WHITESTEIN
Technologies

Living Systems® Process Suite

Management

Living Systems Process Suite Documentation

3.3
Mon Nov 1 2021

Whitestein Technologies AG | Hinterbergstrasse 20 | CH-6330 Cham
Tel +41 44-256-5000 | Fax +41 44-256-5001 | http://www.whitestein.com

Contents

Chapter 1

Management

LSPS provides the management tools to manage the runtime data, LSPS server components and their resources:

- the repository with modules,
- model instances,
- persons, users of the management tools,
- person's roles, which define the way they participate in model execution,
- persons' security rights governing access to management actions,
- the to-dos generated by model instances,
- documents saved by persons,
- logs of model instances
- exceptions that occurred in model instances
- webservice wait points of model-instances,
- server and database settings

To allow you to manage the data efficiently, LSPS comes with three management tools.

Chapter 2

Management Tools

LSPS comes with the following management utilities:

- Management perspective, a dedicated perspective in the PDS development environment, convenient for management during development of modules in PDS since it is integrated in PDS and allows you to work directly with the workspace resources;
- Management Web Console, a web-based front-end application for quick management and tweaks on servers without the need to access local resources;
- Command Line, command-line management tool for automation and bulk operations.

2.1 Management Perspective

Management perspective is a PDS perspective that contains views and tools that allow you to communicate with the LSPS Server and manage its resources.

The data and management tools are available in the management views, which you can display by clicking the Management Views button () in the main toolbar. The data in the views is loaded, when the view opens. Make sure to refresh the views so you work with the most recent data (click the Refresh () button in the toolbar of the view).

Mind that you can benefit from the features provided by perspectives, such as, displaying any views in the perspective (**Window** -> **Show View** -> **Other**), detaching views, filtering data, etc.

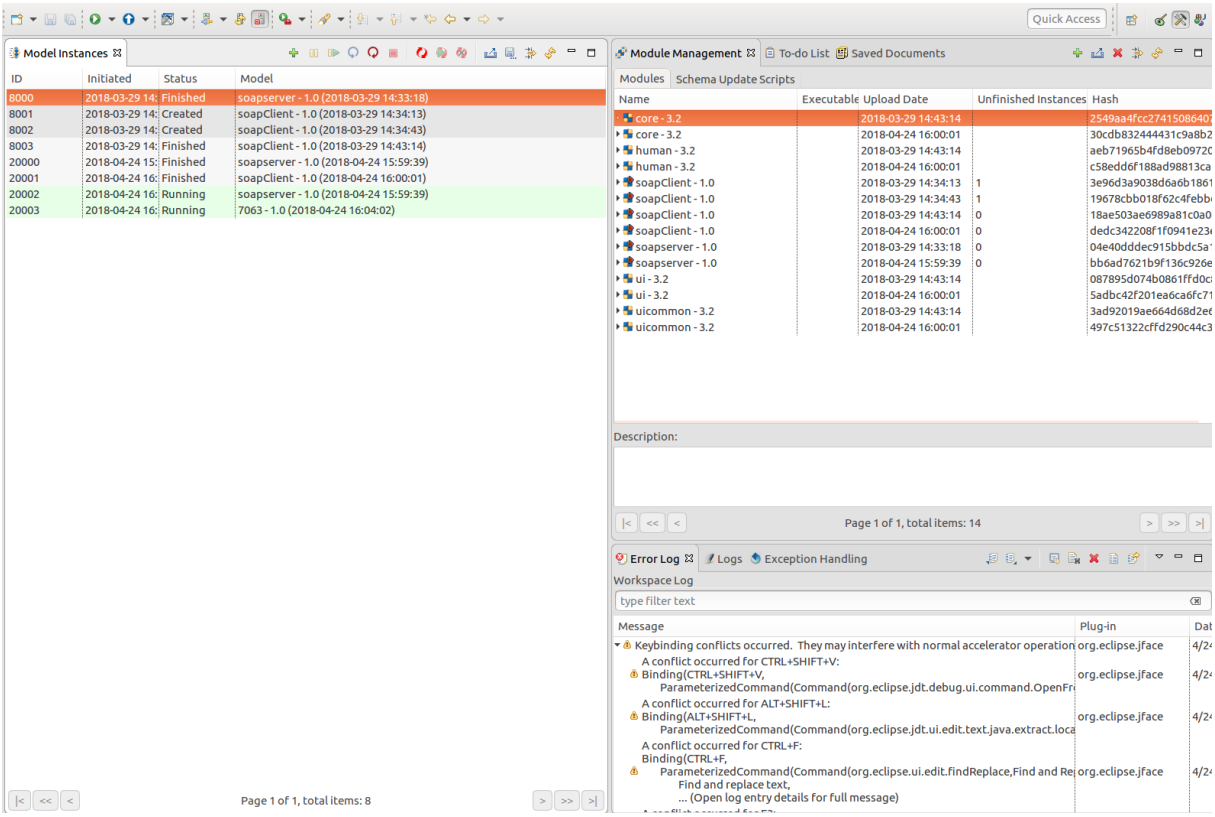
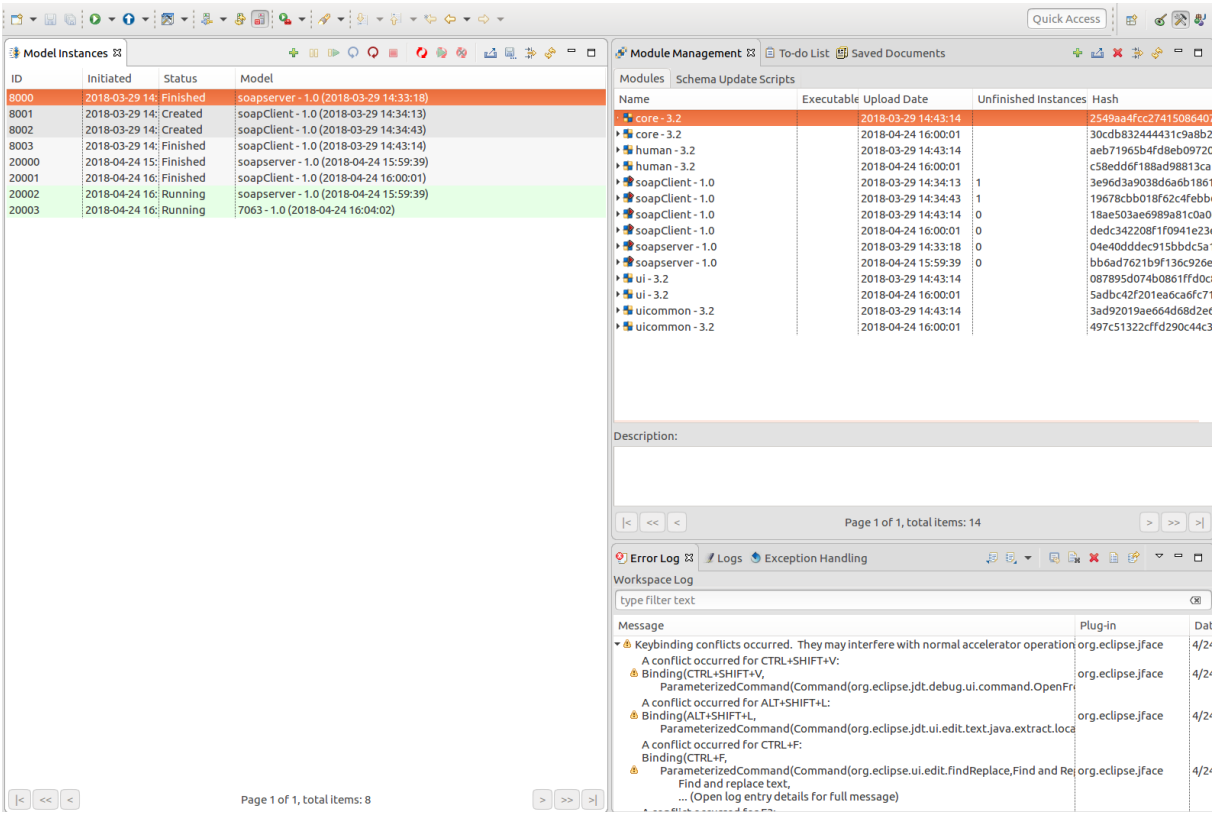


Figure 2.1 Management perspective

2.1.1 Accessing the Management Perspective

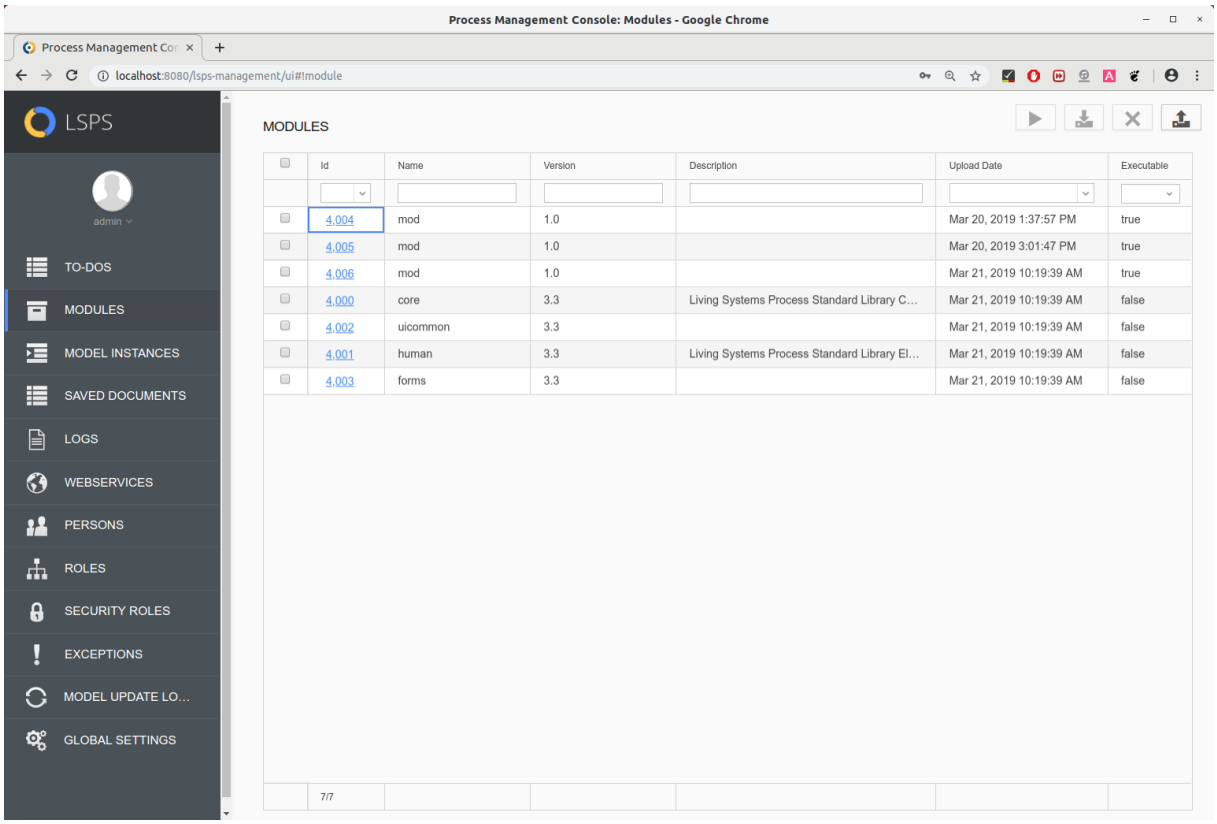
To access the perspective, click the perspective button  or go to **Window** > **Perspective** > **Open Perspective** > **Management**.



2.2 Management Console

Living Systems Process Management Console is a web-based front-end application intended for management of model instances, LSPS users, and the LSPS front-end applications, that is, the *Management Console* itself and the default or customized *Application User Interface*.

The tool is primarily intended for the administrators of the customized *Application User Interace* and the LSPS server. However, also process designers and application developers might benefit from its features.



2.2.1 Accessing Management Console

Note that you need to have the security right `Management:Login` assigned to be able to access Management Console.

You can access Management Console from a supported Web browser:

1. Launch your web browser.
2. Go to the URL of the console.

   By default, the URL is `http://$DOMAIN/lsps-management`.
3. On the login page, enter your credentials.

To log out, click the drop-down icon next to your user name and in the context menu, click **Log out**.

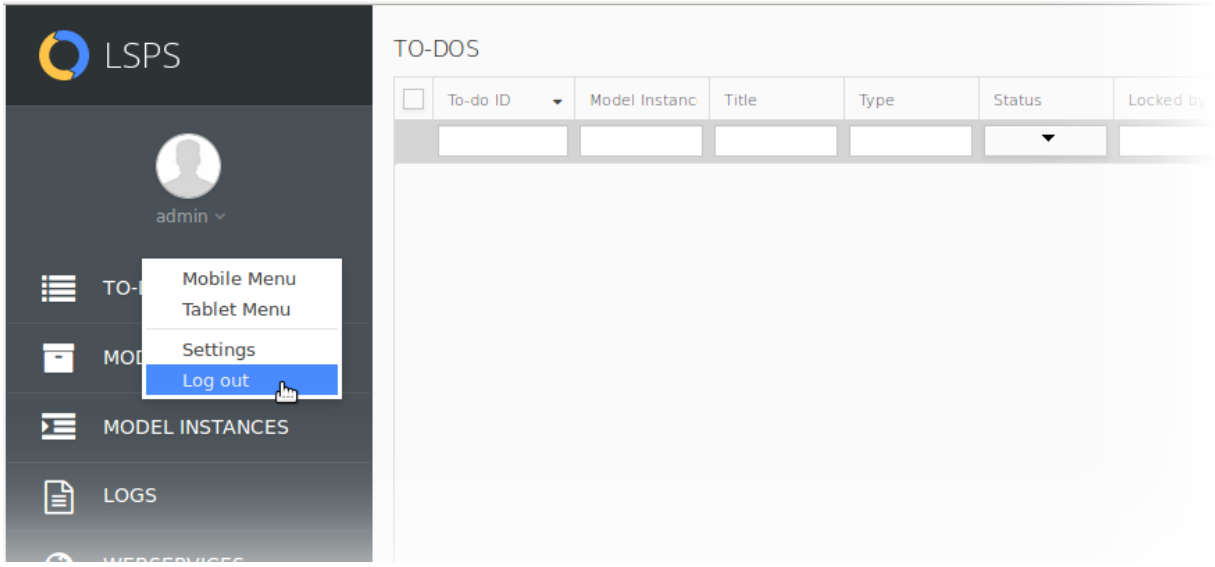


Figure 2.2 Logging out

Related topics:

- Navigating
- Application Settings

2.2.2 Navigating

To navigate through views of Management Console, use the navigation toolbar located on the left. The relevant content is displayed in the area on the right.

