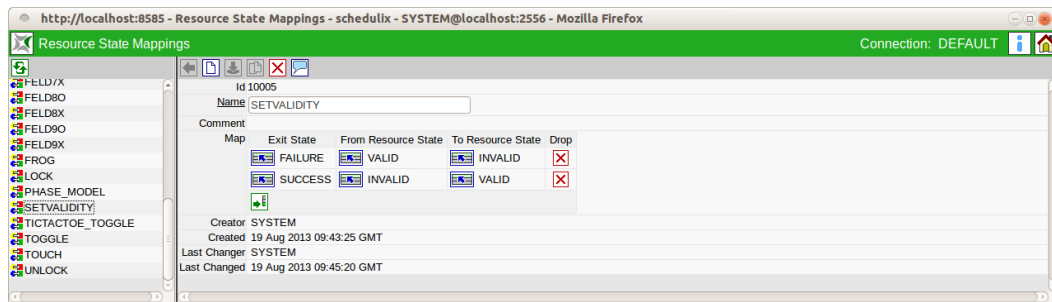
Figure 8.2: Example for a Resource State Mapping

If a state transition is to take place regardless of a Resource State, the value "ANY" needs to be selected in the line "From Resource State".

When implementing the status for a resource, the time of implementation is saved. Reference can be made to this time in the resource requirements, and so it may be practical to define the transition from "VALID" to "VALID".

8.3 Editor
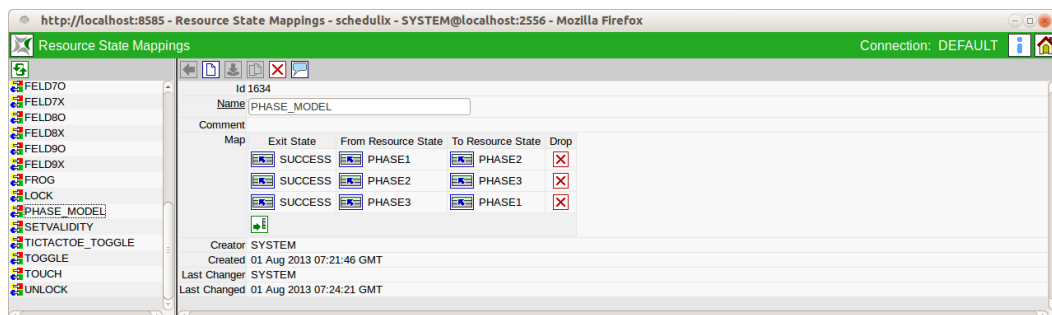
Figure 8.3: Resource State Mapping Editor

The Exit States are assigned to the resource state transitions in this editor.

Resource State Mappings can only be edited by users who belong to the "ADMIN" group. All the input fields are "read only" for all other users.

The fields have the following meanings:

Id System-wide unique number for identifying the object.

Name Unique name of the Resource State Mapping. This name can be freely chosen.

50

Exit State   The Exit State of the job is given in the field *Exit State*.

From Resource State   Either a specific Resource State or ANY is given in the field *From Resource State*. If the current state in combination with the Exit State is explicitly specified in the table, this rule is applied to determine the new state when implementing it. Otherwise, the ANY rule (if present) is used.

To Resource State   The field *To Resource State* is the resultant state of the resource.

9 Named Resources

9.1 View
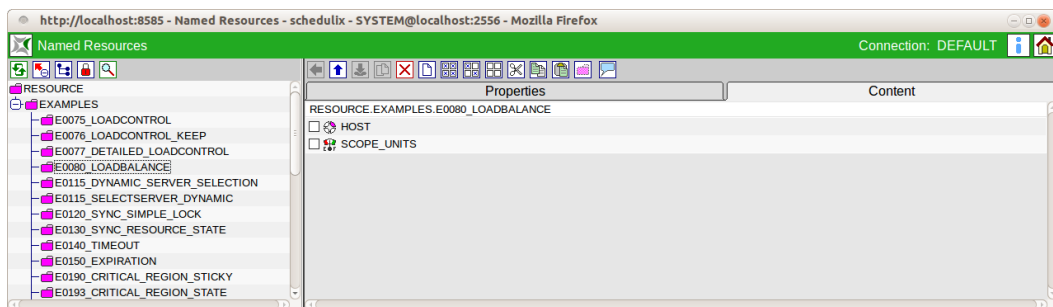
Figure 9.1: Named Resources

9.2 Concept

9.2.1 In short

This window is used to create and administer Named Resources. A Named Resource is the definition of a class of resources.

9.2.2 Detailed description

A Named Resource defines a class of resources. Named Resources can instantiated as resources within a scope, folder or in an Submitted Entity.

Named Resources belong to a group. The right to use (i.e. instantiation and requirement), edit or delete the Named Resource is primarily reserved for users belonging to that group. These privileges can be granted to other groups.

Named Resources are saved in a hierarchy of categories. Categories can be nested as required, but Named Resources cannot be nested. This categorisation serves solely to facilitate the administration of Named Resources, and otherwise has no influence on how the system functions.

9.3 Editor

Named Resources and categories are maintained in the Editor menu. When an entry is selected in the Navigator, the details for this Named Resource or category are displayed here. New Named Resources and categories are created here as well.

9.3.1 Properties tab for categories

This tab is used for maintaining the category properties. It looks like this:
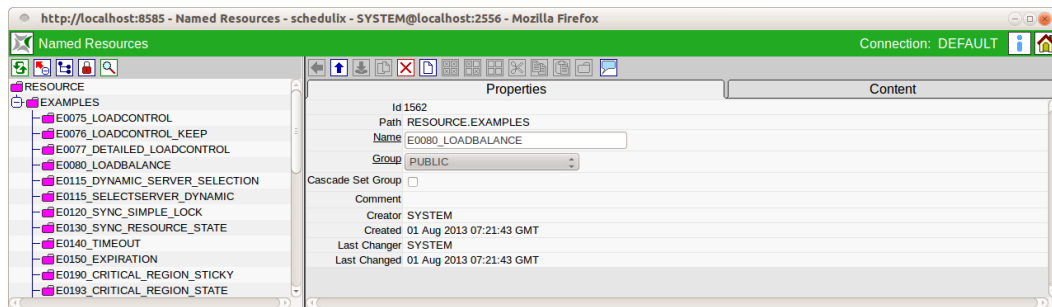
Figure 9.2: Categories; Properties tab

The "Properties" tab for a category and Named Resource has the following fields:

ID   The ID is used to unequivocally identify the category or Named Resource Definition. It is automatically assigned by the system and is unique system-wide.

Path   The path is the entire hierarchy of the viewed entry. The single hierarchy levels are separated by a period.

Name   Any name can be chosen for the category or Named Resource. It must, however, be unique within the parent category.

Group   The group to which the category is assigned can be selected from the "drop-down" list.

Cascade Set Group   If this box is checked, the group is also set in all the Named Resources and categories below this category.

Comment   The *Comment* gives a more detailed explanation about the object.

54

9.3.2 Content tab

The "Content" tab is only displayed for categories and looks like this:
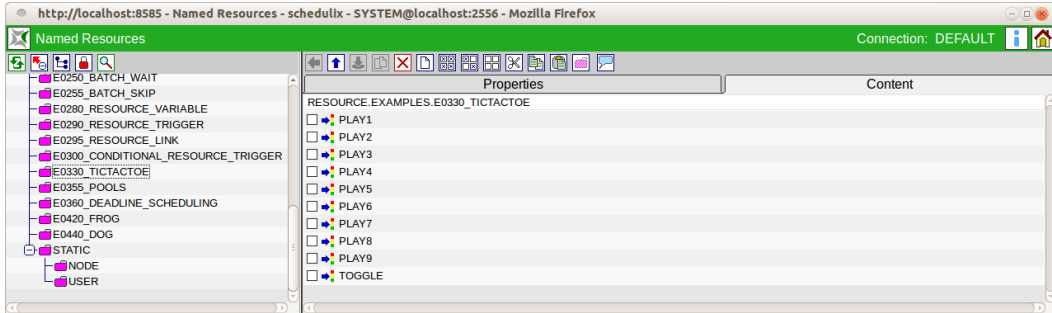
Figure 9.3: Categories; Content tab

This tab contains a list of all the Named Resources in the selected category. The following values are shown in the list:

Name of the Named Resource   The name of the Named Resource is shown here. Selecting the Named Resource's name in this dialog opens the selection window and display of the "Details" data for this Named Resource. In this tab, you can move resources using the standard cut, copy and paste operations.

9.3.3 Properties tab for Named Resource Definitions

The "Properties" tab for a Named Resource Definition looks like this:
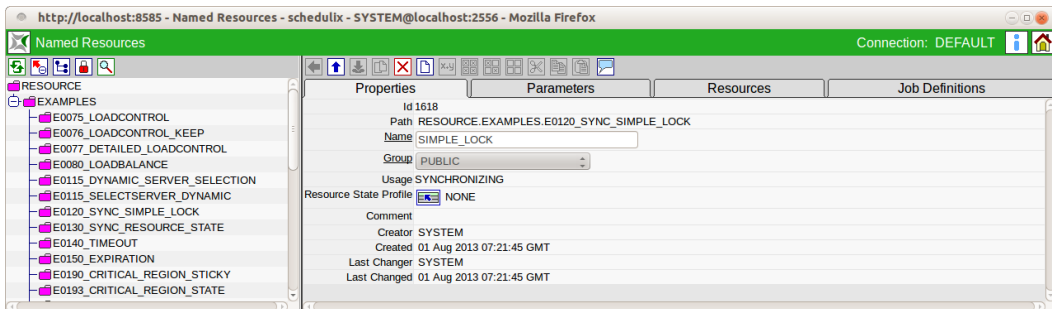
Figure 9.4: Named Resources; Properties tab

When you define a Named Resource, you have to fill in some other fields in addition to those mentioned above.

Usage   The *Usage* field specifies the Named Resource type.
List of options for the usage parameters:

1. *Category*

   Categories behave like folders and can be used to arrange the Named Resources in a clearly organised hierarchy.

2. *Static*

   Static resources describe workflow environments such as those offered by installed software packages or user environments. An instance of this Named Resource must exist and be available in order to be able to start a Submitted Entity that conditionally requires this resource.

3. *System*

   The Named Resource is a resource that maps a system parameter. This can be, for example, a CPU unit, main memory unit or a unit of disk space. This resource also has to be quantified when it is used, i.e. the creator of a job needs to qualify and quantify the requirements that have to be met by the system resources for this object. When defining a job he will have to specify, for example, that 3 CPU units, 3 units of 512 MB main memory and 10 units of 1 GB disk space are required.

4. *Synchronizing*

   Synchronizing Resources are special resources that are used to synchronise concurrently running jobs. Whether a resource requirement can be fulfilled and the resource can be allocated can be made dependable upon the current status of the resource, the last status update time, and the contending allocation of the resource by other tasks. The requirement can be quantified analogue to System Resources.

Resource State Profile   A Resource State Profile that defines the status that the Synchronizing Resource can take on can be assigned to that Synchronizing Resource. No status or timestamp-related requirements can be defined if a Resource State Profile has not been specified. Further information can be found in Chapters 6 and 7.

9.3.4 Parameters tab

Additional information about a resource that can be evaluated by jobs or Resource Triggers can be saved in the "Parameters" tab.
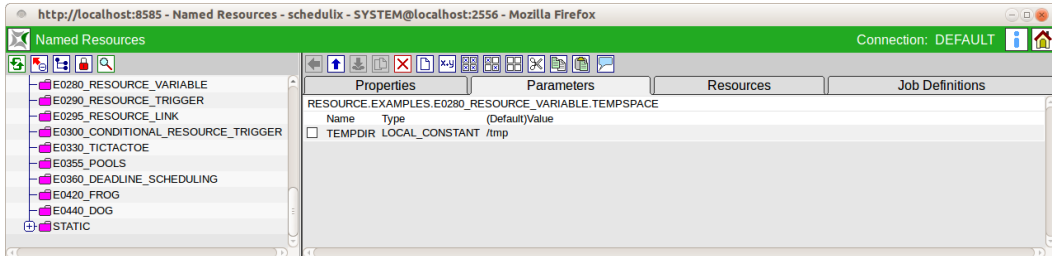
Figure 9.5: Named Resources; Parameters tab

The "Parameters" tab looks like in Figure 9.5.
The "Parameter Details" tab is opened by clicking the parameter name.

9.3.4.1 Parameter Details tab
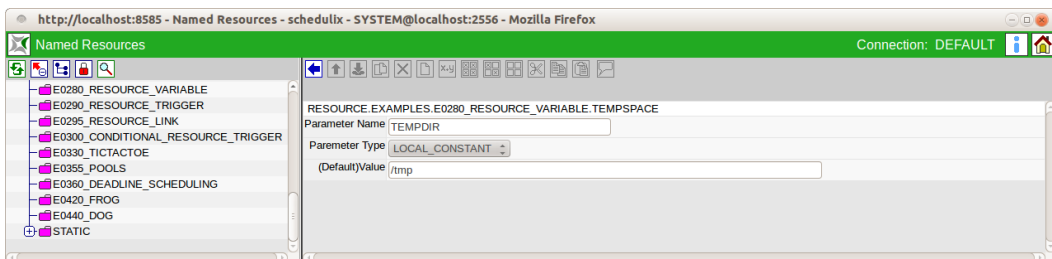
Figure 9.6: Named Resources; Parameter Details

The fields in the "Parameter Details" tab have the following meanings:

Parameter Name   Name of the parameter.

Parameter Type   Type of the parameter.
The following options are available:

- Constant: The Constant has a fixed value and applies for all resources.

- Local Constant: The Local Constant has a fixed value which may vary from resource to resource.

Default Value   With the Default Value, we differentiate between Constants and Local Constants. It is the value of the parameter for Constants and the default value for Local Constants.

9.3.4.2 Standard parameters

Just as there are standard parameters for jobs and batches, standard parameters are also available for resources.
To be able to reference them, a slight trick has to be used analogue to the situation with batches and jobs. If you want to use the contents of the standard variable "STATE" for a processing operation, for instance, you have to create another variable (a constant) with the value "$STATE". Due to the recursive parameter resolution, the constant is given the value of the parameter "STATE".
The following standard parameters are available:

STATE   The state of the resource. This content is empty for non-Synchronizing Resources.

AMOUNT   The total available amount of the resource. This value is empty for static resources.

FREE_AMOUNT   The available free amount of the resource. This value is empty for static resources.

REQUESTABLE_AMOUNT   The maximum amount of the resource that can be requested. This value is empty for static resources.

9.3.5 Resources tab

When you select a Named Resource in the navigation, its instances are displayed in the "Resources" tab. All the scopes, folders, Submitted Entities, Scheduling Entities or jobservers that offer instances of this Named Resource are shown here.
The "Resources" tab looks like this:
The instantiations of the Named Resource are described by the following fields:

Scope   The names of the scopes, Submitted Entities, Scheduling Entities or folders that offer instances (resources) of the respective Named Resource are shown here.

State   If the selected Named Resource has a Resource State Profile, the current state of the resource is displayed here. This is only possible with Synchronizing Resources.
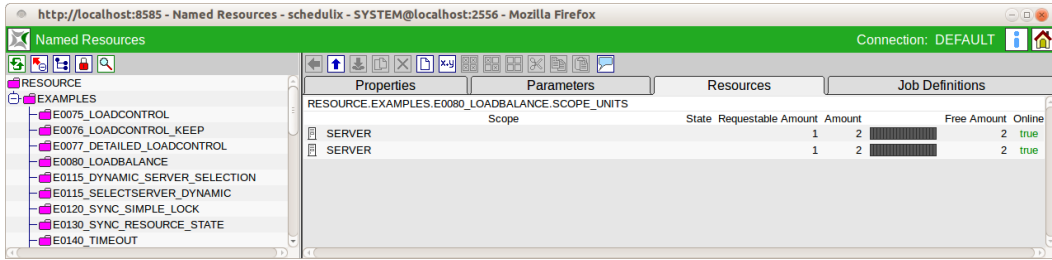
Figure 9.7: Named Resources; instances

Requestable Amount The maximum amount of resources that can be requested by a job.

Free Amount The number of free instances of this resource is shown here. A bar graphically indicates the current allocation rate.

Amount The amount is the number of available units of the resource.

Online The availability status of the resource.

9.3.6 Job Definitions tab

The "Job Definitions" tab shows all the Job Definitions that request this Named Resource. This tab is for information purposes only; no changes can be made here. The tab looks like this:
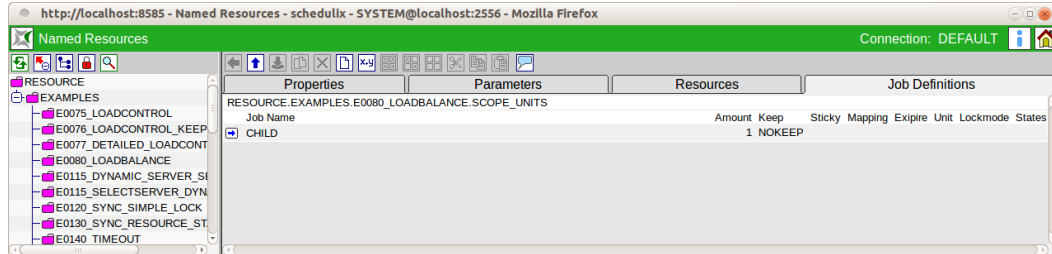
Figure 9.8: Named Resources; Job Definitions tab

The list of jobs is described in the following fields:

Job Name   The names of the Submitted Entities that require the Named Resource are shown here.
Clicking the name opens a Job Editor window.

Amount   The amount of the resource that is required by the job.

Keep   The value of the Keep parameter for the resource request from the job.

Sticky   The value of the Sticky Flag for the resource request from the job.

Mapping   If a Resource State Mapping was specified in the resource request, it is displayed here.

Expire   The Expire value specifies the maximum length of time that may elapse since the last state transition of the resource for it to be regarded as being occupiable. A negative Expire value means that a resource must be at least as "old" as given here.

Lock mode   The Lock mode describes the mode for accessing this resource (exclusive, shared, etc.).

States   All the Resource States that are accepted by this job are shown here. Multiple states that are acceptable for this job are separated by commas.

9.4 Named Resource selector

This navigator is used for selecting single or multiple Named Resource Definitions and is called by other dialogs like Footprint as a search and selection mask. Which resources are displayed varies depending upon the call dialog. If the selector is called from the Footprint editor, only the system resources are displayed. If the selector is called from the Environment dialog, only static resources are displayed.
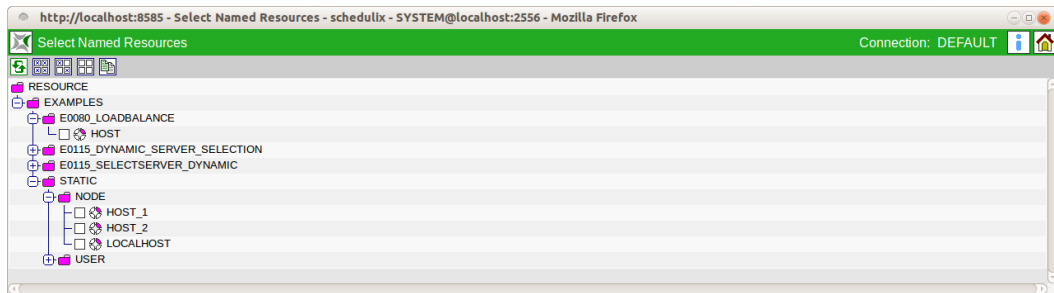
Figure 9.9: Named Resources; selector

10 Environments

10.1 View
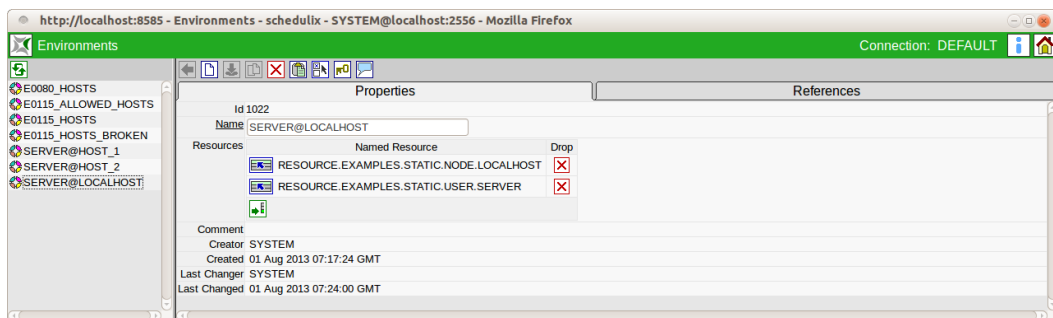
Figure 10.1: Environments

10.2 Concept

10.2.1 In short

The "Environment" dialog is used for administering environment definitions. An environment collates multiple requirements for static resources under one name.

10.2.2 Detailed description

Environments serve two purposes. On the one hand, they simplify the maintenance of resource requirements since they no longer have to be individually added in the Job Definition. All you have to do is to specify the respective environment. On the other hand, environments are used to determine which users or user groups are permitted to use specific runtime environments.

The Environment Definition therefore states which runtime environment is required by a job. For example, which host is to run a job or which users and programs must be available can be controlled with an environment.

The environment is a mandatory parameter in a Job Definition.

Environments can also be entered as a Folder Environment for a folder. This means that all the Job Definitions below such a folder "inherit" the resource requirements from the Folder Environment.

Where separate jobservers are in use for Development and Production, for example, an environment can be used for the job which allows it to be run on both jobservers. Creating two folders for Development and Testing (to which a "Development" and "Production" environment are respectively assigned whose resource requirements can only be fulfilled by development or production jobservers) allows the Development and Production environments to be easily kept separate. If such a job is located somewhere below the "Development" folder, the combination of resource requirements for Job Environment and Folder Environment ensures that the correct Development jobserver is selected. If the job is moved or copied to the "Production" folder and then executed, the Production jobserver is selected automatically. The Job Definition does not have to be modified.

10.3 Navigator
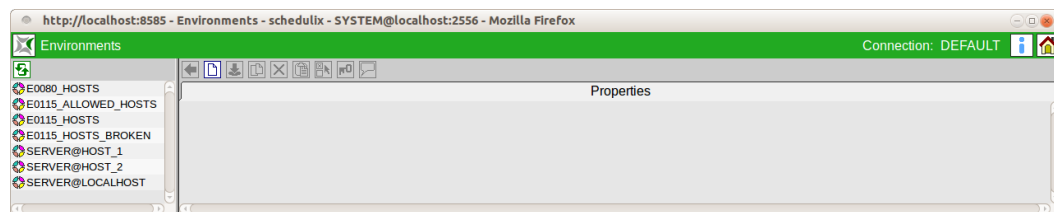
Figure 10.2: Environment navigator

The "Environment" dialog navigation window shows all the existing environments. If the logged in user is a member of the "ADMIN" group, all the environments are displayed. If this is not the case, then only those environments that can be used by the user who is logged in are shown. This means that the logged-in user must be a member of a group that holds the "use" privilege for that particular environment.

10.4 Editor

10.4.1 Properties tab

This tab is used for maintaining the environment properties. Environments can only be edited by users who belong to the "ADMIN" group. All the input fields are "read only" for all other users.
It looks like this:
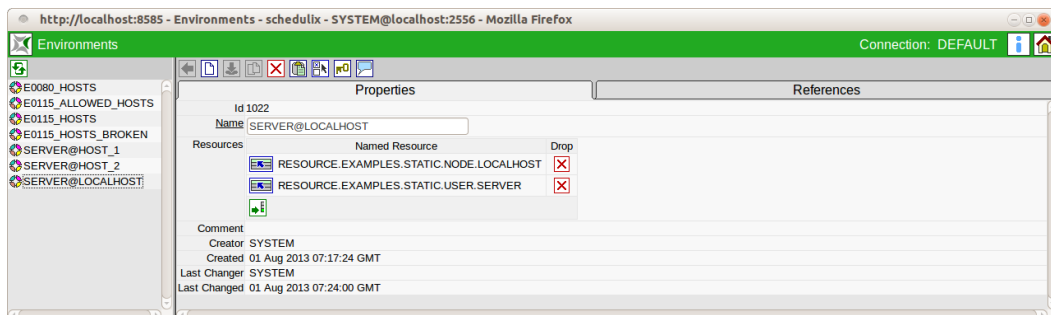
Figure 10.3: Environment properties

The "Properties" tab for environments has the following fields:

ID   The ID is assigned automatically and is used for unequivocal, system-wide identification of the object.

Name   The name of the environment. Any name can be chosen, although it must be unique within the environment.

The "Resources" list shows all the resources that belong to the environment. These are exclusively static resources. The *Named Resource* fields have the following meanings:

Named Resource   The name of the Named Resource is shown here.

Condition   A condition is entered in the *Condition* field that must be fulfilled to ensure that the resource is recognised as being valid. The condition is evaluated in the context of the requesting job.

Drop   The *Drop* button can be used to delete a specific row.

10.4.2 References tab
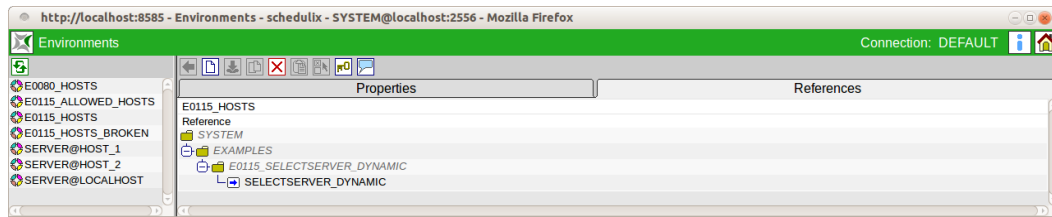
Figure 10.4: Environment references

This tab displays any references to the environment.

The "References" tab shows a tree view that only contains the parent folder of objects that reference the environment.

Folder names are usually shown in italics unless the folder references the environment as a Folder Environment.

Clicking the name of a non-recursively displayed folder or a job opens an editor window for the folder or job.

11 Footprints

11.1 View
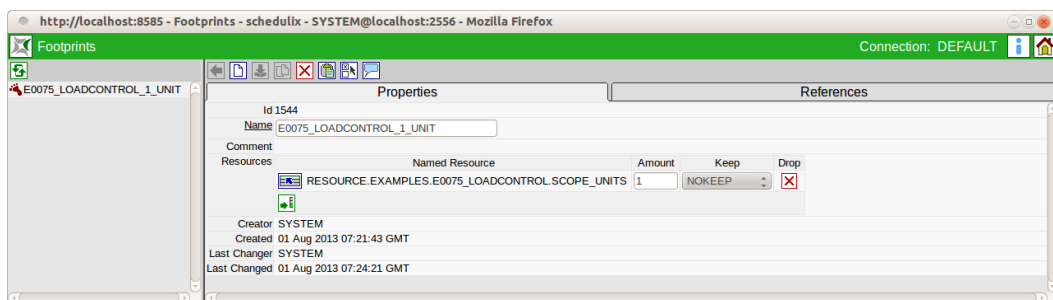
Figure 11.1: Footprints

11.2 Concept

11.2.1 In short

This dialog is used for creating a footprint definition. A footprint collates a group of System Resource requirements under one name.

11.2.2 Detailed description

Collating the resource requirements for footprints makes it easier to maintain the requirements for jobs in the runtime environment because they do not have to be individually specified.

The footprint can be overridden for each job. This is done by using an explicit requirement for a resource (contained in the footprint) with a deviating amount or by changing the setting for the Keep parameter. A resource requirement cannot be removed, i.e. all the resources defined in the footprint are requested. The amount can be reduced to 0, although resource must still be present.

Example:

A System Resource CPU_UNIT with the amount 4 (CPU units) is defined in the footprint. The footprint can be overridden by adding a requirement for the same System Resource (CPU_UNIT) within the Job Definition (Resources tab) for an

amount of 2 (CPU units). It is not possible to completely remove the resource CPU_UNIT, however.

11.3 Editor

11.3.1 Properties tab

This tab is used for maintaining the footprint properties.
Footprints can only be edited by users who belong to the "ADMIN" group. All the input fields are "read only" for all other users.
It looks like this:
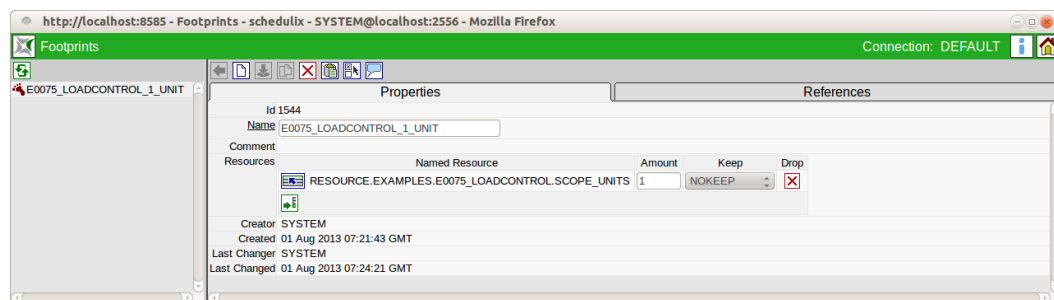
Figure 11.2: Footprint properties

The Properties tab for footprints has the following fields:

ID The ID is assigned automatically and is used for unequivocal, system-wide identification of the footprint.

Name The name of the footprint. This can be freely chosen.
A list of the resources contained in the footprints can be entered in the "Resources" list.
The fields have the following meanings:

Named Resource The name of the Named Resource is shown here. It is selected with the *Chooser* button.

Amount This is the amount of the resource required by a job.

Keep The Keep parameter determines whether the Named Resource is retained after a job has been completed or if it can be released again. The following selection options are available:

68

1. *No_Keep*

   The Named Resource is released after completion of the job. Whether the job has been completed successfully or aborted due to an error is irrelevant as far as the release is concerned.

2. *Keep*

   The resource is not released until the job has attained a final state. The resource is retained in the event of an error (Restartable State).

3. *Keep_Final*

   The resource is not released until the job and all its children have attained a final state.

11.3.2 References tab

This tab displays the references to the footprint. It looks like this:
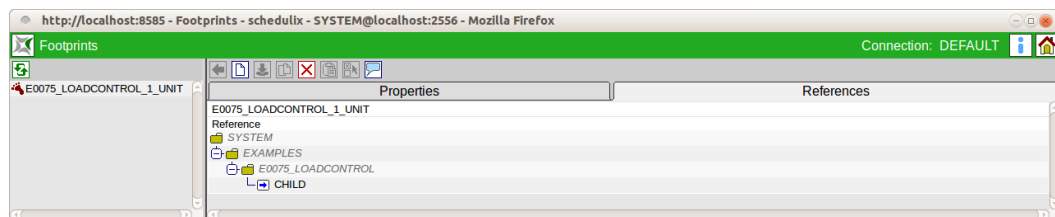
Figure 11.3: Footprint references

The References tab shows a tree view that only contains the folder below which there are jobs that reference the footprint.
Folders are displayed in italics.
Clicking the name of a job opens a window where it can be edited.

12 Jobservers and resources

12.1 View
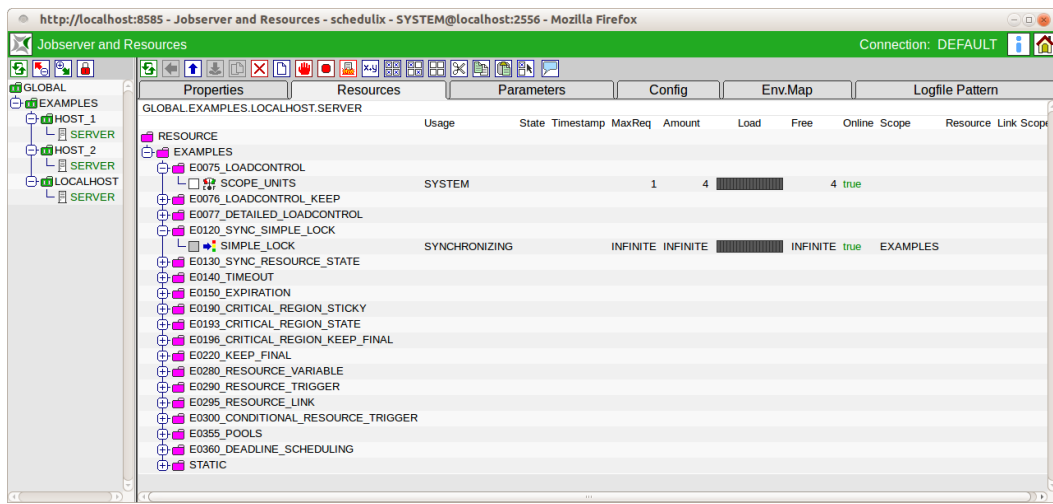
Figure 12.1: Jobservers and resources

12.2 Concept

12.2.1 In short

A jobserver is a schedulix system process that has to run on all the computers that have been defined as executive units of the schedulix Scheduling System. If a jobserver is assigned to perform a task by the schedulix Server, the jobserver executes the script or program specified in the job definition in the required environment and with the necessary parameters. The jobserver monitors the program and sends the return state of the script or program back to the schedulix Server.

12.2.2 Detailed description

The physical distribution (i.e. which jobservers run on which computers and which resources are made available by these jobservers) can be defined and maintained in the "Job Servers and Resources" dialog.

These jobservers must be set up and started at system level.

A differentiation is made between two types of objects in this dialog. On the one hand there are so-called scopes, which can serve as a container for the respective jobserver or child scopes.

On the other hand, those jobservers that reflect the executive system processes are administered in this dialog.

The hierarchy can be subdivided as required and is largely dependent upon the complexity of the system environment and administration structure (e.g. in the data centre).

For example, it is possible to define each physical host as a scope, and all the workflow environments required on this computer can be defined as jobservers below this scope. If the system landscape is more complex, it is practical to use multiple levels for departments, system architectures (Unix, Windows) or similar entities.

The scopes and jobservers can be assigned to resources at every level of this hierarchy. These resources can then be used by all the children jobservers.

Example:

A value of 5 is entered for the System Resource CPU_UNITS at host system level. If 2 jobservers have been configured on the computer, a total of 5 CPU units can be shared between them. If a jobserver occupies 3 CPU units when executing a job, only 2 CPU units remain available to the other jobserver. However, this behaviour is not additive. If the resource CPU_UNITS is also created on one of the aforementioned jobservers (for example with a value of 3), only these 3 CPU_UNITS will be available to that jobserver. Accordingly, 5 CPU_UNITS from the parent scope will be available to the other jobserver.

12.3 Navigator

The Navigator provides a hierarchical structural view in which scopes are defined
as folders. The jobservers are located within the scopes.
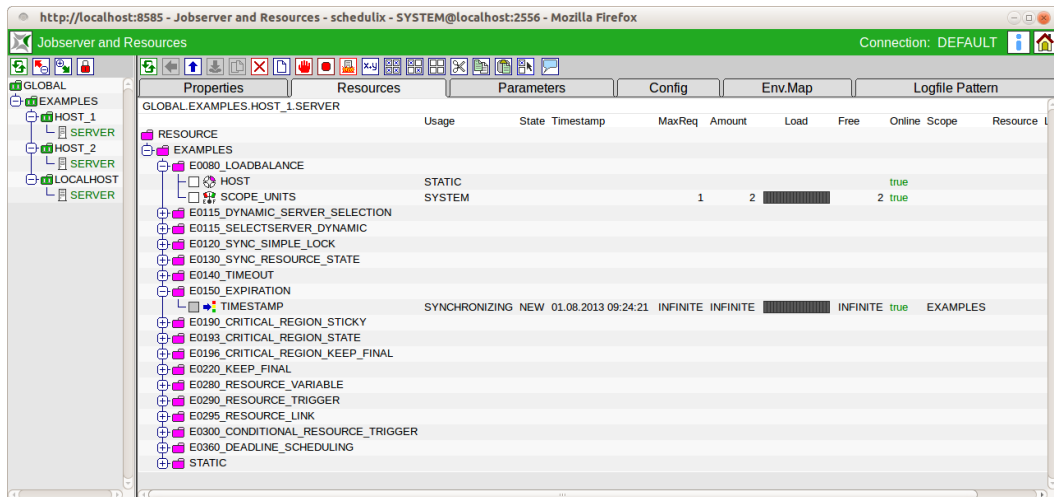
Figure 12.2: Jobservers and Scope Navigator

12.4 Editor

12.4.1 Properties tab

The ″Properties″ tab is used for maintaining the properties of scopes and job-
servers.
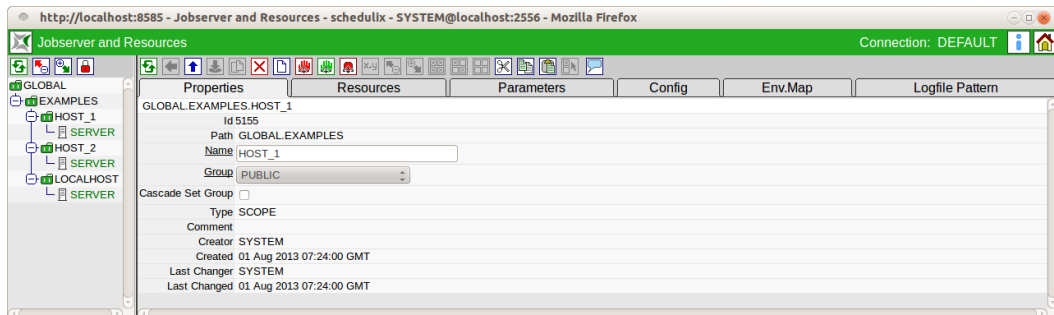
Figure 12.3: Scope properties

″Properties″ tab for scopes is shown in Figure 12.3.
″Properties″ tab for jobservers is shown in Figure 12.4.
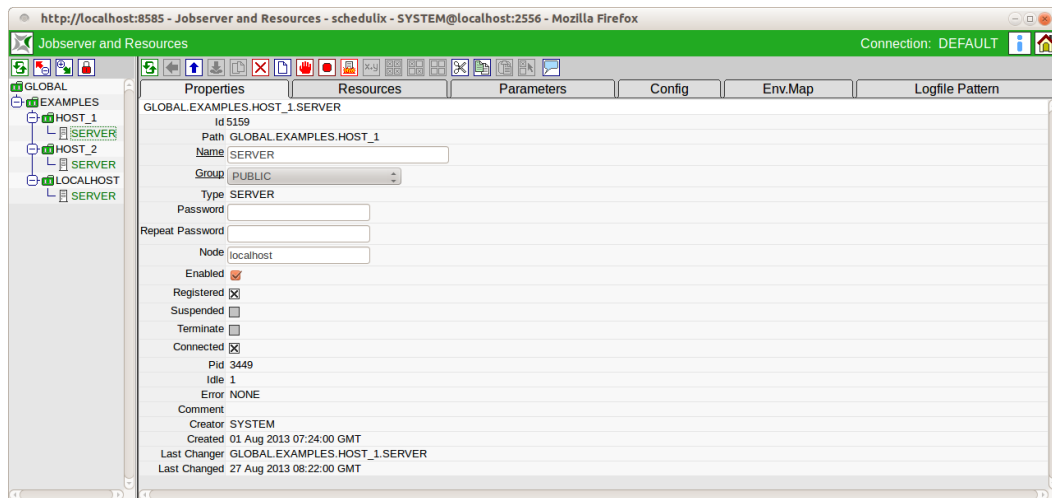
Figure 12.4: Jobserver properties

The fields in the "Properties" tab have the following meanings:

ID   The ID is used for unequivocal, system-wide identification of the object.

Path   This is the complete path to the object within the hierarchy.

Name   This is the name of the scope or the jobserver. The name can be freely chosen, but it must be unique within the hierarchy level.

Group   The group can be selected from a drop-down list. It designates the scope's owner group.

Type   The *Type* field shows the type of the selected object. The type is selected when creating a new server or scope and cannot subsequently be changed.
OPTIONS FOR TYPE

1. *Scope*

   A scope is a category that can be grouped hierarchically organised below the jobserver. This could be a host computer, for example, or, at an even higher level, all the computers in a department or all the Windows-based PCs. The structure and depth of the hierarchy can be modified depending upon the system landscape and should reflect this.

2. *Server*

This object is a physical jobserver. The physical process has to log in to the schedulix system under the selected name and in doing so set up the connection to this jobserver definition.

Password   The *password* used by the jobserver to log on to the schedulix Server. Each jobserver requires a name and password to log on to the schedulix Server. A job can only be handed over to the jobserver to be executed with a valid login.
The password is hidden on the screen and must be identical to the field *Repeat password*.
Since the password is not displayed in the dialog, it is not clear whether a password has already been entered or not. The password does not have to be specified again when changing other properties. If the jobserver is logged in, it is automatically notified about any password changes.

Repeat Password   The *Repeat Password* must be the same as the password you entered in the *Password* field. It is necessary to enter it twice to detect any accidental typing mistakes.

Node   The *Node* specifies the computer on which the jobserver is running. This field has a purely documentary character.

Enabled   If the jobserver has set the Enabled flag, the jobserver process can log onto the schedulix Server. The login will fail if the flag has not been not set.

Suspended   If the Suspended flag has been set, although the jobserver process can log on it will not be given any jobs to run. State changes for previously allocated jobs are accepted by the schedulix Server.

Terminate   The Terminate flag indicates whether the jobserver is to terminate or not. If this flag has been set, the jobserver will receive this message at the next communication step and behave accordingly.
It may take a few seconds before the jobserver terminates. The termination can be verified by clicking the *Refresh* button. Once the jobserver has finished, the value displayed in the *Connected* field changes to "FALSE".

Connected   The *Connected* field shows the connection status of an external jobserver process. If the jobserver process was able to successfully log on to the schedulix system, this value is set to "TRUE".

75

PID   The PID is the process identification number of the jobserver process on the respective host system. It is communicated to the schedulix system during the registration.

The PID and the *Node* field allow the process to be located in the operating system running on the respective server.

Error   An error message is displayed here if an error occurred while the jobserver process was running.

12.4.2 Resources tab

The "Resources" tab looks like this:
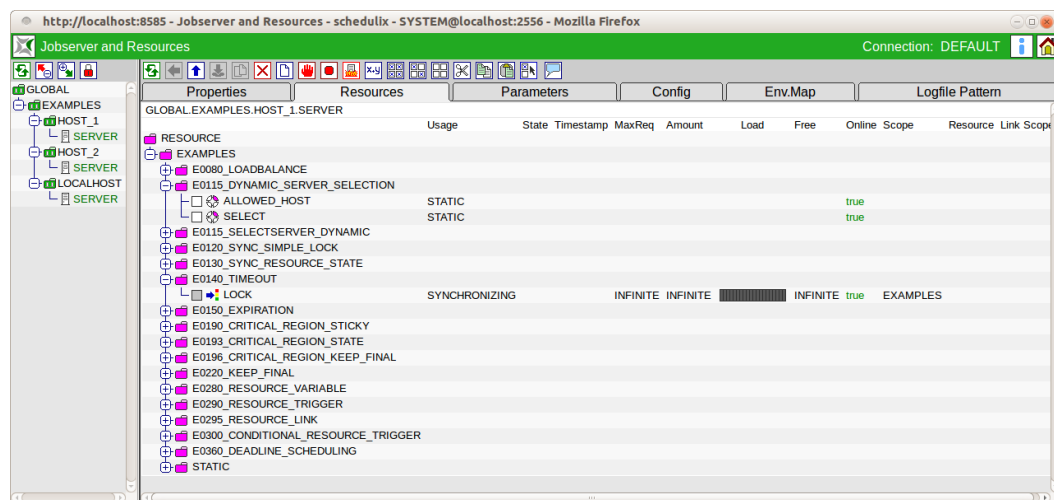
Figure 12.5: Jobserver and scope resources

All the resources offered by the current scope or jobserver are shown in the Resources tab. If any resources (especially of the type Synchronizing) are to be awarded system-wide, this can be done in the "GLOBAL" scope.

If multiple instances of a resources are entered in different scopes or jobservers, these are respectively a separate instance of that resource.

Example:

If a Synchronizing Resource A is entered in Scope X and Scope Y, these two instances of Resource A are mutually independent (in X and Y). This means that a possible synchronisation will only take place among all the jobs in Scope X and Scope Y respectively. If a global synchronisation is to take place here, Resource A must be defined in the GLOBAL scope instead.

The resources are displayed as a map of the hierarchy from the navigation in the Named Resources dialog instead together with additional details. The first field in

the list shows the name of the Named Resource or category. If this is a category, a folder icon stands here as well. The category can be opened and closed by clicking this icon. If it is a Named Resource, clicking the name opens the "Resource Details" tab.

The other fields are explained in the next section.

12.4.2.1 Resource Details tab

The "Resource Details" tab is displayed as soon as a resource is selected. The instance information for this Named Resource is entered and maintained in this tab. Here, an instance is the expression of the Named Resource in the selected scope or jobserver. All instance parameters can be entered and maintained here.