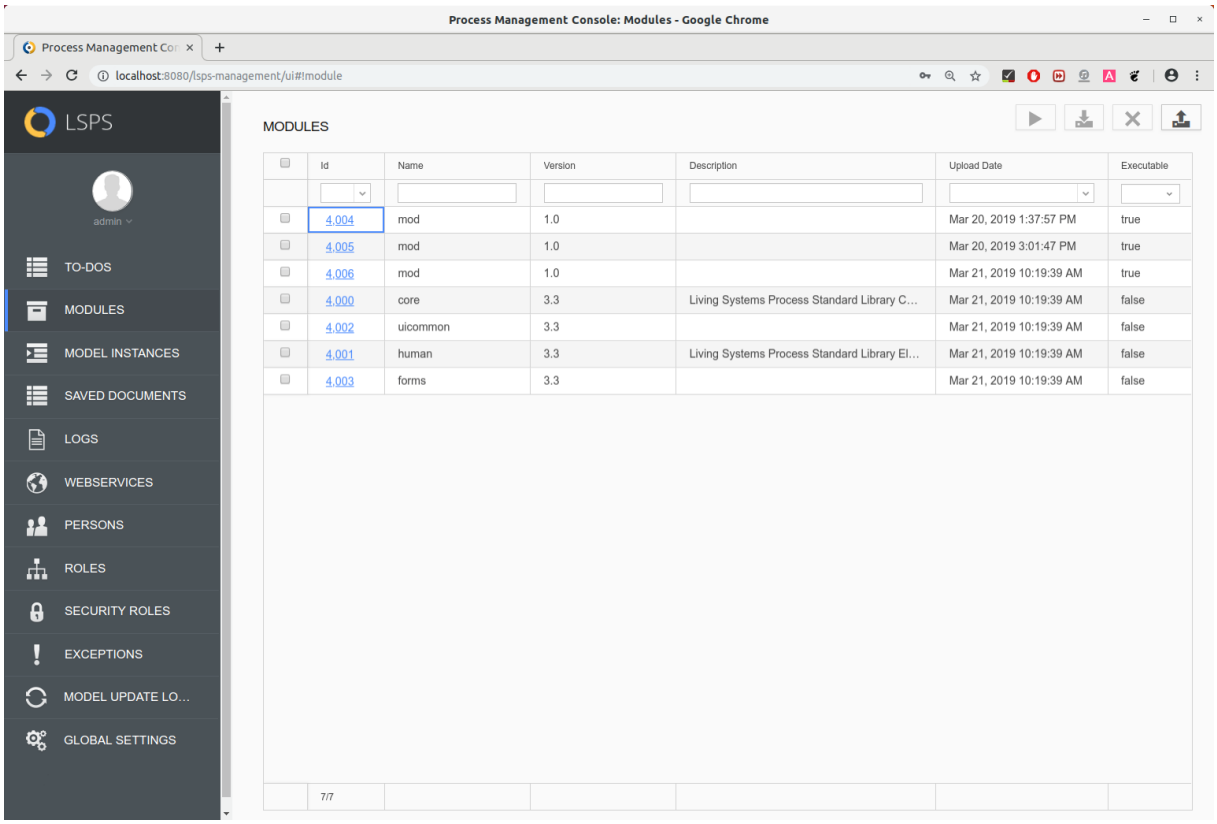


Figure 2.3 Management Console environment

2.2.2.1 To-dos View

The To-dos view contains a list of all to-dos dispatched by the Navigation Engine and their details.

The to-dos are listed in a table with the following data:

- To-do ID: unique to-do ID
- Model Instance ID: ID of the parent model instance (the to-do was generated by a User Task of one of the module instances of this model instance)
- Title: todo title defined by the title parameter of its User Task
- Type: name of the User Task that generated the to-do
- Status: status of the User Task (for information on the task life cycle, refer to GO-BPMN Modeling Language Guide)
- Locked by: name of the person, who has locked the to-do
- Performers: the expression used to obtain the users to whom the to-do was initially allocated
- Interruption Reason: reason why the to-do was terminated, such as, the parent model instance was terminated
- Task Namespace: full path of the User Task
- Issue Date: date when the to-do was issued

- Submitted: date when the to-do was submitted (became *finished*)

From the view, you can search for orphaned to-dos, escalate and reset alive to-dos, and display to-do details.

You can also upload new modules and access module details.

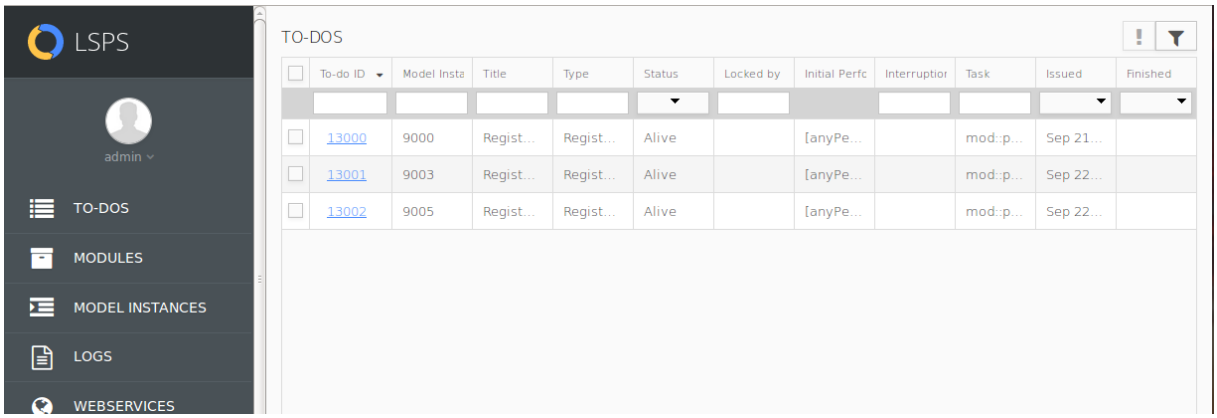


Figure 2.4 To-dos page

2.2.2.1.1 To-Do Detail

To display a To-do Detail, click a to-do ID or name in the To-do List or on the To-do page.

The To-do Detail page contains the following to-do information:

- To-do ID: unique to-do ID
- Model instance ID: ID of the model instance that produced the to-do
- External link: link to the to-do content
- Title: task title
- Task: namespace path and name of the to-do's user task
- Status: task execution status
- Issued: date and time, when the to-do was generated
- Initial Performers: expression used to acquire the initial potential performers for the to-do
- Locked by: person who is currently handling the todo
- Assignees: users who have currently the right to work with the to-do unless the to-do is locked by a user
- Finished: date and time, when the to-do was submitted or interrupted
- Delegation level: current level of delegation

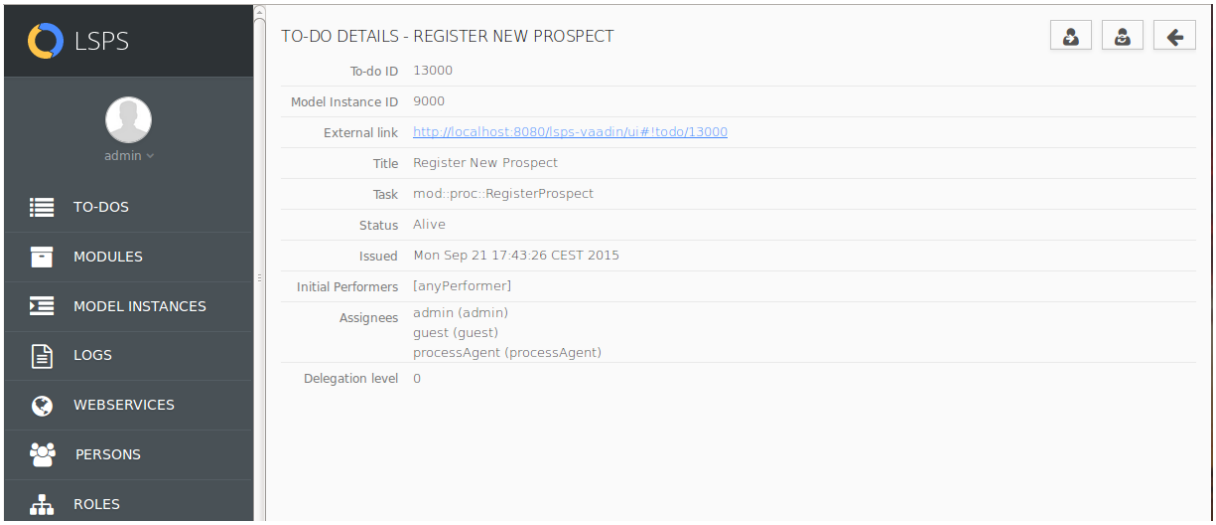


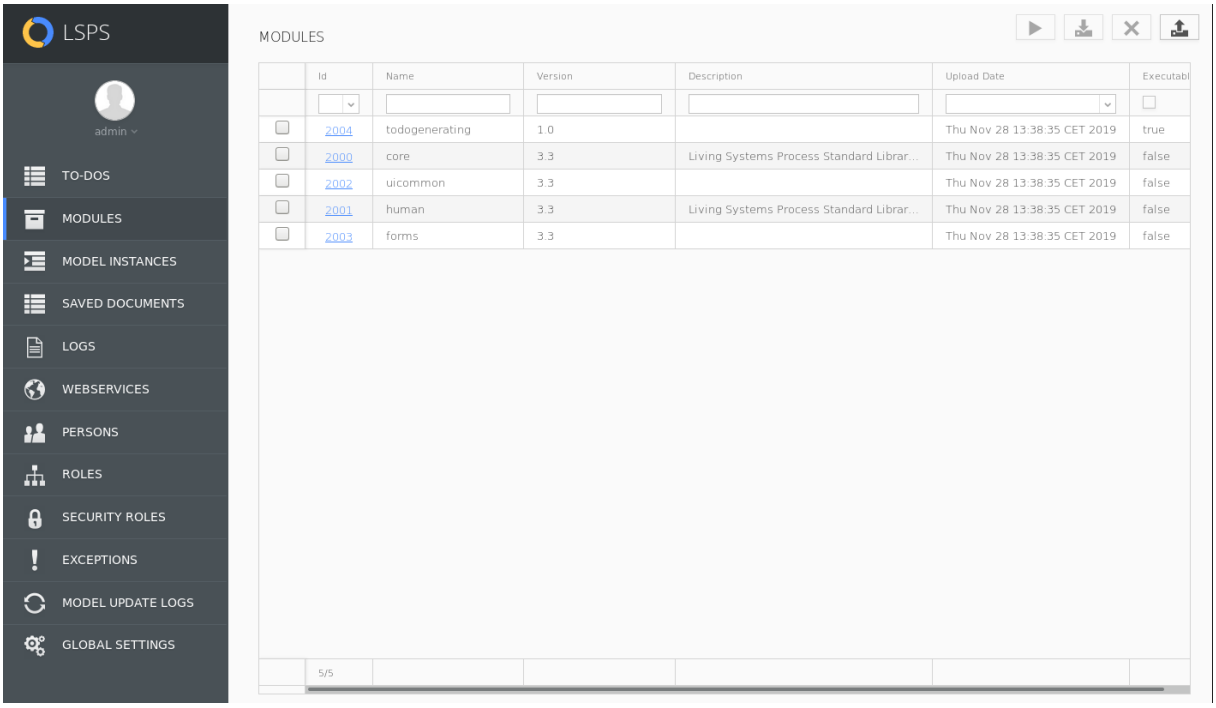
Figure 2.5 To-Do detail

From the to-do detail, you can delegate the to-do or undo a delegation, reassign to-dos, export the state of a saved todo to XML, import an updated state of a saved to-do in an XML.

2.2.2.2 Modules View

The Modules view contains a list of modules uploaded on the LSPS Server. From the view, you can perform module-related actions, including upload, unload, and instantiation.

You can also upload new modules and access module details.



2.2.2.3 Model Instances View

The Model Instances view contains a list of all model instances in the LSPS Server with their details:

- ID: unique ID of the model instance
- Model ID: ID of the underlying model
- Model name: name of the underlying executable module
- Model version: version of the underlying executable module
- Started Date: time when the model instance was created
- Finished Date: time when the model instance finished
- State: current execution `status of the model instance`

You can filter the table content by entering the filter values in the respective header cell and pressing ENTER.

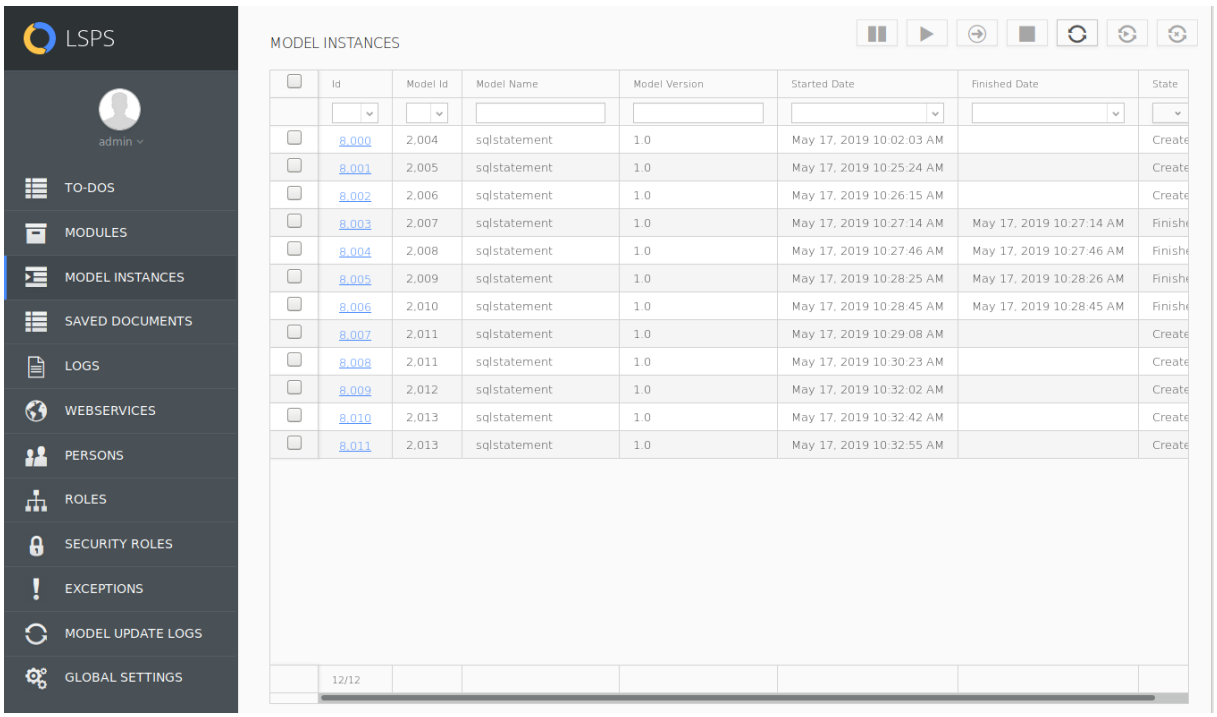


Figure 2.6 Model Instances

Selected model instances can be managed using the buttons in the view toolbar.

2.2.2.3.1 Model Instance Detail

The *Model Instance Detail* view contains multiple tabs with detailed information about the model instance.

To open a detail of a model instance, click the model instance ID in the Model Instances view.

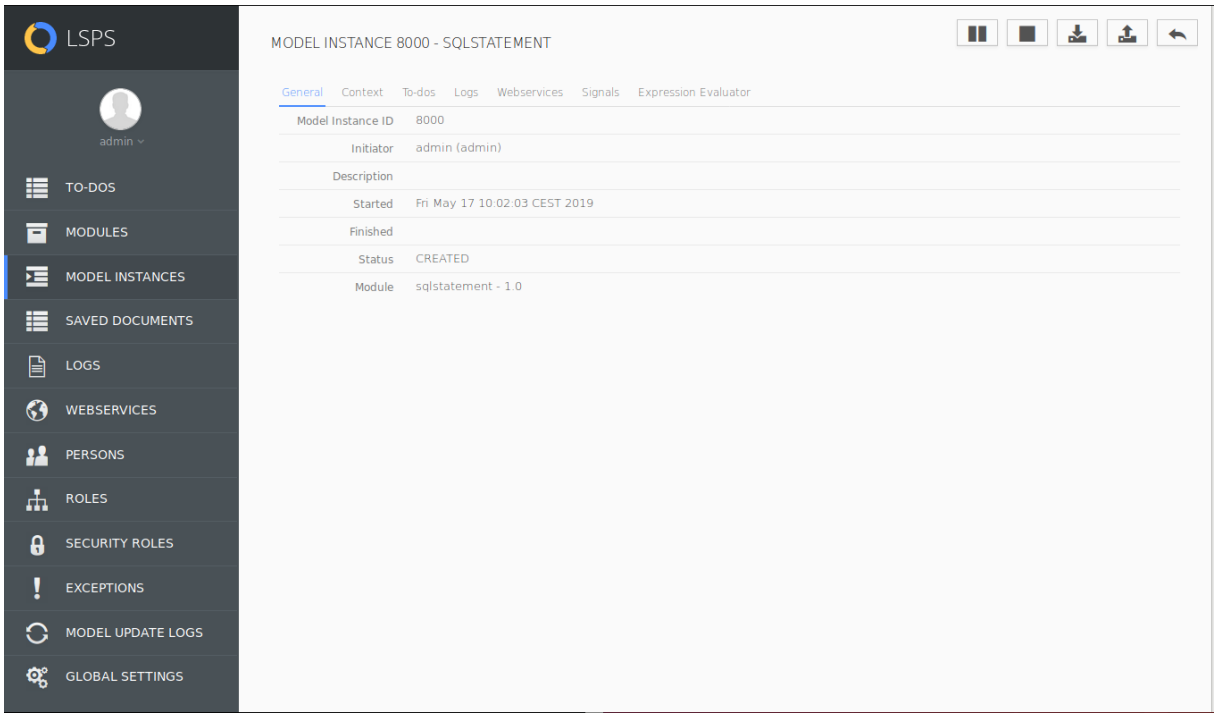


Figure 2.7 Model Instance Detail

General Tab

The General tab contains general information about a model instance.

- Model Instance ID: unique model instance ID
- Initiator: person, who created the model instance (passed on model instance creation as `InitiatorId`)
- Description: free-text description entered when the instance was created
- Started Date: date and time of model instance creation
- Finished Date: date and time of model instance termination
- Status: current execution status of the model instance
- Module: name and version of the executable module that was used to create the model instance

From the tab, you can suspend and finish the model instance, and export and import its state as XML.

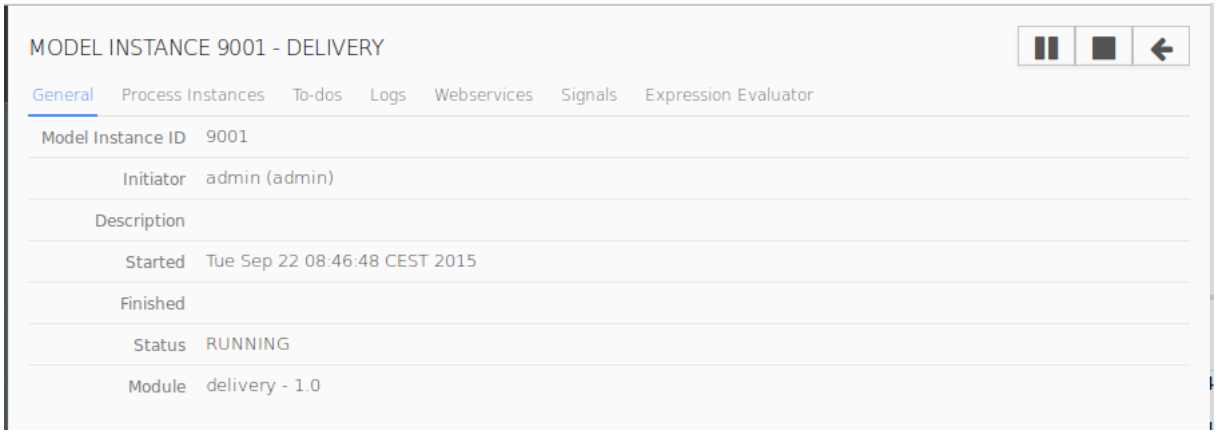


Figure 2.8 The General tab in a Model Instance detail view

Context Tab

The *Context* tab contains the runtime data of the model instance with its child contexts. The context are visualized in a tree structure with the root being the model instance. The details about individual elements are displayed in their respective rows.

From the tab, you can export and import its state as XML.

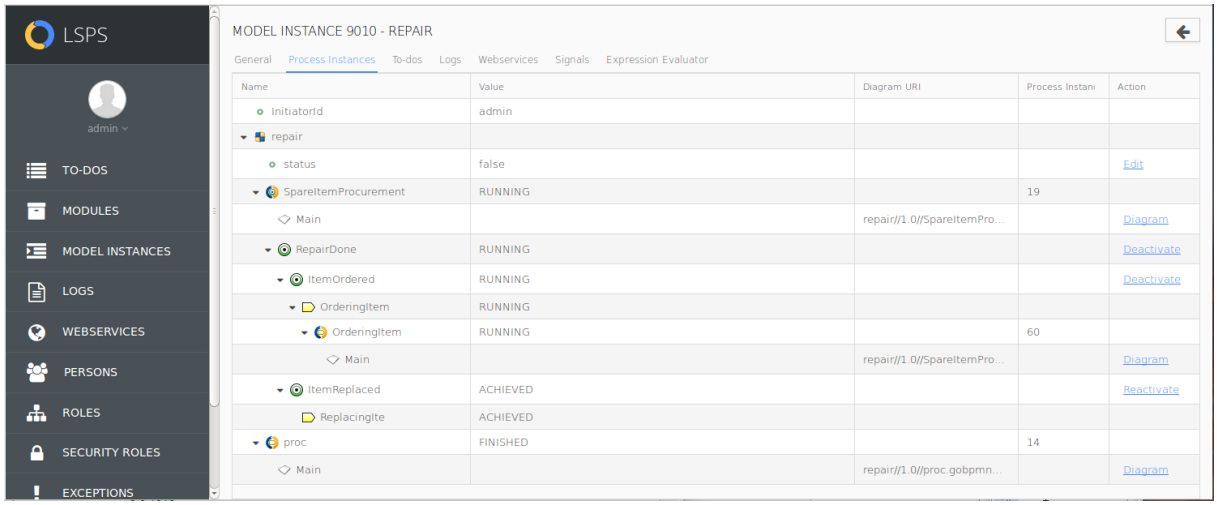


Figure 2.9 Context tab

To-dos Tab

The To-dos tab contains all to-dos generated by the respective model instance and their details.

- ID: unique to-do ID number
- Title: title defined by the title parameter of the underlying user task

- Type: name of the user task
- Status: to-do status (`Alive`, `Accomplished`, or `Interrupted`)
- Performers: names of initial performers
- Interruption reason: reason why the to-do was finished without completion if applicable (for example, due to process deactivation or model update)
- Task Namespace: task path and task name (mostly `module name::process name::plan name::task name`)
- Issued Date: date when the to-do was issued
- Submitted Date: date when the to-do was finished

Clicking a to-do ID opens the to-do detail. You can filter for any orphaned to-dos from this tab: click the Filter button in the upper right corner and define the filter properties.

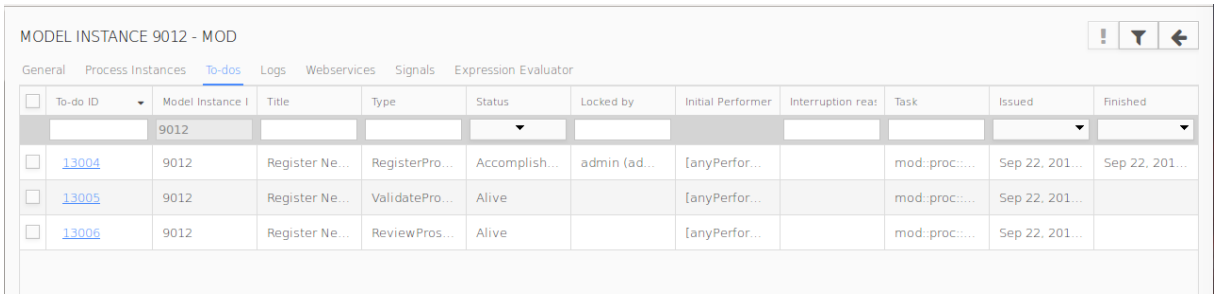


Figure 2.10 To-dos Tab

Logs Tab

The Logs tab contains a table with messages logged by Log tasks, tasks of the Log task type, executed by the model instance.

For each log entry, the table shows:

- Log ID: unique log ID
- Date: date when the message was logged
- Level: severity of the log message defined as an integer value on the Log task
- Description: log message content

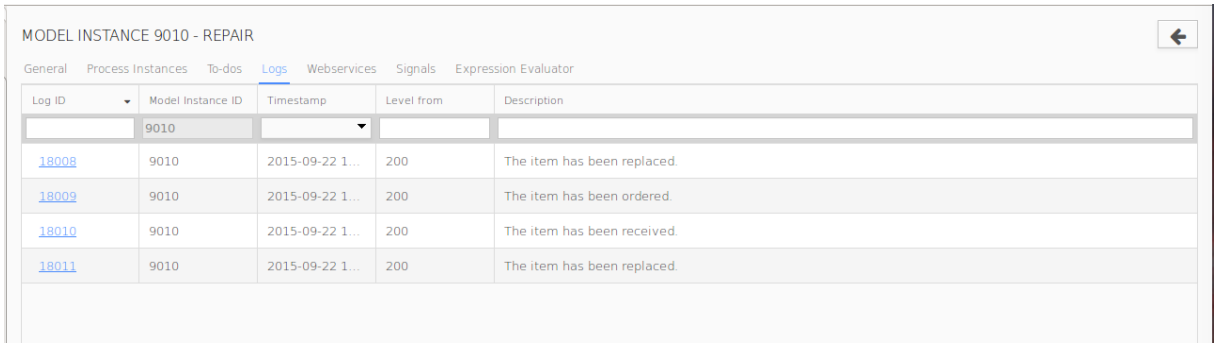


Figure 2.11 Logs tab

Web Services Tab

The Web Services tab contains a list of active and processed wait points (waiting for a web-service request from a client) of a model instance.