The "Resource Details" tab looks like this:
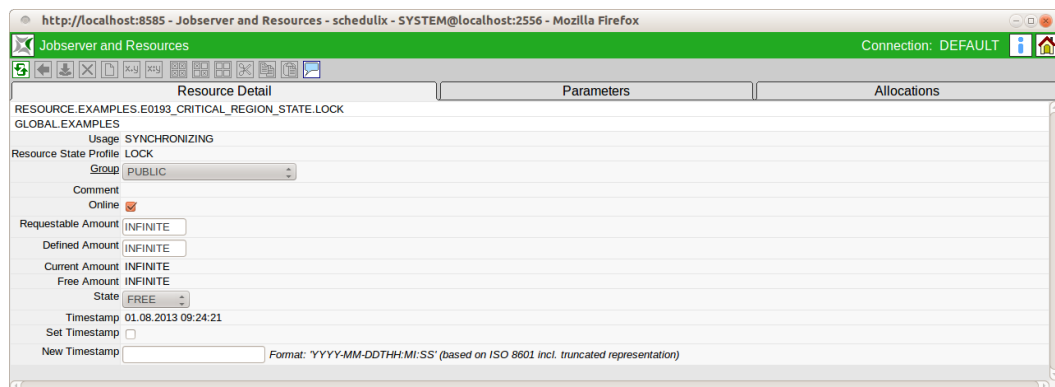
Figure 12.6: Resource Details

The fields in the "Resource Details" tab have the following meanings:

Usage The *Usage* field specifies the "Resource" type. More details about resource types can be found in the Named Resource dialog.

Resource State Profile This field is only visible if the resource usage has been set to "Synchronizing".

This is the Resource State Profile assigned to the resource.

Online *Online* can be used to switch a resource in this scope or jobserver online or offline. An offline resource is temporarily unavailable. Example:

A scheduled computer or database downtime is pending. If the computer or database has been mapped as a resource in the system, the resource can be set as being offline as soon as it is physically no longer available. All jobs that require this resource are automatically no longer executed. If the resource is to be defined in another scope

as well (as a separate instance), all the jobs that would require the resource would deviate to this scope or jobserver.

An offline resource does *not* trigger the error message "Job cannot run in any scope because of resource shortage".

Requestable Amount   The *Requestable amount* is the maximum amount that can be requested.

Defined Amount   This field is only visible if the resource usage has been set to "Synchronizing" or "System".

The *Defined Amount* states the current number of instances of the Named Resource for this scope or jobserver. If the number of instances for this scope changes, you can edit this field accordingly.

Example:

This may become necessary if the hardware has been modified. If the resource represents a CPU unit and 2 CPU units are installed in the current host, the amount is 2. If 2 more CPU units are then installed, this value must be increased to 4.

Current Amount   The *Current Amount* and *Defined Amount* are identical.

Free Amount   This field is only visible if the resource usage has been set to "Synchronizing" or "System".

The *Free Amount* designates the total number of instances of a resource in the selected scope or jobserver that have not yet been allocated to jobs.

State   The *State* field is only visible if the resource usage has been set to "Synchronizing" and a Resource State Profile has been assigned. It designates the current status of the resource in this scope or jobserver. The state can be set or changed with this value. The drop-down list contains all the valid Resource States of the Resource State Profile, which can be selected here. This may be necessary when manually fixing an error to restore a resource to a state that allows further processing by follow-on jobs.

If you manually change this value, the value in the *State* field in the "Resources" tab is not automatically refreshed. Click the *Refresh* button if this is required and necessary.

Timestamp   This field is only visible if the resource usage has been set to "Synchronizing" and a Resource State Profile has been assigned. The *Timestamp* indicates the time of the last change made to a Resource State.

Timestamps play a role if an expiration interval has been defined in an expiry object. If this is the case, the timestamp must not be older than the specified interval. More details about expiration times can be found in Chapter 13.5.8.1.

If the completion of a job causes the Resource State to change, the timestamp is automatically updated by the schedulix Server.

Set Timestamp This field is only visible if the resource usage has been set to "Synchronizing" and a Resource State Profile has been assigned. The *Set Timestamp* flag is a so-called action flag. If you set this flag and then click the *Record* button, the time shown in the *Timestamp* field is set to the current time.
Example:
This may be necessary when manual intervention is required to fix an error and the state has been changed manually in the *State* field .

12.4.2.2 Parameters tab

Additional information about a resource that can be evaluated by jobs or Resource Triggers can be saved in the Parameters tab. The parameters are defined in the Named Resources.
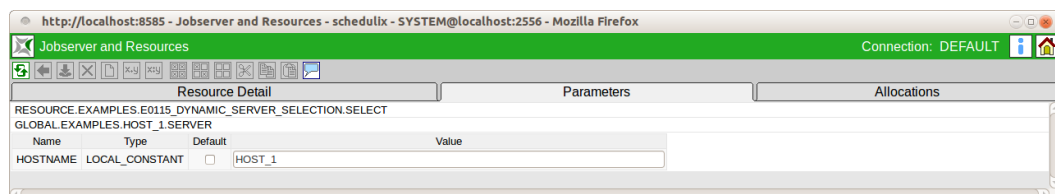
Figure 12.7: Resource parameters

The columns in the list shown above have the following meanings:

Name The "Name" column shows the name of the parameter.

Type This is the parameter type. The following options are available:

- Constant: The constant has a fixed value and applies for all resources.

- Local Constant: The Local Constant has a fixed value which may vary from resource to resource.

The values of the Local Constants can be changed here.
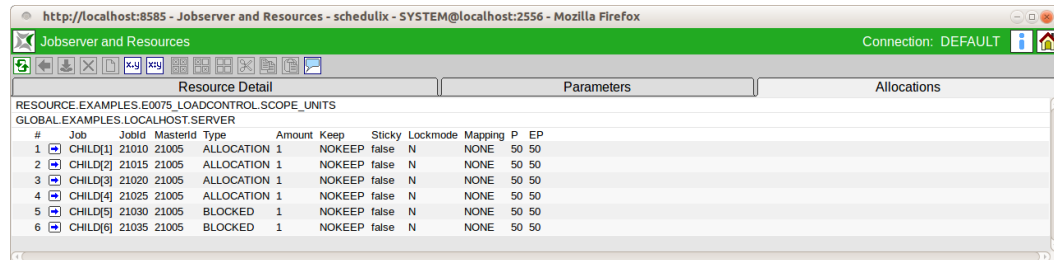
Default The *Default* field indicates whether the displayed value is the default value. The flag has to be set to restore the value to the default value.

Value This is the value of the parameter.

12.4.2.3 Allocations tab

The "Allocations" tab provides information about which tasks have been currently allocated the resource and which tasks require resources but cannot obtain them because they have the incorrect state or due to a lock.
The Sheet tab looks like this:

Figure 12.8: Resource allocations

The columns in the list shown above have the following meanings:

Job   Depending on the setting, either the name of the job or the full path of the folder hierarchy for this job is shown here.

Job Id   This is the ID of the job instance that was started with either a direct submit in the Submit Jobs dialog or a submit of the Master Batch or job.

Master Id   This is the ID of the job instance or batch that was started as a Master Job in the Submit Jobs dialog and contains the current job as a child. If the job is itself a Master Job, then its own Job Id is shown here.

Type   The *Type* refers to the method used by the current job instance to access or attempt to access the resource. The following options are available:

1. Requested

   The job asks the server if the resource is available.

2. Reserved

   The resource is reserved by the job. This has the same effect as Allocated, except that reserved resources can be released again when the job is started. Allocated resources are released at the earliest when the job has finished.

3. Allocated

"Allocated" indicates that the job is currently running and accessing the resource. Other jobs that want to access this resource can, depending upon the type of access lock, be allocated this resource as well or be blocked.

4. Blocked

   If a job is labelled as being "Blocked", it cannot be run at the moment because it is waiting for access to the selected resource but is not able to obtain it due to a non-matching lock mode or other criteria.

   The reason why the resource is not available is indicated by the red lettering.

   As soon as the resource becomes available again, the type changes from "Blocked" to "Requested".

5. Available

   The resource is available.

6. Ignored

   The user has told the system to ignore the resource request.

7. Master Reservation

   A Master Reservation is a reservation of Sticky Resources.

Amount   The *Amount* is the number of resource instances allocated to or requested by the current job. If the value of the amount exceeds the currently available number that can be maintained in the *Amount* field in the "Resource Details" tab and there is no alternative scope available with a sufficient number, the job cannot be executed.

Keep   The *Keep* parameter defines the Keep state of the current resource. More information about the Keep state can be found in Chapter 11.3.1.

Sticky   The *Sticky* parameter defines whether the job is given the current "Sticky" resource or not. More information about the Sticky Flag can be found in Chapter 13.5.8.1.

Lock mode   The *Lock mode* defines the access mode being used in allocating the resource to the current job. The following access modes are available:

1. Exclusive (X)

   No other job has access to this resource.

   Example: When loading a database table.

2. Shared Exclusive (SX)

   These access operations are mutually compatible, but they are not compatible with shared access operations.

   Example: An application performs numerous small write operations on a database table.

3. Shared (S)

   Shared access operations are mutually compatible.

   Example: Evaluations of complete database tables or large parts of them.

4. Shared Compatible (SC)

   These access operations are both mutually compatible as well as being compatible with S and SX access operations.

   Example: An application performs numerous, brief read transactions on a database table. Such applications do not impede one another. Small write transactions or large-scale evaluations are not an impediment either.

5. NoLock (N)

   The NoLock lock mode indicates that the resource is neither being locked nor is a lock to be heeded. A job that locks a resource with NoLock can always access it regardless of any other lock options.

Here is a compatibility matrix summarising all the lock options and their interactions.

(+) means that the lock modes are mutually compatible. Two jobs that request the two lock modes can run concurrently.

(-) means the lock modes exclude one another. A job stays blocked until the other job releases the resource again.

	Exclusive (X)	Shared (S)	Shared Exclusive (SX)	Shared Compatible (SC)	NoLock (N)
Exclusive	-	-	-	-	+
Shared	-	+	-	+	+
Shared Exclusive	-	-	+	+	+
Shared Compatible	-	+	+	+	+
NoLock	+	+	+	+	+

Table 12.1: Lock mode matrix

Mapping   The *Mapping* indicates the current Resource State Mapping that is using the resource.

P (Priority)   The *Priority* defines which start priority the process had at the time of the submit in the Submit Jobs dialog. The values for the priority range from 100 (lowest priority) to 0 (highest priority).

EP (Effective Priority)   The longer a job waits to be executed in the system, the higher the effective priority of its execution in the schedulix system. This is to make sure that previously started jobs are taken into account before the later jobs (depending upon the settings in the Priority field in the Batches and Jobs dialog and the start time).

How the priority rises over time can be set in the server configuration. The default setting causes the priority to rise by one point for every half an hour that elapses since the job was submitted.

Example: If a job was started at 12 p.m. with a priority of 50, the effective priority at 12:30 p.m. is 49, at 1 p.m. it is 48 and so on until the highest level has been reached.

Blocked Processes   If the current job is marked in the list as being blocked, the reason why the job cannot be executed is shown in red.

Example: If the resource is currently exclusively allocated to another job and the current job also wants that resource to be exclusively allocated to it, the *Lock mode* field is shown in red.

12.4.3 Parameters tab

All the parameters defined in the jobserver or scope are administered in the "Parameters" tab. These parameters can be used for the jobs and those programs that are executed in the jobs.

The parameters are inherited in scope hierarchies. A parameter can be defined at the highest level and is then automatically made available to every scope and jobserver located below this level.

The "Parameters" tab looks like this:
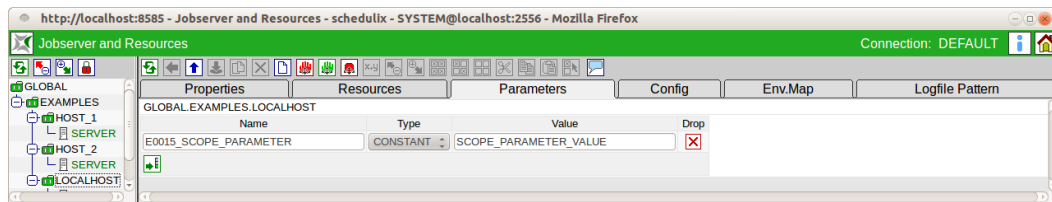
Figure 12.9: Jobserver and scope parameters

A list of all the parameters is displayed here. If a jobserver has been selected in the navigation, all the parameters inherited from the parent scope hierarchy are visible as well in addition to the current parameters. However, they cannot be changed at this level. To change an inherited parameter at this level, it has to be recreated under the same name. This value overwrites the value of the inherited part. The displayed value can now be changed.

The columns in the list shown above have the following meanings:

Name   The name of the parameter is shown here.

Type   This is the type used for the parameter. The following options are available:

1. Dynamic

   This designates a dynamic parameter. The value of this parameter is derived from the value of an environment variable that has to be maintained in the system on which the current scope or jobserver is running.

   This value is only determined when the jobserver is started. If the jobserver is currently running on the computer, the actual value of the identically named environment variables on the computer and the environment in which the jobserver is running is displayed in the *Value* field.

   Important: The environment variable must have been created on the computer and in the jobserver's start environment.

84

2. Constant

   If the parameter type is "Constant", it is a constant expression that needs to be entered in the *Value* field. If a job or its program uses this value, the parameter is replaced by this constant expression when the job or program is executed.

   Example:

   If a script is to be run independently of the current database, the database connection can be mapped as a parameter. If the database connection parameter is simultaneously defined with respectively different values in both Scope 1 (for example the test database) and Scope 2 (for example the production database), the script can run in both environments without having to modify it because the database connection is defined in the parameter.

Value   This is the value of the parameter. If the parameter type is "Constant", the parameter value has to be entered. If it is a dynamic parameter, the current value of the environment variables in the system on which the jobserver is running is displayed here.

12.4.4 Config tab

The configuration of the jobserver is described in the "Config" tab. A table of key/-value pairs is used for the configuration. Please refer to the jobserver documentation for the meanings of the individual values.
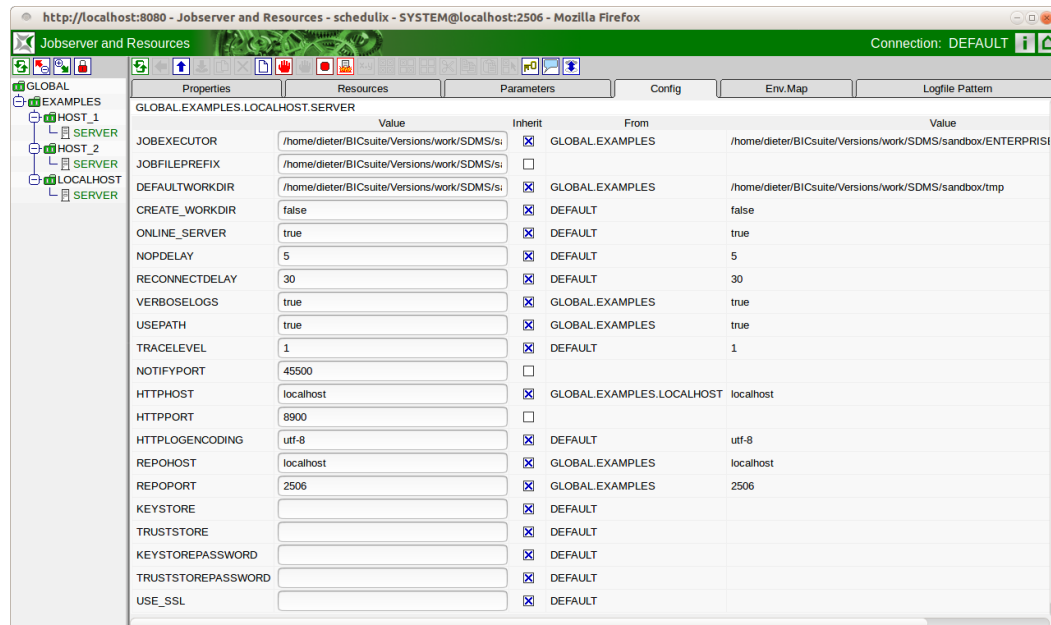
Figure 12.10: Jobserver and scope configuration

The fields in the "Config" tab have the following meanings:

Value   The *Value* field shows the respective key/value pair.

Inherit   The *Inherit* field shows whether the key/value pair value is being inherited from a parent scope. The current entry is deleted by setting the flag in the *Inherit* field. In this case, the value of the key/value pair is determined via the inheritance mechanism. The entry "DEFAULT" means that this is preset by the system.

From   The *From* field shows from which scope the value is being inherited. If *Default* is displayed for the scope, these are the default configuration values.

Value   The inherited value of the key/value pair.

12.4.4.1 Standard configuration parameters

The following table shows the names of the configuration parameters for a job-server and their meanings:

Name	Meaning
JOBEXECUTOR	The fully qualified path of the job executor.
JOBFILEPREFIX	Path and file prefix of the task files.
DEFAULTWORKDIR	Working directory of the jobserver and the default working directory for the jobs executed using the jobserver.
ONLINE_SERVER	Boolean value. If this is set to "False", the jobserver logs off when there is no work to be done.
NOPDELAY	Time between two requests for jobs sent to the scheduling server.
RECONNECTDELAY	Time between two attempts to log on to the scheduling server.
VERBOSELOGS	If this is set to "True", the start and end times of the jobs are written to their log files in addition to the process output.
USEPATH	This parameter defines whether the jobserver uses the path setting when starting processes. If "USEPATH" is set to "False", all the program calls have to be fully qualified.
TRACELEVEL	Defines how "communicative" the jobserver is. • 0: Errors are logged • 1: Errors and warnings are logged • 2: Errors, warnings and information are logged • 3: Debug level; all messages are logged
NOTIFYPORT	The Notify port is the port to which a UDP packet is sent if a job is ready for executing by the jobserver.
HTTPHOST	The jobserver can send log files using the HTTP protocol. The two parameters "HTTPHOST" and "HTTPPORT" have to be set for this.
HTTPPORT	Port for sending log files.
HTTPLOGENCODING	Character set for the log files. When displaying log files, a meta/charset tag with this character set is created in the HTML header.
Continues on next page	

Continued from previous page	
Name	**M**eaning
REPOHOST	Host name or IP address of the scheduling server.
REPOPORT	Port for the scheduling server.

Table 12.2: Description of the jobserver configuration parameters

12.4.5 Env.Map tab

When a job is started, a number of default parameters for the job are sent to the jobserver as key/value pairs. These parameters can be set as environment variables before the process is started. Whether and under what name these variables are made visible is configured in the "Env.Mapping" mask.
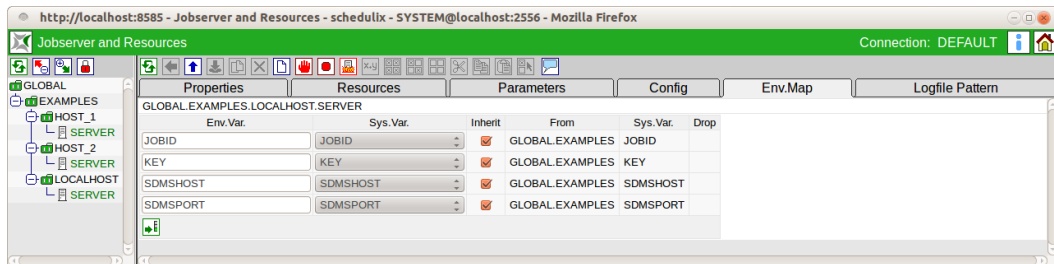
Figure 12.11: Jobserver environment mapping

The columns in the list shown above have the following meanings:

Env.Var.   The name of the environment variable is shown in the *Env.Var.* field.

Sys.Var.   The name of the parameter is shown in the *Sys.Var.* field.

Inherit   The *Inherit* field shows whether the key/value pair value is being inherited from a parent scope. The current entry is deleted by setting the flag in the *Inherit* field. In this case, the value of the key/value pair is determined via the inheritance mechanism.

From   The *From* field shows from which scope the value is being inherited. If *Default* is displayed for the scope, these are the default configuration values.

12.4.6 Logfile Pattern tab

The jobservers can send the contents of log files using the HTTP protocol. To prevent the uncontrolled reading of arbitrary files, only files with names conforming to certain patterns can be requested.

Permitted patterns have to be entered by the administrator. If he does not do so, no files can be requested.
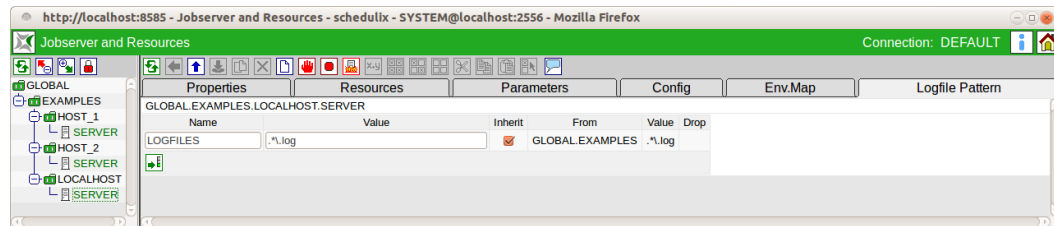
Figure 12.12: Jobserver log file name patterns

The log file patterns are regular expressions with a twist. The character string "/../" or "\..\" (Directory up) cannot usually occur in the name of a requested file. This rule makes it easier for the administrator to define the log file patterns.

It is generally advisable to test the whole name. A pattern such as ".*\.log" allows all files to be read which contain this string somewhere in their name. The pattern ".*\.log$", on the other hand, is an important limitation, since the name has to end with this string.

The pattern "^/tmp/.*\.log$" is being used in the screenshot. All files below the directory "/tmp"

that end with "/.log" can be requested.

Requests for log files that do not match any of the patterns are logged in the jobserver's log file as errors. The messages look something like this:

```
ERROR [Jobserver]  01-12-2009 16:43:43 CET [HttpThread] ERROR: Illegal file request: /etc/passwd
```

together with some additional information (which is logged for each file request):

```
...[HttpThread] Got Request from 1.2.3.4 : GET /?FNAME=/etc/passwd HTTP/1.1
...[HttpThread] Got Request from 1.2.3.4 : Host: localhost:8881
...[HttpThread] Got Request from 1.2.3.4 : User-Agent: Mozilla/5.0 (X11; U; Linux x86_64; en-US; rv:1.9.0.15)
    Gecko/2009102815 Ubuntu/9.04 (jaunty) Firefox/3.0.15
...[HttpThread] Got Request from 1.2.3.4 : Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
...[HttpThread] Got Request from 1.2.3.4 : Accept-Language: en-us,en;q=0.5
...[HttpThread] Got Request from 1.2.3.4 : Accept-Encoding: gzip,deflate
...[HttpThread] Got Request from 1.2.3.4 : Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
...[HttpThread] Got Request from 1.2.3.4 : Keep-Alive: 300
...[HttpThread] Got Request from 1.2.3.4 : Connection: keep-alive
...[HttpThread] Got Request from 1.2.3.4 : Cookie: _ZopeId=''55722729A4JSGHnrljM''; tree-s=''...''
...[HttpThread] Got Request from 1.2.3.4 : Cache-Control: max-age=0
```

An error message is displayed to the user:

```
ERROR: The requested filename doesn't match any of the configured patterns
```

12.5 Resource Links

Resource Links allow resources to be allocated in a different scope to the one in which a job is being executed.

For example, if operations are being performed on a database server in a client/server environment, the resources required by the database operation are located not on the client computer on which the job is being executed, but on the host computer of the database server.

Resource Links can be used to map a resource for one scope in another scope.

A Resource Link is created by copying a resource with the usage STATIC, SYSTEM or SYNCHRONIZING from a scope or server to the clipboard and pasting it into another scope.

If a resource and/or a Resource Link is being pasted from the clipboard, the following mask opens for choosing how the paste operation is to be performed.

Figure 12.13: Copying and creating Resource Links

If a resource is being pasted from the clipboard, the field *Paste Resource Operation* is displayed.

The following selections are possible:

- COPY RESOURCE: The resource is copied.

- LINK TO RESOURCE: A Resource Link is created.

If a Resource Link is being pasted from the clipboard, the field *Paste Link Operation* is displayed.

The following selections are possible:

- COPY RESOURCE: The resource referenced by the link is copied.

- COPY LINK: The link is copied.

- LINK TO LINK: A Resource Link referencing the link is created.

If a Resource Link is being pasted from the clipboard, the field *Overwrite Existing Link* is displayed as well. If this check box is enabled, an error message is not displayed if a link with the same name already exists. Instead, the link is overwritten or modified.

Resource Links are indicated in the resource list by an arrow in the icon.

12.5.1 Resource Details tab

The "Resource Details" tab for a Resource Link shows the details of the actual resource that is addressed by the Resource Link. If the Resource Link is edited, the changes are applied to be addressed resource.
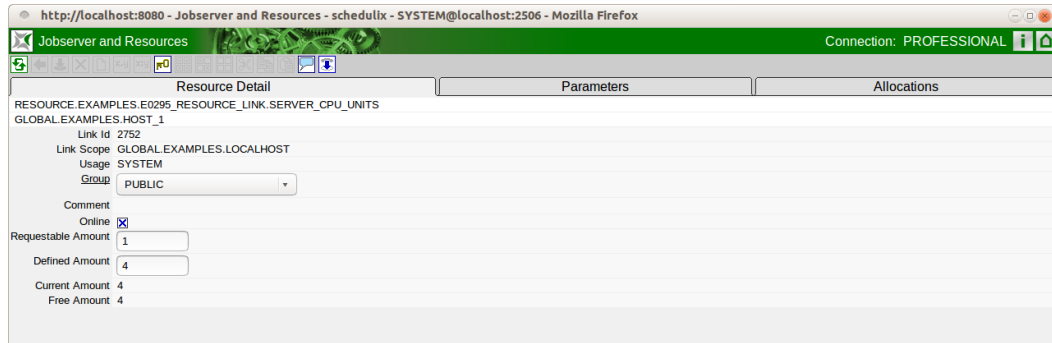
Figure 12.14: Information about a Resource Link

The following 'read-only' fields are also displayed in addition to the attributes of the base resource that have already been mentioned.

Link Id   The Id of the resource referenced by the link.

Link Scope   The scope in which the linked resource is located.

13 Batches and Jobs

13.1 Image
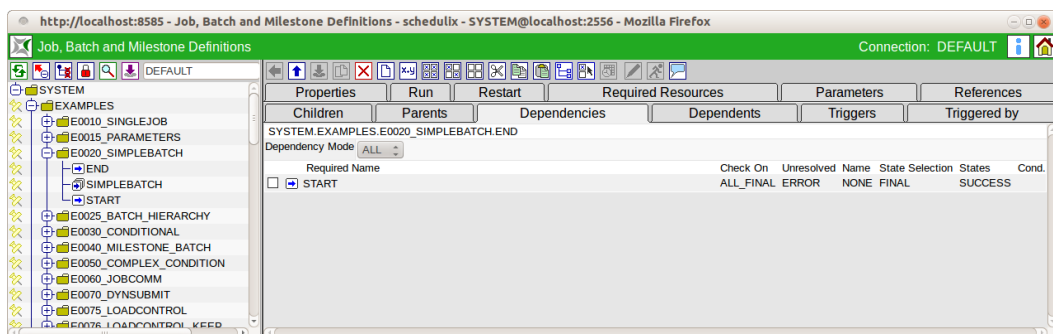
Figure 13.1: Batches and Jobs

13.2 Concept

13.2.1 Brief description

The Batch and Jobs dialog window is where all the Job Definitions in the BICsuite Scheduling System are managed.

The objects "Batches", "Jobs" and "Milestones" are used to define the units to be executed in the Scheduling System.

A job is a container for an executable script or program in the Scheduling System. When a job is submitted and started, the job server runs the specified program or script and sends feedback after the process has been completed as to whether it has succeeded or failed (Exit State).

A batch is a container for other objects and contains so-called children. When a batch is started, all the children are started automatically as well. A batch does not have a program or script that is executed.

The batch (or job) that is the first object started in the Submit Batches and Jobs dialog window is called the Master Job.

A milestone is an object that can be used to send a notification when a certain number of completed jobs has been attained (or not attained) or for mapping complex dependencies.

A milestone does not have an executable program or script either.

13.2.2 Detailed description

Each Scheduling Entity (batch, milestone or job) can belong to a batch run in a parent-child hierarchy. A job is started when its parent has been started. This means that when such an executable object hierarchy is submitted, all the jobs are submitted from top to bottom.

However, this takes place within a transaction so that all the actions, as seen by an outsider, are performed simultaneously.

Each task (batch, milestone or job) can have dependencies. An Exit State dependency describes which executable object (Required Job) must have run beforehand.

If only one specific Exit State is to be taken into account, the Dependent Job is not started until the Required Job has been completed in full and the correct Exit State has been reported back to the system.

This allows dependency models to be created which ensure that all the lead jobs really could be successfully executed in order to start a follow-on process (e.g. a report).

Each task can require different resources. These

This ensures that the runtime environment used to execute the job has all the required resources available (e.g. sufficient main memory, sufficient CPU units, correct database, required system programs, etc.).

Synchronizing Resources can also be used to make sure that processes do not hinder one another or that the resources have specific states and are up to date timewise. Load balancing can also be achieved using the resources.

Each Scheduling Entity can define or use certain parameters. This enables data to be transferred between the individual jobs or from the runtime environment. The parameters can be transferred in both directions within the parent-child relationship (from parent to child, from child to parent).

However, it is also possible to access the parameters of Scheduling Entities that do not belong to the parent-child relationship. This makes it possible to transfer data between all Scheduling Entities of a Master Job. More information about the parameters can be found in Chapter 13.5.10.

Actions can be defined for a task to be performed when a particular event occurs (e.g. completion of the job trigger) or when specific Exit States are attained. These actions are called triggers. A trigger defines which action is to be performed when a specific event occurs. The action that is triggered is an executable object again.

If the event occurs, this trigger is activated and the job (or batch) is submitted. This functionality can be used to send a message to a system administrator, for example.

13.2.3 Recommended convention for batch objects

For the sake of clarity and to facilitate working with batch objects, we advise our users to adhere to the following convention.

94

A batch object should be located in a folder with the same name. This folder should contain the batch and its children (sub-batches and jobs). Sub-batches should, in turn, be located in a separate subfolder. Job objects that have children should be treated like batches, i.e. they should also be located in separate subfolders.

This convention is not mandatory, but it can significantly reduce the time and effort required to grant access privileges or deploy, copy and move sub-workflows, etc., and greatly improve the orientation.

To facilitate the implementation of this convention, the following functional extensions have been implemented in the BICsuite!Web interface:

- When you create a folder, a batch object can also be created at the same time in this folder.

- When you create a batch, you can also create a parent folder for it.

- When renaming batches or folders, the folder or batch with the same name can be renamed as well.

- When creating objects in a folder with a batch or job of the same name, at the same time the new object can be added to the batch or job as a child.

- When cloning jobs, the new job can be added as a child to the batches or jobs of the same name in the folder.

- When cloning folders, the batch or job of the same name in the new folder can be automatically renamed.

- When cloning folders, the batch or job of the same name can be added as a child to the same batch or job in the parent folder.

13.3 Navigator

All objects of the types "batch", "job" and "milestone" are hierarchically managed in the Navigator in containers (so-called folders). These folders are used for clearly organising objects.

In addition, parameters and a default environment can be defined in them. All the parameters for the folder are available to all the jobs in that folder. The folder environment behaves additively to the environment of a job.

13.3.1 Pinning

For greater clarity in the Navigator folder, objects can be pinned by clicking the *Pin* icon .

A pin can be removed again by clicking the *Unpin* icon .
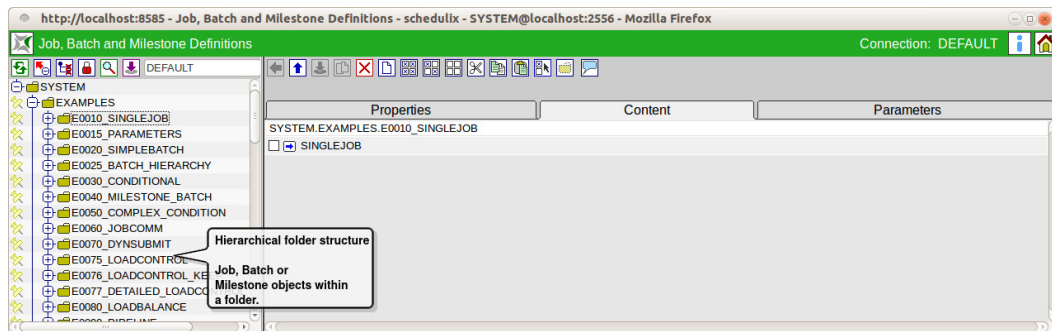
Figure 13.2: Navigator for batches and jobs

Lines with a pin are always displayed regardless of whether the parent folders are expanded or not. This means that the user has to scroll much less frequently in the Navigator.

The "Expand" and "Collapse" icons in the tree view are coloured yellow if there is a pinned object below a folder. This indicates that collapsing a folder will not close it completely, but that the path to the pinned object will always remain visible.
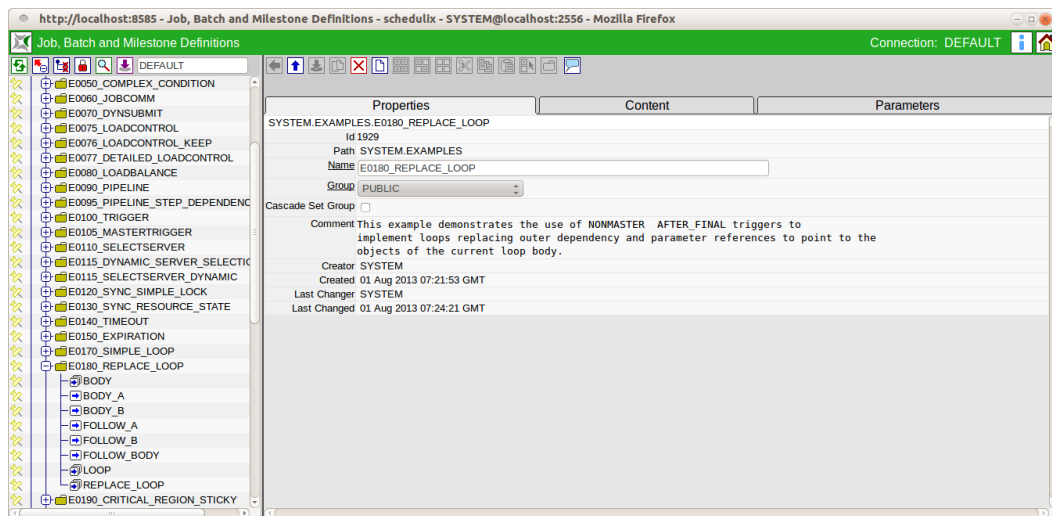
Figure 13.3: Batches and Jobs; view without pin
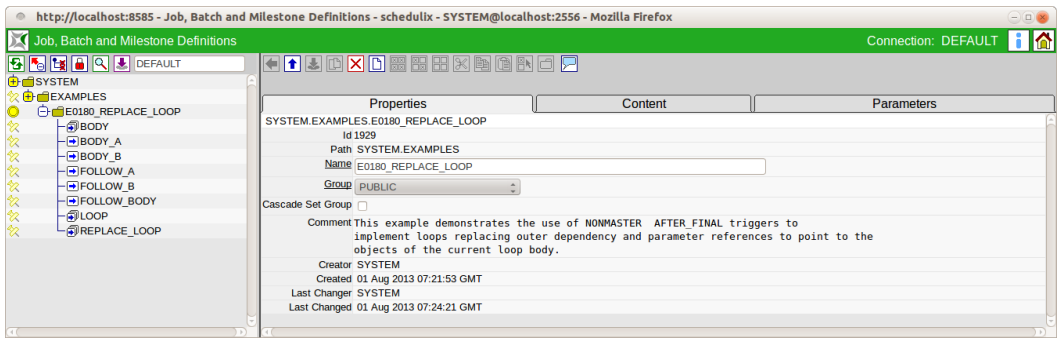
Figure 13.4: Batches and Jobs; view with pin

13.3.2 Folder bookmarks

Store Bookmark

Clicking the *Store Bookmark* button creates a bookmark for the current Navigator setting.

The navigation settings contain information about how the jobs, batches and locked objects are displayed, the current expansion state of the tree and the pinned objects.

The name for the saved bookmark is entered in the box to the right of the *Store Bookmark* button.

The "DEFAULT" bookmark is used when opening the "Batches and Jobs" dialog window from the BICsuite!Web desktop.

Other bookmarks can be opened by clicking Bookmark (folder).

If the current user is a "Web GUI ADMIN", an option field is displayed next to the name of the bookmark which can be used to set whether the bookmark should be saved as a system bookmark (visible to all users) or as a user bookmark (only visible to the current user).

13.3.3 Folder search

Find

Clicking the *Find* button switches the Navigator to search mode.

In the input field, you can enter a search pattern corresponding to the syntax selected in the options field to the right of the input field. The SQL LIKE syntax and regular expressions are supported.

In the next options field, you can define whether this pattern is to apply to the entire path (FULL) or only the end of it (TAIL).

The hits are displayed in the Navigator by clicking the *Find* button in search mode.

Clicking a hit opens the object in the editor pane, the object is pinned, and the search mode closes again.

13.4 Editor for folders

13.4.1 Properties tab

All information pertaining to the general properties of the folder is recorded in the "Properties" tab.
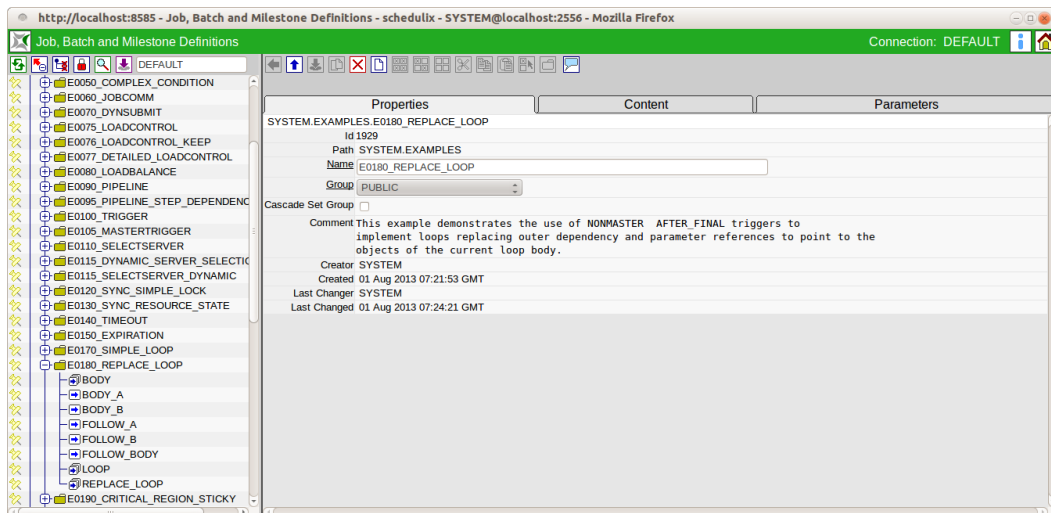
Figure 13.5: Folder Properties

Id   The *Id* is the unique identification number for the object.

Path   The *Path* describes the parent folder hierarchy. All parent folders are displayed separated by periods.

Name   The name serves as a comprehensible and unambiguous identification of the object in the current context. It must be unique in the folder. Identical names in different folders are allowed.

Rename in Content   If the name of a folder is changed and the folder conforms to the "Batch in Folder Convention" (contains a batch or job object of the same name), the *Rename in Content* checkbox is displayed. If this option is selected and the folder is renamed and cloned, the batch or job object in the folder is renamed as well.

Group   The *Group* field defines the user group that owns the object.

Cascade Set Group   The *Cascade Set Group* is used to set the user group for the folder and all the subordinate objects.

Create Batch in Folder   The *Create Batch in Folder* checkbox is displayed when creating a new folder and is used in support of the recommended convention for folder, batch and job objects. If a check mark is set here, a batch object of the same name is also created in the new folder.

Batch Exit State Profile   The *Batch Exit State Profile* field is only displayed if the *Create Batch in Folder* check box has been enabled. It is used to select the Exit State Profile for the new batch.

Add as Child   The *Add as Child* checkbox is displayed in the following cases.
When creating a new folder, the *Create Batch in Folder* checkbox is enabled and the parent folder of the newly created folder conforms to the "Batch in Folder" convention (it contains a batch or job object of the same name).
An existing folder is renamed and both the folder itself as well as its parent folder conform to the "Batch in Folder" convention (they contain a batch or job object of the same name). In this case, the *Add as Child* checkbox only has an effect when cloning the folder. The *Add as Child* checkbox is ignored when the folder is saved (renamed).
The *Add as Child* checkbox can be used to select whether the newly created batch or a copied (cloned) batch or job is also to be added as a child of the parent folder batch or job.

Environment   The *Environment* field is a selection box for choosing a valid environment object. This selection defines the additional requirements for all the jobs in the folder that have to be fulfilled by their execute environment.

Comment   A more detailed explanation about the object can be entered in the *Comment* field.

13.4.2 Content tab

The contents of the folder are shown in the "Content" tab.
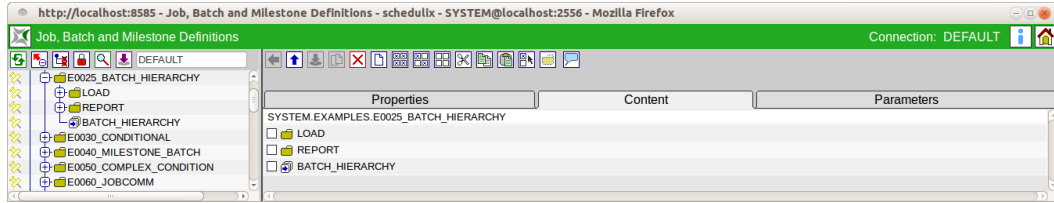
Figure 13.6: Content folder

When copying/cutting and subsequently pasting into "Content" tabs of folders, depending on whether the source and target of the operation correspond to the convention (see also a list is displayed showing the operations that are performed on the parent-child relationships to comply with the convention.

These operations can then be specifically selected before performing the paste operation.
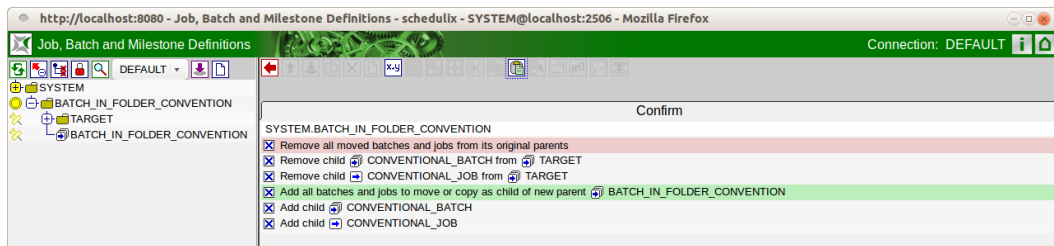
Figure 13.7: Supports the convention for paste

13.4.3 Parameters tab

The parameters can be defined in the "Parameters" tab.
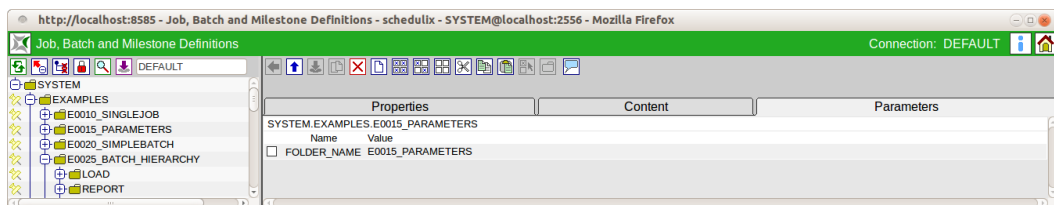
Figure 13.8: Parameters folder

Name   This is the name of the parameter. All the parameter values are exchanged between the individual jobs using this name.

Value   The *Value* field shows the parameter value.

13.4.4 Resources tab

The resources are shown in the "Resources" tab. Clicking the name of the resource opens the Resource Details tab.
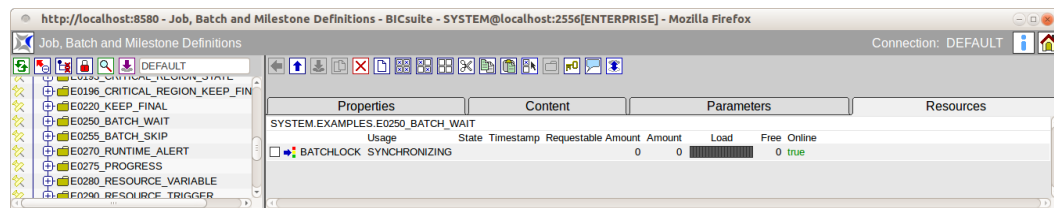
Figure 13.9: Resources folder

Usage   The *Usage* field specifies the "Resource" type. More details about resource types can be found in the Named Resource dialog window.

State   The *State* field is only visible if the resource usage has been set to "Synchronizing" and a Resource State Profile has been assigned.
It designates the current status of the resource in this scope or job server. The status can be set or changed with this value. The drop-down list contains all the valid Resource States of the Resource State Profile, which can be selected here.
This may be necessary when manually fixing an error to restore a resource to a state that allows further processing by follow-on jobs.
If you manually change this value, the value in the *State* field in the "Resources" tab is not automatically refreshed. Click the *Refresh* button if this is required and necessary.

Timestamp   This field is only visible if the resource usage has been set to "Synchronizing" and a Resource State Profile has been assigned. The *Timestamp* indicates the time of the last change made to a Resource State.
Time stamps play a role if an expiration interval has been defined in an expiry object. If this is the case, the time stamp must not be older than the specified interval. More details about expiration times can be found in Chapter 13.5.8.1.
If the completion of a job causes the Resource State to change, the timestamp is automatically updated by the BICsuite Server.

Requestable Amount   The maximum amount of resources that can be requested by a job.

Amount   The *Amount* is the number of resource instances allocated to or requested by the current job. If the value of the amount exceeds the currently available number that can be maintained in the *Amount* field in the "Resource Details" tab and there is no alternative scope available with a sufficient number, the job cannot be executed.

Load   The *Load* shows the graphical load of the resource.

Free   This field is only visible if the resource usage has been set to "Synchronizing" or "System".
The *Free Amount* designates the number of all instances of a resource that have not yet been allocated to jobs within the selected scope or job server.

Online   *Online* can be used to switch a resource in this scope or job server online or offline. An offline resource is temporarily unavailable.

13.5 Editor for Job Definitions

13.5.1 Properties tab

All information pertaining to the general properties of the job, batch, milestone or folder object is recorded in the "Properties" tab. Only the respectively required information is displayed in the "Properties" tab depending on the type you selected in the navigation or while creating the new object.
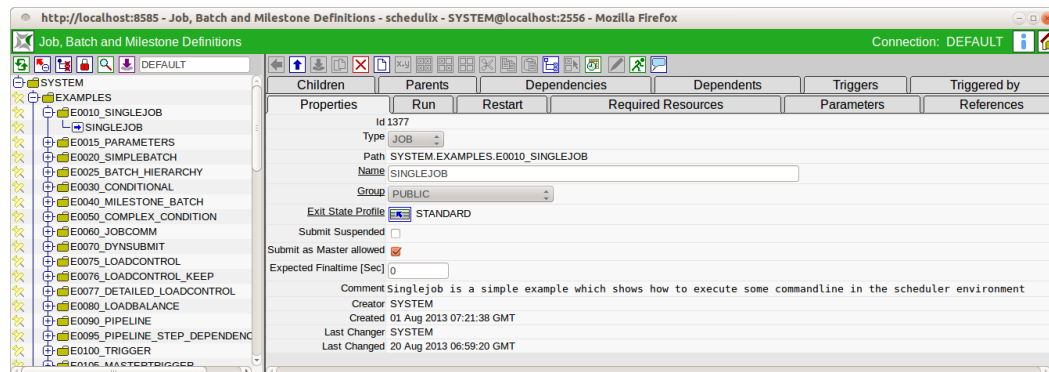
Figure 13.10: Job Definition properties

The fields in the "Properties" tab have the following meanings:

Id The *Id* is the unique identification number for the object.

Type This is the *type* of object. One of the following types can be selected here:

1. Batch

    The object is a batch. A batch is used as a startable container for a number of jobs, batches, or milestones. It does not itself have an executable program, but is only used as the start object for its child objects.

    An example of modelling as a batch is a number of jobs that have to be executed on a daily basis. These can all be grouped under a batch called "Daily". All weekly jobs are likewise grouped under the batch "Weekly", which also includes the "Daily" batch as this obviously has to be executed as a batch at the same time.

    A "Monthly" batch can additionally be created that contains all the monthly executable jobs. This means that only these three batches need to be created in this system in the Time Scheduling and all Child Jobs, Batches and Milestones are automatically started

    at these times.
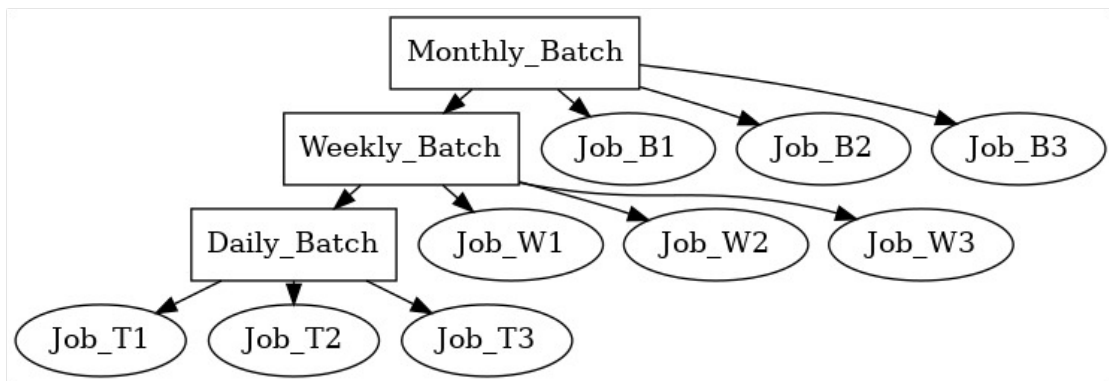
Figure 13.11: Examples of batches

The legend for the graphic can be found in Chapter .

2. Milestone

A milestone is required for several reasons. Firstly, it serves as a placeholder for a particular time during the course of a complex batch run. When this milestone is reached, a trigger can be used to perform an action (for example to send a text message or email).

Secondly, milestones are required to define complex dependencies between single jobs. Since a job can only be dependent upon either all the required objects or at least one required object, it is not possible to directly define more complex dependencies (upon Job1 and Job2 or Job3, etc.).

These options can now be defined using a milestone. By modelling a Milestone1 that is reached when Job1 and Job2 have been completed, the dependency of our job upon Job3 or Milestone1 can be modelled and the condition can now be mapped.

The legend for the graphic can be found in Chapter .

Milestones also have a unique characteristic: Dynamic children can be defined for them. Every time a job which is also a child of a milestone is dynamically submitted, that milestone also receives this job as a child.

This functionality enables "pipeline" processing. Let's suppose, for example, that data from a (partitioned) table needs to be processed. Due to the partitioning, it has been decided to parallelise the work. The relevant data is extracted for each partition and then processed further in some form in parallel operations. As soon as all the extraction jobs have finished, the data is to be processed further.
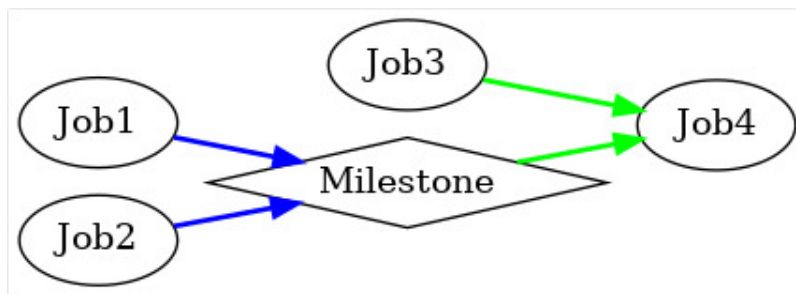
Figure 13.12: Example of a milestone

If the extraction job is defined as the dynamic child of a milestone, then the milestone will attain the final state at precisely the time all the extraction jobs have been completed successfully. Further processing of the table can be made dependent upon the milestone and in turn take place parallel to the processing of the extracted partition data.

3. Job

   All executable objects (programs, scripts, etc.) that are to be executed in the Scheduling System on a job server must be defined as a job

   .

4. Folder

   A folder represents a folder within the folder hierarchy for the executable objects. It is used for organising the other objects because in a normal scheduling system there are many different jobs, batches and milestones that have to be managed, and with a flat hierarchy it is very easy to lose track of them.

   How the hierarchy is structured is decided by each user. It could be logically organised (all objects on the same theme in one folder, etc.) or it could map the departmental or human resource structure (all objects relating to user/department X are placed in the same folder).

   The user should just be able to quickly find their way around the hierarchy and easily find the required objects.

   Folders cannot be executed and cannot have any dependencies. It is possible to define an environmentfor a folder. Furthermore, parameters can be defined which can be used by all the jobs located in the folder.

Path   The *path* describes the parent folder hierarchy. All parent folders are displayed separated by periods.

Name   The name serves as a comprehensible and unambiguous identification of the object in the current context. It must be unique in the folder. Identical names in different folders are allowed.

Decomposite   The *Decomposite* checkbox is displayed if an object type is changed from "job" to "batch". If a check mark is set here, a so-called decomposition is carried out. Not only is the object type changed to "batch", but a job is also created as a child of this batch which then takes over the job properties of the original job.
The batch object produced by the decomposition (with a job as a child) behaves, in the context of the Scheduling System, as the previously existing batch object. All the time schedules, dependencies, parameter references, etc. are retained.
The result is an ideal starting point for decomposing jobs into smaller single steps (process decomposition).

Job Name   The *Job Name* is only displayed if the *Decomposite* checkbox has been enabled. The name of the job object to be created during the decomposition can be entered here.

Rename Parent Folder   If the name of a batch is changed and the batch conforms to the "Batch in Folder" convention (it is located in a folder of the same name), the *Rename Parent Folder* checkbox is displayed. If this checkbox is enabled, when you save (rename) the batch the parent folder of the same name is renamed as well. The *Rename Parent Folder* checkbox is ignored when cloning a batch.

Create Batch Folder   The *Create Batch Folder* checkbox is displayed when a new batch is created or the type of an object is changed to "batch". It serves to support the recommended convention for folder, batch and job objects. If a check mark is set here, a folder with the same name as the batch is created for it and the new batch is created in this folder. The *Create Batch Folder* checkbox is also displayed if an object type is changed from "job" to "batch" (decomposition).

Add as Child   The *Add as Child* checkbox is displayed in the following cases:
When creating a new batch, job or milestone object where the parent folder of the new object to be created conforms to the "Batch in Folder" convention (contains a batch or job object of the same name).
In the case of an existing job or milestone where the name is changed and the parent folder conforms to the "Batch in Folder" convention. In this case, the *Add as Child* checkbox only has an effect when cloning the job or milestone. The *Add as Child* checkbox is ignored when the object is saved (renamed).
The *Add as Child* checkbox can be used to select whether the newly created object or a copied (cloned) job or milestone is also to be added as a child of the parent batch or job folder.

Group   The *Group* field defines the user group that owns the object.

Exit State Profile   An Exit State Profile can be selected in this field.

Submit Suspended   The *Submit Suspended* switch can be used to delay the actual start of a workflow, for example where external authorisation by a third person is necessary for a job. Only when this authorisation has been given can the job be started.

Resume   This field is also only displayed if the option *Submit Suspended* has been enabled.
Here you can choose whether a workflow should be resumed automatically.
The following possibilities are available:

- NO: Deselects this function

- AT: Selects an automatic resume action at a fixed time. The *Resume Time* input field is displayed.

- IN: Selects an automatic resume action after a time period has expired. The *Resume In* and *Unit* input fields are displayed.

Resume Time   This field is only displayed if "AT" was selected for *Resume*. The required resume time is entered here in the format "YYYY-MM-DDTHH:MI:SS".
This format is based on the ISO standard 8601 and also permits incomplete entries.
If you enter 'T16:00', the job will be resumed at 16:00 hours (based on the current time).

Resume In   This field is only displayed if "IN" was selected for *Resume*.
Here you can specify how many time units (see *Unit*) the system is to wait for until the resume action is triggered.

Unit   This field is only displayed if "IN" was selected for *Resume*.
This field is used to define whether the entry in *Resume In* is in minutes (MIN), hours (HOUR) or days (DAY).

Nice Value   This field is only visible if the object is a batch.
The *Nice Value* indicates the priority to be applied for running children of this object.
It is an offset value which is added to the priority of the children.
Example:
If the process has a Nice Value of -50 and the child has a priority of 100, then the priority of the child process of the batch is 50. If the child process were to be started on its own, the priority would be 100.

Submit as Master Allowed   This field is only visible if the object is a batch or a job.

This switch specifies whether it is possible to submit this job as a Master Job in the Submit Batches and Jobs dialog window. If the switch is not enabled, this is not possible and the job or batch cannot be executed on its own, but only as a child of any parent job.

Comment   A more detailed explanation about the object can be entered in the *Comment* field.

13.5.2 Run tab

The "Run" tab is only shown if an object of the type "Job" is in the navigation or if the type "Job" has been selected in the selection box when creating a new object. All information about the command lines of a job object that is to be executed is managed in the "Run" tab.
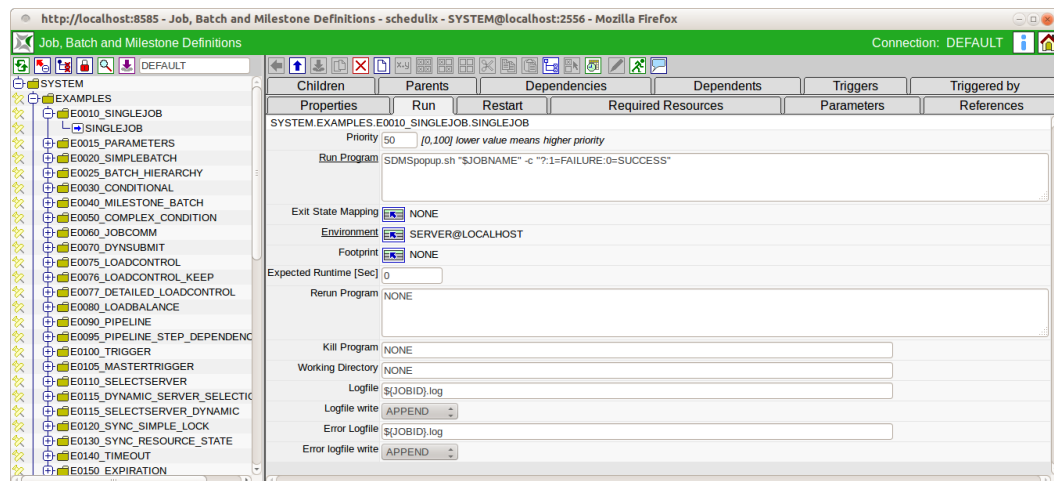
Figure 13.13: Job Run tab

The fields in the "Run" tab have the following meanings:

Priority   The field *Priority* states the urgency with which the process is taken into account by the Scheduling System if it is to be started. The range of values for priorities ranges from 100 (very low) to 0 (very high). All startable jobs and their priorities are taken into account in the system and then the job with the highest priority is started.
If two jobs have an identical priority, the job with the lowest ID is started.
The longer the waiting time for a job before it is started in the Scheduling System, the higher its priority. This means that if a process with a priority of 50 is started, its priority increases per time unit. This is set in the Scheduling System configuration. For example, a job can increase by one priority point per half hour. This means that after half an hour the priority is 49, after an hour it is 48, and so on. Consequently, older jobs get a higher priority and are considered before newly submitted jobs.

Run Program   A command line to be executed by a suitable job server is entered in the *Run Program* field. This command line is first interpreted by the Scheduling System and then decomposed into the program call and single parameters. The Bourne shell rules with regard to quoting are followed here.

If environment variables or parameters are to be specified, these must be quoted conform to the respective runtime environment (depending on the shell or command line interpreter of the environment, for default parameters see Environment).

A list of all the usable standard parameters can be found in the paragraph 13.5.10.2. Please refer to the documentation for your command line interpreter (shell) for more information about the quoting to be used.

If the target system is a Windows system, some other difficulties are very likely to arise. Under Windows, if a table consisting of executables and parameters is offered for the execution, Windows first constructs a command line from the table and then interprets this command line again.

The use of quoting is also problematic under Windows, as the called programs remove or do not remove the quotes depending on the type of executable (`.exe` or `\verb`.bat).

If the program to be executed (first element of the command line) is a valid integer, the command line is not executed by the job server, but the job is handled as if it had ended with the integer as the Exit Code.

Dummy Jobs with 'true' or 'false' as the program can now be implemented as '0' instead of 'true' or '1' instead of 'false', and can therefore be processed much more efficiently and quickly by the system.

If it is actually necessary to execute an executable with a number as its name, this can be achieved using a path prefix ('./42' instead of '42').

Exit State Mapping An existing Exit State Mapping can be entered in the *Exit State Mapping* . This defines the mapping of the Exit Code for a job after an Exit State. In the absence of a specified mapping, the default mapping from the Exit State Profile is used. The mapping must be compatible with the profile.

Environment In the *Environment* field it is necessary to select an existing Environment . All the Named Resources in this environment must be present on a job server so that the job can be run on that server.

Footprint An existing footprint can be selected in the *Footprint* field. All the Named Resources for this footprint must be present on a job server in a sufficient quantity so that the job can be run on that server.

Expected Runtime The *Expected Runtime* describes the expected runtime for a job. It must be entered manually and can be used, for example, for monitoring the runtime.

Expected Finaltime The *Expected Finaltime* describes the expected runtime for a job (from when it is submitted until it attains its final state). It must be entered

manually and is used for showing the job in the calendar. Of course, there is also a default parameter for this so that the expected value is available to the job.

Rerun program   The *Rerun Program* field specifies the command that is to be executed when the job is run again after an error state (Rerun). This may be a different command than the run program, as it may be necessary to perform clean-up work on a repeated start before the original command can be set up again.

If no value was entered in the Rerun program, the Run program is called when restarting the job.

See 'Run Program' above for details regarding processing (quoting, treating integers as a program name).

Kill Program   The *Kill Program* determines which program is to be executed to end a currently running job.

If no value is entered in this field, it will not be possible to prematurely terminate the job from the BICsuite interface.

An example of a Kill program in a Unix environment is the **"kill"** command. The process ID of the current job can also be accessed in the Kill program using the predefined system variable "PID" (example: "kill -9 $PID").

Working Directory   The *Working Directory* specifies the directory in which the run program is to be started. The job server switches to the specified directory before the Run program is started.

If no value is entered in this field, the default directory of the job server is used as the working directory.

It is also possible to specify a parameter, folder, scope or job server parameter as the working directory.

Logfile   The *Logfile* field specifies the file in which all the normal outputs of the Run program are to be returned. These are usually all the outputs that use the standard output channel (STDOUT under UNIX).

Parameters can also be replaced in the *Logfile* field. For example, the JOBID parameter can be used to allocate a job a unique name or a TIMESTAMP can be used for the date.

If no value is entered, all the outputs will be discarded.

If no path is specified in the log file, it is written to the working directory.

Logfile Write   The *Logfile Write* field tells the system where a log file is to be created. The following options are available:

- Append

112

The Append option appends all the created log files to an existing log file. If no log file exists yet, a new one is created.

- Truncate

  With the Truncate option, the log file is deleted and a new one is created for each new action.

Error Logfile   The *Logfile* field specifies the file in which all error outputs from the Run program are to be returned. Error outputs are all the outputs that use the error output channel (stderr under UNIX).

Parameters can also be replaced in the *Error Logfile* field. For example, the JOBID parameter can be used to allocate a job a unique name or a TIMESTAMP can be used for the date.

If no value is entered, all the outputs will be discarded.

If no path is specified in the log file, it is written to the working directory.

If the same name as the one in the *Logfile* field is being used, these outputs will also be written to this file. However, the sequence of the outputs (Error or Normal) is random due to the buffering. However, regardless of the content of the Error Logfile Write field, outputs from the normal output channel (stdout) are not overwritten.

Error Logfile Write   The *Error Logfile Write* field tells the system where the error log file is to be created. The following options are available:

- Append

  The Append option appends all the created error log files to an existing error log file. If no error log file exists yet, a new one is created.

- Truncate

  With the Truncate option, the error log file is deleted and a new one is created for each new action.

13.5.3 Restart tab

The restart behaviour for a job can be defined in the "Restart" tab.

A job will only be restarted if its Job State is "FINISHED".

If a job has a "RESTARTABLE" Exit State due to an "ERROR" Job State, NO restart is performed!

The fields in the "Restart" tab have the following meanings:
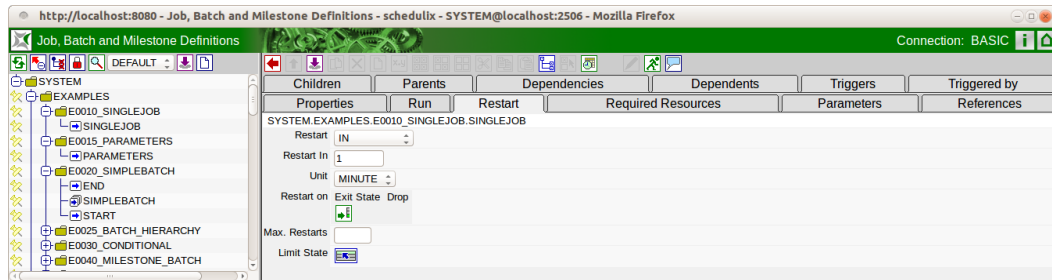
Figure 13.14: Job tab Restart

Restart This determines whether the job is to be automatically restarted if it attains a "RESTARTABLE" Exit State.
The following selections are possible:

- NO: No restart. No other fields are displayed in the "Restart" tab.

- IMMEDIATE: Immediate restart.

- AT: Restart at a specified time (e.g. "T16:00" at 16:00 hours).

- IN: Restart after a delay (e.g. 10 minutes).

Restart Time This field is only displayed if "AT" was selected in the *Restart* field. Time when the job is to be restarted using the format "YYYY-MM-DDTHH:MI:SS". This format is based on the ISO standard 8601 and also permits incomplete entries. If you enter 'T16:00' here, the job restart will be suspended and resumed at 16:00 hours (based on the submit time of the job).

Restart In This field is only displayed if "IN" was selected in the *Restart* field. Number of time units (see *Unit*) for the delay until the job restarts.

Unit This field is only displayed if "IN" was selected in the *Restart* field. Units for the delay interval (see *Restart In*).

Restart On Table of Exit States that are to trigger a restart.
If the table is empty, jobs are started for all "RESTARTABLE" Exit States.

Condition Optional condition (from BICsuite PROFESSIONAL) for triggering a restart.

114

Max. Restarts   Maximum number of times that the job is restarted.
The system parameter "TriggerSoftLimit" applies if nothing is entered here. Entering 0 will restart the job any number of times.

Limit State   An Exit State can be set here that the job should assume if the maximum number of restarts has been reached.

13.5.4 Children tab

In the "Children" tab, all executable objects that have been defined as a child of the current job are displayed as a list. Additional objects can be added as children using Copy and Paste.
The sequence in which the children are run is not fixed, but can be defined in the dependencies.
The "Children" tab looks like this:
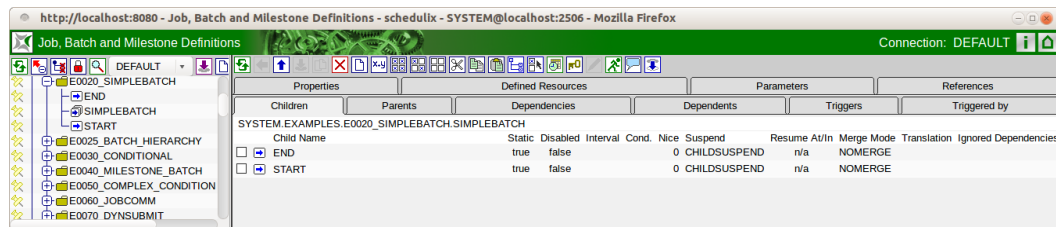
Figure 13.15: Batches and Jobs; Children tab

A list of all job, milestone and batch objects that are subordinate to the selected job as children is shown here.
The columns in the list shown above have the following meanings:

Child Name   This is the name of the child. Clicking the name opens the "Child Details" tab.
The other columns are explained in the next section.

13.5.4.1 Child Details tab

The "Child Details" tab is displayed after selecting the name of a child object in the "Children" tab. Here, all the data already displayed in the table as elements is listed again and can also be modified here. The tab looks like this:
The fields in the "Child Details" tab have the following meanings:

Child Name   The *Child Name* is always displayed with the full path to the parent folder hierarchy.
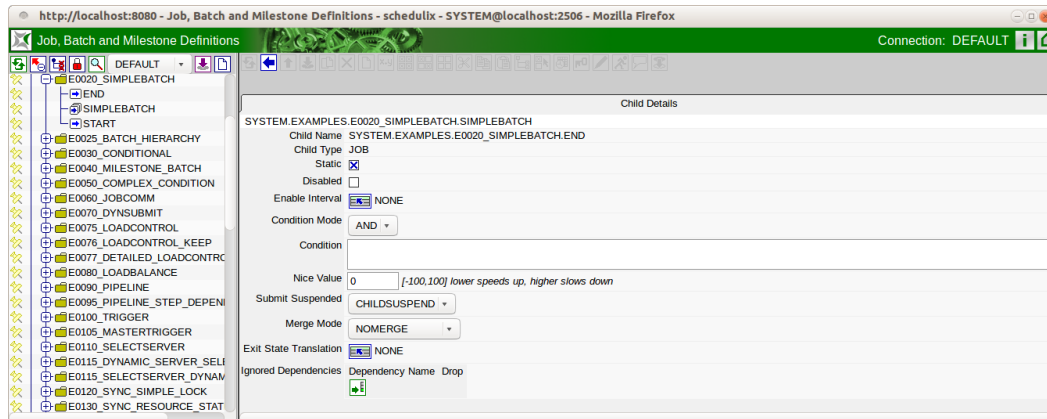
Figure 13.16: Batches and Jobs; Child Details

Child Type   The *Child Type* specifies the type of child (batch, job or milestone).

Alias   A child can be assigned a new logical name by entering it in the *Alias* field. This designation must be unique within all children of the selected job. The alias is only significant for dynamic children. The Parent Job can create additional instances of the children by stating the alias in the submit. This means that the parent does not need to know where the child has been saved. This abstraction makes it considerably easier to reorganise the folder structure.

Static   The *Static* switch indicates whether a child is to be submitted statically or dynamically. In the case of a static child, the child instance is instantiated when the parent is submitted. If it is a dynamic child (the Static switch has not been set), one or more instances of the child can be dynamically created at runtime.
Children are often used for dynamic parallelisation tasks. Here, the run program only decides at runtime whether and how many dynamic children should be created; these are created by the BICsuite API.
This makes it possible to divide a process into any number of sub-processes working in parallel.

Disabled   If the *Disabled* switch is set, the child is submitted as being "disabled". All children of those batches or jobs submitted as being "disabled" are likewise submitted as being "disabled". "Disabled" jobs are not executed, but go straight to their default Exit State if all their dependencies have been fulfilled.
This makes it possible to exclude individual jobs or batches of a processing chain from being processed while retaining the dependency relationships and the processing sequence of the processing chain.

116

Enable Interval Optionally, an "Enable" interval can be set. If this interval is set, a check is run when the child is submitted to determine whether the submit time for the master is a valid time for the "Enable" interval. If this is not the case, the child is submitted as being "disabled".

Condition mode If a condition has been entered, Condition mode can be used to check whether the condition is linked with any Enable Interval AND or OR that may have been set.

Condition If a condition is entered here, it is checked when the child is submitted. If the condition is not fulfilled, the Child Job is disabled. Condition mode determines how the condition is linked to any set Enable Interval.
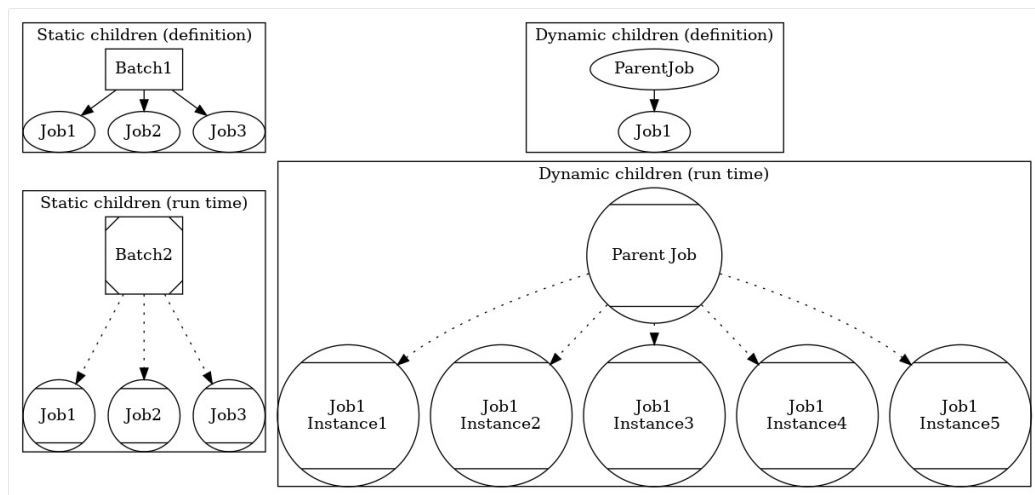
Figure 13.17: Static and dynamic children

The legend for the graphic can be found in Chapter 1.8.

Nice Value The *Nice Value* indicates the priority with which the selected child is to run. It states an offset value which is added to the priority for the child.
Example: If the process has a Nice Value of -50 and the child has a priority of 100, the job instance started as a child of this job has a priority of 50.
If the child process is started on its own, it has a priority of 100.
If the parent is a batch, the Nice Value of the child is added to that of the parent.

Submit Suspended The *Submit Suspended* parameter specifies the form in which the child object is delayed when being started or if it can be started immediately. The following options are available:

1. YES

   The child is created as being suspended when it is submitted and it has to be started by being explicitly released.

2. NO

   The child is not suspended when it is submitted and can be started immediately.

3. CHILDSUSPEND

   Whether a delay takes place or not depends on the *Suspend* field for the Child Job. This means that the setting defined in the Child Job is applied.

Resume   This field is only displayed if the option *Submit Suspended* has been set to "YES".
Here you can choose whether a workflow should be resumed automatically.
The following possibilities are available:

- NO: Deselects this function

- AT: Selects an automatic resume action at a fixed time. The *Resume Time* input field is displayed.

- IN: Selects an automatic resume action after a time period has expired. The *Resume In* and *Unit* input fields are displayed.

Resume Time   This field is only displayed if "AT" was selected for *Resume*.
The required resume time is entered here in the format "YYYY-MM-DDTHH:MI:SS".
This format is based on the ISO standard 8601 and also permits incomplete entries.
If you enter 'T16:00', the job will be resumed at 16:00 hours (starting from the Job Submit time).

Resume In   This field is only displayed if "IN" was selected for *Resume*.
Here you can specify how many time units (see *Unit*) the system is to wait for until the resume action is triggered.

Unit   This field is only displayed if "IN" was selected for *Resume*.
This field is used to define whether the entry in *Resume In* is in minutes (MIN), hours (HOUR) or days (DAY).
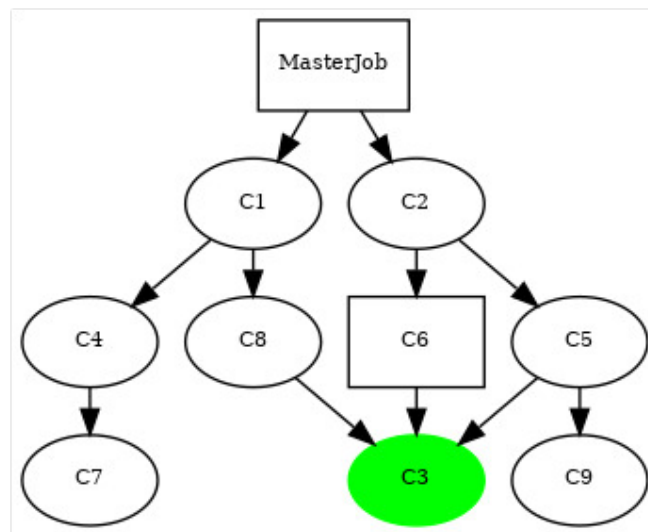
Figure 13.18: Example of Merge Mode

Merge Mode   Merge mode specifies whether a child object is started multiple times within a Master Job run or not.

The legend for the graphic can be found in Chapter 1.8.

The following options are available:

1. No Merge

   Jobs are not merged. The Child Job is started without checking whether this job has already run in the current Master Batch.

   In the example shown in the screenshot 13.18, Job C3 is started three times if No Merge is set in each parent-child relationship.

2. Failure

   At the submit time for the job, the system checks whether the Child Job has already been started and, if this is the case, an error message is returned and the start of the Master Job is cancelled.

   In the example shown in the screenshot 13.18, the job cannot be submitted.

3. Merge Local

   A merge is performed in the child hierarchy of the parent object. This means that if the job in the child hierarchy has already been started once, it will not be started again.

   If the job has already been started within the Master Job but outside the child hierarchy of the parent, this is not taken into account and the job is submitted again.

The dependencies of this job are taken into account in the same way as if this job had been started in this child relationship.

If the Merge Mode Local is set for the child relationship C2→C6 and C2→C5 in the example above, Job C3 is only started twice as the merge would be performed in the child hierarchy of C2.

As the relationship C8→C3 is not in this child hierarchy, C3 is run there as no merge is performed here.

4. Merge Global

The entire Master Job is checked to see whether this job has already been started before and, if this is the case, the job is not started again.

This may be necessary if a job should only run once, but it is needed as a requirement by several jobs. It can be defined as a child of multiple parent processes. The parent process that is run first starts the process. The other parent processes use the Merge Global to check whether this job has already been run and they don't start it again.

The dependencies of this job are taken into account in the same way as if this job had been started in this child relationship.

In the example above, Job C3 is only started once as a global merge takes all relationships in the Master Job into account and therefore performs the merge for the relationships (C8→C3, C6→C3 and C5→C3).

Which relationship is used to actually start the job is immaterial.

Exit State Translation   In this selection mask, you can select the Exit State Translation which is used to tell the parent object the Exit State of the child object.
More details about Exit State Translations can be found in Chapter 5.

List of Ignored Dependencies   A list of dependencies can be added here which are to be ignored by the child within this parent-child relationship.
All these dependencies are now no longer taken into account for the child and the child can be started without the dependency of the parent having been fulfilled.
An example of Ignored Dependencies is shown in the screenshot 13.19 . The legend for the graphic can be found in Chapter 1.8.
Only those dependencies that have entered a value in the *Dependency Name* field can be ignored. Unnamed dependencies cannot be selected in the selection box and therefore cannot be ignored.
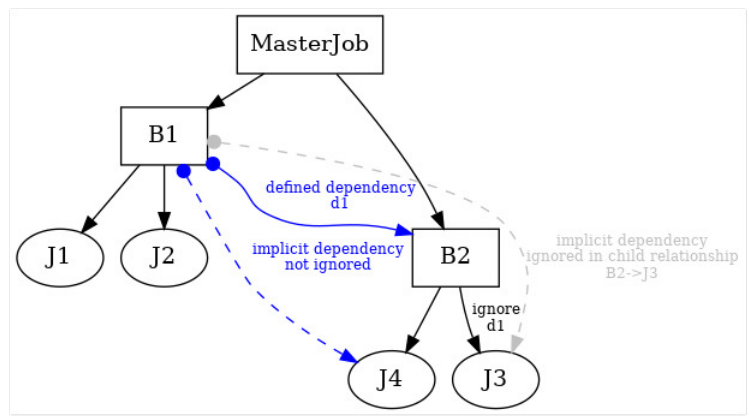
Figure 13.19: Example of Ignored Dependencies

13.5.5 Parents tab

The "Parents" tab displays a list of parent objects that use the selected object as a child. This therefore figuratively represents the view upwards while the children represent the view downwards as all children of the currently selected objects in the   are displayed there.
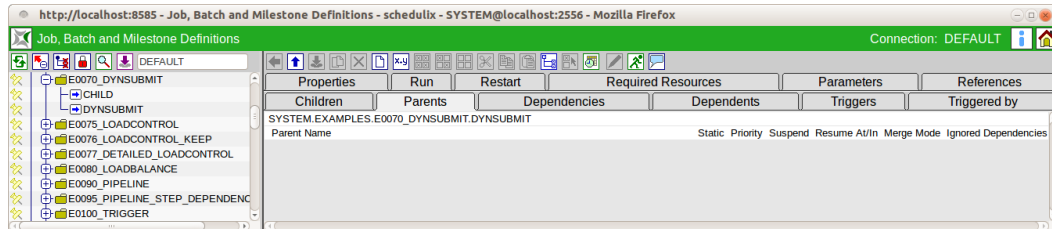
Figure 13.20: Batches and Jobs; Parents tab

The tab is for information purposes only; no values can be changed or new lists added here. This is only possible in the direction parent-to child.
The columns in the list above are explained in the "Children" tab.

13.5.6 Dependencies tab

The "Dependencies" tab displays a list of all objects on which the current object is dependent. A dependency exists if the current object is not allowed to start before all objects on which the job is dependent have been executed and ended with a certain Exit State.
Additional objects can be added as required Scheduling Entities using Copy and Paste.
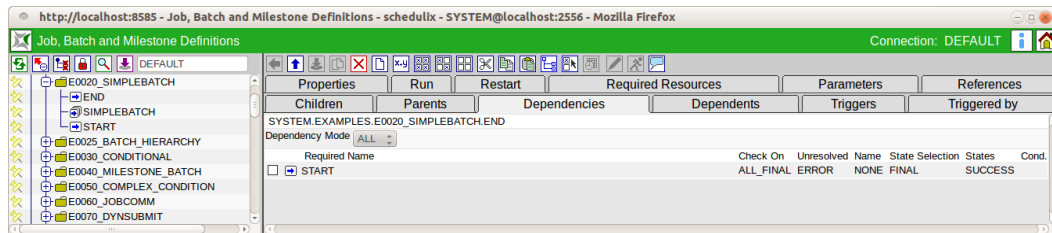
Figure 13.21: Batches and Jobs; Dependencies tab

The field in the list shown above has the following meaning:

Dependency Mode   The "Dependency Mode" indicates the context in which the list of dependencies must be viewed. The following options are available:

1. ALL

   All the dependencies must be collectively fulfilled before the object can start. This corresponds to linking all the conditions with **"AND"**.

   Example: Object C is dependent on Job A and Job B (i.e. it should only start when A and B are finished), and so the ALL Dependency Mode must be selected.

2. ANY

   At least one of the dependencies must be fulfilled before the object can start. This corresponds to linking all the conditions with **"OR"**.

   Example: Object C is dependent on Job A or Job B (i.e. it should start when A and B are finished), and so the ANY Dependency Mode must be selected.

The dependencies of a job can contain a link to either ANY or ALL. If more complex dependency links are to be mapped (for example, dependent of A or B and C or D),
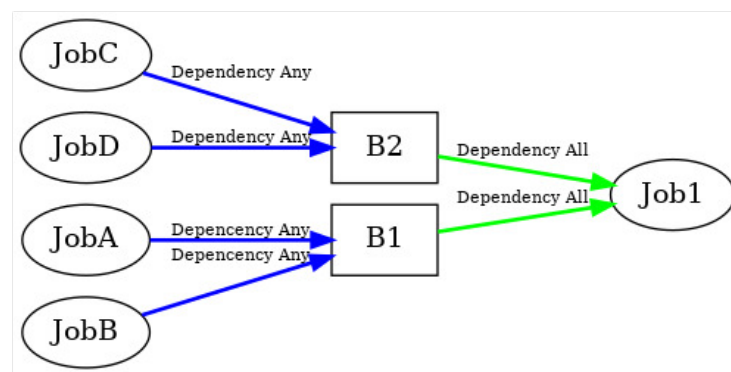
Figure 13.22: Beispiel für Dependency Modes

this is done as shown in the screenshot 13.22 by using a milestone or batch object. The legend for the graphic can be found in Chapter 1.8.

In the example above, Batches B1 and B2 were generated. B1 has linked the dependencies A and B using ANY. and B2 has linked the dependencies C and D with ANY as well. Our object (Job1) can then link the two batches as dependencies using ALL and has therefore mapped the complex condition (dependent on A or B and C or D).

The "Dependencies" list shows all those objects upon which the currently selected object is dependent. All the elements in the list are linked using Dependency Mode. The columns in the list are explained in the next section.

13.5.6.1 Dependency Details tab

The "Dependency Details" tab opens when an entry has been selected in the "Dependencies" tab by clicking the Required Name.
All the aspects of a dependency between the two jobs can be configured here. The tab looks like this:
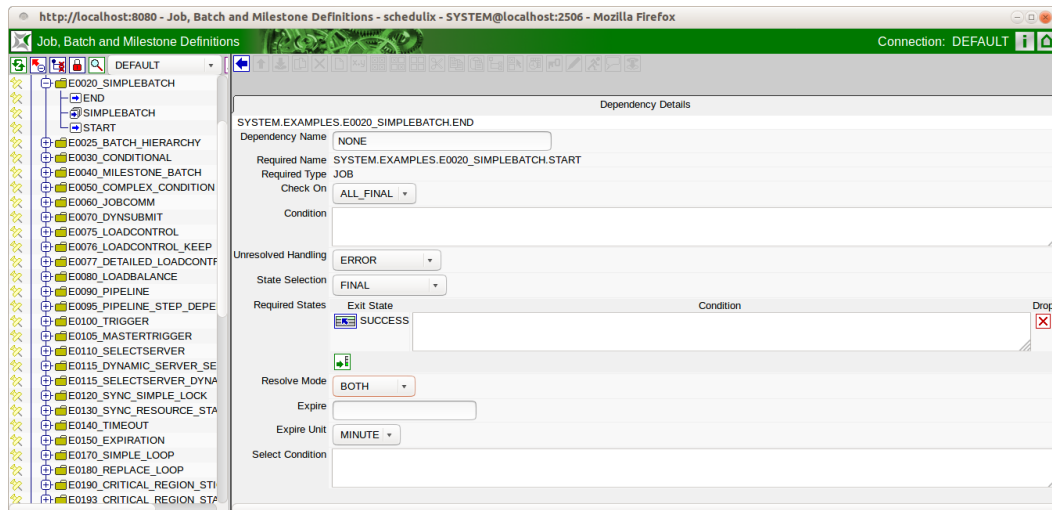
Figure 13.23: Batches and Jobs; Dependency Details

The fields in the "Dependency Details" tab have the following meanings:

Dependency Name   The *Dependency Name* is optional and is the prerequisite for ignoring dependencies.

Required Name   This is the name of the object upon which the currently selected job is dependent. It corresponds to the full name with the parent folder hierarchy.

Required Type   The type of the required object is displayed in the *Required Type* field.

Check On   The *Check On* field describes the type of completeness check for the required object. The following options are available:

1. ALL_FINAL

    If this option is selected, both the required job and all its children must have attained a Final State.

124

2. JOB_FINAL

   If this option is selected, the system only checks whether the job has attained a Final State. The state of the child objects is not checked.

Condition   The *condition* is a condition that must also be fulfilled for the dependency to be considered as being fulfilled. Parameters of the type "Resource Reference" cannot be used because no resources are assigned at the time the condition is evaluated.

Unresolved Handling   The selection field *Unresolved Handling* describes what to do if the current Master Batch does not contain a dependent object instance.
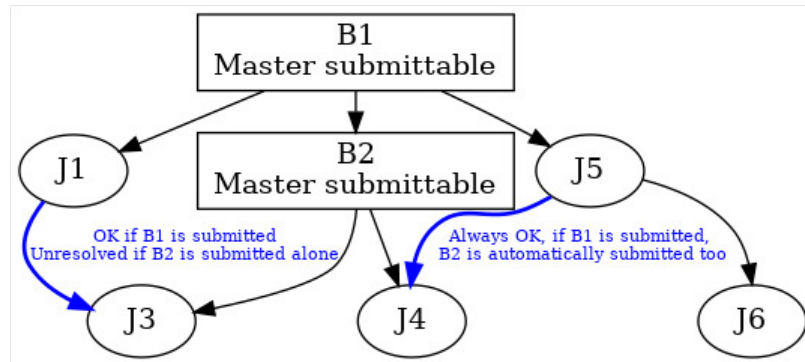
Figure 13.24: Example of Unresolved Handling

The legend for the graphic can be found in Chapter 1.8.
The following options are available:

1. IGNORE

   The dependency is ignored if the required job does not exist. If the job exists, it is handled in the normal way.

   An IGNORE may be necessary if smaller batches have been merged to form larger ones. If the small batch runs on its own, it can ignore all dependencies that only need to be observed during the bigger run. If it runs as part of the large batch, the named jobs then exist and the dependencies must be observed.

   In the example above, an Ignore must be entered in the dependency J1→J3 if batch B2 is also allowed to start on its own.

2. ERROR

If the required job is not present, the submit operation is terminated with an error.

3. SUSPEND

The current job switches to the Suspend State. This means that it must be started by an external program or manual intervention as soon as the required job is present.

In the example above, Job J3 would go into the Suspend State if J1 were a dynamic child of B1 and it is not clear when the job will be submitted.

The idea behind SUSPEND is that neither an unattended start nor an "ERROR" occur.

4. DEFER

The dependency is resolved at a later time if the required job is generated in the workflow by a trigger or dynamic submit.

If a workflow becomes inactive and only contains jobs with the status DEPENDENCY_WAIT, all jobs that are only waiting at this time for DEFER dependencies become UNREACHABLE.

5. DEFER_IGNORE

The dependency is resolved at a later time if the required job is generated in the workflow by a trigger or dynamic submit.

If a job is only waiting for DEFER_IGNORE dependencies, these are ignored.

State Selection   The *State Selection* field defines which Exit States of the required job fulfil the dependency and the dependent job can start. The following options are available:

1. FINAL

If this option is selected, the required job must have attained a Final State. This option allows the required Exit States to be selected in the following *Required States* table.

2. ALL_REACHABLE

If this option is selected, all Final Exit States that were not designated as being Unreachable in the Exit State Profile of the required job are valid.

3. UNREACHABLE

If this option is selected, only the Exit State that was designated as being Unreachable in the Exit State Profile of the required job is valid.

4. DEFAULT

   If this option is selected, all Final Exit States that were designated as a Dependency Default in the Exit State Profile of the required job are valid.