The Web Services tab contains a table with all wait points of web service clients in the model instance:

- ID: unique wait-point ID
- Operation URI (Unified Resource Identifier): identifier of the resource (if several wait-point requests are processed by the same task type, value of their operation URI is identical. However, their wait-point IDs differ)
- Created: time when the wait point was created
- Request: link to the XML received from the client
- Request Received: time when the client request was received
- Response: link to the XML sent to the client and information on the response type (`time-out`, `error`, `finished`, `terminated`)
- Finished: time when the Web service response was sent
- Response timed out: if the response time was exceeded
- Terminated: if the wait point was erased due to parent-element termination
- Error response: if the received response was an error response

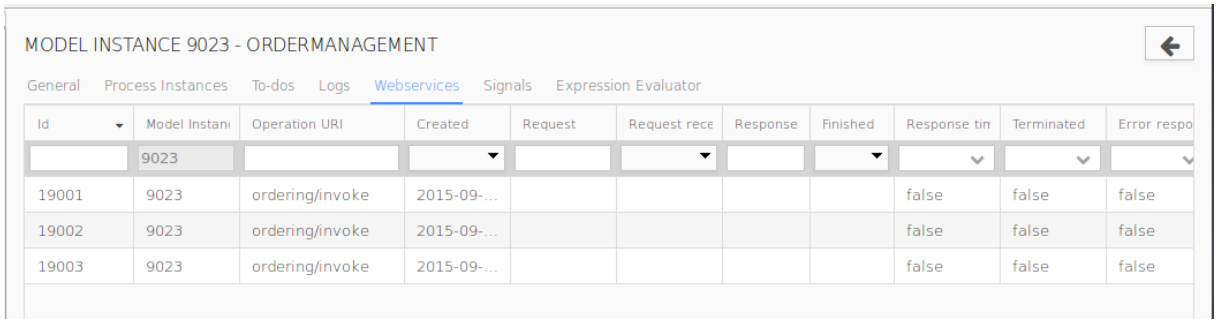


Figure 2.12 Webservices Tab

Signals Tab

The Signals tab contains a list of signals which were produced by a module instance and were not consumed.

Waiting signals may be removed.

Figure 2.13 Signals tab

Expression Evaluator Tab

The Expression Evaluator tab enables you to evaluate expressions in the `Expression Language` in a context of the model instance.

By default, the evaluated expressions have no impact on the persisted data, that is, on shared Record instances. If you want to persist the outcome of the expressions, select the **Persistent** option.

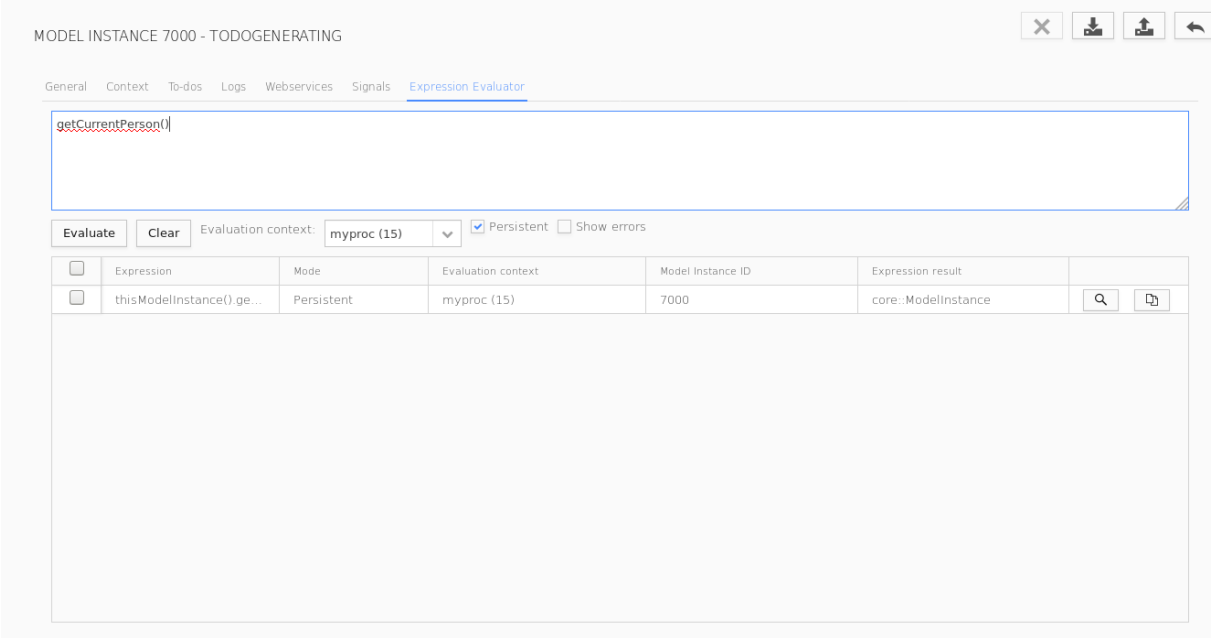


Figure 2.14 Expression Evaluator

Show Errors enables you to display detailed error messages in case some of the expression could not be evaluated.

2.2.2.4 Saved Documents View

The *Saved Documents* view lists the documents the users have saved and are hence probably working on. You can export the documents as XMLs and import the modified documents back. The export and import features are available only to users with the security right `Document:Write`.

2.2.2.5 Logs View

The Logs view contains the messages logged by Log Tasks, tasks of the Log task type, executed by model instances.

For each log entry, the following details are displayed:

- Log ID: unique log ID
- Model Instance ID: ID of the parent model instance of the Log task
- Date: date when the message was logged
- Level: severity of the log message

- Description: log message content

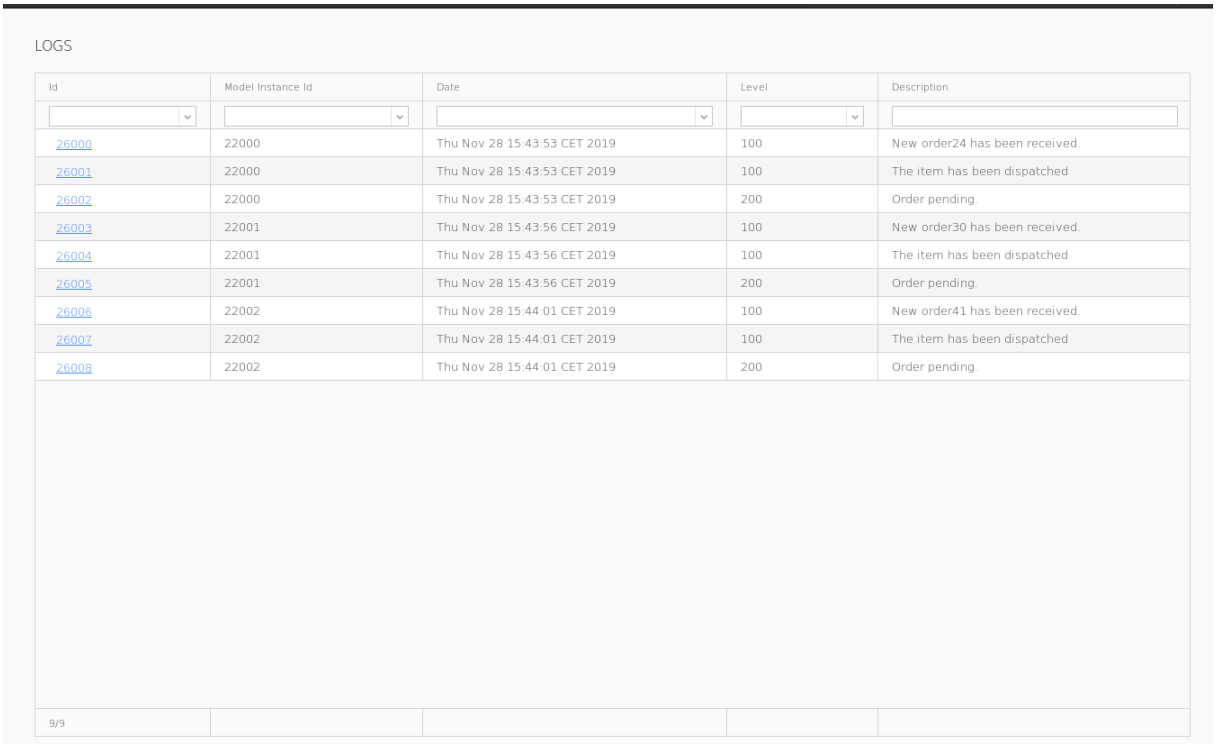


Figure 2.15 Logs page

2.2.2.6 Web Services View

The Web Services view contains a list of active and processed wait points waiting for a Web service request from a client with the following details:

- ID: unique wait point ID
- Model Instance ID: ID of the model instance processing the Web service request
- Operation URI (Unified Resource Identifier): identifier of the resource

  Note: If several wait-point requests are processed by the same task type, value of their operation URI is identical. However, their wait-point IDs differ.

- Created: time when the wait point was created
- Request: link to the request XML received from the client
- Request Received: time when the client request was received
- Response: link to the response XML sent to the client and information on the response type (`time-out`, `error`, `finished`, `terminated`)
- Finished: time when the Web service response was sent
- Timed Out: selected if the response time limit was exceeded

- Terminated: selected if the wait point was erased due to parent element termination
- Error: selected if the received response was an error response

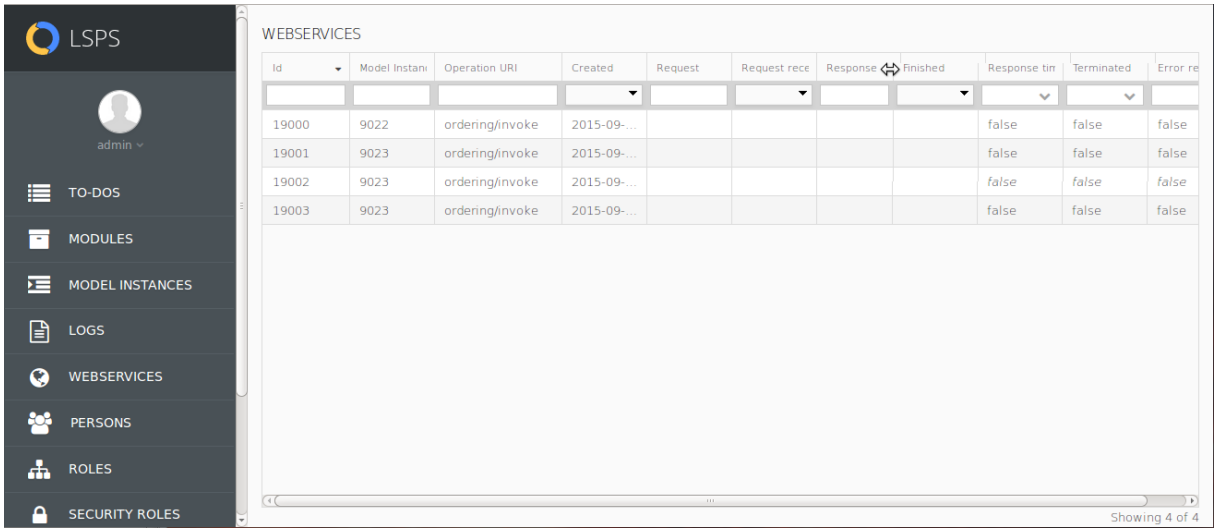


Figure 2.16 Web Services view

2.2.2.7 Persons View

The Persons view shows a list of persons, that is, LSPS users, as defined on the LSPS server.

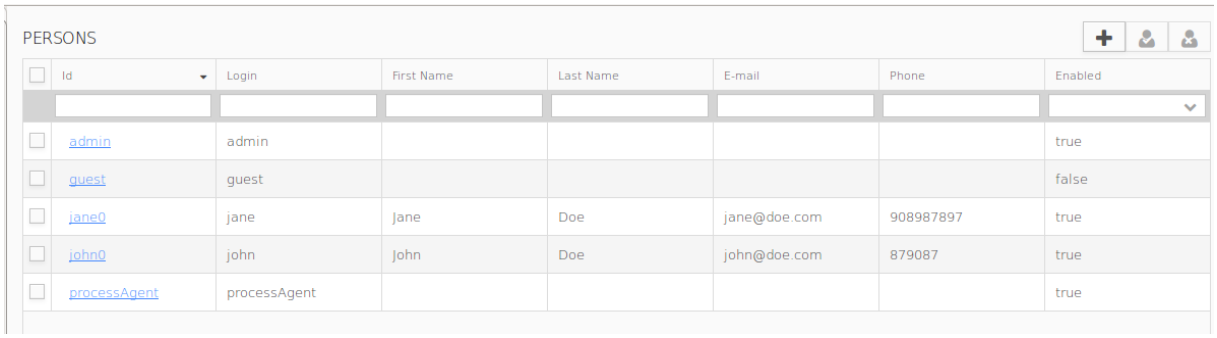


Figure 2.17 Persons View

By clicking a person id, you can display the person detail.

Using the buttons in the view toolbar, you can add a new person, disable selected persons, or enable selected persons.

2.2.2.8 Person Detail View

The Person Detail view shows details of a person including their substitution setting, security and modeled roles.

The detail appears after clicking a person id in the Persons or Roles view.

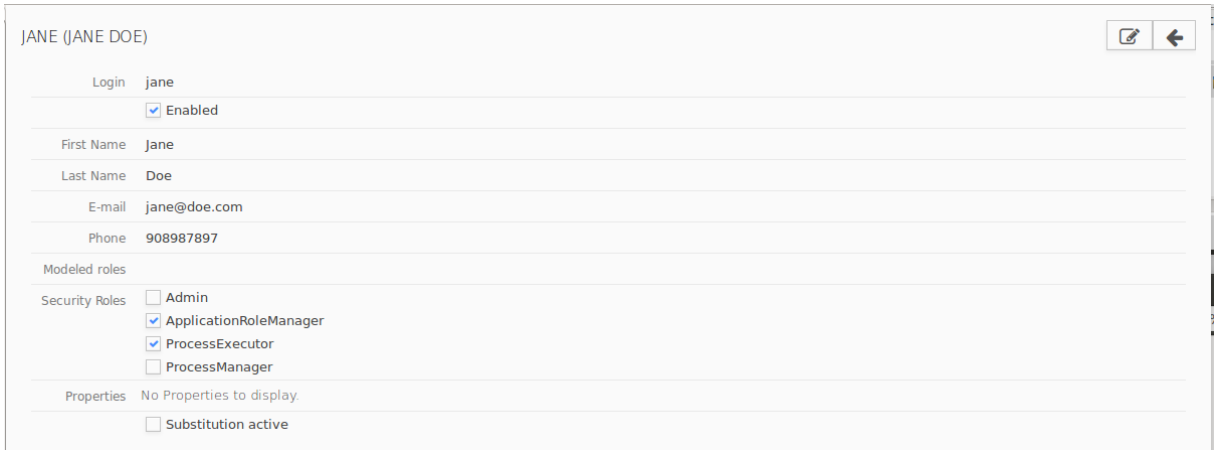


Figure 2.18 Person Detail view

2.2.2.9 Roles View

The Roles view contains lists of runtime and modeled roles created based on the organization definitions of uploaded modules.

If a runtime role is selected, the persons with the role are shown in a table on the right.

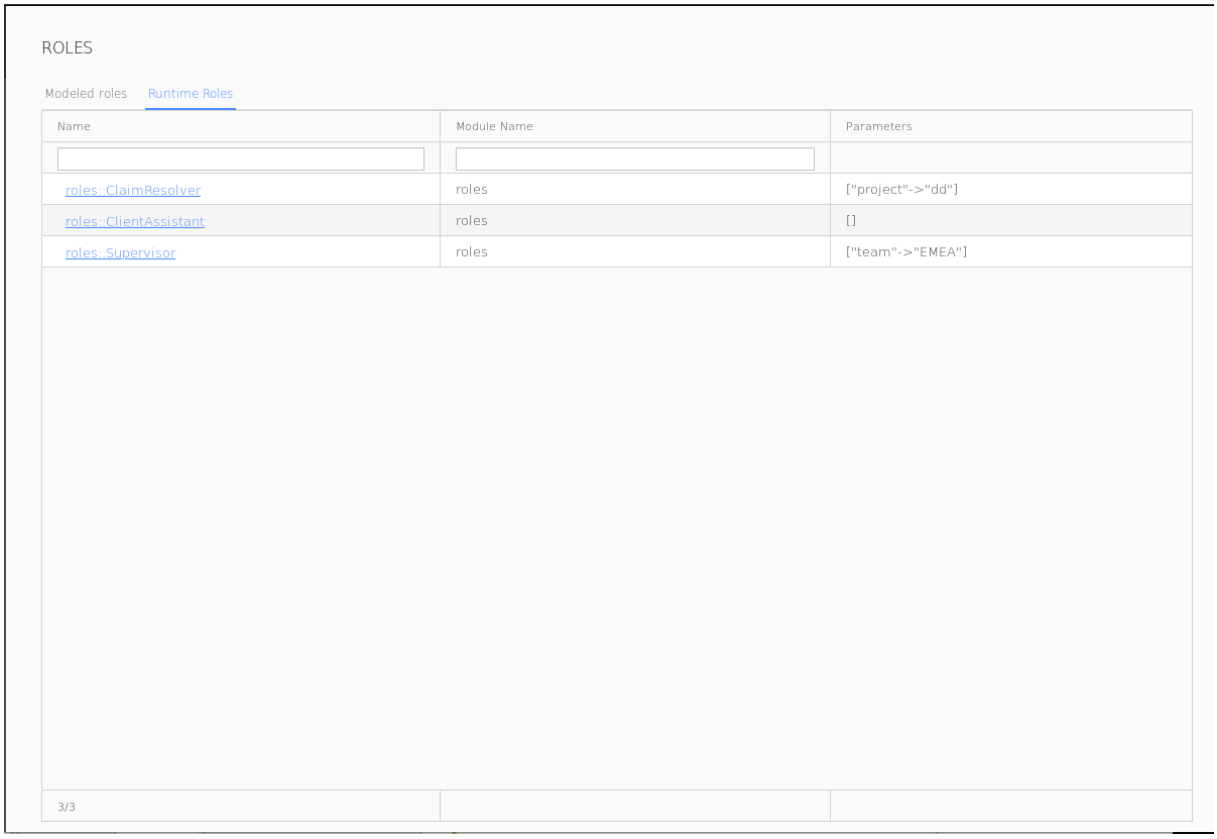


Figure 2.19 Roles view with runtime and modeled roles

2.2.2.10 Security Roles View

The Security view contains the list of available security roles, which restrict the availability of the feature a user can use in the application and Management Console itself. Each security role defines a particular set of security rights: A security right is the right to perform an action. A security role is a grouping of security rights.

Figure 2.20 Security view

2.2.2.11 Exceptions View

The Exceptions view contains a list of all server exceptions and their details:

- ID: ID of the exception
- Exception: first line of the exception
- Create Date: time when the exception occurred
- Resend Date: date when the resend has successfully taken place

  To resend an exception, select it in the list and click the Resend button.
- Service: component which generated the exception

To view the stack trace of an exception, click the exception id in the table.

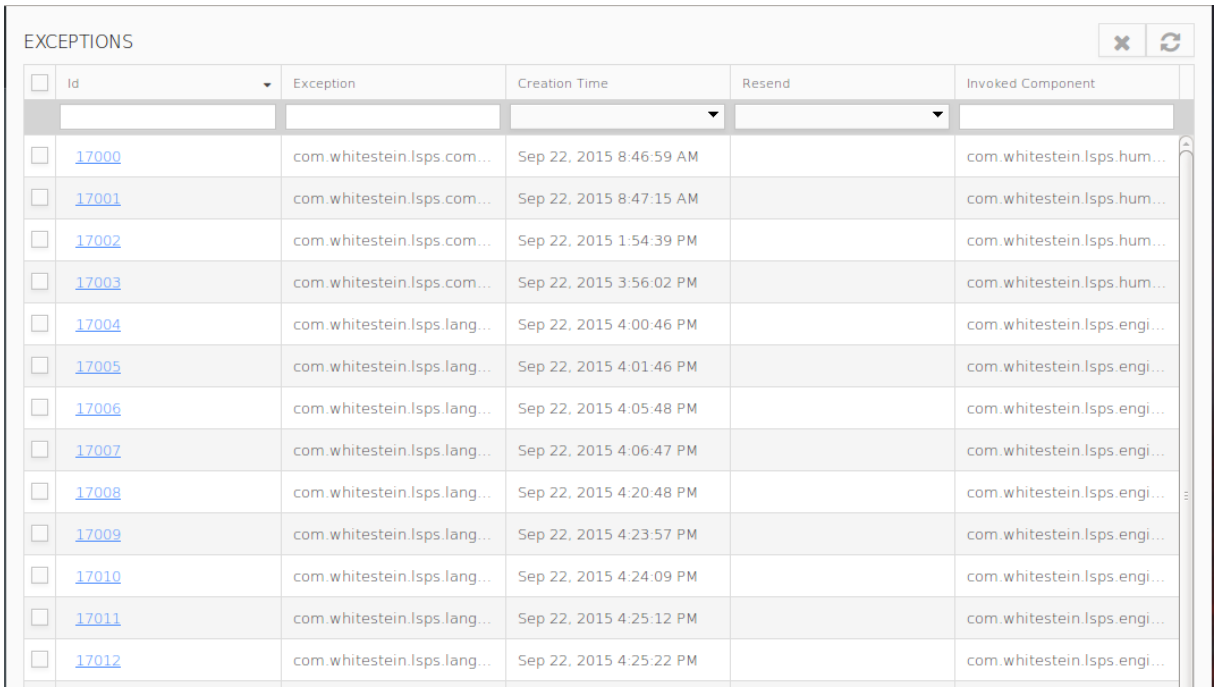


Figure 2.21 Exceptions view

2.2.2.12 Model Update Logs View

The Model Update Logs view contains a log entry for every updated model instance. You can display the log content by clicking the log ID.

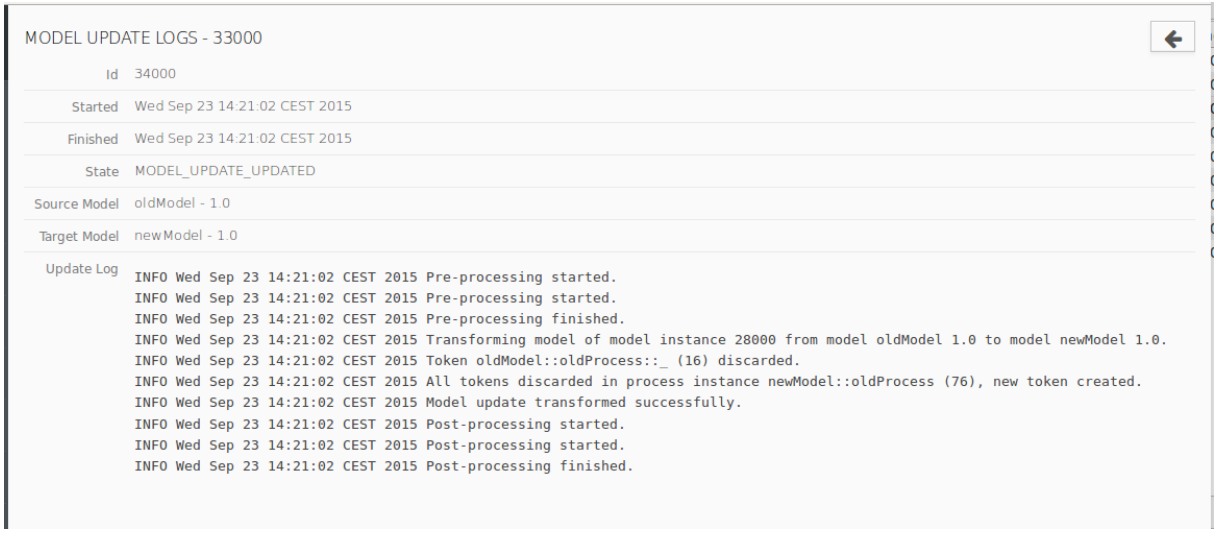
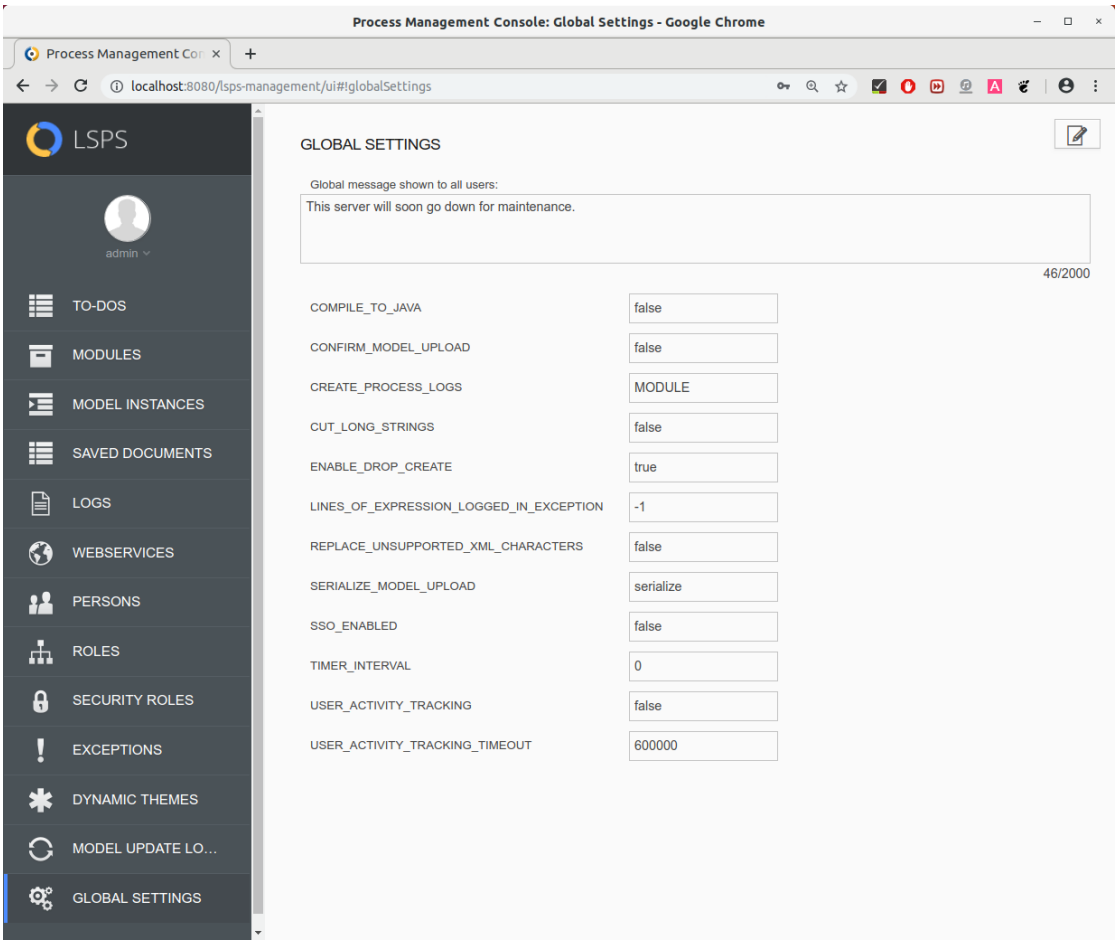