Required States   Here is the list of all valid Exit Statesthat the required object must have so that the dependency is fulfilled and the dependent job can start.
This list is only visible if the option FINAL was selected in the *State Selection* field.
Example:
A Job A required by Job B can take on two Exit States (SUCCESS or WARNING). Since Job B should only start if Job A has ended with an Exit State SUCCESS (i.e. if it has run successfully), there must only be one entry with SUCCESS in the list of Required States.
Once Job A has been completed and has received the SUCCESS Exit State, Job B can then start. If Job A finishes with WARNING, Job B cannot start.
This list is necessary because a dependent job can accept multiple Exit States as a requirement. In our example, this means that Job B would have entered both Exit States (SUCCESS and WARNING) as Required States if it is to start in any case regardless of whether Job A was completely successful or it ended with a warning.

Resolve Mode   How the dependency is to be resolved can be selected in the *Resolve Mode* field. The following options are available:

1. INTERNAL

   If this option is selected, the dependency is only resolved within the master batch of the dependent job or batch.

2. BOTH

   This option is used to first attempt to resolve the dependency within the master batch of the dependency job or batch. If this does not succeed, the dependency is resolved externally.

3. EXTERNAL

   This option defines an external resolution in other master batches. If all non-external dependencies are not fulfilled, an attempt is made to determine the requested job. The respective jobs are determined in other master batches, and it is verified whether the expiration and select conditions are fulfilled. In the case of multiple candidates, the most recently submitted job or batch is selected and assigned.

Expire The *Expire* and *Expire Unit* fields define the duration of the submit time of the dependent job or batch after the submit time of the requested job or batch in order to be able to perform a resolution of the dependency.

The *Expire* and *Expire Unit* fields are only visible if BOTH or EXTERNAL have been selected in the *Resolve Mode* field.

Select Condition The *Select Condition* defines a condition that must be fulfilled for a resolution of the dependency to be possible. Parameters of the requested job or batch can be addressed with $REQUIRED.<parametername> and $DEPEN-DENT.<parametername> for parameters of the job or batch to be executed.

The *Select Condition* field is only visible if BOTH or EXTERNAL have been selected in the *Resolve Mode* field.

13.5.7 Dependents tab

The "Dependents" tab shows all the Scheduling Entities that are dependent upon the selected Scheduling Entity. This is the reverse view of the "Dependencies" tab, as the dependent jobs are now displayed and not the required jobs.

The tab looks like this:
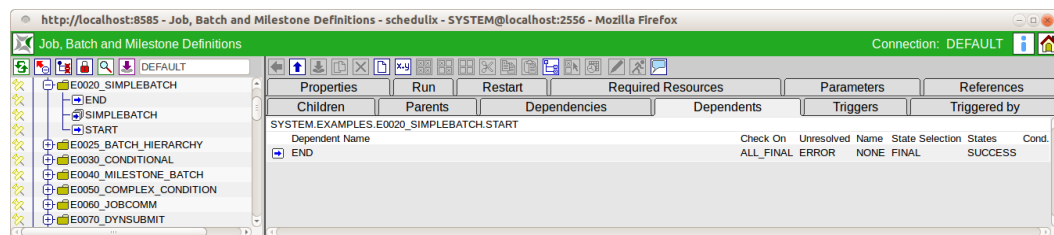
Figure 13.25: Batches and Jobs; Dependents tab

The list shows all the objects (jobs, milestones, batches) which require the selected Scheduling Entity. The list is purely informative and cannot be edited. The field values cannot be changed in this view either. To make a change, navigate to the object displayed in the list and select the "Dependencies" tab. The change can be made here.

The list columns are explained in the "Dependency Details" tab.

13.5.8 Required Resources tab

The "Required Resources" tab shows all the Resources that are required to execute this object. Additionally required resources can be added here.

All the resources that have already been defined by the environment or a footprint are displayed here. These can only be removed by changing the corresponding environment or footprint.

128

The "Required Resources" tab looks like this:
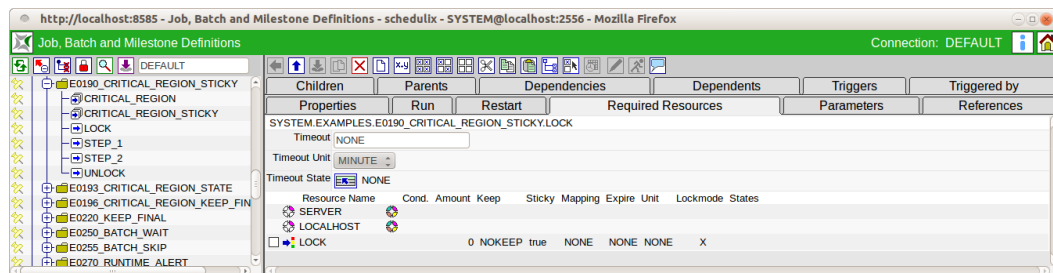
Figure 13.26: Jobs; Required Resources tab

The fields and columns in the "Required Resources" tab have the following meanings:

Timeout   The *timeout* specifies how long the job is to wait for all the resources after it has been submitted. If no value is entered here, the job will wait for them indefinitely.

The value in this field must be regarded in conjunction with the *Timeout Unit* field. The value for the timeout is entered in the *Timeout* field. The unit is entered in the *Timeout Unit* field.

Example: If the *Timeout* field contains the value 5 and the timeout unit is **MINUTES**, a job waits 5 minutes for a resource and, if this is not available during this time,

this job cannot be started and ends with the Exit State defined in the *Timeout State* field.

Timeout Unit   The *Timeout Unit* is the unit that is to be used for the selected value from the *Timeout field.* If no value is entered in the *Timeout* field, the value in the *Timeout Unit* field is irrelevant.

Timeout State   Here you can select the Exit State which the job is to take on should a timeout occur. By clicking the selection button you can choose those Exit States from the list that were made available by the *Exit State Profile*.

For example, you could define a separate Timeout State or use the Failure State.

If no value is entered in the *Timeout* field, the value in the *Timeout State* field is irrelevant.

The "Resources" list shows all the resources that are currently required by the selected job for it to be executed. The list contains the following columns:

Resource Name   This is the name of the resource that is needed to start the current job.

Next to the field name is an icon which indicates the context from which the resource originates. The following icons are used:

- Footprint

  The resource originates from the *Footprint* field. An individual request can be created for each resource. This overwrites the original request.

- Environment

  The resource originates from the *Environment* field. It is only available for information purposes and cannot be changed in this tab.

- No icon

  If the resource name is not followed by an icon, the resource was added in this tab and can also be modified here. By clicking the name, you can switch to the "Resource Details" tab and change the values in the list. It is also possible to mark and then delete or move this line by clicking the *Selections* button in front of the name.

Cond. A condition can be entered in the *Cond.* field that must be fulfilled to ensure that the resource is recognised as being valid. The other columns are explained in the next section.

13.5.8.1 Resource Details tab

The "Resource Details" tab opens when you select a list entry in the "Required Resources" tab.
All the details for an allocated resource can be entered and changed in this tab.
The tab looks like this:
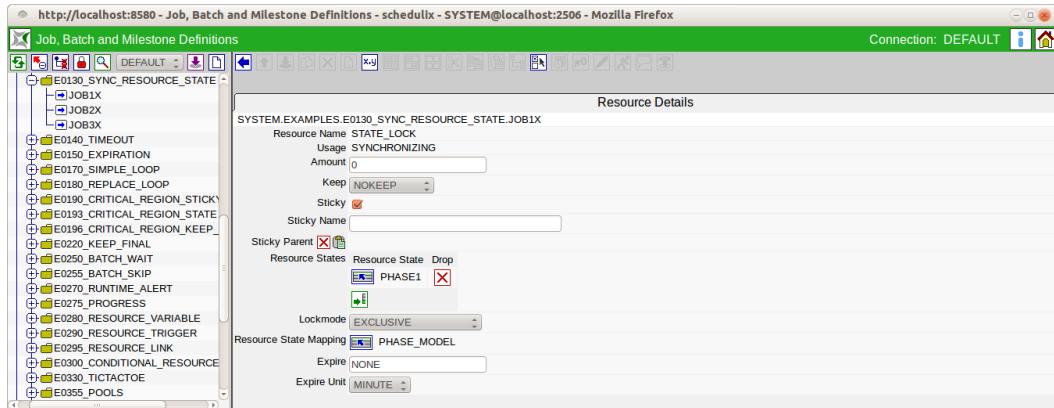
Figure 13.27: Batches and Jobs; Resource Requirement details

The fields in the "Resource Details" tab have the following meanings:

Resource Name   This is the name of the required resource. The full name together with the parent folder hierarchy is displayed here.

Usage   This is the usage of the resource. More information about the usage can be found in Chapter 9.3.3.

Amount   This field is only displayed if the usage type is either "SYSTEM" or "SYNCHRONIZING".
This is the amount of resources required by the job. Maximum amounts of the resource are made available on the job server (or in the parent scopes, see Job server). Every job that requires an amount of this resource reduces the number of free amounts.
Example:
A job server makes the amount '5' available to Resource A (for example 5 CPU units on this server).
Job A starts now and requires the amount '3'. The job server has now been allocated 3 CPU units, leaving 2 still available.
Job B wants to start, but it also requires an amount of '3'. The job cannot be handed over to the job server to be started because at this moment only 2 CPU units are available.

Job B can only start when Job A has finished and at least three CPU units are available.

Keep   The value of the Keep parameter is set here. More information about the Keep parameter can be found in Chapter 11.3.1.

Sticky   This switch is used to control how resources are returned. If this switch is set, the resource is retained within a Master Job until the last job that requires this resource with an activated Sticky flag has been completed.
This functionality is for keeping reserved resources within a Master Job while preventing them from being used in the interim by other jobs.
Example:
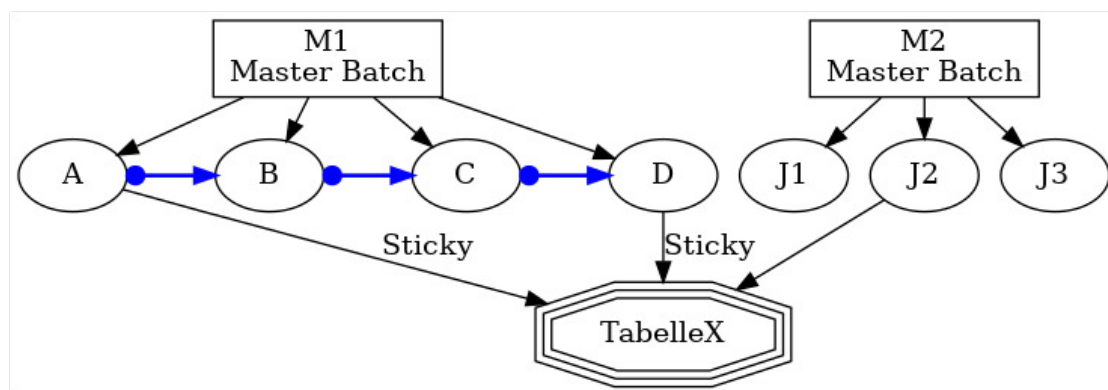
Figure 13.28: Example of Sticky Handling

Master Job M1 comprises the jobs A, B, C and D, which are executed in succession. Job A generates Table X, which is required by Job D. Jobs B and C don't need this Table X. Other Master Batches (M2) also want to access the table and modify it. However, as long as Job D has not yet started, this should be prevented, as these actions would corrupt the result of job A and render it unusable. The Table X is now mapped as a Synchronizing Resource.
If a Sticky flag is not set, the resource would be returned after Job A has been completed and the table could be modified by J2. With a set Sticky flag (which has to be done in Job A and Job D), the resource is not returned following completion of Job A, but is retained in the Master Job as being allocated.
Other jobs cannot change this resource now. Only once Job D (the last job in this Master Job that requires the resource with a set Sticky flag) has been completed is the resource released again and it can be used by other jobs.

With the Sticky flag, all the necessary processes can be run on the same scope or job server using a Sticky Resource.

Here's another example to illustrate this:
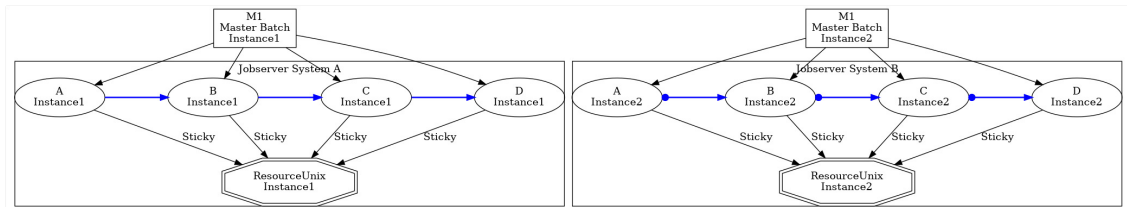
Figure 13.29: Example of load balancing with Sticky Handling

In a system environment there are 2 UNIX workstations with an identical configuration and environment that are used for load balancing. A Master Job can run on one of these two workstations. However, since the intermediate results are to be transferred as data files between the individual jobs in the Master Job, the Master Job must run on the workstation after it has been selected.

This can be achieved by means of the Sticky flag. Job servers (System A and System B) are installed respectively on these workstations and the environment is mapped as a Synchronizing Resource Definition (Resource Unix).

This Resource Definition is now allocated within the two job servers (twice, once on each job server). Now all the children of the Master Job have to be defined so that they require the Resource Definition with the Sticky flag.

The first job (Job A) in the Master Job now allocates one of the two available resources (on one or the other job server). Since this sticky has been allocated, it is not returned following completion of the first job. All the other jobs in the Master Job are defined as belonging exclusively to the selected job server, and therefore to its workstation, by virtue of this Sticky Resource.

In the diagram, one Master Job (Instance 1) is running entirely on job server System A and a second Master Job (Instance 2) is running entirely on System B.

Sticky Name If the Sticky flag described above is set, the *Sticky Name* field is displayed. If the *Sticky Name* is set, the resource is only made available to those jobs in the same Master Run that want to allocate the resource with the same Sticky Name, but not to the other jobs.

Sticky Parent If the Sticky flag described above is set, the *Sticky Parent* field is displayed. If the *Sticky Parent* is set, the resource is only available to those jobs in the same Master Run that want to allocate the resource with the same Sticky Parent, but not to the other jobs.

Resource States list   The R*Resource States List* is only displayed if the resource has the type "Synchronizing" and a Resource State Profile has been allocated.
A list of required Resource States can be entered here. The job can only be started if the required resource is in one of these states.

Lock mode   The *Lock mode* indicates the access mode that is to be used to access the required resource. More information about *Lock mode* can be found in Chapter 12.4.2.3.

Resource State Mapping   This field is only displayed if the resource has the type "Synchronizing" and a Resource State Profile has been allocated.
The *Resource State Mapping* defines how and whether the state of the resource is to be changed after the job has finished. More information about Resource State Mappings can be found in Chapter 8.

Expire   Gives the value of the Expiration interval. More information about Expire can be found in Chapter 9.3.6.

Expire Unit   This is the unit used for the Expiration interval. More information about Expire Units can be found in Chapter 9.3.6.

Ignore Expiration On Rerun   If this flag is set, the Expire condition is not checked again for a rerun. This means that jobs that have already run once, i.e. where the Expiration condition was met, can be restarted after a period of time even if the Expire condition is no longer met.

13.5.9 Defined Resources tab

The "Defined Resources" are instantiated in that moment when the job starts. These resources are only visible for jobs in the same workflow.
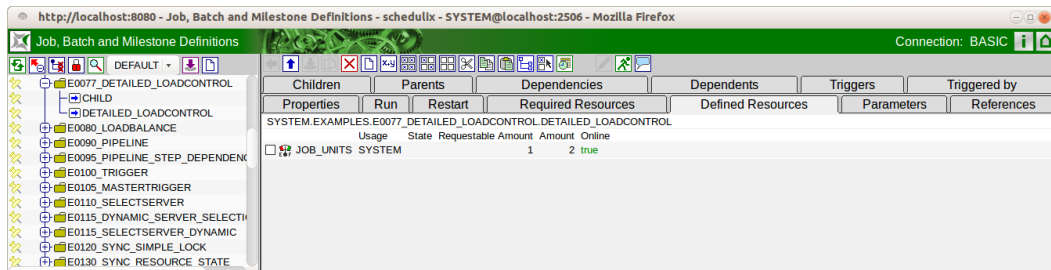
Figure 13.30: Batches and Jobs; Defined Resources

The fields in the list shown above have the following meanings:

Usage   The *Usage* field specifies the "Resource" type. More details about resource types can be found in the Named Resource.

State   The *State* field is only visible if the resource usage has been set to "Synchronizing" and a Resource State Profile has been assigned. It designates the current status of the resource in this scope or job server. The state can be set or changed with this value. The drop-down list contains all the valid Resource States of the Resource State Profile, which can be selected here.
This may be necessary when manually fixing an error to restore a resource to a state that allows further processing by follow-on jobs.
If you manually change this value, the value in the *State* field in the "Resources" tab is not automatically refreshed. Click the *Refresh* button if this is required and necessary.

Requestable Amount   The maximum amount of resources that can be requested by a job.

Amount   The *Amount* is the number of resource instances allocated to or requested by the current job. If the value of the amount exceeds the currently available number that can be maintained in the *Amount* field in the "Resource Details" tab and there is no alternative scope available with a sufficient number, the job cannot be executed.

Online   *Online* can be used to switch a resource in this scope or job server online or offline. An offline resource is temporarily unavailable.

13.5.10 Parameters tab

Parameters that can be used for communication and data transfer between jobs can be defined in the "Parameters" tab. Parameters that originate from the environment or from the scope or folder environment of the job server and job environment can also be defined here for use within the current job.
The "Parameters" tab looks like this:
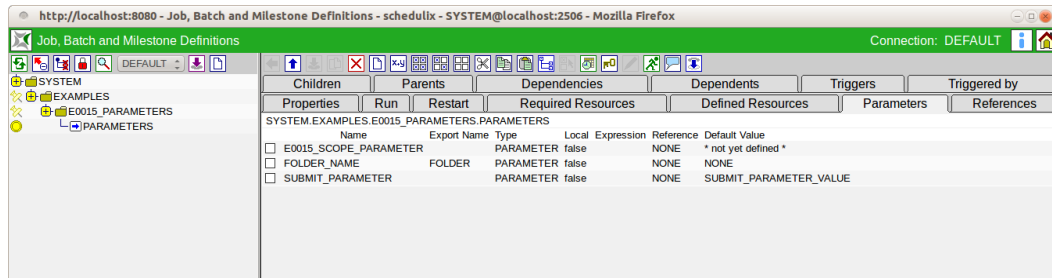
Figure 13.31: Batches and Jobs; Parameters tab

All the parameters that have been currently defined in this Scheduling Entity are shown in the list above. A new parameter can be created by clicking the *New* button. After entering the data and returning to the list, it will also appear here. Selected parameters can also be deleted with the *Delete* button. You can switch to the "Parameter Details" tab by clicking the parameter name.

The columns in the "Parameters" tab are explained in the next section.

13.5.10.1 Parameter Details tab

The "Parameter Details" tab is displayed if a parameter has been selected in the "Parameters" tab or if a new parameter is to be created with the *New* button.
All the details for a parameter can be entered and changed in this tab.
The "Parameter Details" tab looks like this:
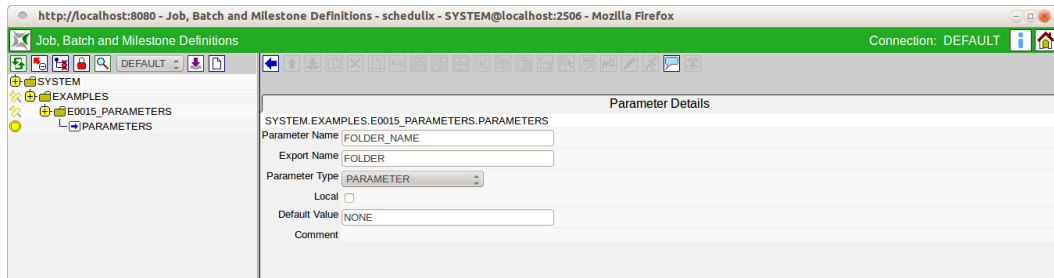
Figure 13.32: Batches and Jobs; Parameter details

The fields in the "Parameter Details" tab have the following meanings:

Parameter Name   This is the name of the parameter. All the parameter values are exchanged between the individual jobs using this name. The parameter value is set by the first job and can be used in the second job under the same name.

Export Name   If a name valid for an OS environment variable is entered here, the value of the parameter will be available to the job under this name at runtime.
If the field is empty or NONE, this parameter is not made available as an environment variable for the job.

Parameter Type   This is the parameter type and indicates how the parameter is used.
Options for "Parameter Type":

1. PARAMETER

   Parameters of the type "Parameter" can be specified at the time of submit if a default value has been specified or must be specified if no default has been specified. However, this only applies to the parameters of the Master Batch. All other parameters behave as if they were of the type "Import".

2. IMPORT

   The value must only be resolvable at its execution time. The usage of this variable is explicitly documented.

3. RESULT

This is the result of a job that can be exported to a follow-on job.

4. CONSTANT

This parameter is a constant that can no longer be changed. The value of the constant is entered in the field *Default Value*. This value is mandatory for parameters of the type "Constant".

5. EXPRESSION

An expression parameter is a special form of parameter for parallel processing with dynamic Child Jobs. To obtain an aggregate across all children in this case, an aggregate function can be selected in the "Expression" field that appears and the child parameter that is to be aggregated can be entered in the "Expression Parameter Name" field.

If the expression parameter is used at runtime, all instances of the Child Jobs are searched for the value of this parameter and the corresponding aggregate function is executed.

Here is an example illustrating this:
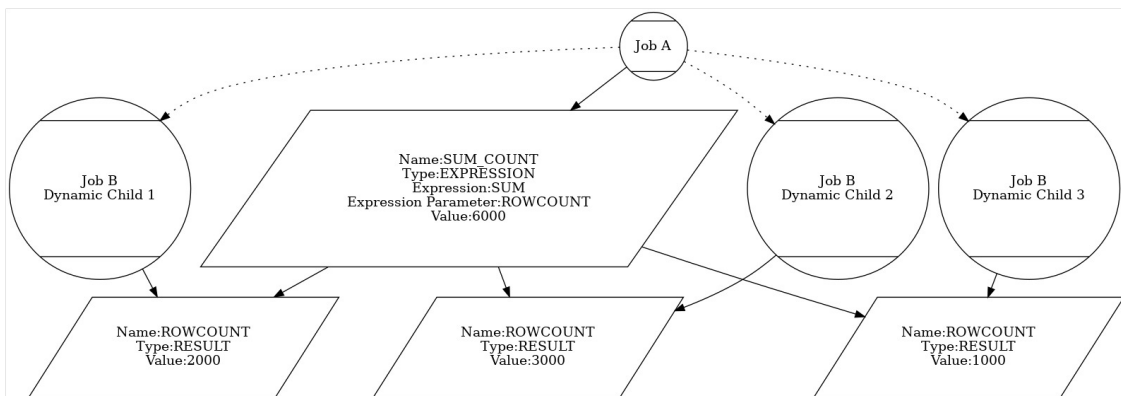
Figure 13.33: Example of an Expression

Job A has a dynamic Child Job B, which is to be simultaneously submitted at runtime. The Child Jobs B write all the x-lines in a table in parallel. Now A would like to know the total number of lines that all the Child Jobs have written.

To do this, Job B must have a defined parameter (Type Result) LINECOUNT that contains the number of processed lines at the end of the job run.

In Job A, a parameter TOTALCOUNT is now created with the type "Expression", the expression option "SUM" and the parameter name "LINECOUNT".

Job A starts now and submits 3 Child Jobs B (B1, B2, B3). B1 writes 2000 lines, B2 writes 3000 lines, and B3 writes 1000 lines. After the children have been completed, Job A (or a follow-on job) queries the TOTALCOUNT parameter and receives the following result: 6000 lines, which corresponds to the sum total of all the children.

6. REFERENCE

   The type "Reference" allows a parameter to be allocated as a reference to another parameter. This parameter is performed with a cross-search of all the jobs on the same level as the current job. This makes it possible to get parameters from jobs that are not a direct parent of the current job. The parameter that is to be referenced must be inserted in the *Reference* field using *Paste (Copy from the "Parameter" tab of another Job Definition)*.

7. CHILDREFERENCE

   The type "Childreference" allows a parameter to be allocated as a reference to another parameter. This referenced parameter is determined by running a search in all the children of the current job. This enables parameters from the job's own children to be identified.

8. RESOURCEREFERENCE

   The type "Resourcereference" allows a parameter to be allocated as a reference to a Resource parameter. The parameter that is to be referenced must be inserted in the *Reference* field using *Paste (Copy from the "Parameter" tab of a Named Resource)*.

Expression This field is only visible if the value "Expression" has been selected in the field *Parameter type*.
The type of aggregate function can be selected here.
The following rule applies for all aggregate functions: If the parameter or the parameter value is not available in some child objects or it is of the wrong type (string instead of number), these values are ignored and only the available and correct values are aggregated.
The following aggregates are available:

1. SUM

   The sum of all the values for the referenced parameter for all the child elements

2. AVG

   The average of all the values for the referenced parameter for all the child elements which contain valid and existing values.

3. MIN

   The minimum of all the values for the referenced parameter

4. MAX

   The maximum of all the values for the referenced parameter

5. COUNT

   The number of child instances that have defined this parameter

6. NONE

   An aggregate function has not been selected yet. None is not a valid value.

Expression Parameter Name   This field is only visible if the value "Expression" has been selected in the field *Parameter type*.
This is the name of the child parameter for which the aggregate is to be calculated. More information about aggregates can be found in the *Expression* field.

Local   If a check mark is set here, this parameter is only visible to the job itself.

Default Value   The *Default Value* specifies the default value. This value is returned in the parameter resolution if no explicitly set value is found during the search. This field is optional. A value must be entered here in the case of a constant.

Variables are evaluated recursively. This means that if a variable contains a string such as $NAME or $NAME, the variable NAME is resolved and the value is inserted in its place. This means it has now also become possible - albeit indirectly - to access system variables for other jobs.
The $ character is represented by \$ and \\ is used for a backslash.

Comment   A more detailed explanation about the object can be entered in the *Comment* field.

Reference   This field is only visible if the value "Reference", "Childreference" or "Resourcereference" has been selected in the field *Parameter type*.
A parameter can be entered here using *Copy and Paste* which is referenced via the selected name.

13.5.10.2 Predefined default parameters for the runtime system

Predefined default parameters in the runtime system can be used by all Submitted Entities in the field *Run Program* as a transfer parameter or in the run program itself. The predefined default parameters contain information about the Submitted Entity and its environment.
The following default parameters are provided by the runtime system:

Job environment parameters   These are all the parameters that relate to the Submitted Entity itself. Job environment parameter names:

1. JOBNAME

   This is the name of the job that was entered in the *Name* field.

2. JOBID

   This is the ID of the Submitted Entity. This makes it possible to unambiguously identify the current instance of the entity.

3. KEY

   The *Key* is the password for the respective job. It is used to log onto the server.

4. MASTERID

   The *MASTERID* is the ID of the Master Job that is executing the current job.

5. PID

   The *PID* is the process ID of the Submitted Entity at operating system level. This can be used for the Kill program, for example.

6. LOGFILE

   The LOGFILE parameter specifies the log file defined in the *Logfile* field in the Batches and Jobs dialog window.

7. ERRORLOG

   The ERRORLOG parameter specifies the error log file defined in the *Error Logfile field* in the Batches and Jobs dialog window.

8. SDMSHOST

   The SDMSHOST parameter specifies the current host of the SDMS server.

9. SDMSPORT

   The SDMSHOST parameter specifies the current port of the SDMS server on the host.

10. JOBTAG

The JOBTAG specifies the Child Tag for the Submitted Entity. This makes it possible to unambiguously identify a dynamically created Child Entity.

11. SUBMITTIME

The SUBMITTIME states the time the Master Job is submitted.

12. STARTTIME

The STARTTIME states the time at which the Submitted Entity actually starts on the job server (if it is a job).

13. SEID

The SEID is the ID of the Scheduling Entity.

14. WORKDIR

WORKDIR is the path to the working directory.

15. JOBSTATE

The JOBSTATE is the Exit State of the job, batch or milestone.

16. MERGEDSTATE

The MERGEDSTATE is the resultant Exit State of a job and its children.

17. EXPRUNTIME

The EXPRUNTIME is the Expected Runtime as stored in the Job Definition.

18. EXPFINALTIME

The EXPFINALTIME is the Expected Final Time as stored in the Job Definition.

19. PARENTID

The PARENTID is the ID of the parent job, milestone or batch.

20. STATE

The STATE is the current state of the job.

21. ISRESTARTABLE

ISRESTARTABLE states whether the job is restartable or not.

22. SYNCTIME

The SYNCTIME is the time at which the job switched to the SYNCHRONIZE_WAIT status.

23. RESOURCETIME

   The RESOURCETIME is the time at which the job switched to the RESOURCE_WAIT status.

24. RUNABLETIME

   RUNNABLETIME is the time at which the job switched to the RUNNABLE status.

25. FINISHTIME

   FINISHTIME is the end time for the process.

26. SYSDATE

   The variable SYSDATE is the current date and time.

27. LAST_WARNING

   This variable contains the text from the last audit entry that resulted in the warning count being raised.

28. RERUNSEQ

   This variable states how often the job has already been restarted.

29. SCOPENAME

   This variable delivers the full path name to the job server that is executing or has executed the job.

Trigger parameters   For use in triggers, there are a number of predefined parameters that can be used in Trigger Job scripts. These parameters are only available if the Submitted Entity was started as part of a trigger. If the entity was submitted normally, these parameters are not filled.
Trigger environment names:

1. TRIGGERNAME

   This is the name of the trigger.

2. TRIGGERTYPE

   This enables a job to perform different actions based on the trigger type.

3. TRIGGERBASE

   The TRIGGERBASE parameter contains the name of the "Triggering Object", i.e. the job that sends the trigger upwards via an Exit State Translation, for example.

4. TRIGGERORIGIN

The TRIGGERORIGIN parameter specifies the name of the entity that contains the trigger. This is the job that is shown the trigger in its "Triggers" tab.

5. TRIGGERREASON

The TRIGGERREASON parameter specifies the name of the object that activates the trigger.

6. TRIGGERBASEID, TRIGGERORIGINID, TRIGGERREASONID

The ID of the Submitted Entity is saved in this parameter for each of the three objects named above (TRIGGERBASE, TRIGGERREASON, TRIGGERORIGIN).

7. TRIGGERBASEJOBID, TRIGGERORIGINJOBID, TRIGGERREASONJOBID

The ID of the Job Definition is saved in this parameter for each of the three objects named above.

8. TRIGGEROLDSTATE

This parameter specifies the Job State prior to the activating moment that is relevant to the trigger.

9. TRIGGERNEWSTATE

This parameter specifies the Job State after the activating moment that is relevant to the trigger.

10. TRIGGERSEQNO

TRIGGERSEQNO is the number of times the trigger was activated.

13.5.11 Approval Settings tab

In the "Approval Settings" tab, you can define whether and what type of approvals are required for certain operator actions. This can be used to implement a 4-eyes principle for certain operator actions.
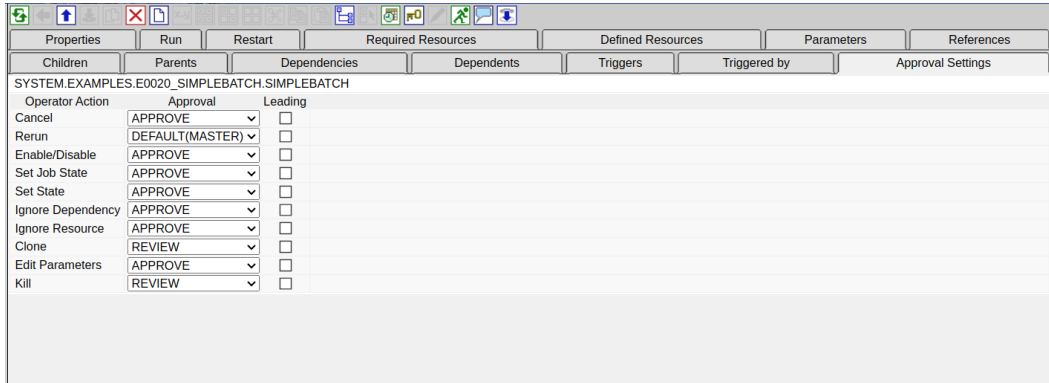
Figure 13.34: Approval Settings job tab

The fields in the "Approval Settings" tab have the following meanings:

Operator Action   The following operator interventions can require an approval or review:

- Cancel

- Rerun

- Enable/Disable

- Set Job State – Always requires ADMIN privileges

- Set State

- Ignore Dependency

- Ignore Resource

- Clone

- Edit Parameters

- Kill

Most operations are self-explanatory. The difference between "Set Job State" and "Set State" is important. With "Set State", the Exit State is set to a value such as SUCCESS or FAILURE.

The operation "Set Job State" is used to set the status of the job. The system normally does this automatically when processing a job. In special cases however, for example as a repair function after a software error has occurred, it may be necessary to set the status of a job manually.

As such an intervention is always critical, only an administrator is permitted to carry it out.

Approval   The following options are available as Approval settings:

- NO

- REVIEW

- APPROVE

- DEFAULT(MASTER)

- PARENT

The "NO" setting means that no approval or review is required for the operation. However, this setting may be overruled by a direct or indirect parent.

The "REVIEW" setting requests a review for the operation. This can still be tightened up by a parent (approval), but not relaxed. A review is more of an organisational measure that retrospectively confirms that the performed operation as such was intended, or perhaps did not lead to the desired result. As this is all recorded in the audit trail, this documentation can help to avoid errors in the future.

The "APPROVE" setting demands an approval for the operation. This cannot be relaxed.

The "DEFAULT(MASTER)" setting is the default setting and means that the job behaves like the master. However, this can still be overruled and tightened up by a parent.

The "PARENT" setting causes the child to behave like its parent.

If nothing is set in the entire workflow, neither an approval nor a review is required. Typically, the desired setting is made at master level and this is inherited by all batches and jobs. Especially critical jobs or sub-workflows can still be configured individually.

Leading   The Leading flag ensures that the selected setting is inherited downwards. This means that no job or batch can evade the setting of a parent. However, it is possible to tighten up the setting. Approval has the highest priority, followed by Review. Logically, the default behaviour, neither Approval nor Review, has the lowest priority.

13.5.12 References tab

All references (parameters of the types REFERENCE and CHILDREFERENCE) from other job or batch objects to parameters for the current object are shown in the "References" tab.
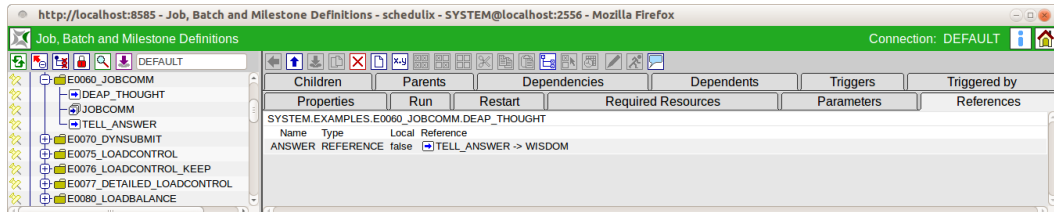
The "References" tab looks like this:

Figure 13.35: Batches and Jobs; References tab

13.5.13 Triggers tab

The "Triggers" tab describes all the triggers that have been defined for this Scheduling Entity. A trigger is used to start another executable object if a configurable event has been reached in the currently selected job (or one of its children, see below). For example, messages can be sent when a job has been completed. Follow-up processing or error handling routines can also be started automatically.

The "Trigger" tab looks like this:
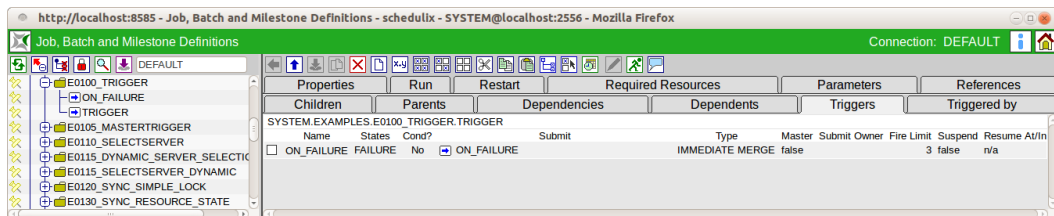
Figure 13.36: Batches and Jobs; Trigger tab

The "Triggers" dialog window shows a list of all the triggers assigned for the selected task. The job to be started by the trigger can be selected and added to the list by copy and paste. The name for the trigger is taken from the name of the job when it is copied. The trigger details can then be viewed and modified by clicking the trigger's name. The trigger is deleted by clicking the *Drop* button. The columns in the list have the following meanings:

Name This is the name of the trigger. When you add a new job to the list of triggers, the trigger name is taken over from the job name. The "Trigger Details" tab

is opened by clicking the trigger name. All the data (including the trigger name) can be viewed and modified here.

States   This is a comma-separated list of all the states that are to be taken into account for the trigger.

Cond?   The *Cond?* field indicates whether a condition has been specified.

Submit   This is the name of the job or batch that is to be submitted with the trigger. Important: This field can also be selected with a mouse click. Doing so opens the edit mask for the respective job. If you have added or modified any data and confirmed the confirmation query, all the modified data is lost.

Param?   The *Param?* field indicates whether a trigger parameter has been specified.
The other columns are explained in the next section.

13.5.13.1 Trigger Details tab

The "Trigger Details" tab is opened by selecting a trigger in the "Triggers" tab. All the detailed information regarding the trigger can be modified or entered here.
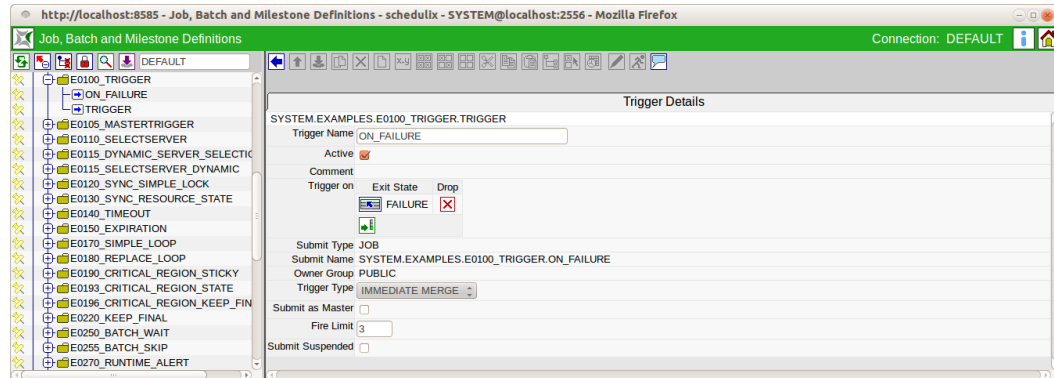

Figure 13.37: Batches and Jobs; "Trigger Details" tab

The tab contains the following fields:

Trigger Name   This is the name of the trigger. When you create a new trigger (i.e. when you add a job to the trigger list), the job name is taken over for it automatically.
The name can then be changed as required in the "Trigger Details" tab.

148

Active   States whether the trigger is active. If this switch is not set, the trigger will not be activated when the trigger event occurs. An inactive trigger is indicated in the trigger list by a black hand (Suspend symbol).

Trigger On list   A list of Exit States for the selected process (not the job that is to be triggered) for which the trigger is to be activated can be defined here. If no list is specified, the job's Exit State is irrelevant and the trigger is activated whenever the event that is to be achieved occurs.
Example:
A trigger is to be activated when a job finishes with an error (EXIT_STATE: FAILURE). In this case the "Trigger On" list must have selected the state FAILURE. Other states (SUCCESS, etc.) are not listed because they are to be ignored when the trigger is activated.
If a trigger is always to be activated when the process ends (regardless of the state, which is perhaps only differentiated in the job that initiates the trigger), the "Trigger On" list can be left empty or filled with any states that the job can take on.

Condition   The condition that has to be fulfilled is entered in the *Condition* field.

Submit Type   This is the type of the object (batch or job) that is started when the trigger is activated.

Submit Name   This is the path and name of the object that is started when the trigger is activated. The full name of the object with all parent folder hierarchies

Parameters   The *Parameters* table makes it possible to specify which parameters are handed over when submitting the batch or job. The parameter name and an expression for determining the value that are to be handed over are defined in the columns *(*Name) and *Expression*. Further details can be found in the documentation for the command language.

Owner Group   The owner of the interval is defined in the *Owner Group* field.

Trigger Type   This is the type of event that is to activate the trigger. The following events are used:

1. Immediate Merge

    The job itself or one of its children has taken on a state in the "Trigger On" list. If no value is entered, the trigger is activated with each state transition.

    Example:

    A text message is to be sent if the job or one of its children contains an error.
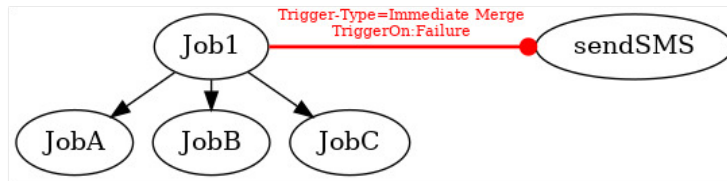
Figure 13.38: Example of an Immediate Merge trigger

2. Immediate Local

The job itself has taken on a state in the "Trigger On" list. If no value is entered, the trigger is activated when the job finishes. The state of the children is ignored.
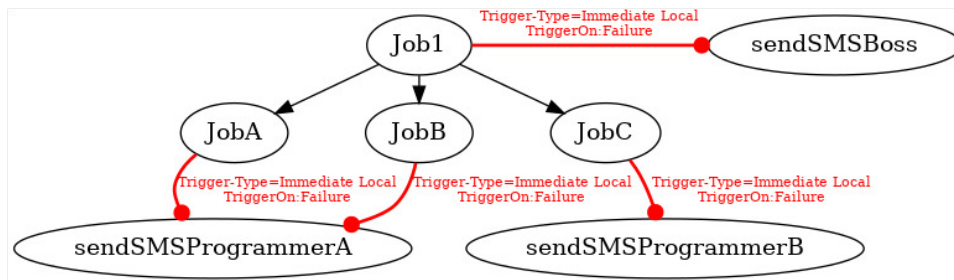
Example:

Figure 13.39: Example of an Immediate Local trigger

A text message is to be sent to the project manager if the job contains an error. If one of the job's children contains an error, a second trigger is implemented in the child which sends a text message to the responsible programmer.

3. Before Final

The trigger condition is evaluated shortly before the Scheduling Entity would reach a Final State. Here, the "Trigger On" list only needs to contain Final States (or none, in which case the trigger is activated for every Final State instance) as otherwise the trigger will not be activated. If the Scheduling Entity that is to be triggered is not submitted as a master, the Scheduling Entity can only attain a Final State after the Scheduling Entity started by the trigger has also attained a Final State.

In the example above, Job C will be called again if the job fails. Since this is a Before Final trigger, Job 1 cannot attain a final state for as long as JobC has been terminated without a FAILURE.
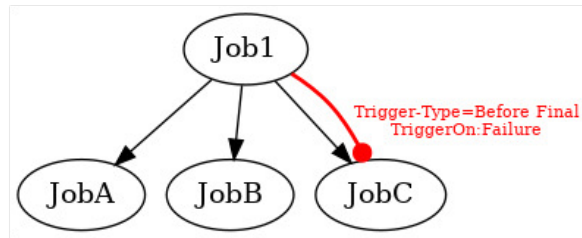
Figure 13.40: Example f)" Before Final trigger

4. After Final

The trigger is activated shortly after the job has reached a Final State. The Here, the "Trigger On" list should only comprise Final States (or none, in which case the trigger is activated for every Final State instance) as otherwise the trigger will not be activated.
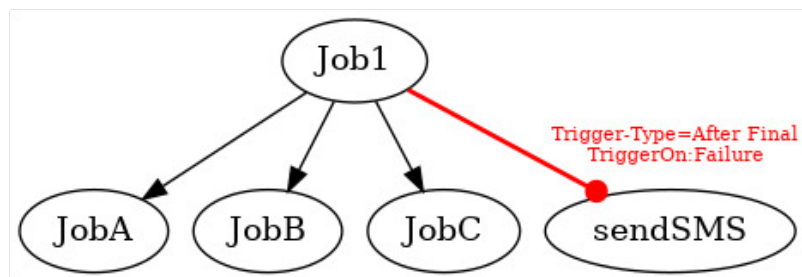

Figure 13.41: Example of an After Final trigger

The legend for the graphic can be found in Chapter 1.8.

5. Finish Child

The trigger is activated when a direct or indirect child of the triggering job has attained a state in the "Trigger On" list. This allows a single point of monitoring to be implemented.

In the example above, a text message with an error message would be sent if one of the Child Jobs takes on a Failure State. If one's own job (Job 1) is successful, the success message is sent by text.

6. Until Finished

The Until Finished trigger periodically checks the condition. If the Condition is "True", the trigger continues to be activated until the "Finished" state has been attained. The condition is then evaluated for the last time and, if it is set to "True", the trigger is activated.
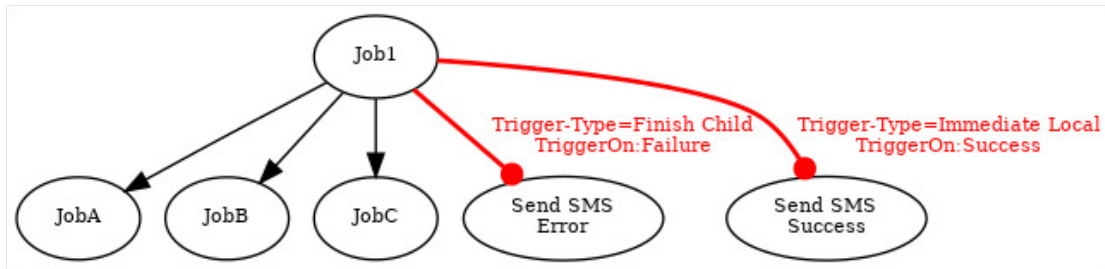
Figure 13.42: Example of a Finish Child trigger

7. Until Final

   The Until Final trigger periodically checks the condition. If the Condition is "True", the trigger continues to be activated until the "Final" state has been attained. The condition is then evaluated for the last time and, if it is set to "True", the trigger is activated.

Submit as Master   If the "Submit as Master" flag has been set, the job that is started by the trigger is submitted as its own Master Job and does not have any influence on the current Master Job run of the triggering job.

If the flag is not set, the triggered job is started as a child of the job to be triggered. This also means that the job in which the trigger is activated is not completed (cannot attain a Final State) as long as the triggered job is still running. The only exception is the "After Final" trigger type, where the trigger is only activated after the triggered job has attained a Final State. If this is the case, the triggered job is started as a child of the parent of the triggering job. If the triggered job is the same Scheduling Entity as that of the triggering job, the triggered job takes the place of the triggering job.

Fire Limit   This defines the maximum number of trigger activations within a workflow. When the maximum number of trigger activations is reached, no more triggers are activated and no more new jobs are started.

Submit Suspended   Defines whether the triggered Scheduling Entity is to be started as being "suspended".

Resume   This field is also only displayed if the option *Submit Suspended* has been enabled.
Here you can choose whether a workflow should be resumed automatically.

The following possibilities are available:

- NO: Deselects this function

- AT: Selects an automatic resume action at a fixed time. The *Resume Time* input field is displayed.

- IN: Selects an automatic resume action after a time period has expired. The *Resume In* and *Unit* input fields are displayed.

Resume time   This field is only displayed if "AT" was selected for *Resume*.
The required resume time is entered here in the format "YYYY-MM-DDTHH:MI:SS".
This format is based on the ISO standard 8601 and also permits incomplete entries.
If you enter 'T16:00', the job will be resumed at 16:00 hours (starting from the time the trigger was activated).

Resume In   This field is only displayed if "IN" was selected for *Resume*.
Here you can specify how many time units (see *Unit*) the system is to wait for until the resume action is triggered.

Unit   This field is only displayed if "IN" was selected for *Resume*.
This field is used to define whether the entry in the *Resume In* field is in minutes (MIN), hours (HOUR) or days (DAY).

13.5.14 Triggered by tab

The "Triggered by" tab lists all the triggers that submit this Scheduling Entity. It
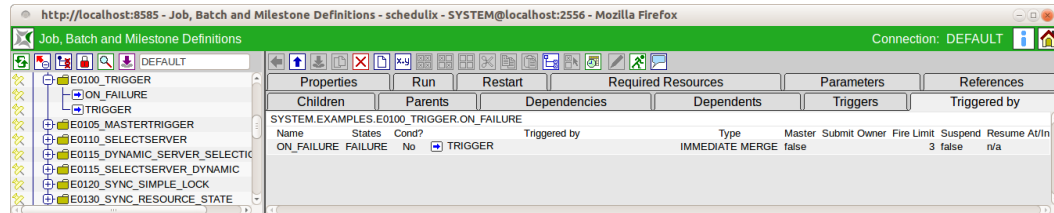looks like this:

Figure 13.43: Jobs and batches; Triggered By tab

The columns in the list have the following meanings:

Name   This is the name of the trigger.

States   This is a comma-separated list of all the states that are to be taken into
account for the trigger.

Cond?   The *Cond?* field indicates whether a condition has been specified.

Triggered by   This is the name of the job, batch or milestone that defines the trig-
ger.
Important: This field can also be selected with a mouse click. Doing so opens the
edit mask for the respective job. If you have added or modified any data and con-
firmed the confirmation query, all the modified data is lost.
The meanings of the other columns can be found in the chapter "Trigger Details
tab".

13.6 Job Hierarchy navigation

The Job Hierarchy navigation window can be viewed for each batch and job in the "Batches and Jobs" dialog window. The navigation screen looks like this:
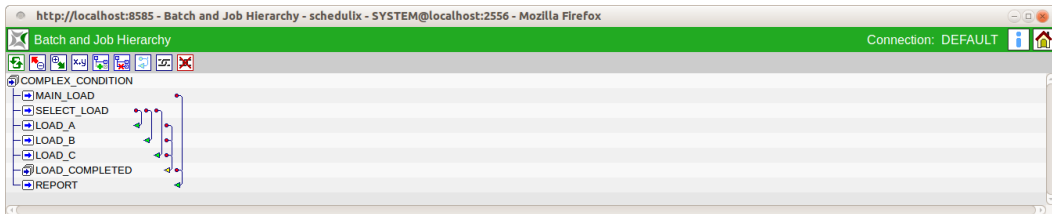

Figure 13.44: Batches and Jobs; Hierarchy view

All the children of the current entity are displayed in the navigation window. If the children also have a child hierarchy, this can be expanded as well until the entire hierarchy of the object is visible.

The displayed standard buttons are supplemented by the following buttons:

Add Children

Children can be inserted directly in the Job Hierarchy navigation window by clicking the *Add Children* button. To do this, click the button followed by the job or batch that is to acquire the children. A mask opens where you can selected one or more children. Having selected the children, close the mask and then click the parent again. The children are added to it automatically.

Remove Child

Children can be deleted by clicking the *Remove Child* button. To do this, click the button and then the parent. The children are deleted automatically after you have clicked "OK".

Show Dependencies

The *Show Dependencies* button has two functions. By default, the dependencies are not displayed in the dialog window and the *Show Dependencies* button appears in the button bar.

The mutual dependencies between the Scheduling Entities can be graphically displayed by clicking the *Show Dependencies* button. Each dependency is indicated by an arrow. The required Scheduling Entity is represented by the start of the arrow (the round part of it) and the dependent Scheduling Entity is the arrowhead.

Example:
In the example above, the job END is dependent upon the job START.
If a job is dependent upon multiple objects, an arrow is displayed with several
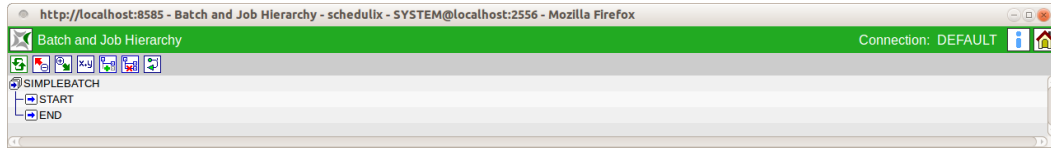
Figure 13.45: Hierarchy view with dependencies

shafts (the round parts of the arrow) representing all the required Scheduling Entities. The arrowhead remains the dependent Scheduling Entity.
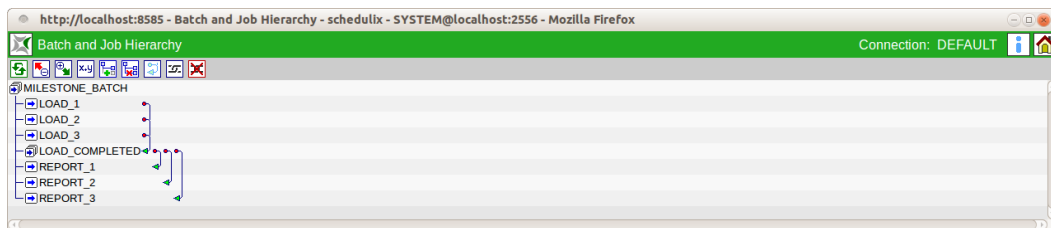Example:

Figure 13.46: Hierarchy view with multiple dependencies

In this example, the batch LOAD_COMPLETED is dependent upon the jobs LOAD_1, LOAD_2 and LOAD_3.

Hide Dependencies

A *Hide Dependencies* function is the second state of this button. It appears if the *Show Dependencies* button has been clicked, thereby activating the display of dependencies The dependencies view can be disabled again by clicking the *Hide Dependencies* button.

Chain

With the *Chain* button you can switch to the Chain mode. Here you can add the job dependencies directly in the Job Hierarchy navigation screen. To do this, first click the *Chain* button followed by the Dependent Job and the Required Job. This creates a dependency relationship between the two jobs.
You can exit the Chain mode by clicking the *Cancel* button.

Unchain

The *Unchain* button is used to delete dependency relationships. To do this, first click the *Unchain* button and then the dependency that you want to delete.
Cyclic dependencies can be easily created with the *Chain/Unchain* buttons. These are identified and reported by the front end. (Figure 13.47)
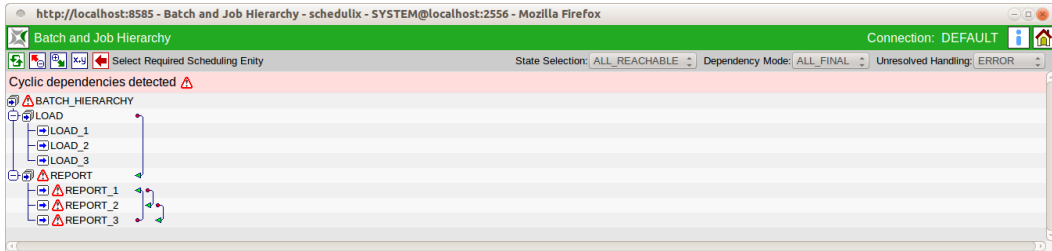
156

Figure 13.47: Cyclic dependencies

13.7 Convention for batches, jobs and folders

To improve working with the system and the overview, we recommend adhering to the following convention:

A folder with the same name is created for each batch which contains the batch and its direct children. If a child has children itself (e.g. a batch), a subfolder is created for this child according to this convention which contains the child and its direct children. The only deviation from this convention is if a job or batch is being used as a child in multiple batches.

The BICsuite!Web GUI supports this convention if the *Enable Batch In Folder Convention Support* checkbox is set in the BICsuite!Web user settings as follows:

1. When creating a folder, a batch with the same name can be created in the folder using the *Create Batch in Folder* checkbox.

2. When creating a batch, a folder with the same name can be created using the *Create Batch Folder* checkbox, which then contains the batch.

3. When creating objects in a folder that follows this convention (i.e. containing a batch or job with the same name), this can be automatically added as a child of this batch or job using the *Add as Child* checkbox.

4. If the name of a folder is changed, the checkbox *Rename in Content* can be used to change the name of the batch or job with the same name in the folder as well.

5. If the name of a batch or job that is located in a folder with the same name is changed, the name of the parent folder can also be changed using the *Rename Parent Folder checkbox.*

6. If the name of an object is changed, a copy of the object with a new name can be created using the *Clone* button. The relevant checkboxes for this convention are therefore displayed in this case as well.

7. When copying/cutting and then pasting in the "Content" tabs of folders, depending on whether the source and target of the operation correspond to the convention a list is displayed with the operations that should be performed on the parent/child relationships in order to adhere to the convention.

   These operations can then be specifically selected before performing the paste operation.
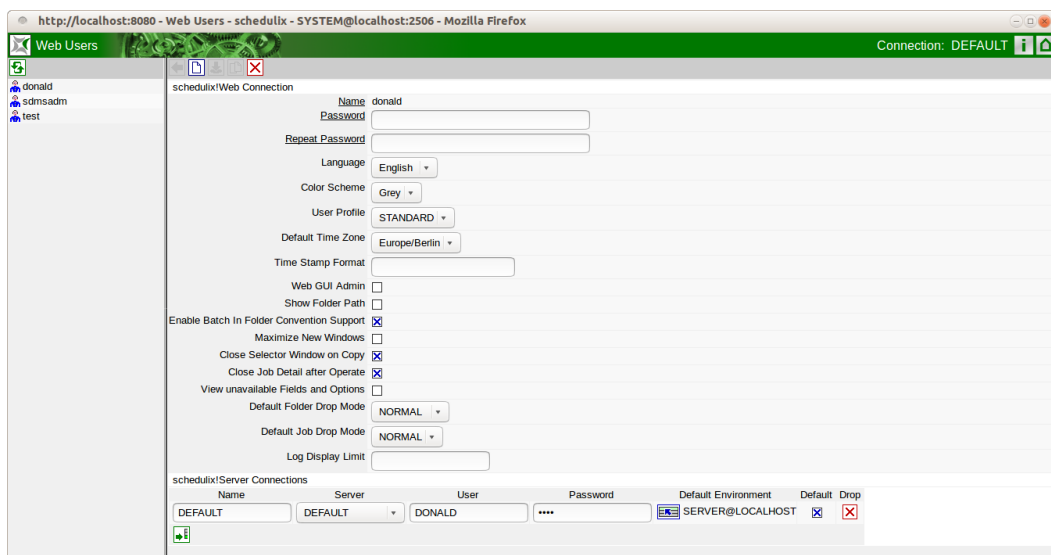
14 schedulix!Web Users

14.1 View

Figure 14.1: User management for the Web front end

14.2 Concept

14.2.1 In short

The user management module is used to define schedulix interface users. All users who want to work with the schedulix interface should have their own user ID. The user must log in with this user ID and a personal password in the Login window.

14.2.2 Detailed description

In addition to the display options and security mechanisms, the connection to the server can also be maintained in this dialog window. The schedulix Server uses its own user management functionality which is independent of the user management module for the interface system. For this reason, it is possible for all interface users

to utilise just one server user, or the interface users can be mapped 1:1 on the server as well. Always notify your system administrator when making changes to your user account.

14.3 Navigator

All the user IDs are displayed in the navigation window.

14.4 Editor

The editor is used to maintain all the data for a selected user.

14.4.1 schedulix!Web Connection

The fields under Web-GUI-Connect have the following meanings:

Name   The user's login name.
The login name is entered as the user name in the Login dialog.

Password   The *Password* authenticates each user who logs into the schedulix System and has to be entered in the Login dialog. The password is hidden on the screen and must be identical to the *Repeat password*.
Since the password is not displayed in the dialog, it is not evident whether a password has already been entered or not. It only has to be entered once, however.

Repeat Password   The *Repeat Password* must be the same as the password you entered in the Password field.
It has to be entered twice because a typing error when entering it just once can cause problems. Since it is unlikely that the user will make the same typing error twice, it is assumed that if the password is twice entered correctly, then the intended term has been used.

Colour Scheme   The colour scheme for the user interface is selected here.

User Profile   The user's profile is entered here.

Default Time Zone   This parameter defines the default time zone when creating new time schedules. All time stamps continue to be displayed in this time zone.

Time Stamp Format   A format other than the default format (%Y.%m.%d %H:%M:%S example 2014.06.18 14:06) for displaying the time stamps can be set here. The format corresponds to the strftime C standard function.

Web-GUI-Admin   This check mark must be set if this user is an administrator for the web interface.

Show Folder Path   If this flag is set, all the folders and Scheduling Entities with their full path names are displayed when opening new windows.

Maximize New Windows   New dialog windows are always maximized when opening them if this check mark has been set. If it is not set, the window always opens in the normal size.

Close Selector Window on Copy   If this check mark is set, windows that are opened for selecting objects are closed automatically after clicking the *Copy* button.

View unavailable Fields and Options   If this check mark is set, the unavailable fields and options are displayed.

Default Folder Drop Mode   This is the mode that is used for deleting a folder in the Batches and Jobs dialog window. The following options are available:

1. NORMAL

   In this mode it is only possible to delete a folder if it is empty, i.e. it does not have any subfolders and no jobs, batches or milestones.

   This is the "default" value for the system.

2. CASCADE

   In this mode, the folder is deleted in a cascade. All the subfolders and all the jobs, batches and milestones in them are deleted.

   Important: This setting is to be used with extreme caution because it is possible to delete a large number of objects at once without actually intending to do so.

Default Job Drop Mode   This is the mode that is used for deleting a job (or a batch or milestone).
OPTIONS

161

1. NORMAL

   In this mode, it is only possible to delete a job, batch or milestone if no other jobs are dependent upon it. The Scheduling Entity must not have any more child relationships either.

   This is the "default" value for the system.

2. FORCE

   In this mode, the scheduling entity is deleted even though other Scheduling Entities are dependent upon it and it still has some child relationships.

14.4.2 schedulix Server Connections

One or more server connections can be configured here.
On the Main Desktop, the server connection that is to be used for the next window to be opened can be selected in an Options field in the window header.
After editing the server connections, it may be necessary to reload the Main Desktop in the browser to make sure that the server connections are displayed correctly in this field.

Name   The name of the server connection for selecting it in the header of the Main Desktop.

Server   The name of the server as entered by the schedulix!Web system administrator under /Custom/SDMSServers (in Zope).

User   The name of the schedulix Server user account. This does not have to be the same as the Web user's name.

Password   This user ID and password are used to connect the schedulix interface with the schedulix Server.
The password is hidden on the screen.

Default Environment   The Default Environment is entered by default for jobs.

Default   If this check mark is set, this server connection is used as the default setting. It must be set for precisely one row.

Server Default   If this check mark is set, this server connection is used as the default setting only if the server is known. It must be set for each server in precisely one row.
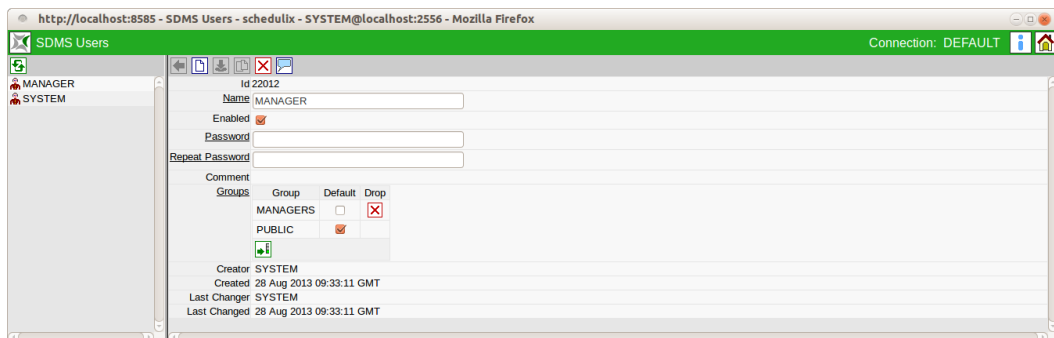
15 schedulix Server Users

15.1 View

Figure 15.1: schedulix Server user management window

15.2 Concept

15.2.1 In short

The schedulix Server user is required for legitimisation toward the Scheduling Server.

15.2.2 Detailed description

Since the Scheduling Server is not just operated using the web interface described in this documentation, but also with the aid of other interfaces such as *sdmsh*, access controls are required on the server. The users and their group memberships can be maintained using the dialog described here.

15.3 Navigator

All the users are displayed in the navigation window.
The Navigator pane is only displayed if the active user is a member of the "AD-MIN" group. All other users will only see the editor that can be used to change their own password and the default group they need to choose.

15.4 Editor

The editor is used to maintain all the data for the selected or new user. SDMS users can only be edited by users who belong to the "ADMIN" group. All the input fields are "read only" for all other users.
The editor contains the following fields:

Name   The user's login name.
The login name is entered as the user name in the Login dialog.

Enabled   If the active user is a member of either the "ADMIN" group, this option is used to define whether the selected user is permitted to log in.

Password   The *Password* authenticates each user who logs into the schedulix System and has to be entered in the Login dialog.
The password is hidden on the screen and must be identical to the *Repeat Password*. Since the password is not displayed in the dialog, it is not evident whether a password has already been entered or not. It only has to be entered once, however.
The maximum length for a password is 64 characters. All characters (i.e. including spaces) are allowed.
The password can always be changed if the active user is a member of the "AD-MIN" group or if the user to be modified is the active user (you can obviously change your own password).

Repeat Password   The *Repeat Password* must be the same as the password you entered in the Password field.
It has to be entered twice because a typing error when entering it just once can cause problems. Since it is unlikely that the user will make the same typing error twice, it is assumed that if the password is twice entered correctly, then the intended term has been used.

Groups   The groups to which the user belongs are listed in the "Groups" table. The table features the following columns:

Group   Groups to which the specified user member belongs.

Default The default group is the one that is joined by a new object in the absence of a defined group.

Drop This button is used to delete the group membership. This action has to be confirmed.
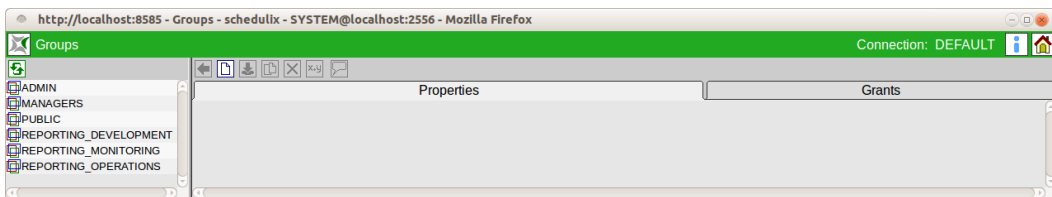
16 Groups

16.1 View

Figure 16.1: Group management

16.2 Concept

16.2.1 Description

The groups are the 'legal entity'. The individual users belong to groups and they have the same privileges held by the groups. A user can belong to several groups. The following groups always exist:

- Public: Every user is always a member of the *Public* group.

- ADMIN: Each user in the *ADMIN* group holds all privileges throughout the system.

16.3 Navigator

All visible groups are shown in a list in the Navigator. Visible in this context means that the user is either a member of the group or he has been granted "view" privileges for it.

16.4 Editor

16.4.1 Properties tab

This tab is used for maintaining the group properties.
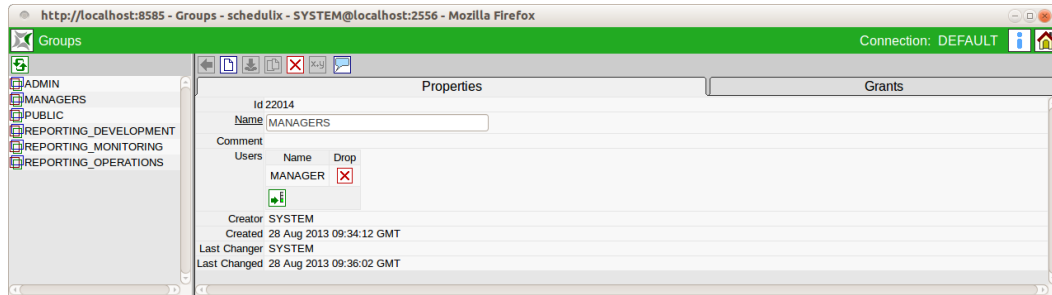
Figure 16.2: Group properties

This window contains the following fields:

ID The Group Id is shown in this field.

Name The name of the group is shown in the *Name* field.

Users The following field is in the Users list.

Name The group user is shown in the *Name* field.

16.4.2 Grants

The "Grants" tab shows all the objects to which group privileges have been assigned. The object type is indicated by the respective icon. The list is always sorted by the object names.

Moving the mouse pointer over the privileges field in a row opens a popup window with an explanation of the privileges string.
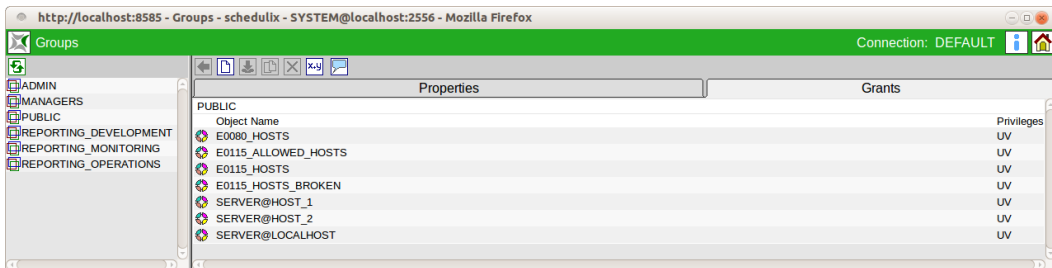
Figure 16.3: Object privileges for a group

17 Time Scheduling

17.1 View
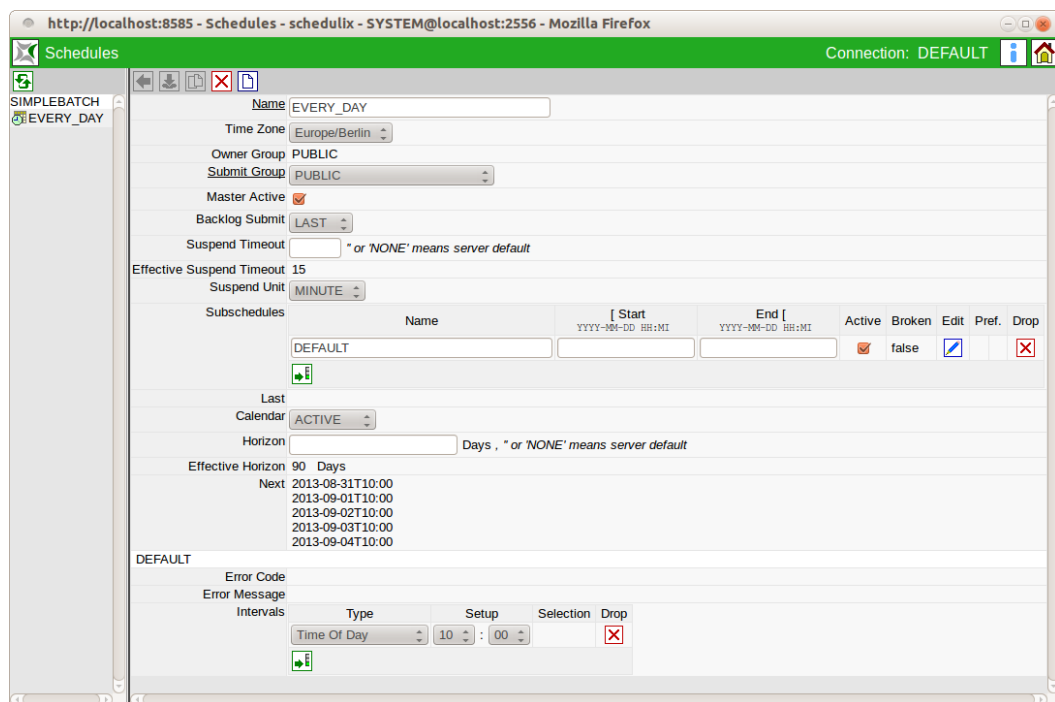
Figure 17.1: Time Scheduling

17.2 Concept

17.2.1 In short

The Time Scheduling module allows batches and jobs to be started at specific times and can be used to easily define complex schedules for jobs and batches.
A number of points in time which determine when a job is to be started is called a schedule.
Schedules always apply to just one Master Job.

17.2.2 Detailed description

In the "Time Scheduling" dialog window, a differentiation is made between Main Schedules, which are shown in the navigation as elements, and Sub Schedules, which can be administered within a Main Schedule.
Main Schedules are mutually independent and no interaction takes place.
Sub Schedules can be defined as being separated from the time interval or as overlapping. This allows different downtimes to be simulated or varying schedules can be set which are therefore always active.
If the schedules are to be changed on demand, this can be entered as an interim schedule. The highest priority has to be assigned to the interim plan to activate it. The original schedule is retained. The original schedule is reactivated by reassigning the highest priority to it.

17.3 Navigator

The Main Schedules for this Scheduling Entity are displayed in the navigation. A Scheduling Entity can have multiple Main Schedules. These Main Schedules are taken into account independently of one another. This means that the start times of all the Main Schedules are factored in.

If multiple Main Schedules have identical start times, the Scheduling Entity is started several times.
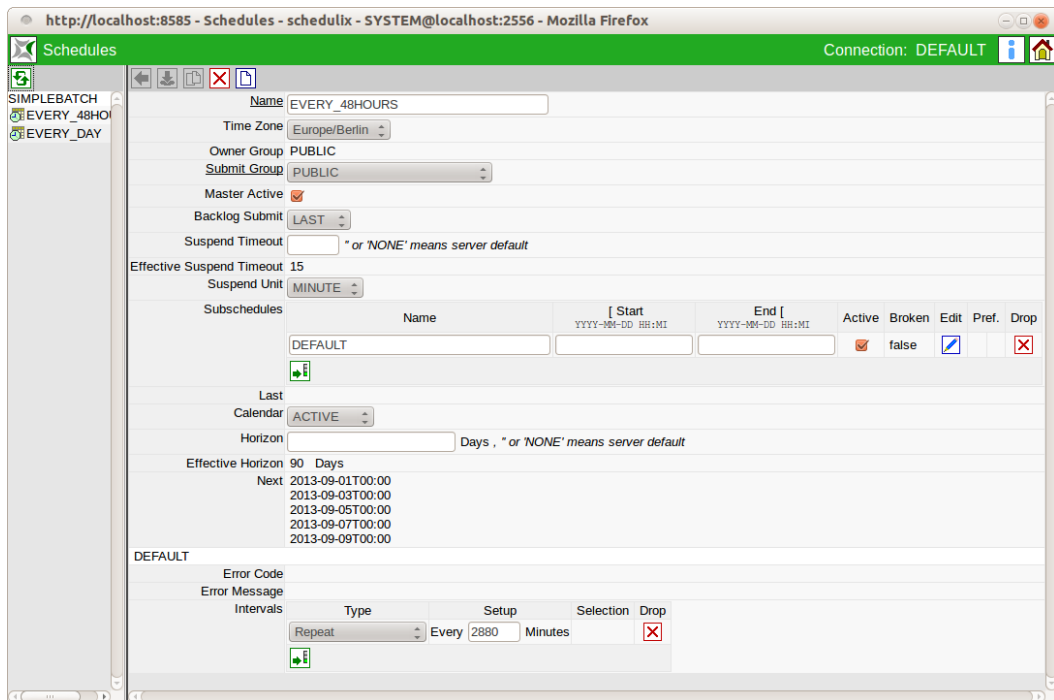

Figure 17.2: Time Scheduling Navigator

17.4 Editor

All the "Detail" elements for the Main Schedule selected in the navigation are displayed in the editor. Here is where Sub Schedules can be created and a list of times maintained for when the selected job or batch is to be started. The "Editor" mask looks like this:
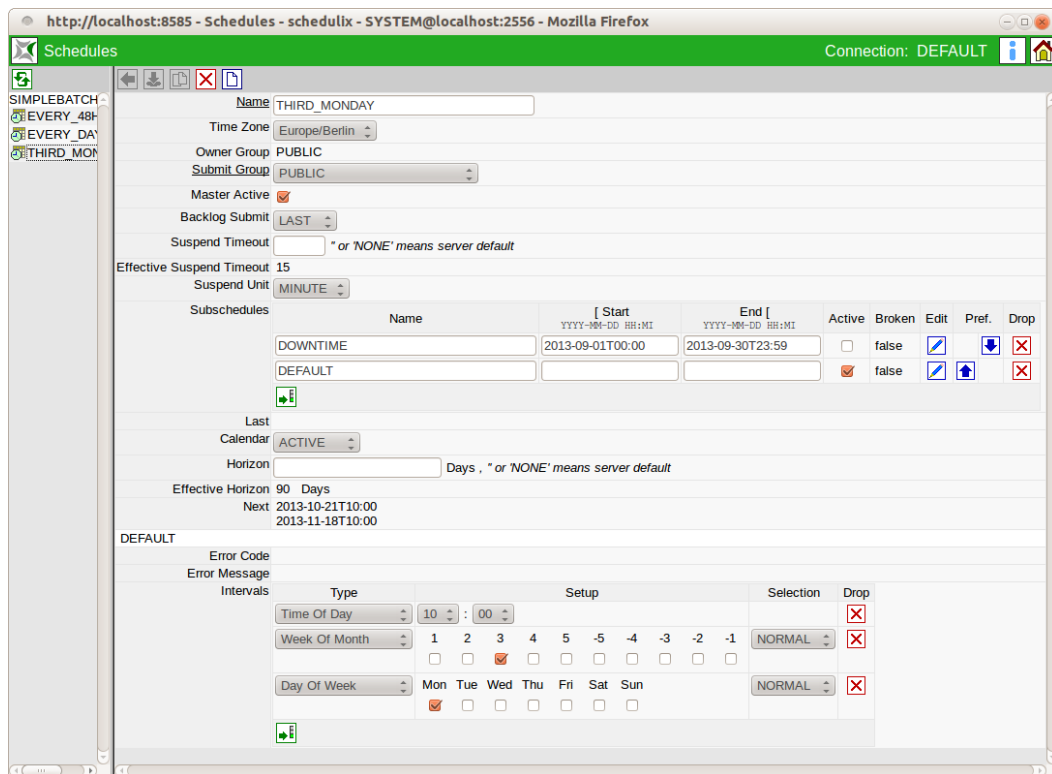
Figure 17.3: Time Scheduling Editor

The fields shown above have the following meanings:

Name   This is the name of the Main Schedule. When you first create a new Main Schedule, it is automatically assigned the name "MASTER", but this can be arbitrarily changed. Each Main Schedule that you create has to have a different name.

Time Zone   The time zone for which the schedule is to be calculated. Which time zones are made available here can be configured by the schedulix!Web administrator.

Owner Group   The owner of the schedule is defined in the field *Owner Group*.

Submit Group   The *Submit Group* determines the owner group of the submitted job.

Master Active   If the flag for *Master Active* is set, the schedule is taken into account by the Time Scheduling System. If the flag is not set, the schedule is ignored.

Backlog Submit   The backlog determines how the Time Scheduling System behaves after a server downtime:

- None: The "missed" jobs are no longer executed.

- Last: Only the last of the "missed" jobs is executed.

- All: All the "missed" jobs are executed.

Suspend Timeout, Suspend Unit   Following a server downtime, a suspended job is submitted if the scheduled submit time is longer than the time entered in the fields *Suspend Timeout* and *Suspend Unit*. If no time is entered, the server-wide default time applies.

Sub Schedules list   All the Sub Schedules of the Main Schedule are shown in the "Sub Schedules" list. The schedules are only valid within the stated timeframe. The Sub Schedules with the highest preference (at the top of the list) are taken into account first. These 'cover' the Sub Schedules further down the list for their timeframe. This means that only one Sub Schedule can be valid at any one time. The fields in the list have the following meanings:

Name   The name of the Sub Schedule is entered in the *Name* field.

Start   The date and time from which this Sub Schedule is valid are entered in the *Start* field. If this field is left empty, the Sub Schedule is considered to be "always valid".

End   The date and time from which this Sub Schedule is invalid are entered in the *End* field. If this field is left empty, the Sub Schedule is considered to be "indefinitely valid".

The following buttons are available in the Sub Schedules list:

Active Jobs are only submitted if this flag is set. Consequently, submits can be prevented by deleting the flag. Downtimes are usually implemented as high-preference Sub Schedules with a non-set Active Flag.

Broken The *Broken* field is used to check whether an error occurred when the job was submitted. If this is the case, TRUE is entered in the *Broken* field for this Sub Schedule.

Edit Clicking the *Edit* button opens the interval description belonging to the Sub Schedule, where you can precisely define when the job is to be started.

Pref. The *Preference* buttons are used to set the preference of the individual Sub Schedules. The rows are shifted up or down by clicking the buttons.

Drop This button is used to delete the object.

Last The last time the job is to be executed by the Scheduling System is shown in the *Last* field. This field is empty if the job has never been started by the Scheduling System.
The date format looks like this:
YYYY:MM:DDTHH:MM
The placeholders correspond to those in the *Start* field.

Calendar This field determines whether the scheduled execution times for the job are to be noted in the calendar. If this is the case, the next two fields are also important.

Horizon The *Horizon* field determines for how far in advance entries are to be made in the calendar. The time period is entered in days.

Effective Horizon The configured default value applies as the horizon if the *Horizon* field is left empty. The *Effective Horizon* field shows the currently valid value.

Next The next scheduled time(s) for the task to be executed by the Scheduling System is/are shown in the *Next* field. If no schedules or times have been defined, this field is left empty. Once a schedule has been created and times have been defined, the closest possible date after this schedule is displayed. If no time is possible because mutually exclusive times have been defined, a value is not displayed here either.
When calendar entries are created, both the next execution time and the following four execution times (providing they lie within the horizon) are displayed.
The date format is identical to the *Start* field.

176

17.4.1 Sub Schedule Details sub-area

This sub-area shows all the "Details" information for the first Sub Schedule or a Sub Schedule you have selected by clicking the *Edit* button. The Sub Schedule Details begin in the mask with a grey line showing the name of the currently selected Sub Schedule followed by the following fields:

Error Code   If an error occurred while the job was being executed in the Time Scheduling, the returned error code is displayed in the *Error Code* field. If no error occurred, this field remains empty.

Error Message   If an error occurred while the job was being executed in the Time Scheduling, the returned error message is displayed in the *Error Message* field. If no error occurred, this field remains empty.

Intervals list   If no interval is entered here (the list is empty) and the Sub Schedule is marked as being active, a job is submitted precisely once at the start time of the Sub Schedule.

A list that is not empty must contain exactly one "driving" interval of the type "Time Of Day", "Repeat" or "Calendar (Driver)". The "driving" interval defines the cycle in which a submit can potentially take place. The other "filtering" intervals determine by virtue of their filtering effect which of the cycles created by the "driving" interval actually lead to a submit.

Type   The *Type* field defines the type of interval. Another mask is displayed in the *Setup* column depending on the interval type.

Setup   A mask for configuring the interval is displayed here depending on the interval type.

Selection In the case of intervals that can be selected (e.g. "Day Of Week"), in this column you can choose either a "NORMAL" or "INVERSE" selection.

The following interval types are used:

- Repeat

  This is a "driving" interval for starting the task at regular intervals according to a schedule. The number of minutes can be entered in the *Setup* field.
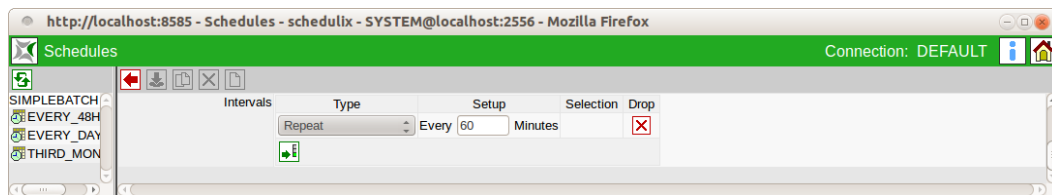
Figure 17.4: Repeat Driver

In the example, the job is started every 60 minutes.

- Time of Day

  This "driving" interval type can be used to start the job at a specified time of day. The time of day is entered in the *Setup* field by selecting the hour (24H) and minutes.
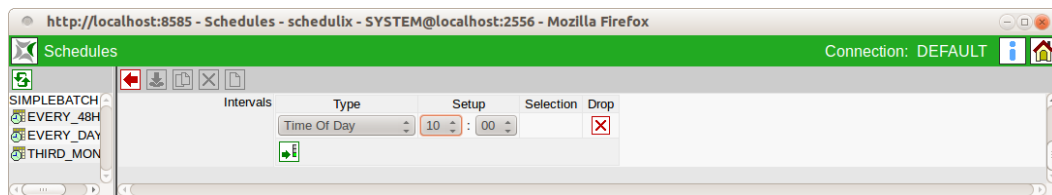
Figure 17.5: Time Of Day Driver

In the example, the job is started every day at 10:00 hours.

Several "Time Of Day" intervals can be created. This allows you to easily define multiple execution times for any one day.

178

- Range of Day

  This "filtering" interval type is only permitted in combination with the "driving" interval types "Repeat" and "Calendar (Driver)". It allows you to set a restriction to one or more time ranges during a day.
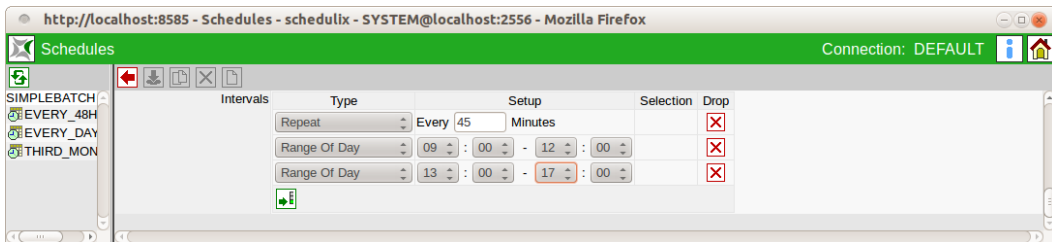
Figure 17.6: Range of Day Filter

  Multiple "Range Of Day" intervals can be created. This allows multiple execution times to be easily defined.

- Day of Week

  The "filtering" interval type "Day of Week" can be used to select the weekdays on which the job is to be executed. This is defined in the *Setup* field by choosing the relevant weekdays.
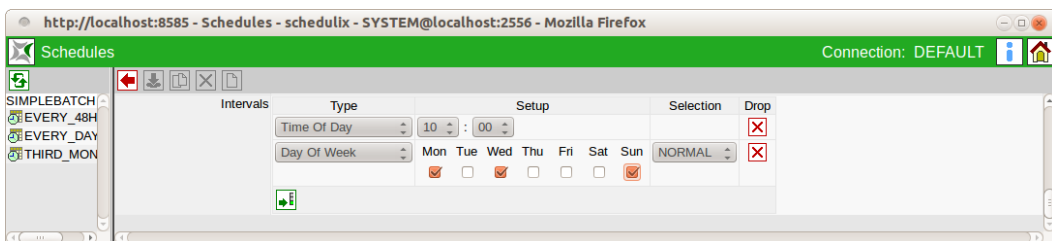
Figure 17.7: Day of Week Filter

  In the example, the job is executed every Monday, Wednesday and Sunday.

- Day of Month

  The "filtering" interval type "Day of Month" can be used to select the days of the month on which the job is to be executed. This is defined in the *Setup* field by choosing the relevant days of the month.
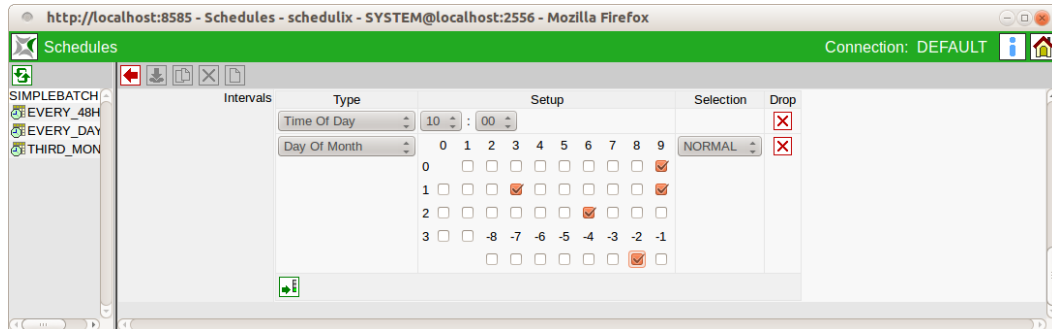
Figure 17.8: Day of Month Filter

  In the example, the job is executed on the 9th, 13th, 19th, 26th and penultimate day of each month.

- Week of Month

  The "filtering" interval type "Week of Month" can be used to select the week of the month in which the job is to be executed. This is defined in the *Setup* field by choosing the relevant weeks in the month.
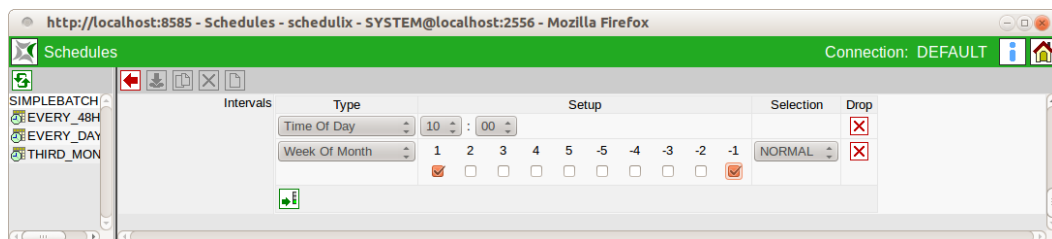

Figure 17.9: Week of Month Filter

  In the example, the job is executed in the first and last weeks of each month. The first week is defined here as being the first 7 days of the month and the last week is the last 7 days of the month.