Figure 2.22 Log content

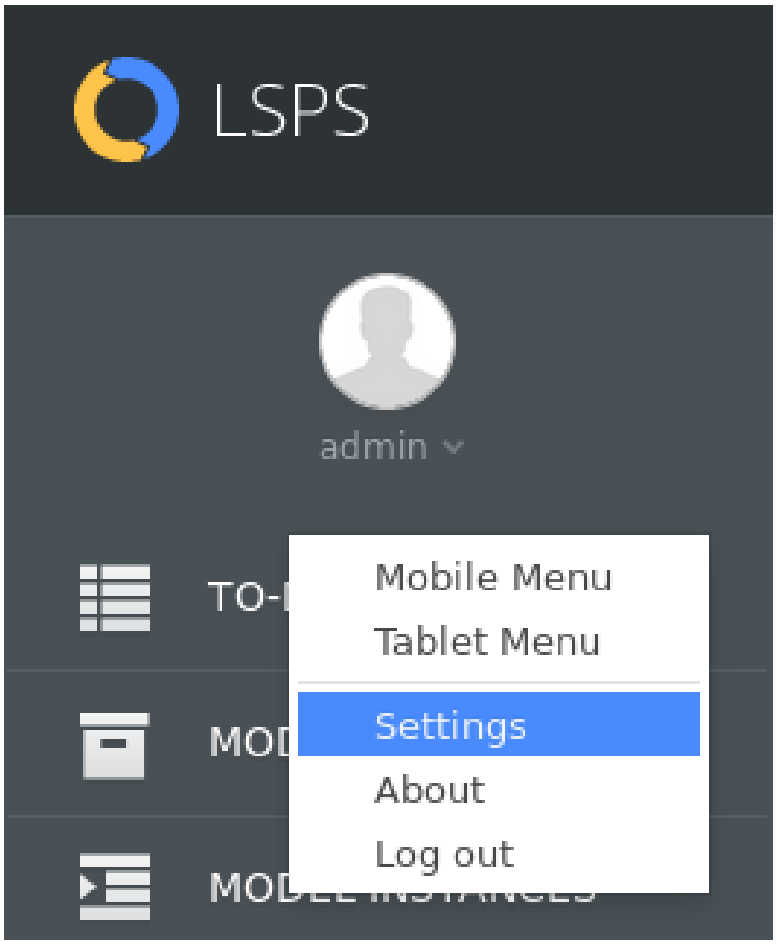
2.2.2.13 Global Settings View

In the *Global Settings* view, you can edit the `server settings` and define the global message displayed in the *Application User Interface* to all users.



2.2.3 Application Settings

Application settings accessed by clicking the user avatar and **Settings**.



2.2.3.1 Switching Languages

The application can be localized into multiple languages. For the Sample Application, German and English localization is available by default. To switch the language, click the arrow next to your user name; change the language on the Settings page and click **Save**.

2.2.3.2 Changing User Details

To edit user details, such as password, email address, phone number, and avatar, click the arrow next to your user name below the avatar and then **Settings**; change the details and click Save.

2.3 Command Line

To manage resource from the command line, you can use the cli client available in the `lsps-tools` package.

Make sure you have installed a `supported JDK` and set up the environment variable JAVA_HOME.

To run a command, do the following:

1. Open your command line interface.

2. On prompt, run *lsps-mconsole-cl* with the respective command and parameters.

To display available commands, `java -jar lsps-cli-<VERSION>-full.jar`.

The *host*, *username*, and *password* parameters take the following default values if not specified:

```
--host http://localhost:8080
--username admin
--password admin
```

Parameters of a command can be displayed with the –help parameter:

```
java -jar lsps-mconsole-cl-3.3.2083-SNAPSHOT-full.jar listSettings --help
```

Example tool run

```
~/lsps-runtime/cli-tools$ java -jar lsps-cli-<VERSION>-full.jar /
savedTodoExport -h http://localhost:8080 -u admin -p admin --todoId 12001 --location .
```

To display help, run the command without parameters, for example:

```
~/lsps-runtime/cli-tools$ java -jar lsps-cli-3.3.2039-full.jar modelUpload
Command: modelUpload
Option                    Description
------                    -----------
-h, --host (req)          host URL (e.g. http://localhost:8080)
-u, --username (req)      username
-p, --password (req)      password
-m, --model (req)         model in ZIP file or directory with model ZIP files
--dbUpdateStrategy        db schema update strategy, possible values: none, validate, update, dropCreate
```

Chapter 3

Module Management

Modules are parts of a business model that are either used to create model instances or are imported and used by other modules: they constitute a logical part of the model providing resources, related data structures, etc. used by other modules.

Modules are created in PDS. Once ready, you can upload them to an LSPS Server using one of the management tools. On upload, a module with all its module imports is uploaded to the Module Repository of the LSPS Server. Management tools allow you to manage the modules in the Module Repository: apart from uploading of modules, you can download and delete them.

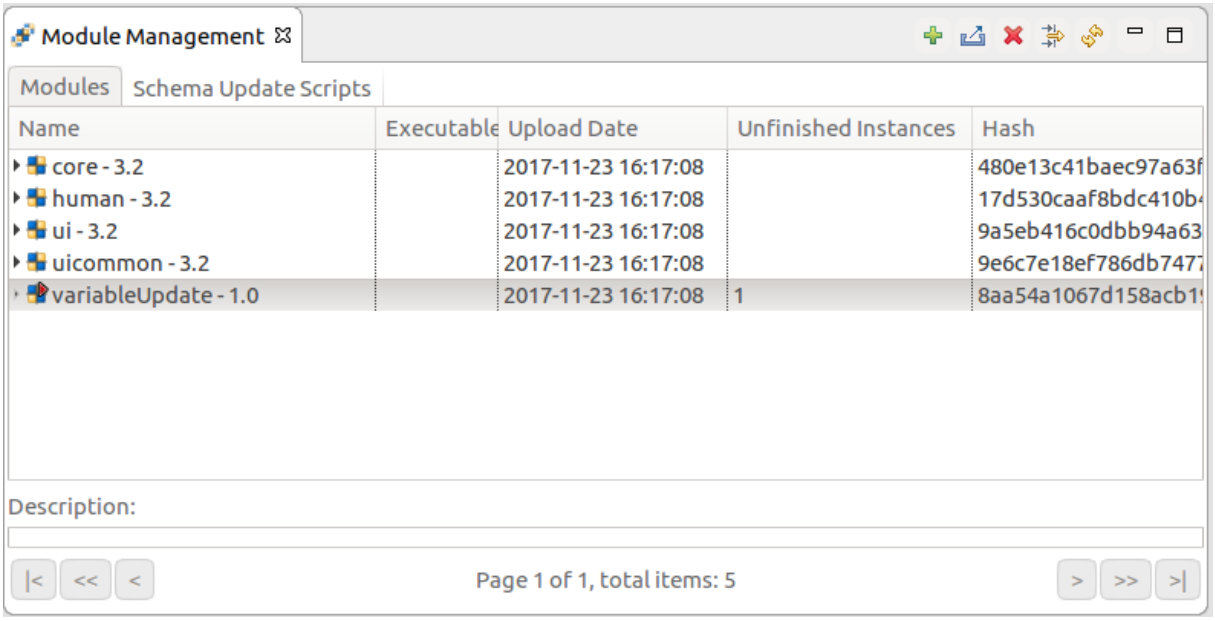
3.1 Module List

When working with modules, the following properties are available to the management tools:

- *Name*: arbitrary name as specified in module properties
- *Version*: version as specified in module properties
- *Executable*: whether the module can be used to create model instances
- *Upload date*: time when the module was uploaded
- *Unfinished instances*: number of unfinished (*running* and *suspended*) model instances based on the module (only on executable modules)
- *Hash*: hash code counted based on the content and properties of the module
- *Id*: id of the module unique on the server

3.1.1 Listing Modules in the Management Perspective

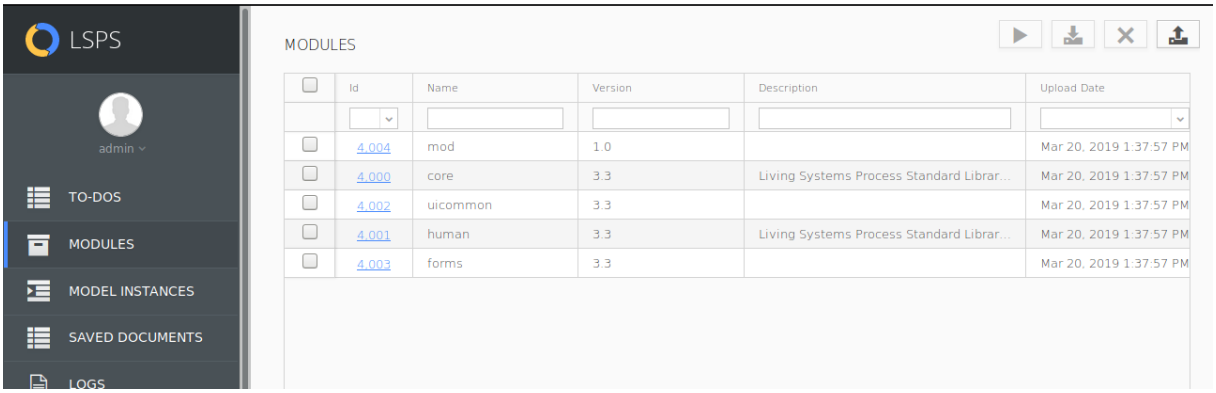
To list the modules uploaded on an LSPS Server in the Management Perspective, connect `PDS to the LSPS Server` and open the Management perspective: the list of modules is available in the *Module Management* view.



If the view is not displayed, go to **Window** > **Show View** > **Module Management**

3.1.2 Listing Modules in Management Console

To list the modules uploaded on an LSPS Server in Management Console, log in to the console and open the *Modules* page.



3.1.3 Listing Modules from the Command Line

To list the modules on the LSPS Server from the Command Line, use the **modelList** command.

Long Option	Short Option	Description
`--host*`	-h	host URL
`--username*`	-u	user name
`--password*`	-p	password
`--name`	-n	model name (supports wildcards)
`--version`	-v	model version (supports wildcards)

∗ required parameters

```
 $ java -jar lsps-cli-<VERSION>.jar modelList -h  http://localhost:8080 -u admin -p admin
Id    Name              Executable Upload Date          Unfinished Instances Hash
2004  formsforms - 1.0  true       2020-03-26 10:11:57  2                    f8cbc787302
2007  uiforms - 1.0     true       2020-03-26 10:23:09  4                    7e384c936f4
2011  uiforms - 1.0     true       2020-03-26 11:34:41  4                    11de15b3fbb
2012  createTodo - 1.0  true       2020-03-26 12:01:41  3                    809ec46fc69
2003  forms - 3.3       false      2020-03-26 12:01:41                       c0505fb1356
2014  uiforms - 1.0     true       2020-03-26 12:22:20  0                    5168fd905ab
```

3.2 Module Upload

When uploading a module to the Module Repository so it can be used by model instances, you need to define how to deal with any database changes the module might be introducing. The behavior is is set by the database update strategy:

- Do not change: Existing schema remains unchanged.
- Update by model: New tables, columns and relationships are added; no data is deleted.
- Validate: Existing schema is validated against the new model schema (if inconsistencies are detected, the model launching fails).
- Drop and create modeled DB tables: Existing schema is dropped and a new schema following the new model definitions is created.

If the database already contains data in the pertinent tables and you want to upgrade the schema to match the schema of new versions of the modules, first generate the schema update scripts, modify the scripts to *migrate the schema with your business data* before you upload the modules.