- ISO Week of Month

  The "filtering" interval type "ISO Week of Month" can be used to select the ISO week of the month in which the job is to be executed. This is defined in the *Setup* field by choosing the relevant   weeks in the month.
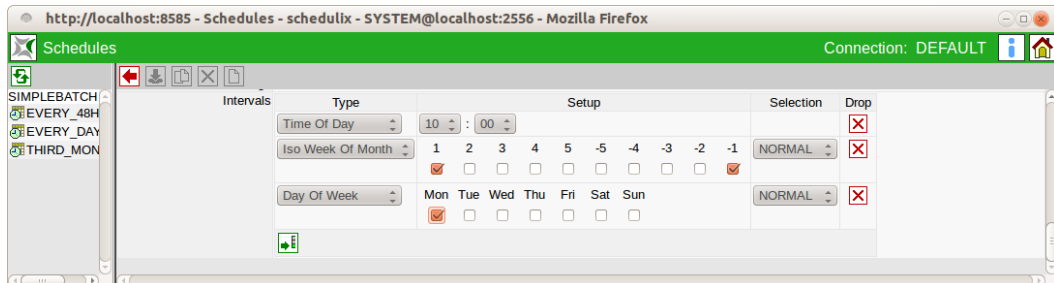
Figure 17.10: ISO Week of Month Filter

In the example, the job is executed on the first Monday of the first ISO week of each month. ISO weeks always begin on Mondays. Weeks are assigned to a months if at least 4 days of the week lie in this month. For example, if the first weekday of a month is a Wednesday, the first ISO week of the month already begins in the preceding month.

- ISO Week of Year

  The "filtering" interval type "ISO Week of Year" can be used to select the ISO calendar week of the year in which the job is to be executed. This is defined in the *Setup* field by choosing the relevant   weeks in the year.
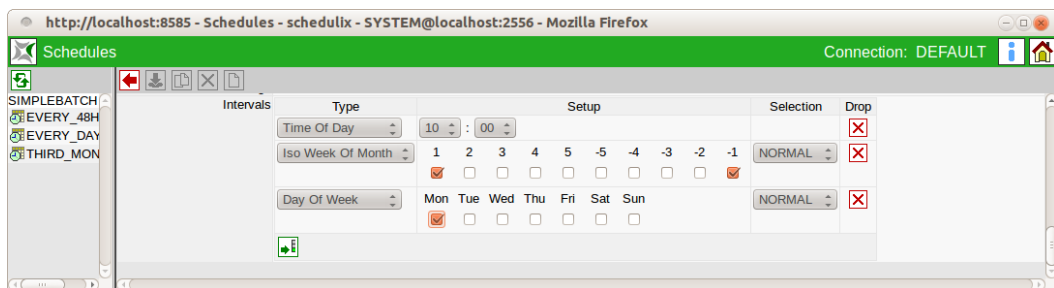
Figure 17.11: Week of Year Filter

In the example, the job is started on the Tuesday of the 31st ISO calendar week. ISO weeks always begin on Mondays. Weeks are assigned to a year if at least 4 days of the week lie in this year. For example, if the first weekday of a year is a Wednesday, the first ISO week of the month already begins in the preceding year.

- Month of Year

  The "filtering" interval type "Month of Year" can be used to select the month of the year in which the job is to be executed. This is defined in the *Setup* field by choosing the relevant   months of the year.
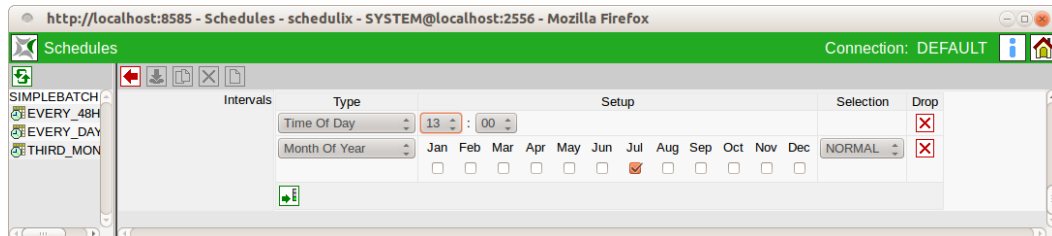
Figure 17.12: Month of Year Filter

  In the example, the job is started every day in July at 13:00 hours.

- Calendar (Driver)

  The "driving" interval type "Calendar (Driver)" can be used to take execution times from a predefined calendar. This option is only available only if the administrator has created such a calendar and registered it in schedulix!Web.

  The calendar can then be selected in the *Setup* field. The option "with select on" determines the unit (DAY, WEEK, MONTH, YEAR) to which the subsequent selection refers.
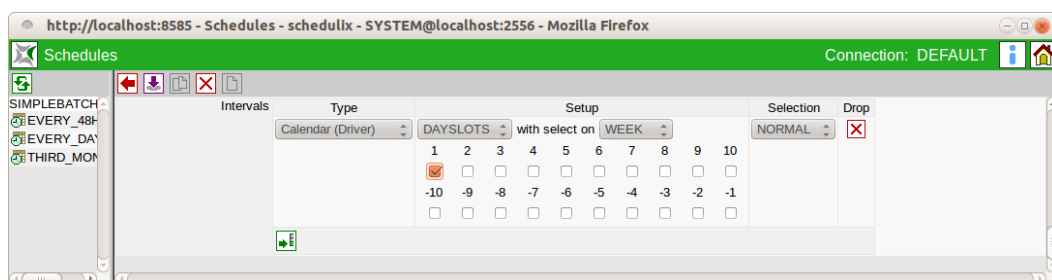
Figure 17.13: Calendar Driver

  In the example above, the first entry in each calendar week is selected from a calendar called "DAYSLOTS".

- Calendar (Filter)

  With the "filtering" interval type "Calendar (Filter)", the execution time points can be restricted by means of a predefined calendar.

  The calendar can then be selected in the *Setup* field. The option "with select on" determines the unit (DAY, WEEK, MONTH, YEAR) to which the subsequent selection refers.
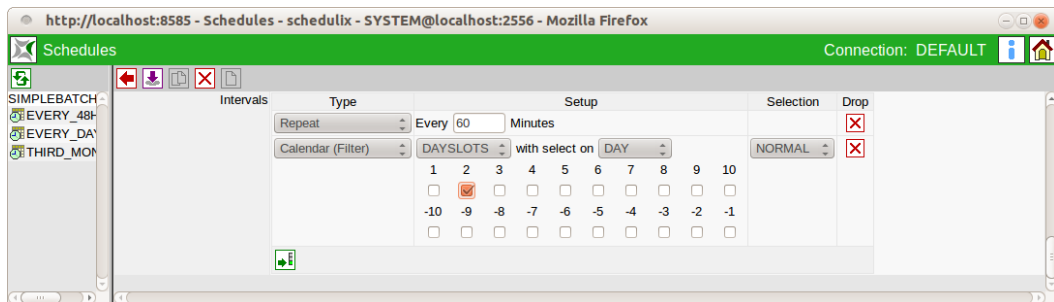
Figure 17.14: Calendar Filter

  In the example above, a submit is performed every 60 minutes if this is in the second block of each day in the calendar "DAYSLOTS".

Setup The *Setup* field defines the interval more precisely. It varies according to the type of interval.

18 Intervals

18.1 Image
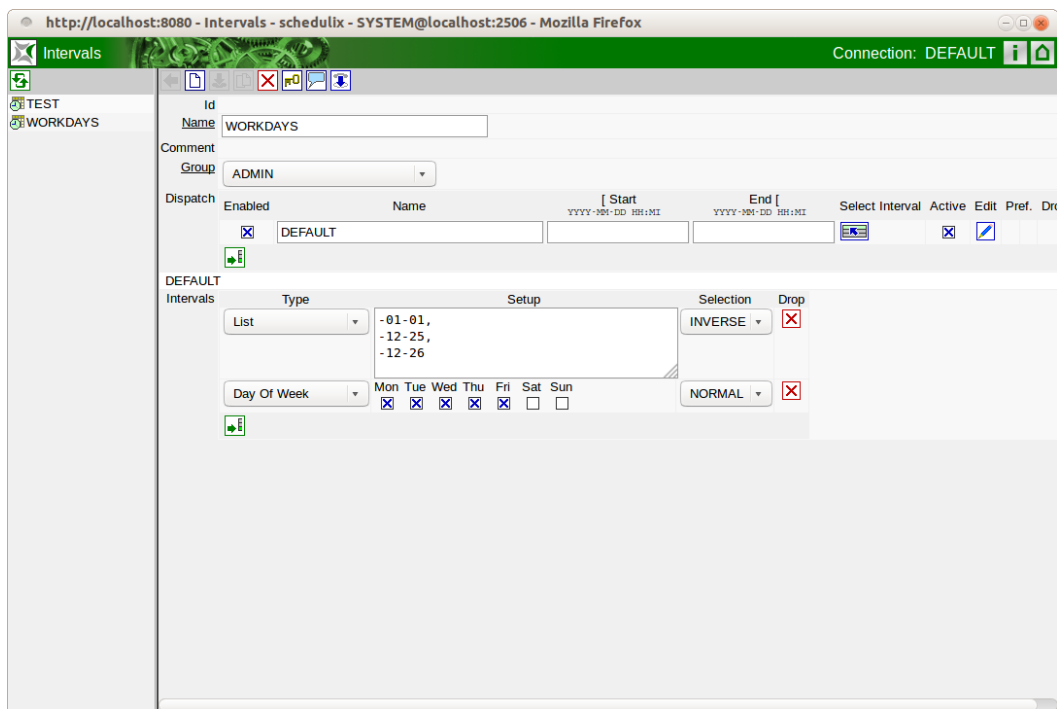
Figure 18.1: Intervals

18.2 Concept

18.2.1 Brief description

Intervals allow the definition of time intervals which can be repeated in cycles. Defined intervals can be used as "calendars" (drivers or filters) in time scheduling for batches and jobs. Whether a child job is to be "disabled" when it is submitted can likewise be controlled by using an interval. Defined intervals can also be used to define other intervals, both as "Calendar" and as "Select" dispatch intervals.

18.2.2 Detailed description

Intervals can be created and edited or deleted in the "Intervals" dialogue window. Intervals comprise one or more dispatchers, each with a filter chain. The order of the dispatchers determines which filter chain is used for a point in time. The functionality of the dispatcher is comparable to the sub schedules in the time scheduling for batches and jobs. For intervals, however, a different defined interval can be used as the dispatcher. Dispatchers can also be temporarily deactivated.

18.3 Navigator

The navigation shows all the intervals visible to the user that were created in the "Intervals" dialogue window. The interval is opened in the editor by clicking it in the navigator.

18.4 Editor

The editor displays the "Interval" selected in the navigation. Dispatchers can be created here, and their filter chains can be edited.
The fields in the editor mask named above have the following meanings:

Name This is the name of the interval.

Owner Group The owner of the interval is defined in the field *Owner Group*.

Dispatcher list All the dispatchers for the interval are shown in the "Dispatchers" list.
The dispatchers are only valid in the entered period and with the set select interval if the point in time to be examined is a valid time for the select interval.
The dispatchers with the highest priority (at the top of the list) are taken into consideration first. These 'conceal' dispatchers for their period and/or select interval

that are further down the list. This means that only one dispatcher can be valid at any one time.

The fields in the list have the following meanings:

Enabled   The *Enabled* field is a flag that determines whether the dispatcher line is to be used. If this flag is not set, the line is treated as if it did not exist.

Name   The name of the dispatcher is entered in the *Name* field.

Start   The date and time from which this dispatcher is valid are entered in the *Start* field. If this field is left empty, the dispatcher is considered to be "always valid".

End   The date and time from which this dispatcher is invalid are entered in the *End* field. If this field is left empty, the dispatcher is considered to be "indefinitely valid".

Select Interval   A "Select" interval can be set in the *Select Interval* selection field. A time is then handled by this dispatcher if it is a valid time for the "Select" interval.

Active   This flag must be set if the dispatcher filter chain is to be used to check whether a time for the edited interval is valid. If this flag is not set, a time that is to be tested which is valid for the "Select" interval of the dispatcher is regarded as being invalid for the edited interval.

The following buttons are available in the list of dispatchers:

Edit   Clicking the *Edit* button opens the interval description for the dispatcher in which the user can precisely define whether a time is valid, respectively the next valid time for this interval.

Pref.   The *Preference* buttons are used to set the preference for the individual dispatchers. The rows are shifted up or down by clicking the buttons.

Drop   This button is used to delete the object.

18.4.1 Sub-area Interval Details

This sub-area shows all the "Details" information for the first dispatcher or for another dispatcher selected by clicking the *Edit* button. The dispatcher details begin in the mask with a grey row containing the name of the currently selected dispatcher, followed by the following fields:

Intervals list   If an interval is not entered here (the list is empty), and if the dispatcher is marked as being active, the edited interval will generate a time precisely once at the start time of the dispatcher, or all times which are valid for the time period and/or "Select" interval will be considered to be valid.

A list that is not empty may only contain one "driving" interval of the type "Time Of Day", "Repeat" or "Calendar (Driver)". The "driving" interval defines the cycle in which a submit can potentially take place.

The other "filtering" intervals determine by means of their filtering effect which cycles generated by the "driving" interval actually result in a submit. In contrast to time scheduling, intervals can also be created without a "driving" interval because intervals themselves can also be used as filters.

Type   The *Type* field defines the type of interval. Another mask is displayed in the *Setup* column depending on the interval type.

Setup   A mask for configuring the interval is displayed here depending on the interval type.

Selection   In the case of intervals which enable a selection (e.g. "Day Of Week"), in this column it is possible to choose whether "NORMAL" or "INVERSE" is to be selected.

The following interval types are used:

- Repeat

    A "driving" interval type which enables a task to be started at controlled regular intervals. The number of minutes can be entered in the *Setup* field.

- Time of Day

    This "driving" interval type can be used to start the job at a specified time of day. The time of day is entered in the *Setup* field by selecting the hour (24 h) and minutes.

    Several "Time Of Day" intervals can be created. This allows multiple execution times to be easily defined for any one day

- Range of Day

    This "filtering" interval type allows for a limitation to one or more time periods in any one day. Multiple "Range Of Day" intervals can be created. This allows multiple validity time periods to be easily defined for any one day.

- Day of Week

    The "filtering" interval type "Day of Week" can be used to select the weekdays on which the interval is to be valid. This is defined in the *Setup* field by selecting the relevant weekdays.

- Day of Month

  The "filtering" interval type "Day of Month" can be used to select the days of the month on which the interval is to be valid. This is defined in the *Setup* field by selecting the relevant days of the month.

- Week of Month

  The "filtering" interval type "Week of Month" can be used to select the weeks of the month on which the interval is to be valid. This is defined in the *Setup* field by selecting the relevant weeks in the month.

- ISO Week of Month

  The "filtering" interval type "ISO Week of Month" can be used to select in which ISO calendar week of the month the interval is to be valid. This is defined in the *Setup* field by selecting the relevant weeks in the month. ISO weeks always begin on Mondays. Weeks are assigned to a months if at least 4 days of the week lie in this month. For example, if the first weekday of a month is a Wednesday, the first ISO week of the month already begins in the preceding month.

- ISO Week of Year

  The "filtering" interval type "ISO Week of Year" can be used to select in which ISO calendar week of the year the interval is to be valid. This is defined in the *Setup* field by selecting the relevant weeks in the year. In the example above, the job is executed on the Tuesday of the 31st ISO calendar week. ISO weeks always begin on Mondays. Weeks are assigned to a year if at least 4 days of the week lie in this year. For example, if the first weekday of a year is a Wednesday, the first ISO week of the month already begins in the preceding year.

- Month of Year

  The "filtering" interval type "Month of Year" can be used to select in which month the interval is to be valid. This is defined in the *Setup* field by selecting the relevant months of the year.

- Calendar (Driver)

  Another existing interval can be used by means of the ″driver″ interval type ″Calendar (Driver)″.

  The calendar can then be selected in the *Setup* field. The option "with select on" determines the unit (DAY, WEEK, MONTH, YEAR) to which the subsequent selection refers.

- Calendar (Filter)

By means of the "filtering" interval type "Calendar (Filter)", the execution time points can be restricted by means of a predefined calendar. The calendar can then be selected in the *Setup* field. The option "with select on" determines the unit (DAY, WEEK, MONTH, YEAR) to which the subsequent selection refers.

- List

  A list of time ranges can be entered that are to be used for filtering purposes with the "filtering" interval type "List". The time periods are separated with ",". All "selections" of intervals described in the syntax documentation are valid. Please note that a selection defined in the command line language " '2019-01-01' - '2019-01-02' " must be written as "2019-01-01 to 2019-01-02" in the input field of the "List" interval type.

Setup The *Setup* field defines the interval more precisely. It varies according to the type of interval.

19 Submit Batches and Jobs

19.1 View
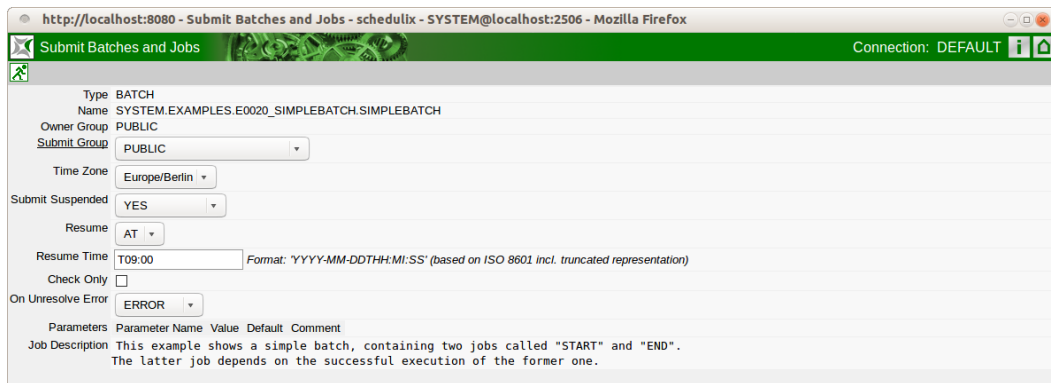
Figure 19.1: Submit Batches and Jobs

19.2 Concept

19.2.1 In short

The "Submit Batches and Jobs" dialog is used for manually starting Master Jobs defined in this dialog. The submit informs the schedulix Server that this Master Job is to be activated.

19.2.2 Detailed description

Manually releasing Master Jobs is one of two methods to start a job. The second method uses the job entry in the Time Scheduling.

Starting a job manually also allows you to set certain parameters and determine whether you really want to start the job or just check if it can actually be started.

You can also define whether the Master Job is to be suspended straight away.

19.3 Navigator

The navigation shows all the available Master Jobs in a folder hierarchy. It corresponds to the navigation pane in the Batches and Jobs dialog window. One difference here, though, is that only objects for which the flag "Submit as Master allowed" has been set are displayed.
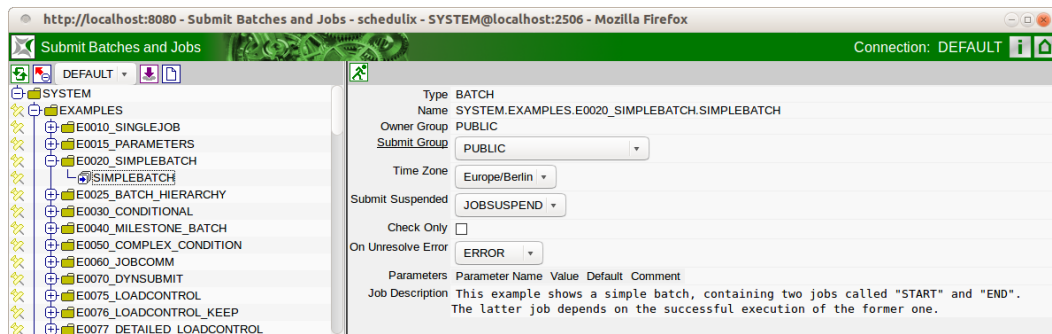
Figure 19.2: Submit navigation

19.4 Editor

The Editor "Frame" can be used to hand over start options and parameters and submit a Master Job.
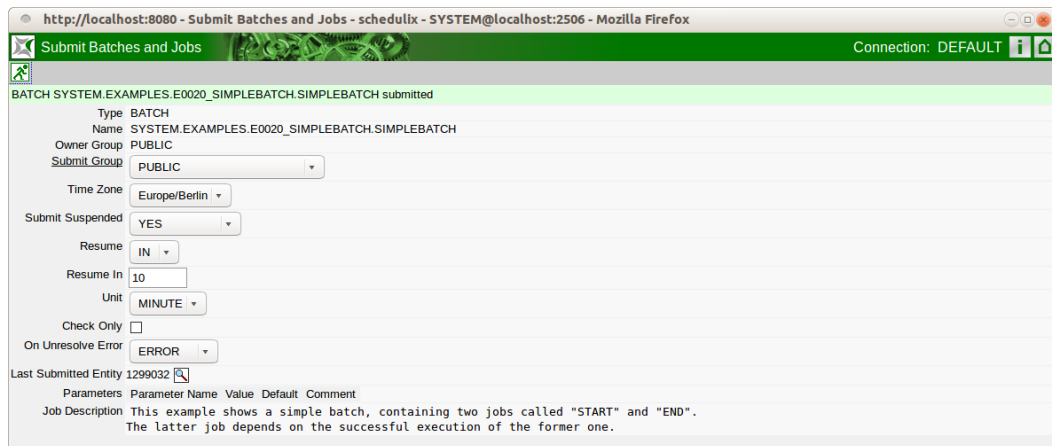
Figure 19.3: Submit editor

The following button is active in this mask:

⚡ Submit

The *Submit* button is used to start the job (the flag "Check Only" is not set) or check the start capability of the Master Job (the flag "Check Only" is set).
The fields shown above have the following meanings:

Type   The type of the Master Job. This can be either a job or a batch.

Name   This is the full name (with its path) of the Master Job.

Owner Group   The owner of the job is defined in the field *Owner Group*.

Submit Group   The *Submit Group* determines the owner group of the submitted job.

Submit Suspended   This field defines whether the job is to take on a Suspend State right after the start or not.
The input fields *Resume*, *Resume Time*, *Resume In* and *Unit* are only displayed if this field has been set to 'YES'.
The following options are available:

- YES

  The child is created as being suspended when it is submitted and it has to be started by being explicitly released.

- NO

  The child is not suspended when it is submitted and can be started immediately.

- JOBSUSPEND

  Whether a delay takes place or not depends on the *Suspend* field for the job to be submitted. This means that the setting defined in the job is applied.

Resume   Here you can choose whether a workflow should be resumed automatically.
The following options are available:

- NO: deselects this functionality and no more input fields are displayed.

- AT: selects an automatic resume action at a fixed time. The *Resume Time* input field is displayed.

- IN: selects an automatic resume action after a time period has expired. The *Resume In* and *Unit* input fields are displayed.

Resume Time   The required resume time is entered here in the format YYYY-MM-DDTHH:MI:SS.
This format is based on the ISO standard 8601 and also permits an incomplete entry. If you enter 'T09:00', the job will be resumed at 09:00 hours (starting from the current set time).

Resume In   Here you can specify how many time units (see *Unit*) the system is to wait for until the resume action is triggered.

Unit   This field is used to define whether the entry in *Resume In* is in minutes (MIN), hours (HOUR) or days (DAY).

On Unresolve Error   When a job is submitted, the *On Unresolve Error* field determines what to do in the event of unresolved dependencies.
The following options are available:

- Error: Dependencies that cannot be resolved are regarded as errors. The missing dependencies cause the submit to be aborted.

- Suspend: With Suspend, the submit takes place as if these dependencies had not been undefined. However, the job is submitted in a suspended state.

- Ignore: With Ignore, the submit takes place as if these dependencies had not been undefined.

Check Only   This flag states whether the job is being submitted or its start capability is being checked.

Parameters list   If the job requires any parameters, the parameter values have to be entered in this list prior to the submit. More information about parameters can be found in Chapter 13.5.10.

Job Description   The description of the job from the Job Definition.

20 Bookmarks

20.1 View
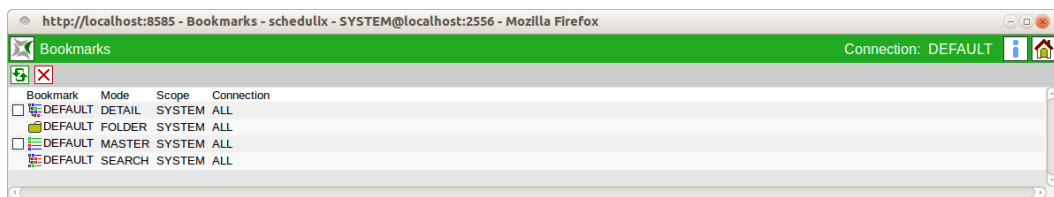
Figure 20.1: Bookmarks

20.2 Concept

20.2.1 In short

Queries from the "Running Master Jobs" and "Search Running Jobs" dialogs can be called in the "Bookmark" dialog. Recurring queries and overview windows can be easily saved and opened again here.

20.2.2 Detailed description

Bookmarks can be made available and modified for the current user or throughout the system. It is also possible to always make certain bookmarks immediately available after logging in to the system. These are then started automatically.
The list of bookmarks contains the following fields:

Bookmark The name of the bookmark is shown here. Selecting the name opens a new window with the request saved as a bookmark.

Mode The *Mode* field shows the type of bookmark. The following types are used:

1. FOLDER

   This is a folder bookmark.

2. MASTER

   This is a bookmark that was saved by the search mask described in Chapter 21. Using this bookmark means that only Master Jobs are visible.

3. SEARCH

   The Search mode designates a bookmark that was saved by the search mask described in Chapter 22. This allows all jobs to be found.

4. DETAIL

   This designates a bookmark that describes the "Detail" view (more information about this can be found in Chapter 21). This CHILD is always related to a MASTER mode bookmark (it is the child of the "Master" view).

   Which bookmark is selected is determined as follows:

   If the Master Job or batch defines a parameter DETAIL_BOOKMARK, then this is used. If this is not the case, the "Detail Bookmark" field for the Master Bookmark is evaluated. If this is not set either, the Detail Bookmark 'DEFAULT' is used.

Scope   The Scope parameter defines whether the bookmark is visible throughout the entire system wide or just to the current user. The following options are available:

1. SYSTEM

   A bookmark with the scope SYSTEM is visible for all users throughout the system. It is displayed in the bookmark list of every user and can be used by all of them. Any changes made to a bookmark from the scope SYSTEM apply system-wide for all users.

   Bookmarks of the type SYSTEM, however, can only be created by users with Web GUI administration privileges (see Web Users). Although a 'normal' user is allowed to modify SYSTEM bookmarks, it is then only saved as a USER bookmark. This means that local changes only overwrite the bookmark for the current user, but not for other users. If this bookmark is deleted, the original SYSTEM bookmark is displayed again.

2. USER

   If the scope has the value USER, then this bookmark was created by the current user and is only available to this user.

Connection   The *Connection* field indicates the validity of the bookmark when using multiple server connections. The following options are available:

1. CURRENT

   The bookmark is only valid for the current connection. This bookmark is no longer visible if the bookmark window is opened for another connection.

2. ALL
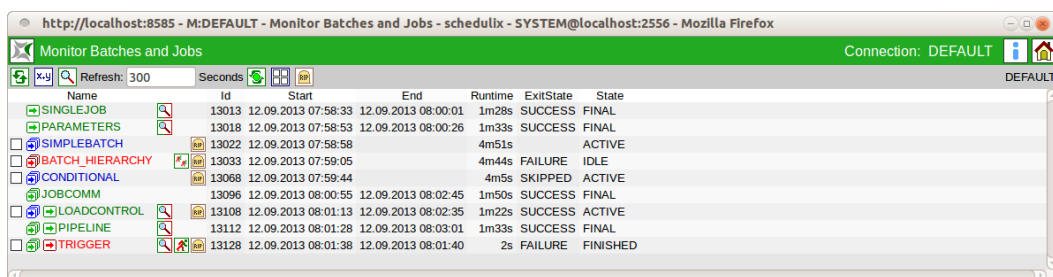
   The bookmark is valid for all server connections.

20.3 Navigation

The Bookmark dialog only features a navigation window. An entry selected here is displayed in a new window.

21 Running Master Jobs

21.1 View

Figure 21.1: Running Master Jobs

21.2 Concept

21.2.1 In short
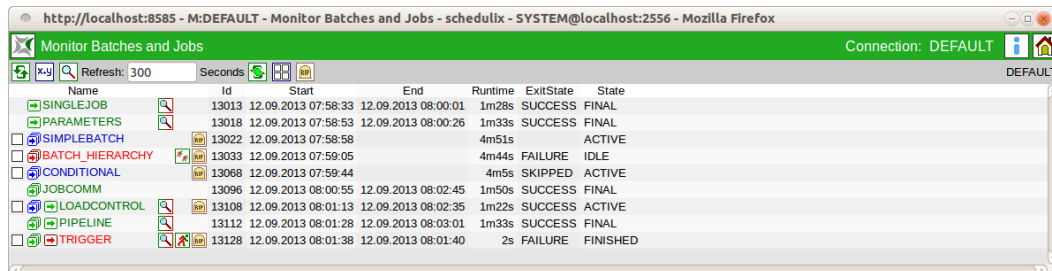
The *Running Master Jobs* dialog is used to display the currently running or (depending on the settings) the historically completed Master Jobs in the system.

21.2.2 Detailed description

The *Running Master Jobs* dialog is where the currently running jobs (Monitoring) are displayed and the active jobs (Operating) are maintained. The display and filter criteria for the Running Master Job dialog can be modified in the bookmark (Default, Master System). This bookmark describes how the entire dialog is displayed.

21.3 Master Navigator

The navigation takes up the whole dialog window. A list of all the currently running 'Master-submittable' jobs and batches is shown here.

Figure 21.2: Running Master Jobs Navigator

Find

Clicking the *Find* button opens the page for entering the search criteria as well as configuring the screen.

Auto-Refresh On/Off

The *Auto-Refresh On* button is also a switch and can be used in two positions. In the position "Auto-Refresh On", a refresh is run automatically in the set time.

The time in seconds between two refreshes is entered in the field to the left of the button. The default value is 300 seconds. The colour of the button indicates the state of the Auto-Refresh function. Green = on, red = off.

The list in the dialog shows either all the Master Jobs that are active in the current system (Default) or, if filter options have been entered in the search mask, the correspondingly filtered results. The following columns are shown in the list:

Name The full path and name of the Master Job or just its name are shown here depending on the position of the Show Job button.

Clicking the name opens the Details mask if the object is a Master Submittable Batch or Job with children or otherwise the Details mask for jobs.

Icons can be displayed in front of the name which describe certain states and situations. The following icons are used:

The displayed object is a job that is either currently running or can run. The icon is blue.

The displayed object is a pending job. The icon is lilac.

- Job icon without a period: The job is in the Dependency Wait state.

- Job icon with a period: The job is in the Synchronize Wait state.

- Job icon with two periods: The job is in the Resource Wait state.

The displayed object is a job that will be submitted in the future by the Time Scheduling.

The displayed object is a job that will be submitted as suspended in the future by the Time Scheduling.

The displayed object is a job that has been cancelled. The state of this job can no longer be changed. The icon is brown.

The displayed object is a job that is in the Final state. The state of this job can no longer be changed. The icon is green.

The displayed object is a job that is in one of the states Finished and Restartable, Unreachable or Error. The icon is red.

The displayed object is a batch that is running at this moment. The icon is blue.

The displayed object is a batch that is waiting at this moment. The icon is lilac.

The displayed object is a batch that will be submitted in the future by the Time Scheduling.

The displayed object is a batch that will be submitted as suspended in the future by the Time Scheduling.

The displayed object is a batch that is in the Final state. The state of this batch can no longer be changed. The icon is green.

The displayed object is a batch that has been cancelled. The state of this batch can no longer be changed. The icon is brown.

The displayed object is a batch that is in the Unreachable state, or which has at least one child that is in one of the states Finished and Restartable, Unreachable or Error. The icon is red.

The displayed object is a pending milestone. The icon is lilac.

The displayed object is a milestone that has been cancelled. The icon is brown.

The displayed object is a milestone that is in the Final state. The icon is green.

The displayed object is a milestone that is in the Unreachable state, or which has at least one child that is in one of the states Finished and Restartable, Unreachable or Error. The icon is red.

These icons tell the user that the object has been suspended. The icon with the clock indicates that an Automatic Resume has been set for the object. The name colours have the same meanings as the matching coloured buttons.

The *Cancel Run* button is displayed if the job is waiting for resources or dependencies or it has still been allocated on a jobserver. The scheduled job run can be cancelled by clicking the *Cancel Run* button. The job then gets the state "Cancelled". Neither a rerun nor a resume are possible. The job doesn't get an Exit State either. Jobs that are waiting for a cancelled job have to ignore this job or be cancelled as well.

This button opens a new window displaying the contents of the log files. Should this not be possible, please contact your system administrator.

 Rerun

The *Rerun* button is displayed if the current job has been finished and it is in a "Restartable" state. This means that it has an Exit State which is not final. Clicking the *Rerun* button calls the Rerun program (if defined, otherwise the Run command). This allows a failed job to be restarted after troubleshooting it.

 Rerun Children

The *Rerun Children* button is displayed if children of the current job are in a Restartable state. This means they have a Restartable Exit State. Clicking the *Rerun Children* button starts the Rerun for all the Restartable children. This allows failed children to be restarted after troubleshooting them.

Id   The *Id* is the unique identifier of the current "Master Job" runtime instance in the system.

Start   The *Start* field shows the time or start of the submit.

End   The *End* field shows the finish time for the job. This field stays empty if the job has not yet finished.

Runtime   This is the current runtime of the job in days (d), hours (h), minutes (m) and seconds (s). If the job is still running, the runtime is displayed until the last refresh of the dialog and is updated after each new refresh.

Exit State   If the job has been finished, its *Exit State* when it finished is displayed. This field stays empty if the job has not yet finished.

State   The *Job State* is the current runtime state of the job. The runtime system of the schedulix Server system assigns and changes these states when the job (i.e. its runtime instance) has passed through the schedulix system. The job can have the following states:

1. Submitted

   The job was submitted manually, by a parent process or by the Time Scheduling and is to be executed. This is the initial state of a job and is usually not visible.

2. Dependency Wait

   The job is waiting for required dependencies that have to be fulfilled.

3. Synchronize Wait

   The job is waiting for the required âĂŹâĂŹSynchronizing ResourcesâĂŹâĂŹ.

4. Resource Wait

   The job is waiting until sufficient System Resources become available.

5. Unreachable

   The job cannot be executed because one or more dependencies have not been fulfilled. This situation can be resolved by ignoring dependencies.

6. Cancelled

   The job has been manually cancelled and is no longer running.

7. Error

   The job cannot be executed due to a definition error. An example of this is a required resource that cannot be provided by any of the jobservers. In this case, the job can be restarted after the error has been remedied.

   What also frequently happens is that a mistake is made when entering the Run program. If the jobserver is consequently unable to start the process, the job is placed in an Error state. The job concerned can be restarted in this case as well.

   If a Master Submit by the Time Scheduling fails, the master is assigned an Error State in the system to identify this error. This makes the error visible for the person responsible. The job or batch is not restartable, however, and has to be submitted manually after being repaired.

8. Runnable

   The job can be started by a jobserver. The jobserver should be checked if the job remains in this state for any length of time.

9. Starting

   The job was handed over to a jobserver to be started.

10. Started

    The job was handed over to a jobserver to be started. The jobserver has acknowledged the job.

11. Running

    The Run command was started by the jobserver.

12. To Kill

    The user has instructed the schedulix Server to run the Kill program associated with the job. The Kill program can be run by clicking the *Cancel Run* button.

13. Killed

    The Kill program associated with the job has been run.

14. Broken Active

    The jobserver has lost the connection to the Run program. Although the process is still running, the result of the job (Exit Code) can no longer be returned to the jobserver and correspondingly the schedulix Server.

15. Broken Finished

    If a job has the state "Broken Active" and it stops, it switches to the state "Broken Finished". This means that the process is no longer running. The Exit Code for the process could no longer be determined by the jobserver. It is necessary to manually check the job results and set the Exit State. The Exit State is set with the *Set State* button.

16. Finished

    The job has been finished. The Exit State is shown in the *Exit State* field.

17. Final

    The job and all its children have finished and have a Final Exit State.

The 21.3 diagram shows the status transitions of the schedulix runtime system.

Variables   Depending on the configuration, the state column can be followed by the variables defined there in the specified order and colour.

Figure 21.3: State chart for batches and jobs

21.3.1 Master Navigator Query mask

The Query mask is opened by clicking the *Settings* button. The query requirements for the current dialog are set in this mask. The filter and display criteria can be edited here.
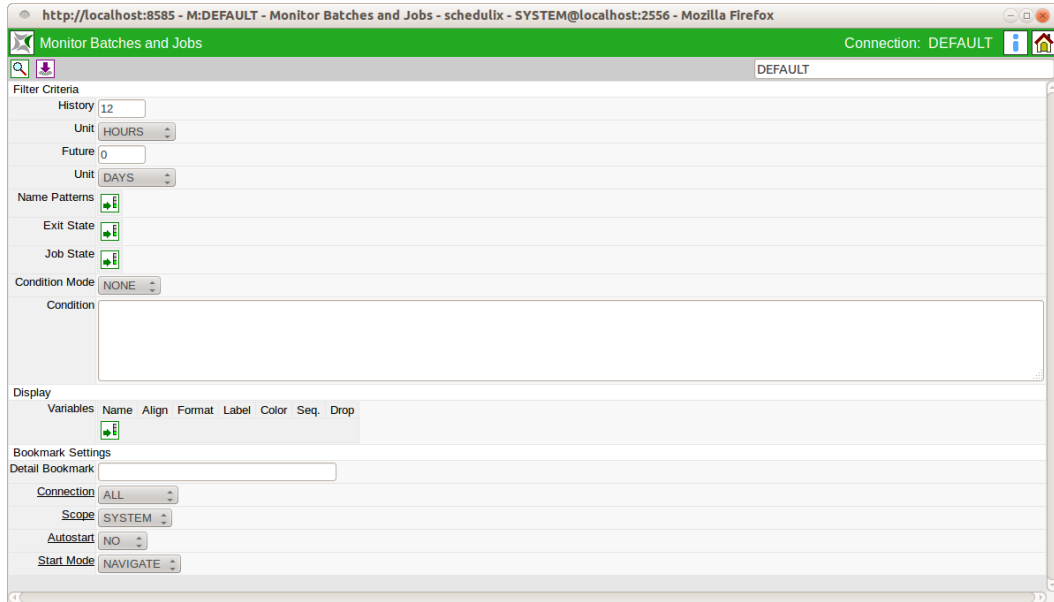
Figure 21.4: Running Master Jobs Query mask

Making changes in the mask and then saving them with the current or a changed bookmark name (entry field in the upper right of the headline) allows them to be used again at a later time. If the changes are only temporary ones, you can simply enter them and jump back to the navigation window. This means that the changes only apply for the current search. The changes will be discarded if the dialog is then closed.

The fields shown above have the following meanings:

History/Unit   The maximum time that may have elapsed since the end of a workflow so that it appears in the display. For example, if all the currently running jobs and all the jobs that have been completed in the last hour are to be displayed, the value 1 has to be entered here and the unit Hours has to be set in the *Unit* field. If all the running jobs and all the jobs that have been completed in the last hour are to be displayed, the value 1 has to be set in the *History* field with Hours in the *Unit* field. The following options are available:

1. Minutes

2. Hours

3. Days

Future/Unit   The setting in the *Future* field determines whether and to what extent batches or jobs that are to be submitted in the future are displayed in the Master List. Regardless of the interval selected in the *Future* field, only the next start can be displayed for schedules without an activated calendar. Only planned starts can be displayed in the calendar horizon of schedules with a calendar. The *Unit* field is handled like the History field.

Name Patterns list   The list of Name Patterns can hold an amount of Name Patterns to be used to filter the returned results. A Name Pattern can be the full name, part of the name or a regular expression.
When searching for multiple entries in the list, all occurrences of these entries are sought. These are linked with "OR".
Example:
You want to search for all occurrences of the name "Pipeline".
The following Name Patterns are possible:

- Full: **pipeline** (the search is not case-sensitive)

- Sub-string: **eline**

- Regular expression: **P.*line**

The list of Name Patterns can be modified using the *Add* and *Remove* buttons.

Exit State list   Exit States can be entered in this list which the sought jobs must have if they are to be shown in the results list.
Example:
Only those jobs with the Exit State "Failure" are to be listed. By adding the Exit State "Failure" to the list of Exit States you can now search for these jobs. If the list does not contain any entries, the Exit States are ignored for the search. The list of Exit States can be modified using the *Add* and *Remove* buttons.

Job State list   Job States in one of the defined states can be searched for in this list. If the list does not contain any entries, the Job States are ignored for the search. The list of Job States can be modified using the *Add* and *Remove* buttons.

Condition mode   The Condition mode defines whether and how the subsequent condition is to be evaluated.
The following options are available:

- None: The condition is ignored.

- And: The search query and condition are linked with ″And″.

- Or: The search query and condition are linked with ″Or″.

- Minus: The search query and negation of the condition are linked with ″And″.

- Only: Only the condition is taken into account.

Condition   The query can be refined using the condition. Please refer to the syntax documentation (List Job) for the exact syntax and the various options that are available to do this.

Variables list   All the job variables and parameters that are to be displayed in the navigation list are defined in the Variables list. This is an easy way to get an overview of the main job parameters. They are displayed at the end of each line in the navigation list.
The following fields have to be populated for the variables:

Name   The name of the variable or parameter has to be entered here.

Align   The *Align* field is for defining how the variables are to be displayed in the list. The following options can be selected:

- LEFT

  The display is left-aligned.

- CENTER

  The display is centred.

- RIGHT

  The display is right-aligned.

Format   The format for the output can be defined here. The following options are available:

- NONE

  No formatting is used.

- NUMBER

  Numerical formatting with thousand separators and decimal points (as necessary) is used.

Label   A heading that is to be shown for the variable in the list header can be defined here. The variable name is used as a heading if a label is not defined.

Colour   The background colour for the variables column can be configured here. A preview of each colour is shown in the field's "drop-down" menu.

Seq.   The list order of the variables can be changed using the *Up* and *Down* buttons. These buttons are only visible if the variables list has more than one entry in it.

Bookmark Settings   The entry fields in the *Bookmark Settings* section are only meaningful when a bookmark is saved. The following fields describe the properties of the bookmark to be saved.

Detail Bookmark   If the name of a detail bookmark is entered in the field *Detail Bookmark*, opening a detail navigation window from a master navigation window opened by this bookmark will use the entered bookmark name for the opened detail navigation window. If this detail bookmark is not yet existing, the detail navigation window will show this bookmark name in square brackets. Those will disappear when the detail bookmark is saved in the opened detail navigation window. The field *Detail Bookmark* is only visible in the *Bookmark Settings* of master navigation windows.

Connection   This field defines the server connections the bookmark can be used with. There are two variants:

1. ALL

   The bookmark can be used with any server connection.

2. CURRENT

   The bookmark can be used with with the current server connection only.

Scope   This field defines the visibility of the bookmark to users. There are two variants:

1. USER

   The bookmark is only visible to the current user. No other users can see this bookmark or, if it is a change made to a system-wide bookmark, only the change is locally visible to this user.

2. SYSTEM

   The bookmark or current change is visible to all users. This means that all users see this new bookmark in their bookmark list. If it is a change made to an existing system-wide bookmark, the change affects all the users.

Autostart   Autostart defines whether a bookmark is to be activated immediately after a user logs on. There are two variants:

1. YES

   The bookmark is started immediately after the login. This means that a window opens on the screen after the user logs on for each bookmark where the autostart option is set to "YES".

2. NO

   The bookmark must be explicitly started by clicking in the Bookmark dialog and it is not automatically activated. This is the default setting.

Start Mode   The Start mode defines whether the query is run immediately after the bookmark has been started and the navigation window is to be opened, or whether the query dialog is to be displayed. There are two variants:

1. NAVIGATE
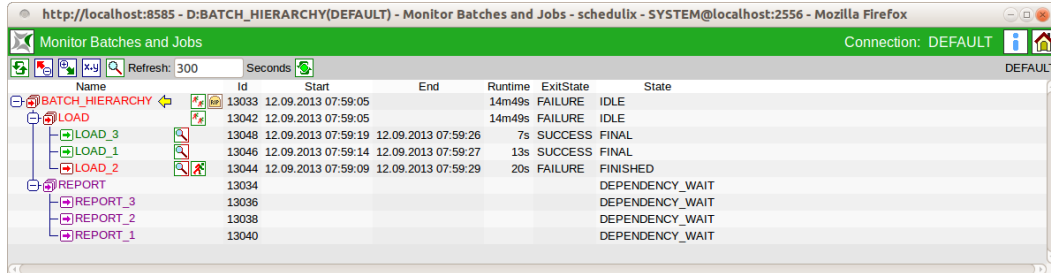
   The query is run immediately and the navigation screen opens with all the results.

2. QUERY

   The Query dialog is displayed first.

21.4 Detail Navigation

Clicking one of the Master Jobs (which has children) opens the "Details" window for the Master Job.

Figure 21.5: Running Master Jobs "Details" window

The "Details" window is similar to the Master window but with the following differences:

21.4.1 Tree view

Instead of the flat list in the Master window, all the children of the current batch and their own children are displayed in a tree hierarchy. The first level is expanded, but the others are not. The exception here is the search using 'Name Patterns', since here every found job is listed and the respective level and parent levels are also automatically expanded.

21.4.2 Search results

The results of a search are indicated by an arrow symbol

.

This enables you to differentiate between found entries that were expanded because another child entry meets the search criteria, and search hits. Only results are displayed with the arrow icon.

21.4.3 Detail navigation query mask
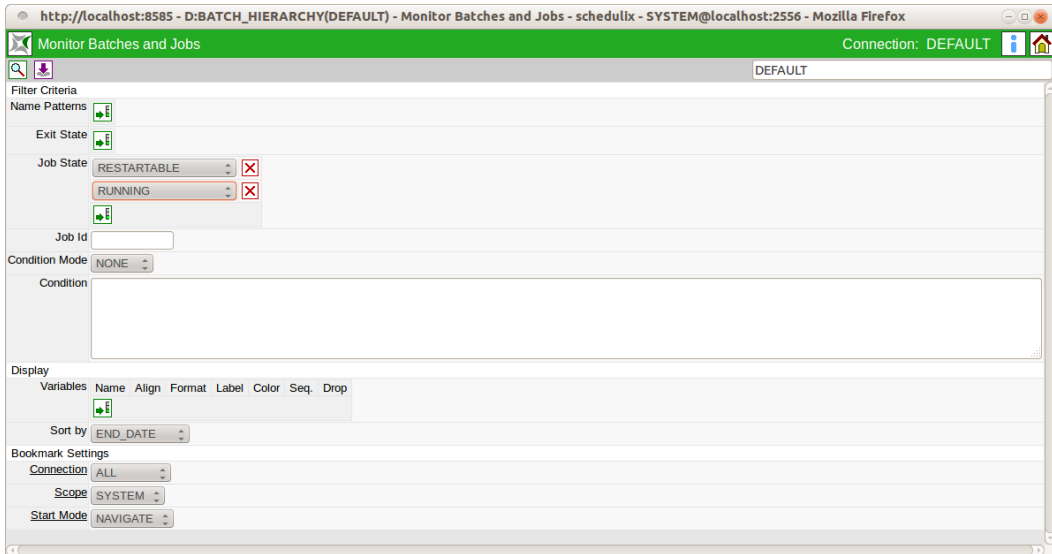
Figure 21.6: Navigation Details query mask

The query mask for the Navigation Details corresponds to the query mask for the Master Navigation, but with the following deviations:

The fields *History*, *Unit*, *Bookmark Details* and *Autostart* are not shown here and some other fields have been added. The mask contains the following fields:

Job State list In the Job State list, jobs which have the entered Job State can be filtered. Each of the states described in the *State* field can be selected and used as a filter.

If no Job States have been entered, they are ignored by the search.

Job Id A specific Job Id that is to be searched for can be entered in this field. A maximum of one job can be returned as a result.

Sort By The list can be sorted by various criteria with this function. The following criteria can be used:

1. Name

   The results are sorted alphabetically by job name.

2. Start Date

   The results list is sorted chronologically by the job start time.

3. End Date

   The results list is sorted chronologically by the job finish time.

Enable Bulk Operations   This flag whether checkboxes for selection and bulk operations will be available in the navigator view.

Bulk Select Hits only   If this flag is set, only search hits can be selected for bulk operations.

21.5 Details mask for jobs

Clicking a job without children in the "Details" or "Master" window automatically opens the Details mask for jobs. It looks like this:
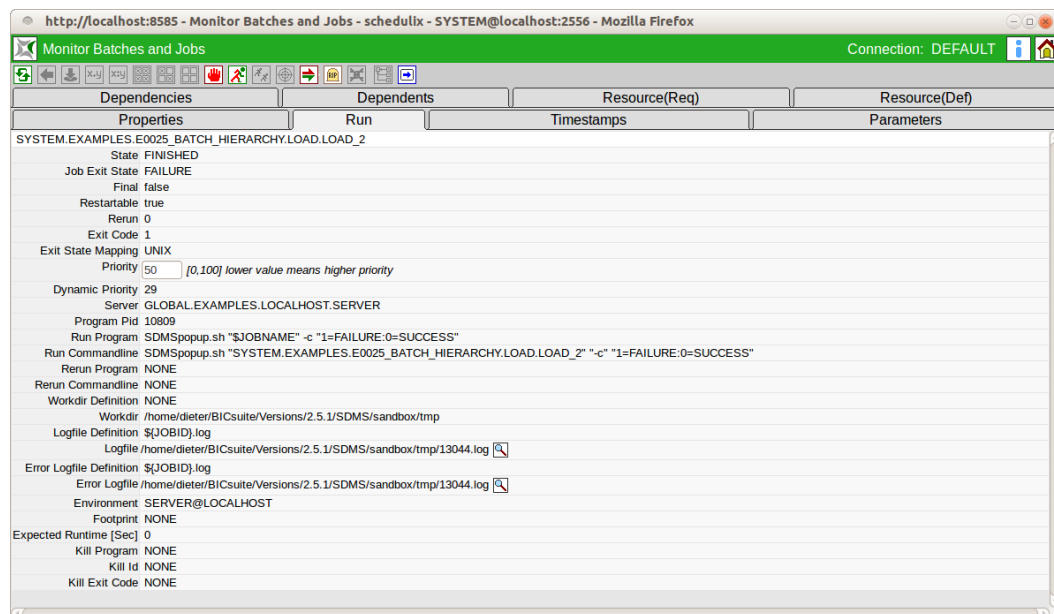
Figure 21.7: Details mask for jobs

All the runtime details are displayed in this mask and some of them can also be modified here. It also features some action buttons which allow the current objects to be manipulated.

21.5.1 Buttons

The following buttons are all action buttons. This means that these buttons are used to perform a certain   action on the selected job. To do this, a Confirm window

214
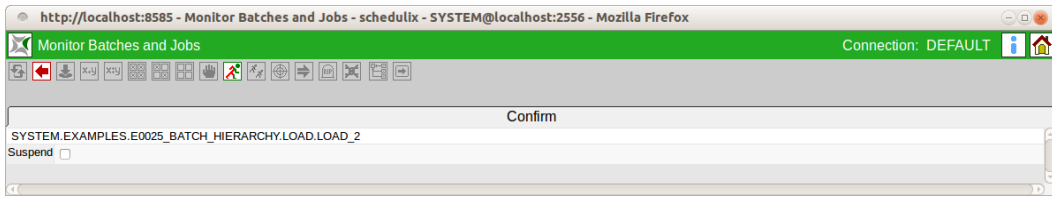
Figure 21.8: Confirm mask

opens like in Figure 21.8 after you click the button.

You can now enter a comment in the text box to describe the reason for the action or some other remarks. This text is later shown in the Audit tab in the field *Reason/Comment*. The action is performed when you click the respective action button again. Clicking the *Cancel* button aborts the action and opens the previous screen again. The Confirm screen contains additional input fields for controlling the action (these vary according to the action).

Suspend

Jobs can be suspended using the *Suspend* button. This is only possible if the jobs do not already have the "Suspended" state and they have not yet been completed. The jobs are placed in the "Suspended" state if this action is successful.

Suspend Local If the *Suspend Local* checkbox is checked, the children of the suspended job or batch will not be suspended. This way the execution of the children will continue, but the the batch or job itself will not go FINAL. Dependent batches or jobs will have to wait until the batch or job is resumed.

Suspend Restricted The Suspend Restricted checkbox is only editable for users of group ADMIN. If checked only ADMIN users are allowed to resume the batch or job.
Additional input fields: *Resume*, *Resume Time*, *Resume In*, *Unit*

Resume Here you can choose whether a workflow should be resumed automatically.
The following options are available:

- NO: deselects this functionality and no more input fields are displayed.

- AT: selects an automatic resume action at a fixed time. The *Resume Time* input field is displayed.

- IN: selects an automatic resume action after a time period has expired. The *Resume In* and *Unit* input fields are displayed.

Resume Time The required Resume Time in the format "YYYYMMDD HH:MM" is entered here.

Resume In Here you can specify how many time units (see *Unit*) the system is to wait for until the resume action is triggered.

Unit This field is used to define whether the entry in *Resume In* is in minutes (MIN), hours (HOUR) or days (DAY).

Resume

Jobs in the "Suspended" state can be reactivated with the *Resume* buttons. This means that the resources and dependencies are checked again, and if all these checks are successful the job is started on a jobserver.
Additional input fields: *Resume*, *Resume Time*, *Resume In*, *Unit*

216

Resume   The resume type can be selected here.

- OFF: This option is only available if the job is suspended and an Autoresume (IN/AT) is active. It disables this auto-resume function.

- NOW: A resume takes place immediately. No other input fields are displayed.

- AT: selects an automatic resume action at a fixed time. The *Resume Time* input field is displayed.

- IN: selects an automatic resume action after a time period has expired. The *Resume In* and *Unit* input fields are displayed.

Resume Time, Resume In, Unit   See above (Suspend action)

Rerun

The *Rerun* button is displayed if the current job has been finished and it is in a "Restartable" state. This means that it has an Exit State which is not final. Clicking the *Rerun* button calls the Rerun program (if it has been defined, otherwise the Run command). This allows a failed job to be restarted after troubleshooting it.
Additional input fields: *Suspended*, *Resume*, *Resume Time*, *Resume In*, *Unit*

Suspended   If this flag is set, the job is suspended and the *Resume* check box is displayed. If this field is not set, no other fields are displayed either.

Resume, Resume Time, Resume In, Unit   See above (Suspend action)

Rerun Children

The *Rerun Children* button is displayed if children of the current job are in a "Restartable" state. This means they have a Restartable Exit State. Clicking the *Rerun Children* button starts the Rerun program for all the Restartable children. This allows failed children to be restarted after troubleshooting them.
Additional input fields: See *Rerun*

Kill

The *Kill* button is only displayed if the current job is running at that moment (state Running) and a Kill program has been defined in the job definition. Clicking the *Kill* button calls the respective Kill program and the job's state is set to "TO KILL". If the Kill program was run successfully, the job's state changes to "KILLED". This enables the job to be terminated during the program runtime.

⮕ Set State

This allows you to manually set an Exit State for a job. The *Set State* button is displayed if a job fulfils at least one of the following conditions:

- The job is in an Error State (restartable)

- The job is in a Pending State

- The job is suspended and not active and not in a FINAL State

Additional input fields: *Set Exit State*, *Force*, *Resume*

Set Exit State   A valid Exit State can be selected here in the *Set Exit State* field.

Force   Here you can select whether you want to allow an Exit State that cannot be attained by the job itself using its Exit Code.

Resume   Here you can select whether a suspended job is to be resumed after the Set State.

Cancel Run

The *Cancel Run* button is displayed if the job is waiting for resources or dependencies or it has still been allocated on a jobserver. The scheduled job run can be cancelled by clicking the *Cancel Run* button. The job then gets the state "Cancelled". Neither a rerun nor a resume are possible. The job doesn't get an Exit State either. Jobs that are waiting for a cancelled job have to ignore this job or be cancelled as well.

Disable

The *Disable* button is displayed if the job or batch is or was not yet active. Clicking the *Disable* button disables the job or batch including all the children. All the disabled jobs are no longer executed, and take on their default Exit Status as soon as their dependencies have been fulfilled. This operation cannot be undone.

218

![Comment icon] **Comment**

You can enter a comment regarding the job by clicking the *Comment* button. Apart from entering the comment, no other action is performed on the job.

![Edit Job icon] **Edit Job**

The *Edit Job* button is not an action button because no changes can be made here to the current runtime object. Clicking the *Edit Job* button opens a new window with the Batches and Jobs Definition Screen for the job definition. The data shown here refers to the definition of the task at the time of the submit.

21.5.2 Properties tab

The "Properties" tab contains all the information regarding the job definition and the current state. It looks like this:
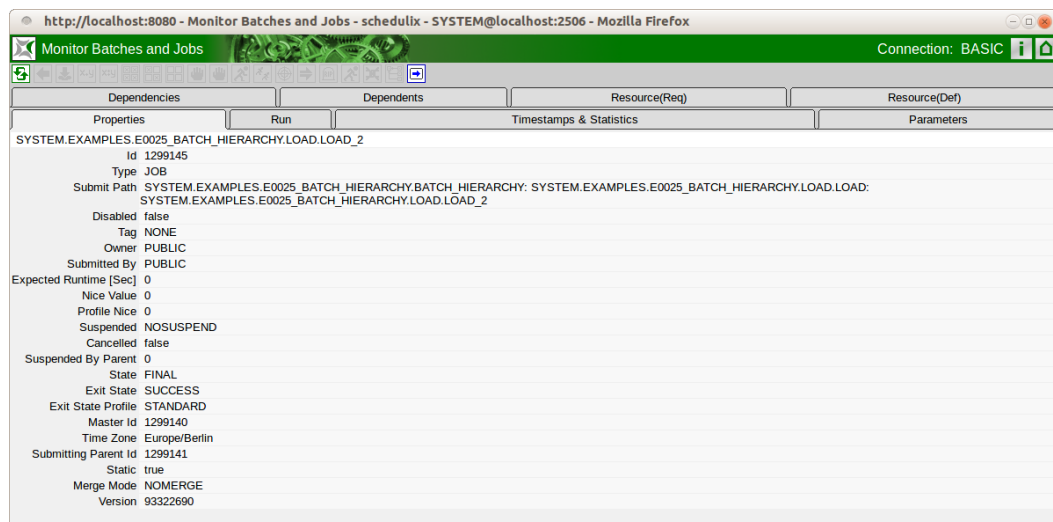
Figure 21.9: Job and batch properties

The fields in the "Properties" tab have the following meanings:

ID This is the unique identification number of the job runtime object.

Type This is the type of runtime object. More details about types can be found in Chapter 13.5.1.

Submit Path This is the full submit path to the runtime object. This means that the comma-separated path used for the submit is given here.

Tag This is the name of the Child Tag. This is only required for a dynamic child and uniquely identifies the current child instance. More details about dynamic children can be found in Chapter 13.5.4.1.

Owner This is the name of the group that created the job definition for this runtime object.

Submitted By This is the name of the group that submitted the runtime instance. If the job was dynamically submitted by a parent program, this is the user who submitted the parent.

Unresolved Handling The *Unresolved Handling* field is only shown in the Master Job mask. It shows the setting of the *On Unresolved Error* field from the submit mask.

- Ignore: Unresolved Dependencies are always ignored.

- Suspend: If errors occur due to Unresolved Dependencies, the entire process is submitted as being "suspended".

- None: Default behaviour; in the event of Unresolved Dependencies, the behaviour is as defined in the Dependency Definition.

Nice Value This is an input field where the Nice Value can be entered. More information about Nice Values can be found in Chapter 13.5.4.1. Since this is the only input field in the dialog box, you have to press the tabulator key after entering the data to enable the *Save* button for saving your changes.

Suspended This field indicates whether the runtime object is currently suspended (TRUE) or not (FALSE). The Suspend State can be changed using the *Suspend* and *Resume* buttons.

Suspend by Parent This indicates whether the runtime object was suspended by the parent (1) or not (0).

State This shows the current state of the runtime object. More details about states can be found in Chapter 21.3.

Error Message If the state "ERROR" was reported in the *State* field, a detailed description of the error is displayed.

Exit State If the job has finished, its resultant Exit State is shown here.

Exit State Profile   The Exit State Profile entered in the job definition is shown here. More details about Exit State Profiles can be found in Chapter 13.5.1.

Master Id   This is the Id of the Master Job that was submitted in order to create this runtime object. If the object itself was submitted as the Master Job, this is identical to the Id.

Submitting Parent Id   This is the Id of the parent runtime object that submitted the current job. If the job does not have a parent, "NONE" is displayed here.

Static   *Static* is the descriptor indicating whether the current runtime object is a statically (TRUE) or dynamically submitted child (FALSE). More details about dynamic children can be found in Chapter 13.5.4.1.

Merge Mode   This is the Merge Mode that is currently being used. More details about the Merge Mode can be found in Chapter 13.5.4.1.

Version   This is the version number of the Job Definition current at the time of the submit.

21.5.3 Run tab

The "Run" tab contains all the information relating to the current details about the Run program, the runtime environment and the computer. The tab looks like this:
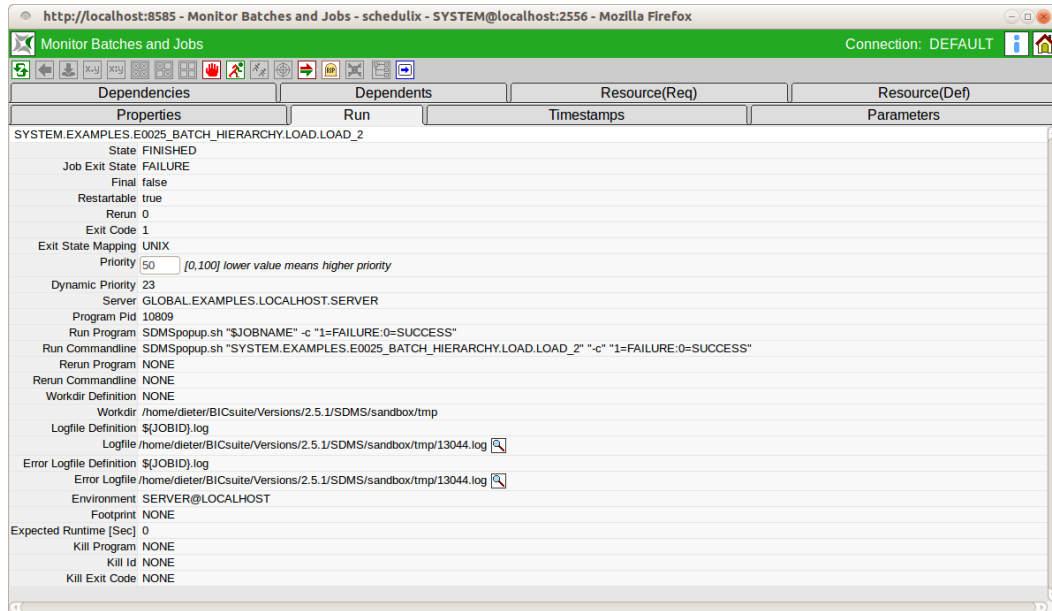
Figure 21.10: Batch und Job Run information

The fields in the "Run" tab have the following meanings:

State   This shows the current state of the runtime object. More details about states can be found in Chapter 21.3.

Job Exit State   The Job Exit State shows either the job's current Exit State (if it has finished) or "NONE".

Final   If the Job Exit State is a Final State, the value in this field is "TRUE"; if it is "NONE" or not a Final State, the value is "FALSE".

Restartable   If the job is restartable, the value in this field is "TRUE", otherwise it is "FALSE". If the value is TRUE, the *Rerun* button is enabled.

Rerun   The *Rerun* field shows the number of times the job has been restarted.

Exit Code   The Exit Code is the exit value that the Run program had when the process finished. More details about the Exit Code can be found in Chapter 13.5.2.

Exit State Mapping   This is the Exit State Mapping defined for the job in the Batches and Jobs dialog that was used to convert the Exit Code into the Job Exit State. More details about Exit State Mapping can be found in Chapter 13.5.2.

Priority   This is an input field It shows the current priority, which can also be changed here. The new priority is saved by pressing the tabulator key and clicking the *Save* button. More details about priorities can be found in 13.5.2.

Dynamic Priority   This is the dynamic (effective) priority for the job. It rises as the number of submits increases until the job is executed. More details about the dynamic priority can be found in Chapter 12.4.2.3.

Server   The *Server* field shows the current jobserver on which the process is running or has been run.

Program PID   The *Program PID* is the process Id of the Run program.

Run Program   This is the command line for the Run program defined in the Batches and Jobs dialog. The variables and parameters are still displayed here under their names and are not substituted.

Run Command Line   This is the command line as defined in the Batches and Jobs dialog for the Run program with substituted values. This is the view as it was handed over to the processing shell. All the variables and parameters were substituted for their current values.

Rerun program   This is the command line as defined in the Batches and Jobs dialog for the Rerun program. The variables and parameters are still displayed here under their names and are not substituted. NONE is shown here if a Rerun program has not been defined.

Rerun Command Line   This is the command line as defined in the Batches and Jobs dialog for the Rerun program with substituted values. This is the view as it was handed over to the processing shell. All the variables and parameters were substituted for their current values. NONE is shown here if a rerun has not taken place.

Workdir Definition   The *Workdir Definition* field shows the directory from where the process is being run. If "None" is shown here, the default working directory for the executing jobserver is used.

Workdir   This is the Workdir that is being used (or is to be used) in the current job. The substituted parameters and variables are displayed here if the job has finished or it is still running (otherwise just the parameters).

Logfile Definition   The *Logfile Definition* corresponds to the definition of the log files in the Batches and Jobs dialog.

Logfile   The name and path of the currently used log files are shown here (if the job has finished or it is still running), otherwise NONE is displayed. Possible variables and parameters are substituted for the actual values. Clicking the button

opens a new window displaying the contents of the log files. Should this not be possible, please contact your system administrator.

Error Logfile Definition   The Error Logfile Definition corresponds to the definition of the error log files in the Batches and Jobs dialog.

Error Logfile   The name and path of the currently used error log files are shown here (if the job has finished or it is still running), otherwise NONE is displayed. Possible variables and parameters are substituted for the actual values. The button

has the same function as described above for log files.

Environment   This field corresponds to the definition of the environment in the Batches and Jobs dialog.

Footprint   This field corresponds to the definition of the footprint in the Batches and Jobs dialog.

Expected Runtime   This corresponds to the definition in the Batches and Jobs dialog.
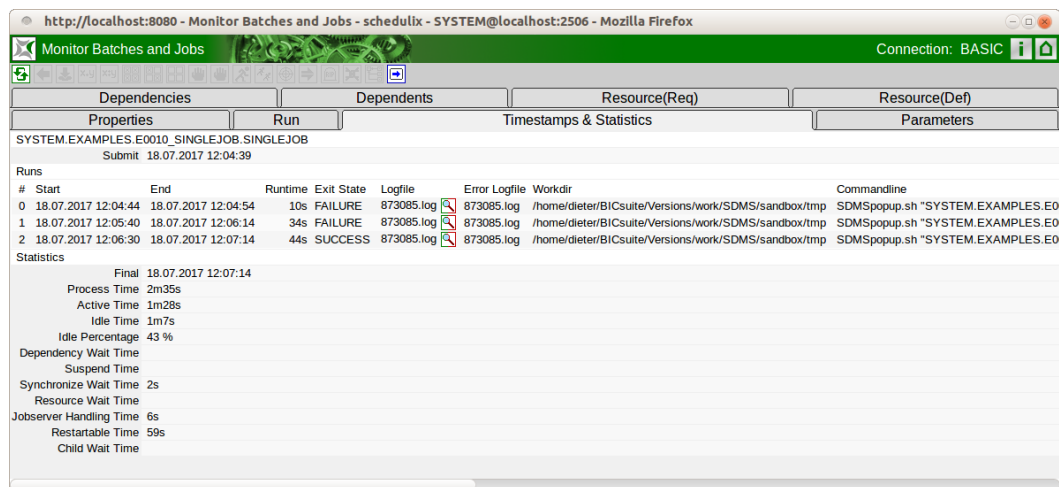
Kill Program   This corresponds to the definition of the Kill program in the Batches and Jobs dialog.

Kill Id   The Kill Id contains the process Id of the Kill program if this has been started. If a Kill action has not yet been performed, the Kill Id is NONE.

Kill Exit Code The Kill Exit Code is the result returned to the jobserver by the Kill program after the kill process has finished. This can be used to determine whether a kill attempt has succeeded or failed if this is configured in the Kill program.

21.5.4 Timestamps & Statistics tab

The "Timestamps & Statistics" tab shows the different dates and times of the respective jobs. These times give an overview of the temporal behaviour and duration of the job runs. The tab looks like this:

Figure 21.11: Batch and job timestamps

The tab is for information purposes only; no entries can be made here.
The fields in the "Timestamps & Statistics" tab have the following meanings:

Submit The submit time for the current job is shown in the *Submit* timestamp. If it is a static job this is the submit time for the Master Job; in the case of dynamic jobs it is the time when the parent job performed the dynamic submit. The submit date is the same as the time when the job switched to the "Dependency Wait" state. The *Runs* table lists every execution of the job. Normally this table only contains one line. However, if the job was restarted due to any errors, the table can ontain multiple lines.

Start The column *Start* shows when the Run program was started by the jobserver.

End The column *Finish* shows when the job was completed.

Runtime   The column *Runtime* shows the runtime of the run.

Exit State   The column *Exit State* contains the Exit Status of the run.

Logfile   The column *Logfile* contains the name of the logfile for standard out. A click on the icon (Magnifying Glass) after the Name opens the logfile for display.

Error Logfile   The column *Error Logfile* contains the name of the logfile for standard error. A click on the Icon (Magnifying Glass) after the Name opens the error logfile for display. The icon is only displayed if the name of the logfile and error logfile differ.

Workdir   The column *Workdir* shows thr working directory of the job.

Commandline   The column *Commdline* shows the commandline executed.

Jobserver   The column *Jobserver* shows the name of the Jobserver agents an, which executed the jon.

Exit Code   The column *Exit Code* contains the Exit Code of each execution.

PID   The column *PID* contains the system process id of the executed program.

Synchronize   The couln *Synchronize* indicates the time when the job switched to the "Synchronize Wait" state. That is the time when all the job's possible dependencies were fulfilled.

Resource   The couln *Resource* shows the time of the status transition to Resource Wait. This means that all the required Synchronizing Resources had been fulfilled and allocated to the job at this time.

Runnable   The column *Runnable* contains the time when the job entered the "Runable" state. This means that all the dependencies and resource requirements could be fulfilled. From now on the job can be processed by a server job.

Final   The *Final* timestamp shows when the job that has created a Final State finished. If the job was finished but it has a Restartable State, NONE is entered here.

Process Time   *Process Time* is the time, the jobs has spent after all pendencies have been fulfilled.

Active Time   *Active Time* is the time, the job or one its child was active.

Idle Time   *Idle Time* is the time, the job was inaktive and none of its children were active.

Idle Percentage   *Idle Percentage* is the precentage of the *Idle Time* of the *Process Time*.

Dependency Wait Time   *Dependency Wait Time* is the time, the jobs waited on dependencies.

Suspend Time   *Suspend Time* is the time, the job was suspended.

Synchronize Wait Time   *Synchronize Wait Time* is the time, the job spent waiting on synchronizing resources.

Resource Wait Time   *Resource Wait Time* is the time, the job spent waiting on system resorces.

Jobserver Handling Time   *Jobserver Handling Time* is the time overhead of the jobserver agent.

Restartable Time   *Restartable Time* is the time, the job spent waiting for a restart after an error.

Child Wait Time   *Child Wait Time* is the time, the job waited for its children to become FINAL.

21.5.5 Dependencies tab

All the current job's dependencies on other jobs are displayed in the "Dependencies" tab. The tab looks like this:
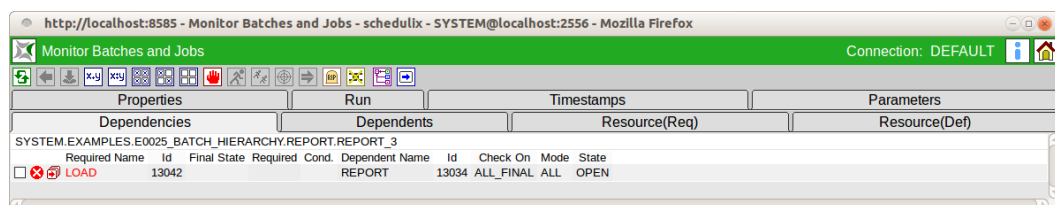
Figure 21.12: Batch und job dependencies

This dialog gives an overview of all the required dependencies, their current state and the possibility for deliberately ignoring dependencies.

The mask contains the following buttons:

✖ Ignore Dependencies

The *Ignore Dependencies* button is used to deliberately ignore one or more dependencies. All the dependencies that are to be ignored must have been previously selected in the mask. When you now click the button, a new window opens where you can select the Ignore mode and enter a remark. The window looks like this:
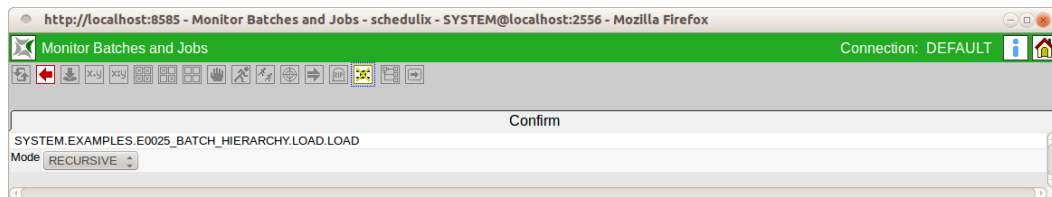
Figure 21.13: Batches and Jobs Ignore Dependencies confirmation mask

The ignore mode has to be entered in this mask. By default, all the children of a Submitted Entity inherit its dependencies: if B is required by A, each of A's children (CA) is likewise dependent on B.

The Ignore mode can now be used to determine the behaviour of the inherited dependency at the children.

The following options are available:

1. Recursive

    This is the default behaviour.

    The Ignore refers not only to the current job, but also to the inherited dependencies on the required job held by the children of the current job. This means that the parent job ignores this inherited dependency and all its children ignore it as well. If a child has an explicitly stated dependency on the required job which was not inherited, this is not ignored but is observed.
    Figure 21.14 shows an example.

2. Job Only

    Only the job itself ignores the dependency. All the children observe the inherited dependency and wait for the required job. This allows the parent job to be executed while forcing its children to wait.
    Figure 21.15 shows an example.

Having entered the Ignore mode and then a comment in the Comment field, you can perform the action by clicking the *Ignore Dependency* button again. The tab list
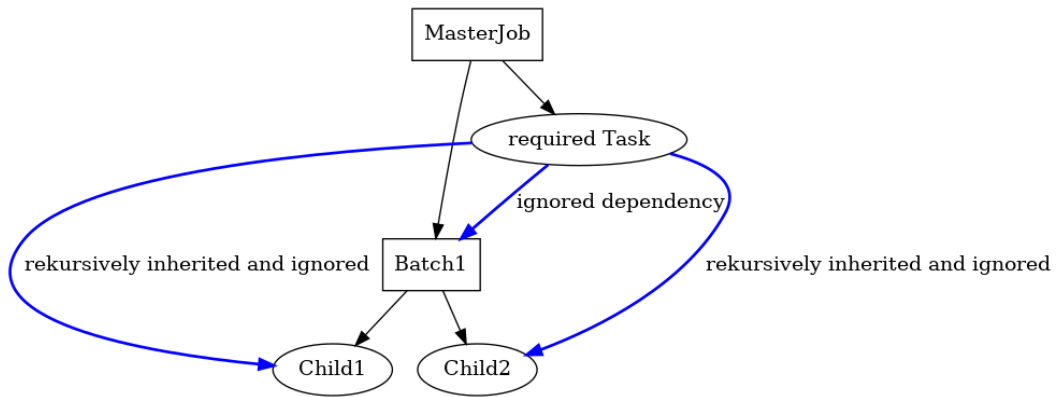
Figure 21.14: Example of a Recursive Ignore
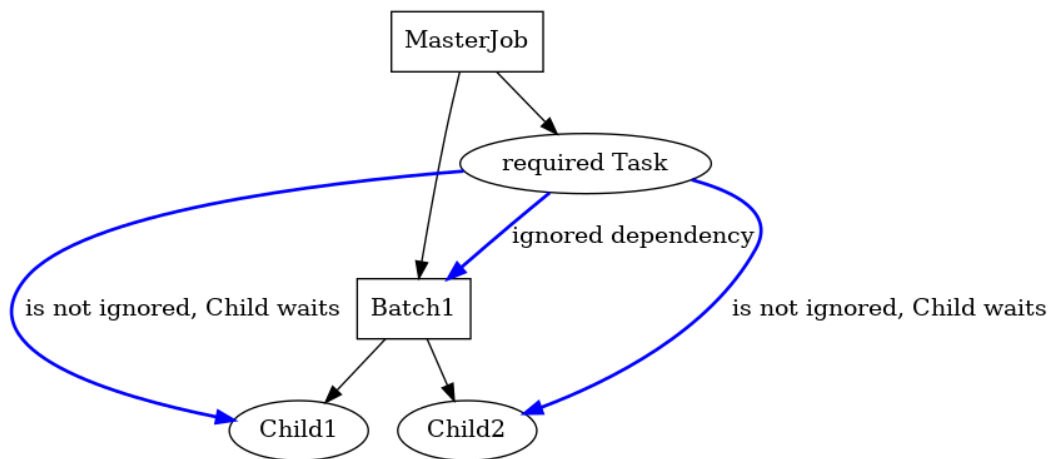
Figure 21.15: Example of a Job Only Ignore

shows all the dependencies that have to be fulfilled when the job starts and their current states. In this list, you can make a selection that can be used to ignore dependencies by clicking the *Ignore Dependencies* button.

The following fields are shown in the list:

Icon field   The current state of the dependency is shown in the *Icon* field The following options are available:

The dependence has not yet been fulfilled or can no longer be fulfilled. Should this icon be visible next to a dependency with the Dependency Mode ALL, the current job cannot be executed by this dependency.
In the Dependency Mode ANY, at least one dependency must not be red for the job to be executed.

The dependency has been fulfilled. In the Dependency Mode ANY, at least one dependency must have the state Green for the job to be able to run. In the Dependency Mode ALL, all the entries must be green for the job to be able to run.

Required Name   This is the name of the job that has to be a prerequisite for starting the current job run. You can switch to its runtime definition and check its state by clicking the name.

ID   The runtime Id of the Required Job is shown in this field.

Final State   If the Required Job has been completed, the job's Final State is shown here. If a Final State is entered here and the dependency is not fulfilled, the reason for this happening must an incorrect Final State for this dependency. This means that there is a difference between a Final State and a Required State. If the job is to be executed now after all, this can only be done with an Ignore by clicking the *Ignore Dependency* button. If the dependencies in the list are in the Dependency Mode ANY, at least one another dependency has to be fulfilled.

Required   The required Final State for the job is shown in this field. For the dependency to be fulfilled, the Final State must be the same as the Required State or the Required State must be NONE.

Dependent Name   The job demanding this dependency is named in this field. This is normally the currently selected job, but it can also be one of the job's parents.

Check On   The *Check On* field describes the relation of the completion of the Required Job. The following variants are used:

1. All Final

   The Required Job and all its children must be final.

2. Job Only

   Only the Required Job has to be final; the children are ignored.

More details about the Check On function can be found in Chapter 13.5.6.1.

ID This is the Id of the dependent job.

Mode The *Mode* defines how the list of jobs is to be handled. The following variants are used:

1. ALL

   All the list's dependencies must be fulfilled before the job can start.

2. ANY

   All least one of the list's dependencies must be fulfilled before the job can start.

More details about Mode can be found in Chapter 13.5.6.

State This is the current state of the dependency relationship. The following variants are used:

1. OPEN

   The Required Job has still not attained a Final State (it has not yet run, is still running, or an error has occurred). The dependency has not yet been fulfilled, although it can be fulfilled by attaining the correct Final State.

2. FULFILLED

   The Required Job has finished and attained the required Final State. The dependency has been fulfilled.

3. FAILED

   The Required Job has finished and attained an incorrect Final State. The dependency can no longer be fulfilled. In the Dependency Mode ALL, the current job can only be made to start by ignoring the dependency. In the Dependency Mode ANY, at least one of the other dependencies is being fulfilled.
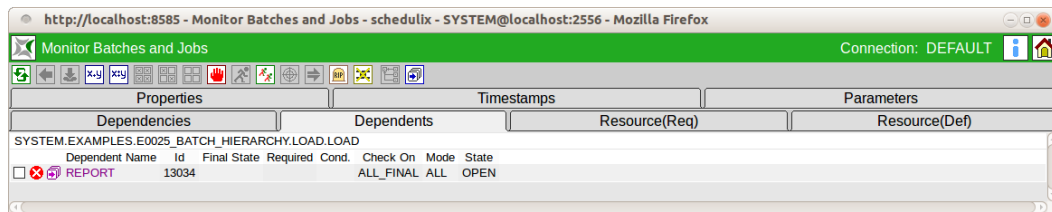
Figure 21.16: Batch and Job Dependents tab

21.5.6 Dependents tab

All the other jobs' dependencies on the current job are displayed in the "Dependents" tab. The tab looks like this:

This dialog gives an overview of all the existing dependencies, their current state and the possibility for deliberately ignoring dependencies.

The mask contains the following buttons:

Ignore Dependencies

The *Ignore Dependencies* button is used to deliberately ignore one or more dependencies. All the dependencies that are to be ignored must have been previously selected in the mask. When you now click the button, a new window opens
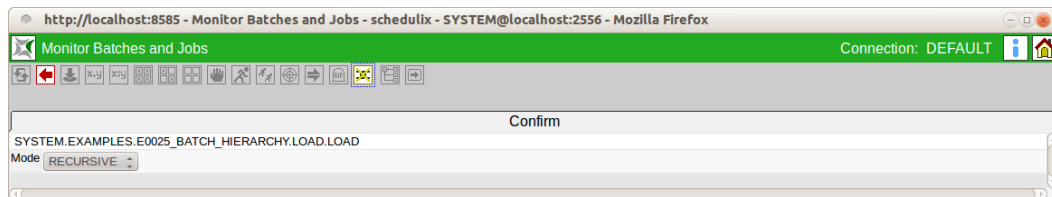
Figure 21.17: Batch and Job Ignore Dependencies

where you can select the Ignore mode and enter a remark. The window looks like in Figure 21.17.

The ignore mode has to be entered in this mask. By default, all the children of a Submitted Entity inherit its dependencies: if B is required by A, each of A's children (CA) is likewise dependent on B. The Ignore mode can now be used to determine the behaviour of the inherited dependency at the children.

The following options are available:

1. Recursive

   This is the default behaviour.

   The Ignore refers not only to the dependent job, but also to the inherited dependencies on the current job held by the children of the dependent job. This

means that the parent job ignores this inherited dependency and all its children ignore it as well. If a child has an explicitly stated dependency on the required job which was not inherited, this is not ignored but is observed.

2. Job Only

   Only the dependent job itself ignores the dependency. All the children observe the inherited dependency and wait for the current job. This allows the parent job to be executed while forcing its children to wait.

Having entered the Ignore mode and then a comment in the Comment field, you can perform the action by clicking the *Ignore Dependency* button again. The tab list shows all the dependencies that have to be fulfilled before other jobs are able to start together with their current states. In this list, you can make a selection that can be used to ignore dependencies by clicking the *Ignore Dependencies* button. The following fields are shown in the list:

Icon field   The current state of the dependency is shown in the *Icon* field The following options are available:

The dependence has not yet been fulfilled or can no longer be fulfilled. Should this icon be visible next to a dependency with the Dependency Mode ALL, the dependent job cannot be executed by this dependency.
In the Dependency Mode ANY, at least one other dependency (which is not visible here) of the dependent job must not be red for the dependent job to be able to run.

The dependency has been fulfilled. In the Dependency Mode ANY, at least one dependency must have the state Green for the dependent job to be able to run. In the Dependency Mode ALL, it may be necessary for other dependencies (which are not visible here) of the dependent job to be fulfilled for the dependent job to be able to run.

Dependent Name   This is the name of the job that is dependent upon the fact that the current job has been executed. You can switch to its runtime definition and check its state by clicking the name.

ID   The runtime Id of the dependent job is shown in this field.

Final State   If the dependent job has already been completed, the job's Final State is shown here.

Required   The required Final State for the current job is shown in this field.

Cond   This field shows whether an additional condition has been defined for the dependency.

Check On   The *Check On* field describes the relation of the completion of the current job. The following variants are used:

1. All Final

   The current job and all its children must be final.

2. Job Only

   Only the current job itself has to be final; the children are ignored.

More details about the Check On function can be found in Chapter 13.5.6.1.

Mode   The *Mode* defines how the list of jobs is to be handled. The following variants are used:

1. ALL

   All the current job's dependencies must be fulfilled before the job can start.

2. ANY

   At least one of the current job's dependencies must be fulfilled before the job can start.
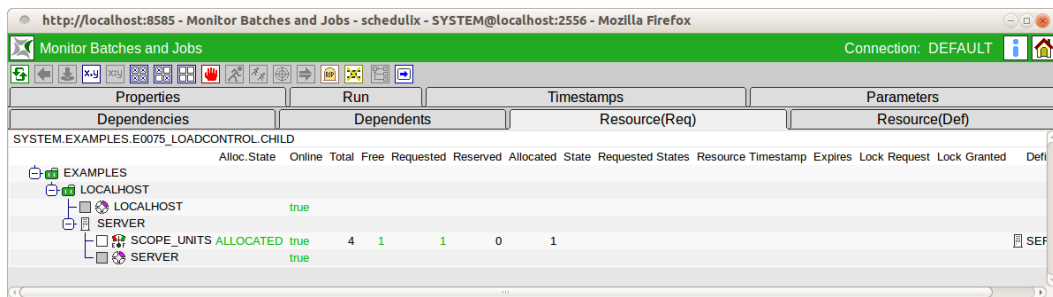
More details about Mode can be found in Chapter 13.5.6.

State   This is the current state of the dependency relationship. The following variants are used:

1. OPEN

   The current job has still not attained a Final State (it has not yet run, is still running, or an error has occurred). The dependency has not yet been fulfilled, although it can be fulfilled by attaining the correct Final State.

2. FULFILLED

   The current job has finished and attained the required Final State. The dependency has been fulfilled.

3. FAILED

   The current job has finished and attained an incorrect Final State. The dependency can no longer be fulfilled. In the Dependency Mode ALL, the dependent job can only be made to start by ignoring the dependency. In the Dependency Mode ANY, at least one other dependency (which is not visible here) of the dependent job must be fulfilled.

21.5.7 Resource(Req) tab

All the information about the current state of the requested resource is displayed in the "Resource(Req)" tab.

Figure 21.18: Job Resource Requirements

All the resource instances and their allocations to scopes can be viewed in this tab. It is also possible to ignore requests for resources.

The *Ignore Resources* button has the following meaning:

![Ignore Resources icon] **Ignore Resources**

The *Ignore Resources* button is used to deliberately ignore one or more resource requests. All the resource requests that are to be ignored must have been previously selected in the mask. When you now click the button, a new mask opens where you can enter a remark.
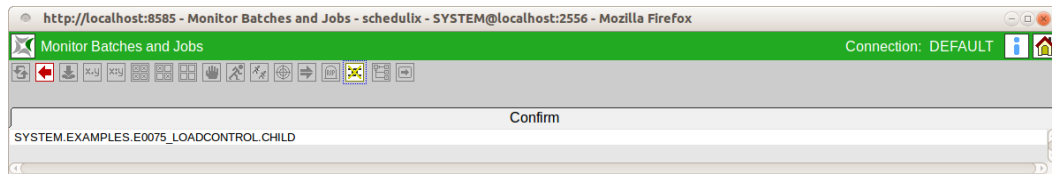
Figure 21.19: Job Ignore Resource Requirement

After you have entered a comment in the Comment field, clicking the *Ignore Resource* button again performs the action.

All the requested resources of the job are displayed in the Resources list. These are displayed hierarchically using the scopes and jobservers where the requested resources are located. The following columns are shown in the list:

All the requested resources together with their names are displayed in the first column of the list in a hierarchy (if the resources are located in scopes and jobservers). The resources for an Ignore can be selected by clicking the preceding *Ignore Dependencies* button. If the preceding field is grey this is not possible (static resources).

Clicking the resource name opens the 12.4.2.3 dialog, where the current allocation of the resource can be examined.

Allocation State The Allocation State is the current state of the resource request. The following variants are used:

1. None

   These are static or system resources that are checked at the time of the submit. If these are no longer present or are offline, an error message is returned at the submit time.

2. Blocked

   The resource is being used by another job and cannot be allocated. The job is now waiting for this resource. There may be different reasons for a block. The criteria preventing an allocation are shown in red in the list.

3. Allocated

   The resource has been allocated to the current job, fulfilling the requirement.

4. Available

   The resource is available but has not yet been allocated.

21.5.8 Resource(Def) tab

All the information about the current state of the resources defined by this Submitted Entity is displayed in the "Resource(Def)" tab. The tab looks like this:

Figure 21.20: Batch and Job Defined Resources

All the defined resources of the job are displayed in the Resources list. The following columns are shown in the list:

All the defined resources together with their names are displayed in the first column of the list. Clicking the resource name opens the 12.4.2.3 dialog, where the current allocation of the resource can be examined.

Usage   The *Usage* field specifies the "Resource" type. More details about resource types can be found in the Named Resource dialog.

State   The *State* field shows the current state of the resource if the usage has been set to "Synchronizing" and a Resource State Profile has been assigned.

Timestamp   The *Timestamp* shows the time of a resource's last status transition if the resource usage is "Synchronizing" and a Resource State Profile has been assigned.

Requestable Amount   If the resource usage is "Synchronizing" or "System", the *Requestable Amount* is the maximum amount that can be requested by a job.

Amount   If the resource usage is "Synchronizing" or "System", the *Amount* is the total amount that is available for all the jobs.

Load   If the resource usage is "Synchronizing" or "System", *Load* graphically displays where the resource is allocated.

Free   If the resource usage is "Synchronizing" or "System", *Free* is the non-allocated amount that is still available for all the jobs.

Online   This indicates whether the resource is "online", i.e. can be requested at this time.