3.2.1 Uploading a Module from the Management Perspective

To upload a module from your workspace or a module zip file using the Management perspective of your PDS, do the following:

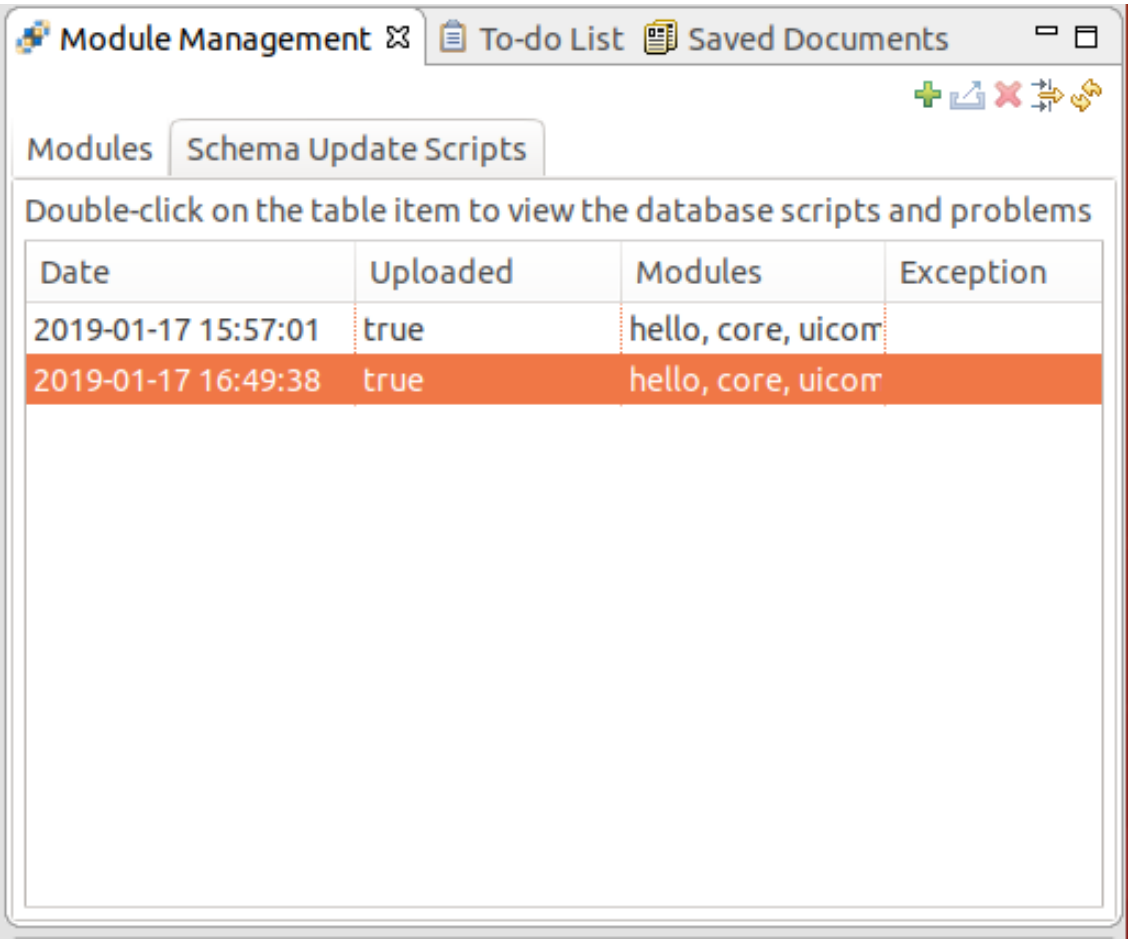
1. Make sure your PDS is connected to the correct LSPS server.
2. In the Module Management view, click the **Upload** button.
3. In the Module Upload dialog, define the upload details:
   (a) Select either a zip file with an exported module on your filesystem or a module from your workspace.
   (b) Define the database update strategy.
   (c) Unselect the *Do not execute schema update* scripts flag.
4. Click **OK**.

3.2.1.1 Generating Schema Update Scripts

When upgrading the data model of your model to a version which is not backward compatible and you want to migrate the existing data or migrate the underlying database schema manually, generate the schema update scripts for your models, modify them as required and apply them on the database.

To generate the schema update scripts for your modules do the following:

1. Open PDS and connect to the LSPS Server with the current modules.
2. Switch to the Management perspective.
3. In the Module Management view, click the **Upload** button.
4. In the Module Upload dialog, define the details:
   (a) Select either a zip files with the new modules or modules from your workspace.
   (b) Set *Database Schema Strategy*: Set *Update by model* to generate scripts that will update the current schema or *Drop and Create Modeled DB Tables* to generate scripts that will create the new schema from scratch.
   (c) Select the *Do not execute DB scripts* scripts flag: This prevent the schema update scripts execution and makes the scripts available in the tab *Schema Update Scripts*. You can find details such as lists of problems and conflicts discovered in the current schema in the script details.
5. Adapt the scripts as necessary:
   - Include the data stored in the tables of the old schemaUpdateScripts
   - *Do not remove* any tables or columns used by either the new or old models if their instances will be updated as part of migration (the new and old schema must be backward-compatible).



3.2.2 Uploading a Module from Management Console

Important: You can upload only Modules exported with the GO-BPMN export.

To upload a Module or a Model to the LSPS Server, do the following:

1. In the Modules view, click the Upload  button.
2. Define the database update strategy.
3. In the Model Upload dialog box, locate and select the model, and click Upload.
4. Click the Upload button.

3.2.3 Uploading a Module from the Command Line

To upload a module from the Command Line, use the **modelUpload** Command over a module zip file; the zip file must be created with GO-BPMN Export.

Important: You can upload only Modules exported with the GO-BPMN export.

Long Option	Short Option	Description
--host*	-h	host URL
--username*	-u	user name
--password*	-p	password
--model*	-m	list of model filenames or directories
--dbUpdateStrategy		strategy for database handling possible values: none, update, validate, drop

* required parameters

```
 ~/lsps-runtime/cli-tools$ java -jar lsps-cli-<VERSION>-full.jar modelUpload --host  http://localh
Info: Uploading files from /home/eko/lsps-workspace/Queries
Uploading files from /home/eko/lsps-workspace/Queries has finished.
```

3.3 Module Details

3.3.1 Displaying Module Details in the Management Perspective

To view the details of a Module in the Management Perspective, do the following:

1. Open the Module Management view.
2. Click the Module ID.
3. Expand the nodes to examine the Module structure.

3.3.2 Displaying Module Details in Management Console

To view the details of a Module in Management Console, do the following:

1. Open the Modules view.
2. Click the Module ID.
3. Expand the nodes to examine the Module structure. For Goal, Plan, and Process diagram node, you can click the *Diagram* link to display the diagram content.

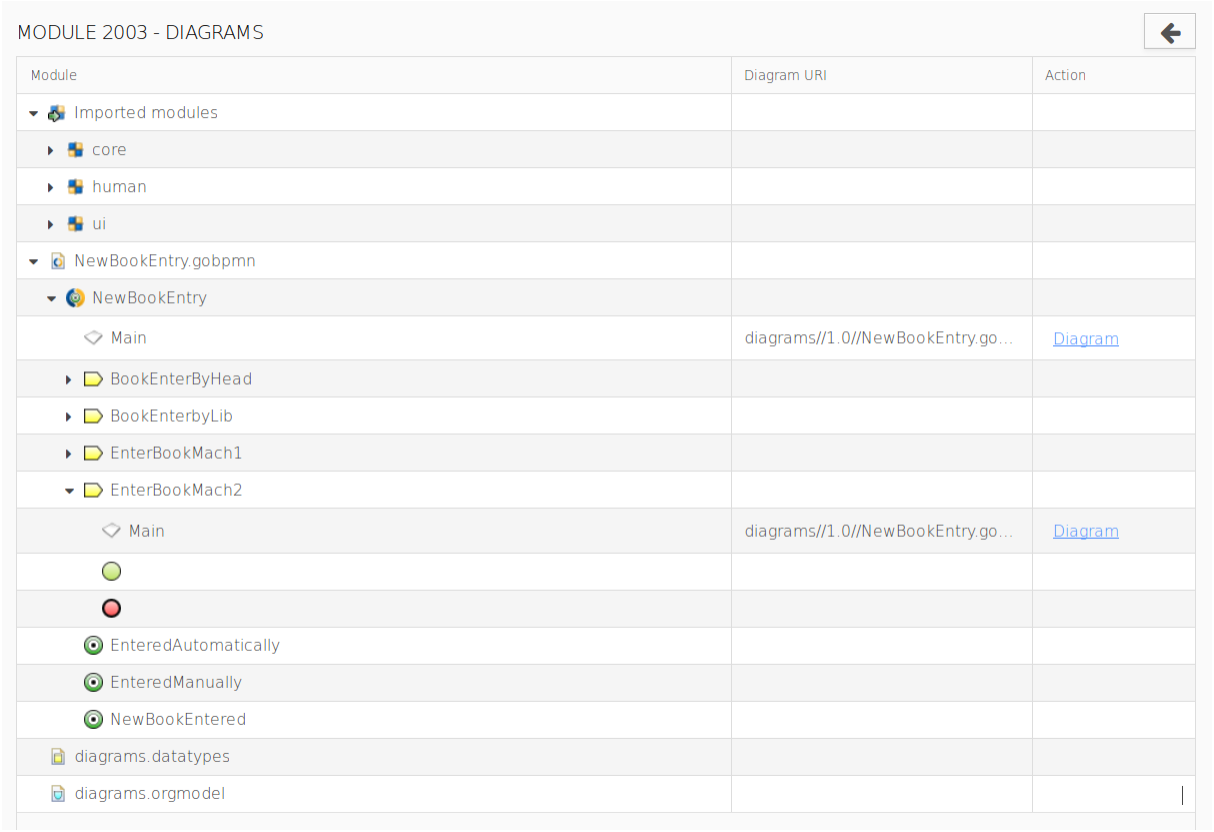


Figure 3.1 Details of a Module

3.4 Module Unload

On unload, the module is deleted from the Module Repository, and if a module is also a model, its running model instances are terminated. Roles and role assignments are preserved (for further information, refer to Role Management). Exploring features may become unavailable.

3.4.1 Unloading a Module from the Management Perspective

To unload a Module from the LSPS Server, do the following:

1. In the Module Management view select the model you wish to unload.
2. Click the Unload ( ) button in the view toolbar.
3. In the Warning dialog box, click Yes.

The Module is unloaded and no longer available. Any Model instances based on the Model are terminated.

3.4.2 Unloading Modules from Management Console

To unload a module, do the following:

1. On the Modules page, select the module.
2. In the view toolbar, click Unload  .
3. In the displayed dialog box, click Unload.

The model is unloaded from the server and any model instances based on the model are finished. Roles and role assignments remain unchanged.

3.4.3 Unloading Modules from the Command Line

To unload the modules from the LSPS Server from the Command Line, use the **modelUnload**.

Note that roles and role assignment remain unchanged.

Long Option	Short Option	Description
`--host*`	-h	host URL
`--username*`	-u	user name
`--password*`	-p	password
`--name`	-n	module name (supports wilcards)
`--version`	-v	module version (supports wilcards)
`--force`		do not prompt on unload

∗ required parameters

```
~/lsps-runtime/cli-tools$ java -jar lsps-cli-<VERSION>-full.jar modelUnload --host http://localhost:8080
      --username admin --password admin --name <MODULE_NAME>
Found more models:
Are you sure to remove model '15224 (1.0)' from repository? (Y/N)y
Model 15224 (1.0) successfully unloaded.
Are you sure to remove model '15224 (1.0)' from repository? (Y/N)y
Model 15224 (1.0) successfully unloaded.
Unloading models from repository has finished.
```

3.5 Module Download

You can download an uploaded Model or Module from the Model Repository as an archive file and then import the Module into the Process Design Suite with the GO-BPMN Import and edit it.

3.5.1 Downloading a Model or Module from the Management Perspective

To download a Module from an LSPS Server, do the following:

1. Make sure PDS is connected to the correct server.
2. In the Module Management view, select the model.
3. Click the Download button.
4. In the Save As dialog box, define the target location and edit the file name if necessary.

3.5.2 Downloading a Model or Module from Management Console

To download a module from the Module Repository of an LSPS Server, do the following:

1. Open Management Console of the server.
2. Open the Modules view.
3. In the modules table, select the module.
4. Click the Download  button.
5. In the dialog box, click Download.

Chapter 4

Model Instance Management

Model instances are runtime versions of models–executable modules with all their module imports. When you create a model instance, you "run a model". You can run multiple instances of a model possibly with different parameters to have the model instance adapt to different circumstances.

Each model instance holds a set of properties along with its context and context data. For detailed information on behavior of Model instances, refer to the GO-BPMN Model Language Guide.

You can temporarily stop the execution of a model instance or terminate it when required, update the model it is based on, and update its data.

You can also export the runtime data as a raw XML and import such data back.

4.1 Model Instance Listing

4.1.1 Listing Model Instances from the Management Perspective

To list the model instances on an LSPS Server in the Management Perspective, connect PDS to the LS↩ PS Server and open the Management perspective: the list of modules is available in the *Module Instances Management* view.

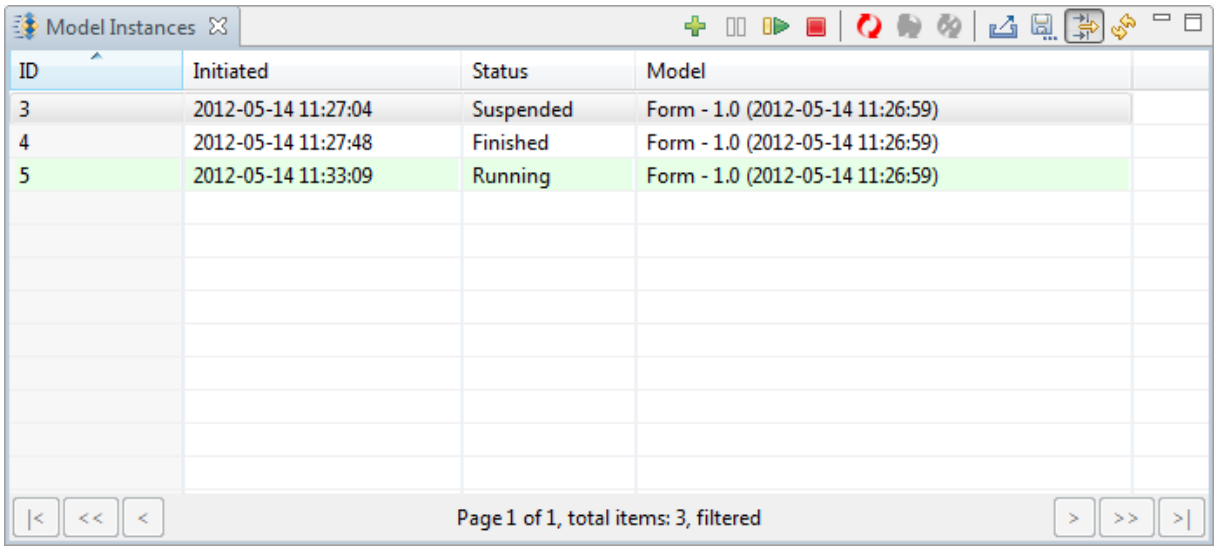


Figure 4.1 Model instances view

If the view is not displayed, go to **Window** > **Show View** > **Module Instances**

Note that you can copy the displayed model instance information with the **Copy to Clipboard** option in their context menu. To copy multiple model instances to the clipboard, select them and press Ctrl + C.