21.5.9 Parameters tab

All the input and output parameters currently defined in this job and their actual values are shown in the "Parameters" tab.
The tab looks like this:
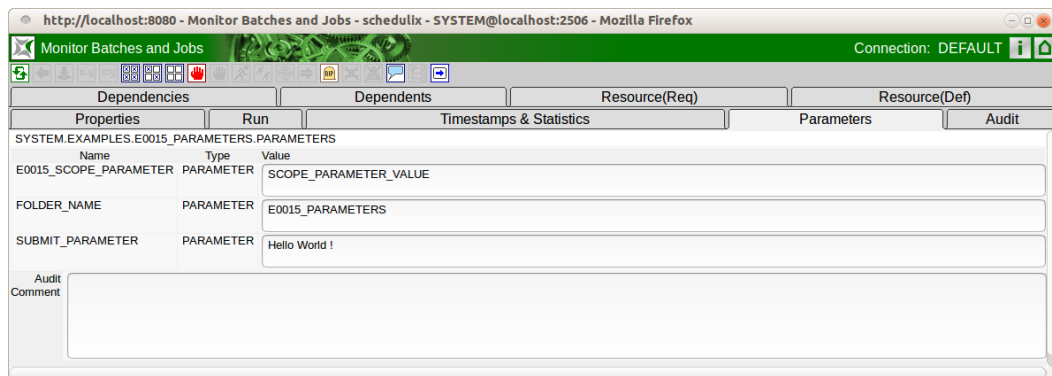
Figure 21.21: Batch and job parameters
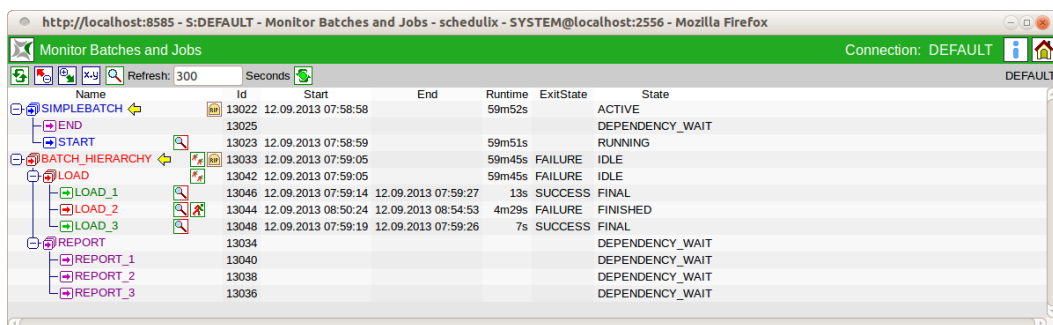
This tab is for information purposes only; no changes can be made here.

The "Parameters" list shows all the parameters that are currently defined in the job. If the active user is a member of either the "ADMIN" group or another group with "operate" privileges for the job or batch, parameters of type PARAMETER and RESULT can be edited. In this case the field *Audit Comment* is displayed. The reason for the modification can be entered here and will be recorded in the Audit Trail.

Information about parameters can be found in Chapter 13.5.10.

22 Search Running Jobs

22.1 View

Figure 22.1: Search Running Jobs

22.2 Concept

22.2.1 In short

In the *Search Running Jobs* dialog you can search for and view all the currently running or completed jobs, batches and milestones.

22.2.2 Detailed description

In this console you can check all the jobs, restart failed jobs, ignore resources, etc.

The Search Running Jobs window is also just a "default" view (of the bookmark "Default Search System") and can be customised and modified by the user accordingly. The "default" implementation is described here.

22.3 Navigator

The navigation window is the same as the "Details" mask for the Running Master Job. More information can be found there.

22.4 Navigator Details query mask

When you call the dialog, in contrast to the Running Master Jobs dialog the query mask opens first.
The query mask looks like this:
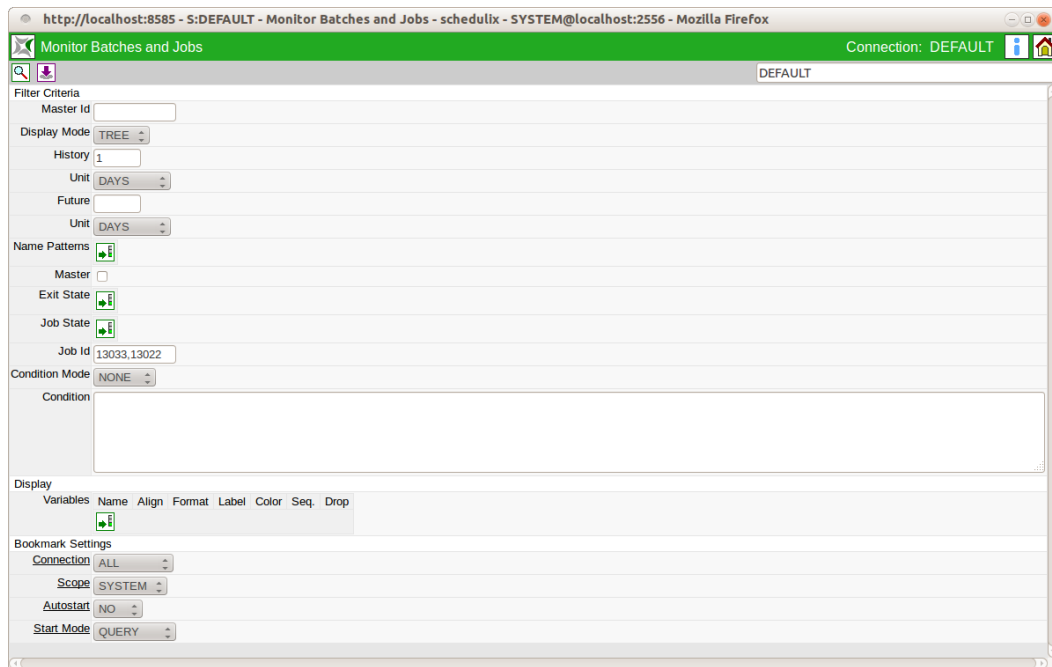
Figure 22.2: Search Running Jobs query mask

The query mask for the Navigation Details corresponds to the query mask for Running Master Jobs.

The following fields are additionally defined here, however:

Master Id   A runtime Id for a Master Job can be entered in the *Master Id* field. All the found jobs must be have been submitted by this Master Id.

Display Mode The list view can be configured in the *Display Mode* field. The following options are available:

1. LIST

   The output is displayed as a list.

2. TREE

   The output is shown as a hierarchy based on the parent-child hierarchy. The hierarchy levels can be expand and closed again in a tree.

Master The *Master* field is a filter (if the switch has been set) for displaying just Master Submittable jobs.

Job State list In the *Job State list*, jobs which have the entered Job State can be filtered. If no Job States have been entered, they are ignored by the search.

Job Id In this field you can enter a specific *Job Id* that you want to search for. A maximum of one job can be returned as a result.

Enable Bulk Operations This flag whether checkboxes for selection and bulk operations will be available in the navigator view.

Bulk Select Hits only If this flag is set, only search hits can be selected for bulk operations.

23 Calendar

23.1 View
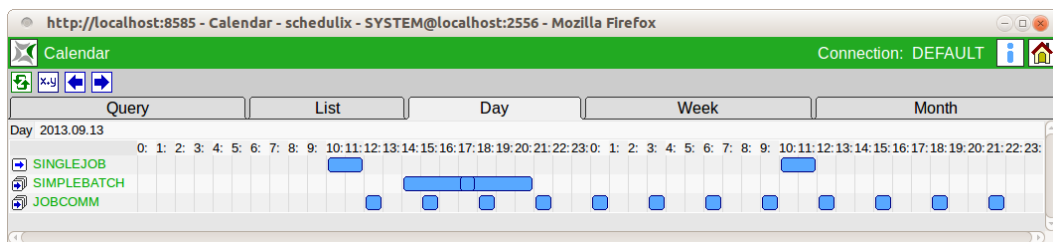
Figure 23.1: Calendar

23.2 Concept

23.2.1 In short

The calendar function gives an overview of scheduled executions of jobs and batches configured using the Time Scheduling module.
For a scheduled job or batch to be included in this overview, the calendar function must have been activated (Calendar = active) when the schedule was created.

23.3 Detailed description

The calendar is displayed in different ways. Firstly, there is the simple, chronologically ordered list in which the pending submits are listed. The evaluation period can be changed to prevent the list from becoming unnecessarily long and thus confusing. By default, only the next 24 hours are initially displayed.
The second view is the day view. Here the day is displayed as a period of 24 hours. Bars indicate when which job is to be started and, provided that this information is maintained by the user, how long it lasts.
The third and fourth views correspond to the day view, but here the period is displayed with a lower granularity. The week view shows seven days split up into 6-hour periods. The month view shows (maximum) 31 days.

In the week and month views, the jobs or batches that are to be executed are also shown as bars. However, the runtime of the objects is not as clearly displayed as in the day view because the bars are at least as long as the smallest unit in the respective view.

This means that a batch with an expected runtime of eight hours is actually displayed as 8 hours in the day view. In the week view, however, it is represented by two fields (i.e. 12 hours). In the month view it visually takes up a whole day.

23.3.1 Query tab
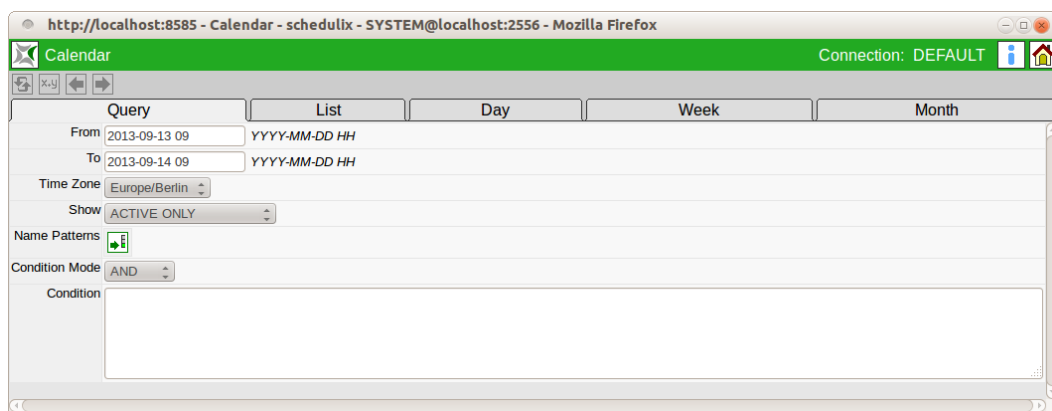
Figure 23.2: Calendar Query tab

In the "Query" tab you can specify search criteria for the Job Definitions that are to be displayed. All the other tabs are affected by these restrictions.

The *From* and *To* fields define the period for the start times of the jobs and batches that are to be displayed.

The time zone to be used for the calendar can be selected in the *Time Zone* field.

Simple conditions linked with Boolean operators can be entered in the *Condition* field. In the simple conditions you can test parameters with specific contents. The parameters are addressed with **JOB.***parametername*. The following example shows a (syntactically) valid condition:

```
job.developer == 'ronald' or
job.developer == 'dieter'
```

23.3.2 List tab
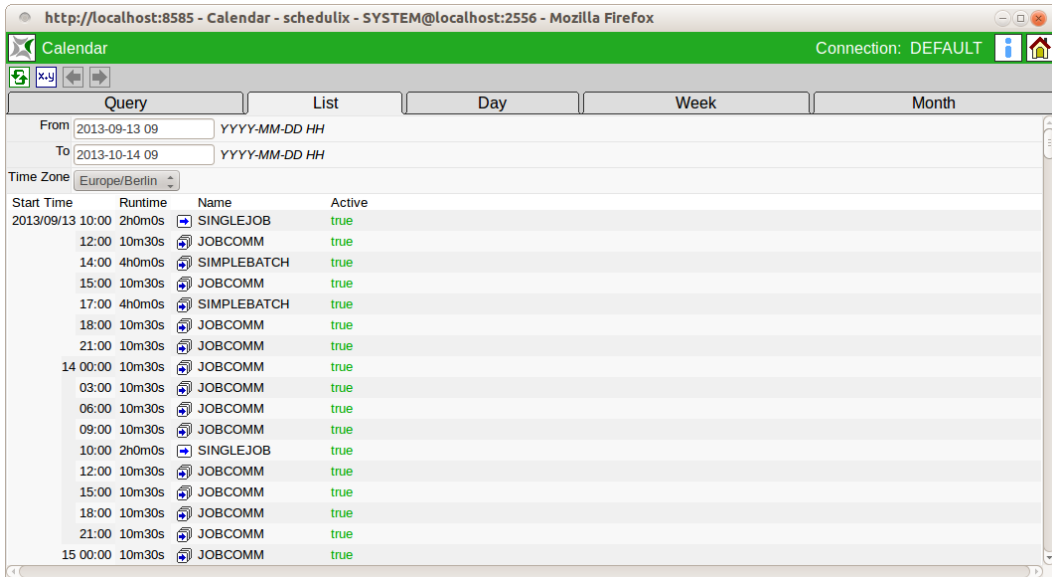
Figure 23.3: Calendar List tab

The "List" tab shows all the start times of the scheduled jobs and batches for the period set in the *From* and *To* fields in a simple list.

The time zone to be used for the calendar can be selected in the *Time Zone* field.

The scheduled start time is shown in the column *Start time*. For reasons of clarity, only the data that has changed in comparison with the previous row is displayed in this column and row. If two successive rows refer to the same day and hour, only the minutes of the time are shown in the second row.

23.3.3 Day tab
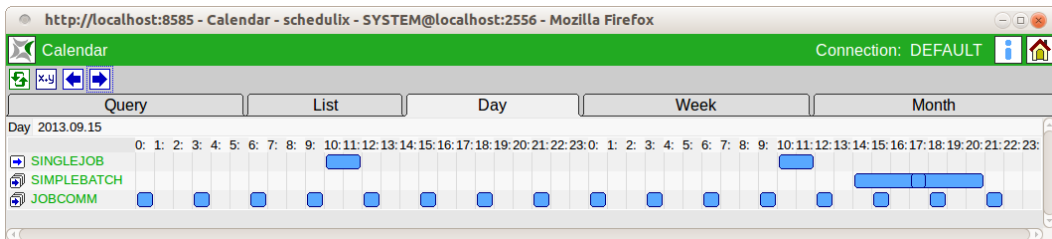
Figure 23.4: Calendar Day tab

The "Day" tab graphically shows all the scheduled jobs and batches for both the selected day and the following day.

If a job or batch is probably still active at the beginning, it is shown in the row
"KEEP_RESOURCE" as illustrated here. The representation of overlapping runs
can also be seen in the row.

23.3.4 Week tab
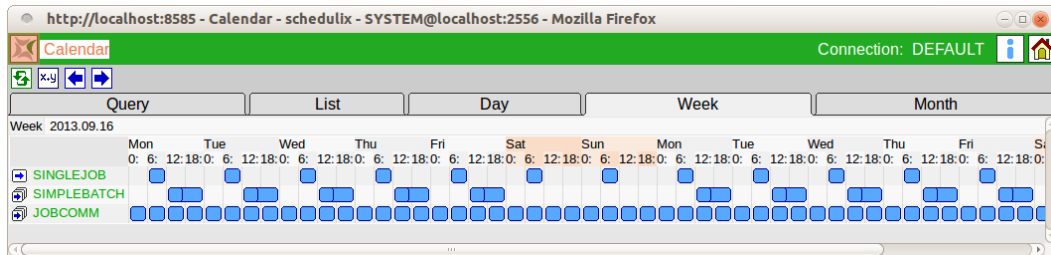
Figure 23.5: Calendar Week tab

The "Week" tab graphically shows all the scheduled jobs and batches for both the
selected week and the following week.

23.3.5 Month tab
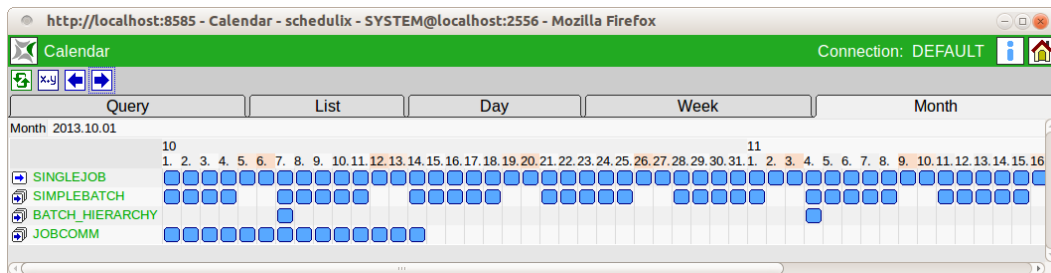
Figure 23.6: Calendar Month tab

The "Month" tab graphically shows all the scheduled jobs and batches for both the
selected month and the following month.

24 System Information

24.1 View
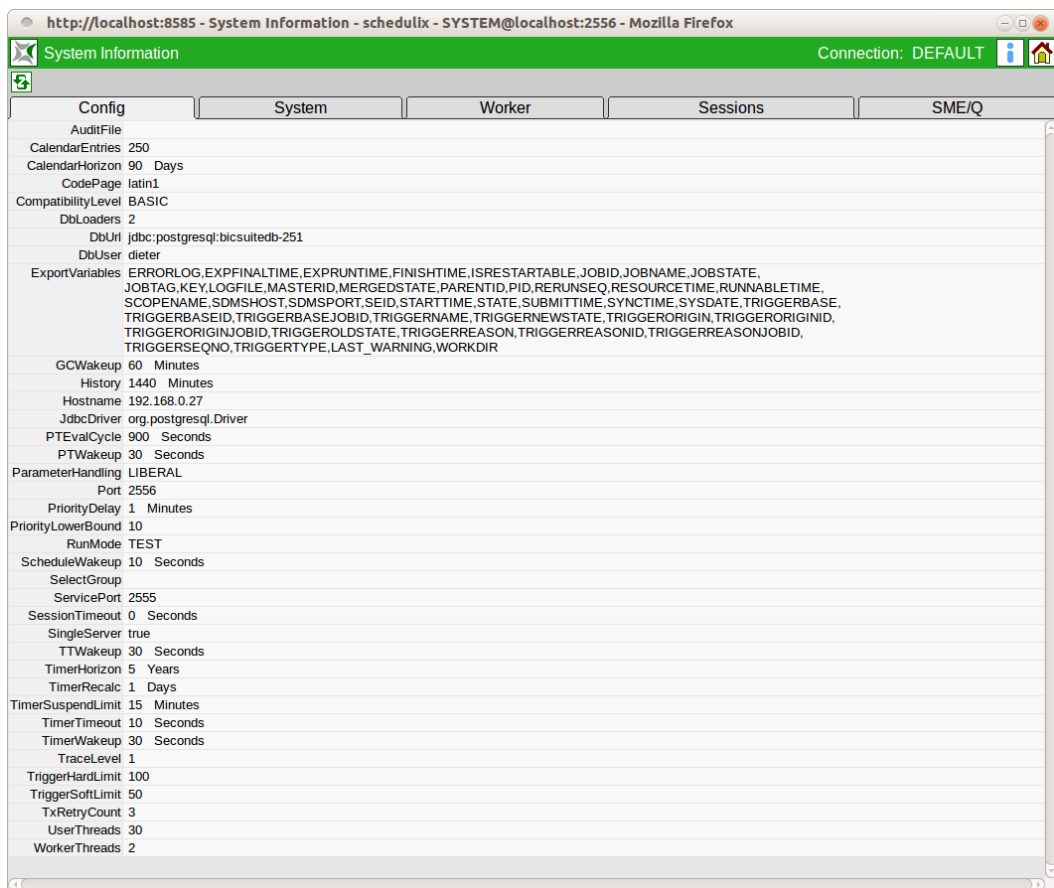
Figure 24.1: System Information

24.2 Concept

24.2.1 In short

The menu option "System information" in the main menu provides an overview of the server and system parameters, a list of worker activities, as well as a list of the current sessions.

24.2.2 Config tab
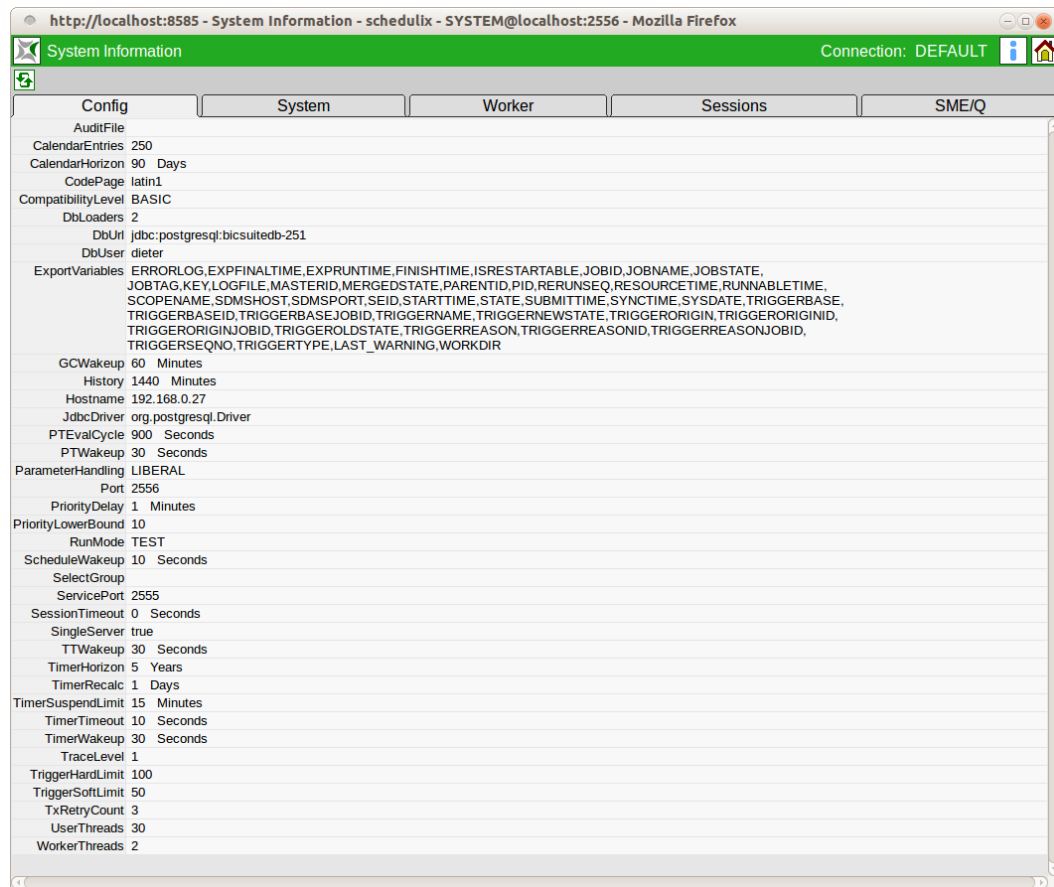
Figure 24.2: System configuration

The "Config" tab shows the current server configuration. The parameters are sorted here in alphabetical order. To change a parameter, you have to modify the $BICSUITECONFIG/server.conf file and restart the Scheduling Server.

If the environment variable BICSUITECONFIGhas not been set, $BICSUITEHOME/etc is used as the configuration directory.

For security reasons, not all the server parameters are displayed here. In particular this includes the access information for the database system. The currently used server parameters are as follows:

CalendarEntries

The parameter **CalendarEntries** defines the maximum number of calendar entries per Scheduled Event. The parameter **CalendarHorizon** normally takes effect first, but if there is an extremely large number of entries in the horizon, this parameter prevents the creation of nonsensical volume of calendar entries. As the horizon can be defined for each schedule, a sort of Denial of Service attack could be made on it. If this parameter is not defined, the default value of 300 entries applies.

CalendarHorizon

The parameter **CalendarHorizon** defines the default horizon for the calendars.
If this parameter is not defined, the default value of 62 days applies (always at least two months).

CodePage

The parameter **CodePage** defines the active character encoding. This parameter is now obsolete.

CompatibilityLevel

The parameter **CompatibilityLevel** defines which options and commands are recognised by the server as being valid. The possible values are: **BASIC**, **PROFESSIONAL** and **ENTERPRISE**. The maximum level is dependent upon the licensed version.
By default the CompatibilityLevel is set to the licensed version.

DbLoaders

The parameter **DbLoaders** determines how many threads for loading the object memory are started when the server starts. Depending on the number of available processors, a higher number of threads can make the server start more quickly.
By default the number of available processors (but no more than 5) is taken here.

DbUrl

The parameter **DbUrl** defines which database the server is to use. The valid syntax is dependent upon the database system.
There is obviously no default value for this parameter.

DbUser

The parameter **DbUser** defines the user name that is to be used for working with the database.
There is no default value for this parameter either.

ExportVariables

The displayed parameter **ExportVariables** is a composition of two configuration parameters: **ExportVariables** and **UserExportVariables**. These two variables are handed over to the jobserver as a list of key/value pairs when fetching a job. In turn, the jobserver has the option of setting the variables in the environment of the process that is to be started. Whether it does this, or under what name it sets the variables in the environment, is dependent upon the jobserver's configuration.
This functionality has been split into two parameters because the default entry for the parameter **ExportVariables** is the list of standard variables. To prevent the administrator from having to type the entire list just to export one more parameter, the additional tags can be specified in the parameter **UserExportVariables**.

GCWakeup

"Obsolete" objects are deleted from the object cache at regular intervals. The parameter **GCWakeup** defines the interval between the cleanups. It is not practical to enter a small value for this parameter because the effort is only dependent upon the number of existing objects (essentially the Submitted Entities) and not upon the number of objects to be cleaned up.
The default value for this parameter is 240 (minutes).

History

The parameter **History** defines the time period for keeping information about master instances of job or batch definitions. Older jobs or batches are only kept in the object cache if they are still active in some way (i.e. if they are neither final nor cancelled).
By default, information is kept for the last 10 days (14,400 minutes).

History Limit

The parameter **History Limit** defines the maximum time period for keeping information about master instances of job or batch definitions that are no longer active. This limit applies if the parameter **MinHistoryCount** is set to a figure other than 0. If History is set to 10 days, History Limit to 30 days and MinHistoryCount to 10, only the last 4 to 5 versions are kept in the object cache for a weekly job or batch.
By default, information is kept for the last 10 days (14,400 minutes).

Hostname

The parameter **Hostname** is the name of the Scheduling Server. This is also the content of the standard job parameter **SDMSHOST**. The default value is "localhost".

JdbcDriver

The parameter **JdbcDriver** defines which class is to be used as the JDBC driver. The name of this class is dependent upon the database system that is being used. This class must be present in the CLASSPATH. The CLASSPATH environment variable is set in the file $BICSUITECONFIG/java.conf. There is obviously no default value for this parameter.

MinHistoryCount

The parameter **MinHistoryCount** defines the minimum number of master instances of a job or batch definition that are kept in the object cache. This allows a job or batch that is only seldom executed to be kept in the object cache for much longer than is defined in the parameter **History**. However, the maximum time that such a job or batch is kept in the object cache is still restricted by the parameter **History Limit**. By default, this parameter has the value 0 and is therefore not active.

MaxHistoryCount

Der Parameter **MaxHistoryCount** defines the maximum number of master instances of a job or batch definition that are kept in the object cache. This allows a job or batch that is frequently executed to be kept in the object cache substantially shorter than is defined in the parameter **History**. An excessive memory usage caused by such jobs can be prevented this way.
By default, this parameter has the value 0 and is therefore not active.

ParameterHandling

The parameter **ParameterHandling** defines how the server is to respond when parameters are addressed that have not been declared. The possible values are:

Value	Meaning
STRICT	An error message is returned if an undeclared parameter is used.
WARN	A warning is written to the server's log file. The value of the variable is returned (if found).
LIBERAL	The value of the variable is returned (if found).

Table 24.1: ParameterHandling options

From the perspective of software engineering, **ParameterHandling** should be defined as STRICT. From the perspective of convenience, WARN or LIBERAL are preferable.
The default setting is LIBERAL.

Port

The parameter **Port** defines which TCP port the server uses for communicating with the clients. Any port can be chosen as long as it is not a port below 1024, since only highly privileged users are able to use these. Because the Scheduling Server has been designed so that no special privileges are required to operate it, this server shouldn't receive any special privileges either.
The default port is 2506.

PriorityDelay

The parameter **PriorityDelay** defines the time after which the (effective) priority of a job is raised. Automatically raising the priority prevents jobs from waiting indefinitely for resources.
The default value is 30 (minutes).

PriorityLowerBound

The parameter **PriorityLowerBound** determines the highest priority that can be attained through the normal ageing of jobs. If the parameter is greater than 0, a number of priority levels will remain free which can then be used by the administrator to guarantee specific jobs a place at the front of the queue.
PriorityLowerBound is set to 10 by default.

RunMode

The parameter **RunMode** should be set to PRODUCTION in productive environments. In the development environment at independIT this parameter is occasionally set to TEST. Setting RunMode to TEST enables the command "RUN TEST . . . ". The impact of using this command is not defined as it is used for testing purposes. It is quite possible that using the command will cause the server to crash.
RunMode is set to PRODUCTION unless it has been explicitly set to TEST.

ScheduleWakeup

The parameter **ScheduleWakeup** defines when the Resource Scheduling thread is to wake up. Since the Resource Scheduler uses up practically no time at all if there is nothing to do, the interval between two activations can be kept relatively short.
The default value is 30 (seconds).

SelectGroup

The parameter **SelectGroup** defines which group is allowed to use the Select statement. If this parameter is not set or it contains the name of a non-existent group, only members of the ADMIN group and members of groups with the SELECT privilege may use the Select statement. This parameter exists for backward compatibility reasons.

ServicePort

The parameter **ServicePort** is the port via which (only) an administrator can connect to the server. It can be used to still gain access (usually read access) to the server in the event of a malfunction.
The default value is 2505.

SessionTimeout

The parameter **SessionTimeout** defines how long a session can be idle by default before it is terminated by the server. Normally, this parameter only plays a role with interactive sessions using `sdmsh`.
The default value is 30 (seconds).

SingleServer

The parameter **SingleServer** is a Boolean value. If it is set to "true", the server assumes that no other server is able to access the repository. When it boots up after an "ungraceful shutdown", it ignores any set tickets and sets its own ticket. Should a second server be unexpectedly active, that server will realise that the repository now belongs to another server and will stop working. If both servers have the parameter set to "true", this will result in a kind of ping-pong effect.
The default setting is "false".

TTWakeup

The parameter **TTWakeup** defines how often the tickets in the repository are to be at least renewed. The parameters should be coordinated with one other (i.e. if possible be the same) in an environment with multiple servers.
The default value is 30 (seconds).

TimerHorizon

The parameter **TimerHorizon** defines how far into the future the Timer thread is to look to find the next execution time for a job.
The default value is 5 (years).

TimerRecalc

The parameter **TimerRecalc** determines after how long another attempt is to be made to find the next execution time for a job after the previous search has been terminated because the TimerHorizon was reached.
The default value is 5 (days).

TimerSuspendLimit

The parameter **TimerSuspendLimit** defines the time after which a job is submitted as being suspended if the server was not active at the scheduled Submit time. This limit can be set individually for each Scheduled Event. If this is not done, however, the value for the TimerSuspendLimit applies.
The default time is 15 (minutes).

TimerWakeup

The parameter **TimerWakeup** determines at what intervals the Timer thread is activated. A higher value will result in inaccuracies concerning the actual Submit time as well as potentially long runtimes for the Time Schedule process. A low value is significantly less harmful.
The default value of 30 (seconds) has proved to produce good results.

TraceLevel

The parameter **TraceLevel** determines how much and what kind of logging information the server writes to the log file. The table below shows what type of messages are logged on which Trace Levels:

Type	Trace Level	Remark
FATAL	All	FATAL messages are logged on the server in the event of a serious malfunction and should be reported immediately to the independIT support team.
ERROR	All	ERROR messages are logged when errors occur. These types of errors are significantly less critical than FATAL class errors. It is advisable to investigate these types of errors and to eliminate their causes where possible.
Continues on next page		

254

Type	Trace Level	Remark
INFO	All	INFO messages contain important or interesting information. Examples of such messages are those that are written during the server startup phase.
WARNING	> 0	WARNING messages are interesting messages that frequently deal with operating inaccuracies. An example of this are the warnings that occur when the parameter ParameterHandling is set to WARN or STRICT.
MESSAGE	> 1	MESSAGE messages are relatively unimportant but can nonetheless be very interesting. For example, all executed commands and their execution times are logged as MESSAGEs.
DEBUG	3	DEBUG messages can be helpful for troubleshooting purposes. However, Trace Level 3 will produce a huge flood of messages. In normal operation, these messages are less meaningful and are, more than anything else, distracting.

Table 24.2: Overview of the Trace Levels

TriggerHardLimit

The parameter **TriggerHardLimit** defines the maximum number of times that one and the same trigger can be initiated.
The default setting is 100.

TriggerSoftLimit

If triggers are created without a FireLimit being specified, the parameter **TriggerSoftLimit** sets the maximum number of times that an instance of this trigger can be initiated.
The default setting is 50.

TxRetryCount

In some cases the performance of a transaction may fail even though the transaction is semantically and syntactically correct. In this case, the server will try to perform the transaction again. How often this takes place is defined by the parameter **TxRetryCount**.
By default, a maximum of three attempts are made.

UserThreads

An important parameter in the server configuration is the parameter **UserThreads**. This parameter defines the maximum number of sessions that can be connected simultaneously. Users who are using the web front end do not have to be included in the full number because the Zope server sets up a new connection to the Scheduling Server for each communication step and then breaks it again, respectively reuses cached connections.
The jobservers usually remain connected with the Scheduling Server, however, and they each occupy one thread.
The default value for this parameter is 10.

WorkerThreads

The parameter **WorkerThreads** defines how many threads are to be started to respond to read queries. In contrast to write transactions, many read transactions are extremely complex. In particular this includes monitoring-related read transactions.
By default, two such threads are started. In larger environments it makes sense to raise this number.

24.2.3 System tab
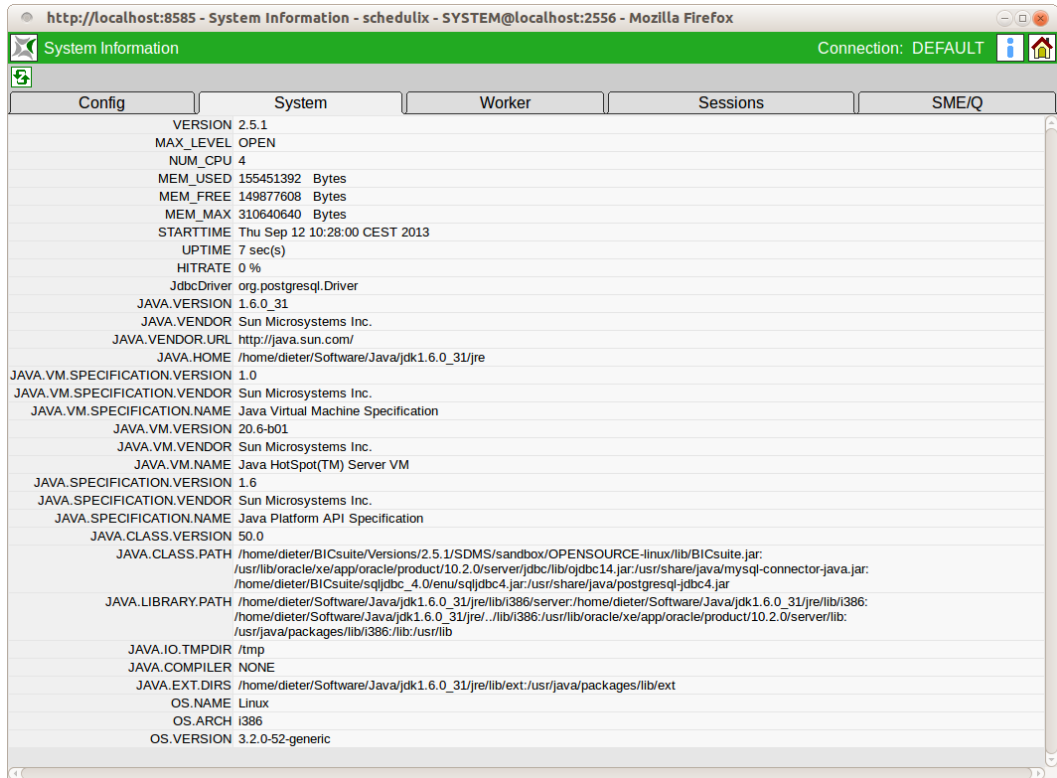
Figure 24.3: System runtime environment

This tab shows some runtime information together with details of the Java Virtual Machine. The information may differ to the information shown here depending upon the hardware and operating system or Java Virtual Machine that you are using. You should therefore to the relevant documentation for a correct interpretation of this information.

VERSION

The *VERSION* field shows the software version of the implemented Scheduling Server.

MAX_LEVEL

The *MAX_LEVEL* field shows the licensed version of the Scheduling Server.

NUM_CPU

The *NUM_CPU* field shows the number of processors reported by the Java Virtual Machine.

MEM_USED

The *MEM_USED* field shows how much memory is currently being used by the Java Virtual Machine.

MEM_FREE

The *MEM_FREE* field shows how much of the occupied memory is still available for the server.

MEM_MAX

The *MEM_MAX* field shows the maximum amount of memory that the Java Virtual Machine is allowed to occupy. If *MEM_MAX* and *MEM_FREE* have the same value, this means that no more memory is available apart from the free disk space reported under MEM_FREE. In itself this is not a problem. However, if the capacity MEM_FREE is almost used up, the Virtual Machine will be increasingly frequently occupied with garbage collection, which can have a serious negative impact on the system performance. In this case, restarting the server (with a possible modification of the memory configuration in `$BICSUITECONFIG/java.conf`) is the lesser evil.

STARTTIME

The *STARTTIME* field shows the time when the server was started.

UPTIME

The *UPTIME* field is for convenience only and shows how long the server is already running.

HITRATE

The *HITRATE* field shows the effectiveness of a caching algorithm in the Resource Scheduling thread. Since this cache only demonstrates how effective it is in case of a complete reschedule, even relatively low values will give an indication of the effectiveness.
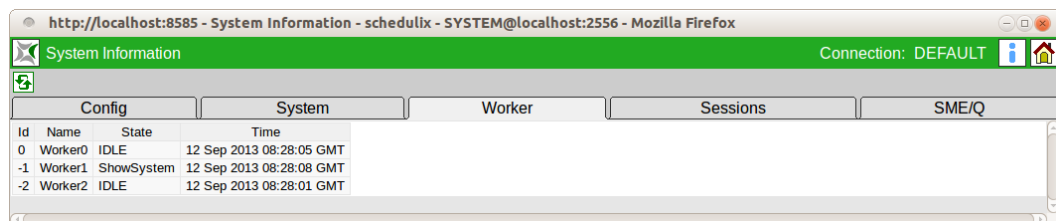
JdbcDriver

The entry under "JdbcDriver" was only included in this tab for the sake of completeness because the other Java-related configuration parameters are given here as well. This is the same value as the one in the "Config" tab.

Other entries

All other entries are dependent upon the hardware, operating system and Java Virtual Machine that you are using. Although in principle the values are relatively easy to interpret, you should refer to the Java documentation for a more detailed description of their meanings.

24.2.4 Worker tab

Figure 24.4: Worker thread information

The activity of the Worker threads is shown in a table in the "Worker" tab. The worker with the ID 0 is always the worker responsible for processing the write transactions. All other workers are responsible for processing the read transactions. If all the Worker threads are regularly active, this indicates that there are too few of them and their number should be increased.

Id

The worker's internal number is shown here.

Name

The worker's name is shown here.

State

The *State* column shows what the respective worker is doing at the moment. The entry here is either IDLE, if the worker has nothing to do, or the class name of the object which he is currently processing. Normally, the type of statement can be easily derived from the name of the class.

Time

This column shows the start time of the most recently executed statement or the statement that is being executed.

24.2.5 Sessions tab

Figure 24.5: Session information

A table of all the currently connected sessions and their activities is shown in the "Sessions" tab.

This

In this column, an asterisk (*) indicates which of the displayed sessions is the user's own session.

Session Id

This column shows the number of the session. This is required if you want to terminate the session using the "KILL SESSION" command.

Port

This column shows the port via which the session is connected with the server.

Start

This column shows the time when the session was started.

Type

This column shows what type of client the connection is. The possible values are USER, JOBSERVER or JOB.

User

The user's name is shown here. In the case of a user, this is the username. If it is a jobserver, this is the server's fully qualified name. The JobId is shown for a job.

UserId

This column shows the object Id of the user, jobserver or job.

IP

This column shows the IP address of the computer from which the connection has been set up.

TxId

This column shows the number of the most recently performed transaction. If a statement is just being executed, it is the number of active transaction.

Idle

This column shows the number of seconds that have elapsed since the last activity. Ideally, for jobservers this value should not be higher than their NOP_DELAY.

State

This column shows the state of the session.

Wert	Meaning
IDLE	The session is waiting for user input.
QUEUED	The session has a statement in the queue that is ready to be executed
ACTIVE	A statement in the session is currently being executed.
COMMITTING	A statement in the session is in the final phase of being executed.
CONNECTED	A session has been initiated but the user has not yet provided authentication.

Table 24.3: Session Status

Timeout

This column shows the number of seconds of idle time that can elapse before the server connection is terminated. A value of 0 means that the server connection will not be terminated.

Statement

This column shows the statement that is currently being executed by the connection provided, of course, the connection is not IDLE. It goes without saying that passwords in Connect statements are made unrecognisable.
On the other hand, the statement is only visible to users with administrator privileges.

24.2.6 SME/Q tab

The "SME/Q" tab shows a table of the Submitted Entities created in each quarter. This tab is only visible if the user belongs to the "ADMIN" group.

Year

This column shows the year.

Quarter

This column shows the quarter.

Total

This column shows the total number of Submitted Entities which were created in this quarter.

Expected Total

This column contains for the current and not yet completed quarter (1st row in the table) the extrapolated expected number of Submitted Entities by the end of the quarter. This number is not binding and is intended solely for orientation purposes. For all previous quarters, the expected total is the same as the total.

Avg. SME / Day

This column the average number of Submitted Entities created each day.

25 Shell

25.1 Image
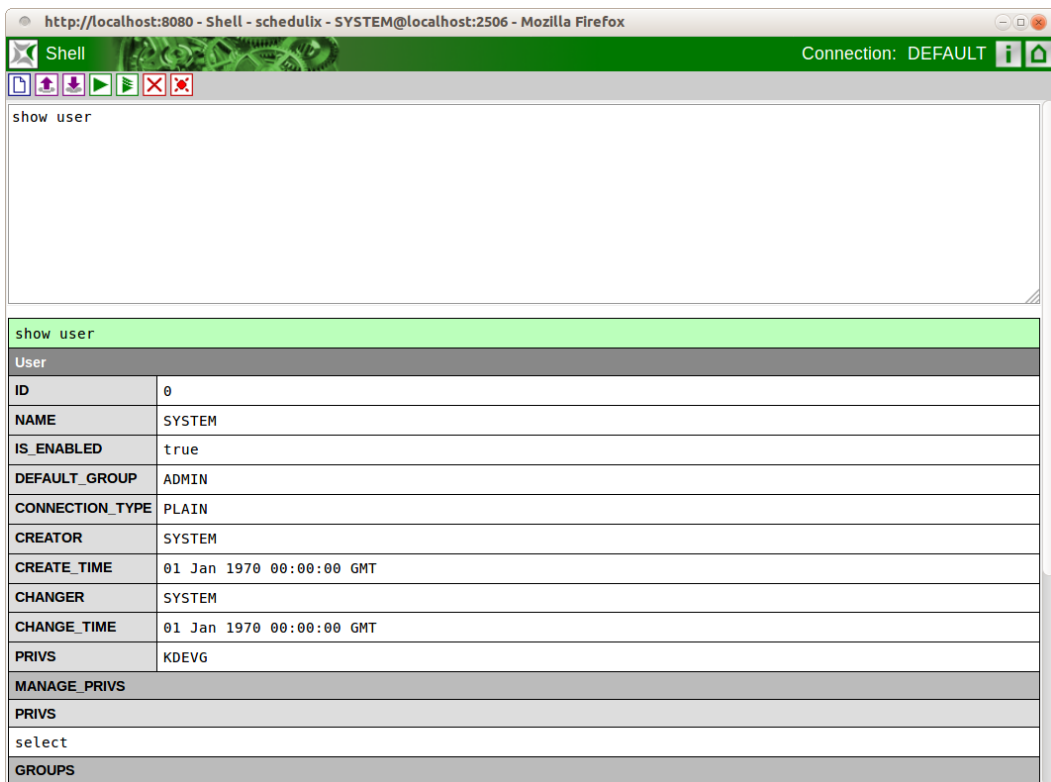
Figure 25.1: Shell

25.2 Concept

25.2.1 Brief description

API commands can be executed from the web front end with the ″Shell″ menu item in the main menu.

25.2.2 Detailed description

The shell window comprises a button bar, an input field for API commands, and an area below the command input field where the executed command is outputted.

25.2.3 Buttons

The following list describes the buttons in the window and their functions:

Delete Entry

The command input field is cleared.

Load

Read in commands from a file.

Save

Write the commands to a file.

Execute

Executes the command over which the cursor is hovering. If a text has been selected, it is executed.

Execute as script

Executes all commands in the input field.

Delete all outputs

All results of commands outputted below the input field are deleted.

Auto clear off

Deactivates the automatic deletion of outputs before commands are executed.

Auto clear on

Activates the automatic deletion of outputs before commands are executed.

Index