4.1.2 Listing Model Instances from Management Console

To list the model instances on an LSPS Server in Management Console, log in to the console and open the *Model Instances* page.

4.1.3 Listing Model Instances from the Command Line

To list the model instances on the LSPS Server from the Command Line, use the **modelInstanceList** command.

Long Option	Short Option	Description
`--host*`	-h	host URL
`--username*`	-u	user name
`--password*`	-p	password
`--modelInstance`		model instance ids, for example `8002 9000`
`--modelInstanceFile`		file with model instance ids
`--filterModelId`		IDs of the models
`--filterModelName`		model name pattern (supports `?` and * wildcards)
`--filterModelVersion`		model version (supports `?` and * wildcards)
`--filterStatus`		models in the defined execution status (CREATED, RUNNING, SUSPENDED, FINISHED)
`--filterFrom`		model instance(s) initiated after TIMESTAMP, format: yyyy-MM-dd HH:mm:ss
`--filterTo`		model instance(s) initiated before TIMESTAMP, format: yyyy-MM-dd HH:mm:ss
`--filterProps`		model instance properties defined as `key1 value1 key2 value2`
`--filterExpression`		boolean expression; Only the model instances for which the expression evaluates to true are returned.

* required parameters

```
java -jar lsps-cli-<VERSION>-full.jar modelInstanceList --host http://localhost:8080 --username admin
       --password admin --filterProps "my property" value "another property" "another value" --filterFrom "2017-10-31
       12:14:00"
```

Example with a text file with model instance IDs

```
lsps-runtime/cli-tools$ cat id.txt
18001
8000
lsps-runtime/cli-tools$ java -jar lsps-cli-3.3.2039-full.jar modelInstanceList --host http://localhost:8080
       --username admin --password admin --modelInstanceFile ./id.txt
18001       2019-01-02 16:25:54   RUNNING            [2004] mod - 1.0 (2019-01-02 16:25:54)
8000        2019-01-02 11:29:29   FINISHED           [2004] mod - 1.0 (2019-01-02 16:25:54)
Done.
```

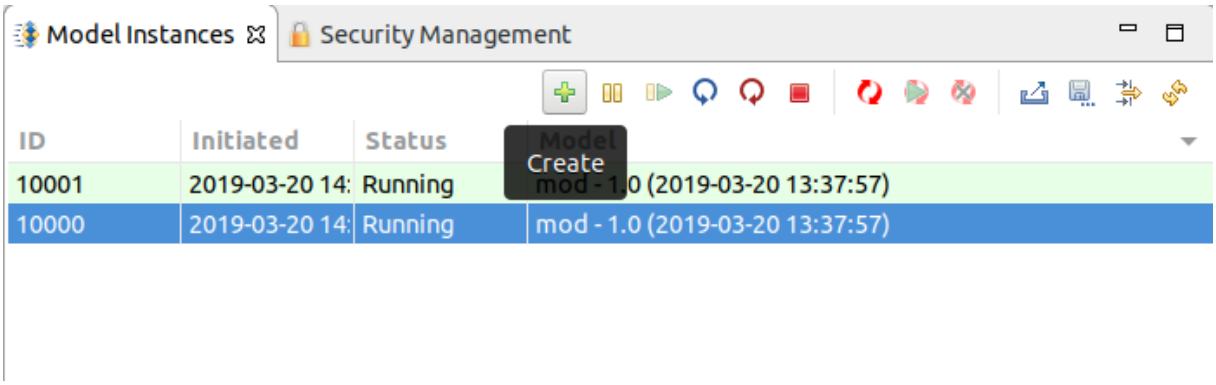
4.2 Model Run

When you run a model, a *model instance* based on the executable module a model instances is created. In the model instance, a module instance of the executable module and module instances of all its imported modules recursively are created. For more details, refer to section Model Instance in the GO-BPMN Language Guide.

4.2.1 Creating a Model Instance from the Management Perspective

To create a Model instance of an uploaded model from the Management perspective, do the following:

1. Open the Model Instances view.
2. Click the **Create** button in the view toolbar.
3. Select one of the uploaded models and click **Finish**.



The new model instance appears in the list of model instances in the *Model Instances* view.

4.2.2 Creating a Model Instance from Management Console

To create model instances of uploaded modules from Management Console, do the following:

1. Open the Modules view.
2. Select the executable modules.
3. Click **Run Model**  in the view toolbar.
4. In the *Run Model* dialog box, optionally enter the description and add properties.
5. Click **Run**.

Model instances of the selected modules are created.

4.2.3 Creating a Model Instance from the Command-Line Tool

To create a model instance of an uploaded model from the Command-Line Tools, use the **startModelInstance** command.

Long Option	Short Option	Description
`--host*`	-h	host URL
`--username*`	-u	user name
`--password*`	-p	password
`--modelName`		model name
`--modelVersion`		model version
`--modelId`		model ID
`--properties`		model instance properties defined a list of key-value pairs

∗ required parameters

```
java -jar lsps-cli-<VERSION>-full.jar startModelInstance --host http://localhost:8080 --username admin
       --password admin --modelName orderModule --properties customerType enterprises compliant yes
# alternatively:
java -jar lsps-cli-<VERSION>-full.jar startModelInstance --host http://localhost:8080 --username admin
       --password admin --modelName orderModule --properties "customer type" "enterprises" "compliant" "yes"
```

4.3 Start Monitoring of a Model Instance

When restarting the server of the application, model instances need to be restarted as well. If your models make use of the `start monitoring mechanism` you can check if the instances started up correctly from all the

management tools.

4.3.1 Checking the Start Status of a Model Instance from the Management Perspective

To inspect the start status of model instances from the Management perspective, display the *Application Restart* view: go to **Window** > **Show View** > **Application Restart**

4.3.2 Checking the Start Status of a Model Instance from Management Console

To inspect the start status of model instances from Management Console, log in to the console and open the *Application Restart* page.

4.3.3 Checking the Start Status of a Model Instance from the Command Line

To list the start status of model instances from the Command Line, use the **appRestartInfo** command.

Long Option	Short Option	Description
`--host*`	-h	host URL
`--username*`	-u	user name
`--password*`	-p	password
`--onlyStatus`		returns only overall restarting status

∗ required parameters

```
cli-tools $ java -jar lsps-cli-<VERSION> appRestartInfo --onlyStatus
IN PROGRESS - SUCCEEDING
```

4.4 Model Instance Detail

A model instance holds a set of properties and details of its execution status. You can view the details and manage the data where possible.

4.4.1 Displaying a Model Instance Detail from the Management Perspective

To display the details of a model instance, double-click the model instance in the *Model Instances* view.

You can visualize the current status of Model instance resources in their live diagrams: Live diagrams are diagrams of Process resources as defined in their Model (process, goal, plan, or sub-process diagrams). Individual execution states are indicated by colors. Diagram legend can be displayed by clicking the legend button  in the view toolbar.

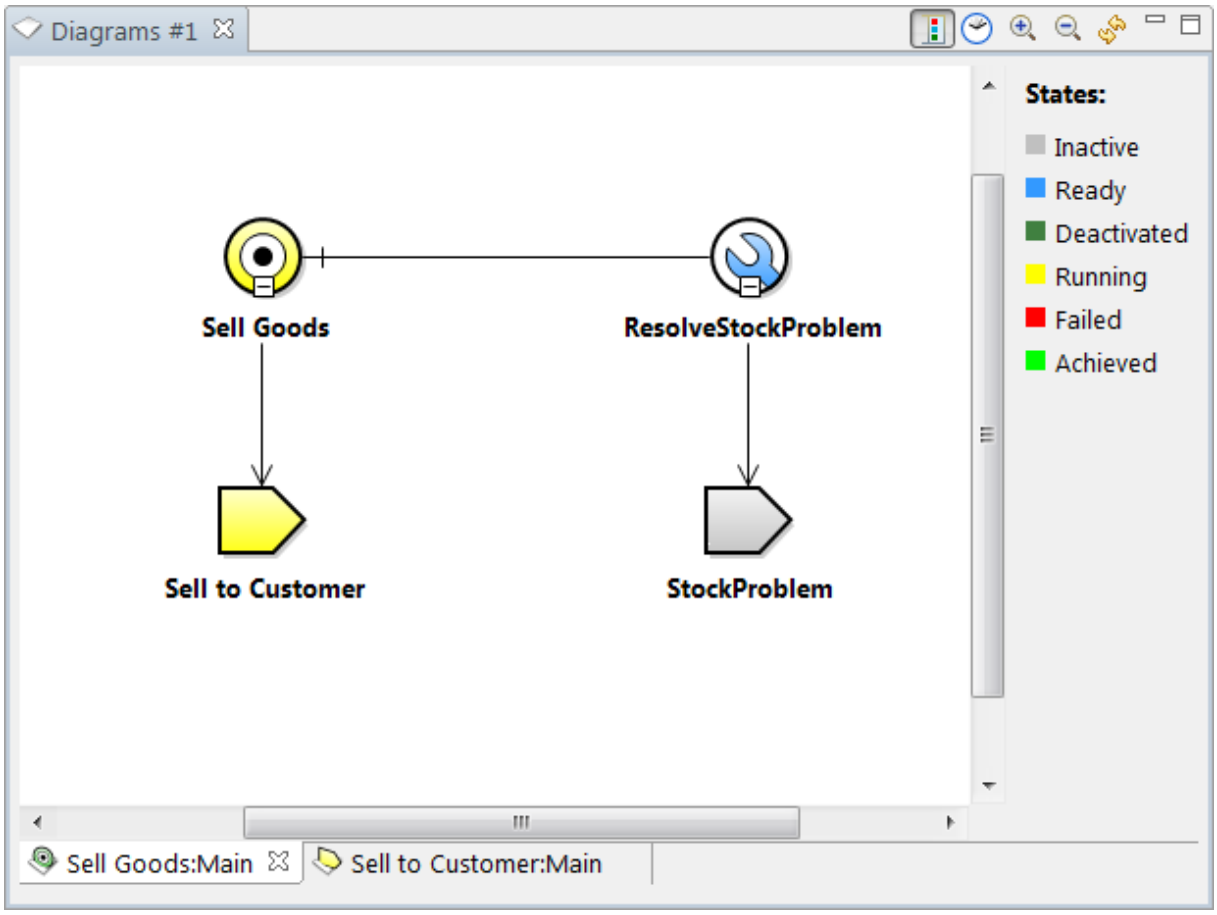


Figure 4.2 Live Goal diagram with legend

Goals with an implicit hyperlink (indicated by a small arrow in the top-left corner of the icon on the canvas), plans, and sub-processes are clickable: click the element to open the respective linked live diagram.

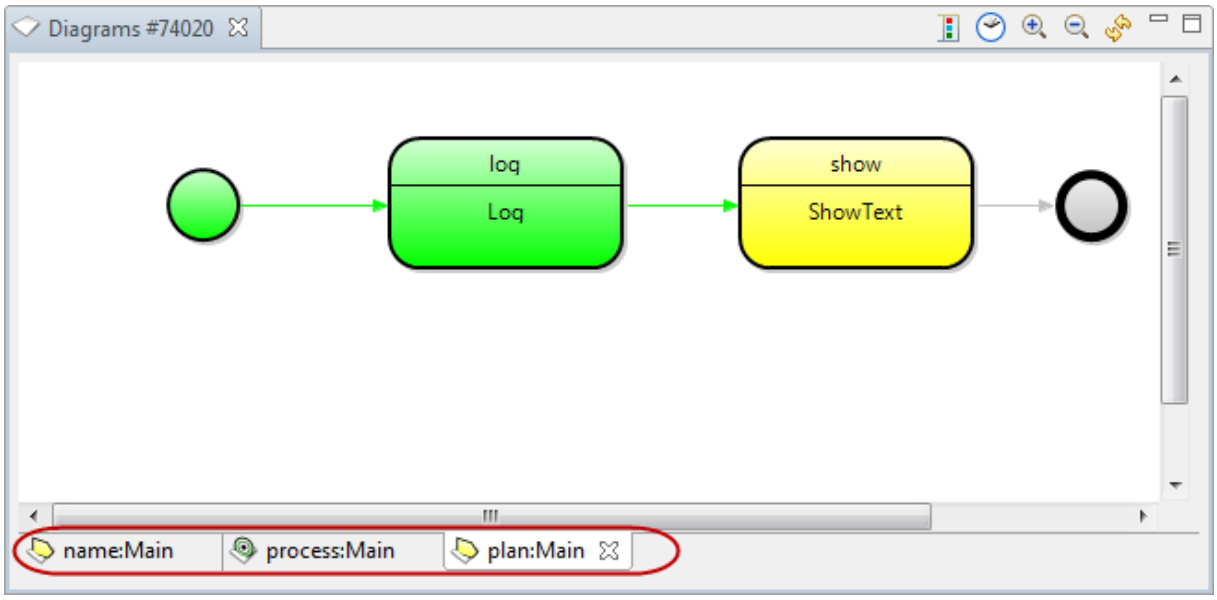


Figure 4.3 Live diagram with multiple pages

To display a live diagram, open the respective model instance detail view and double-click the diagram icon under Module Instances or Model Update History in the tree (also available for old module instances from before model update if applicable).

To visualize the history of the execution flow on the live diagram, click the History button in the view toolbar: You can then use the slider at the bottom of the view to view the execution workflow.

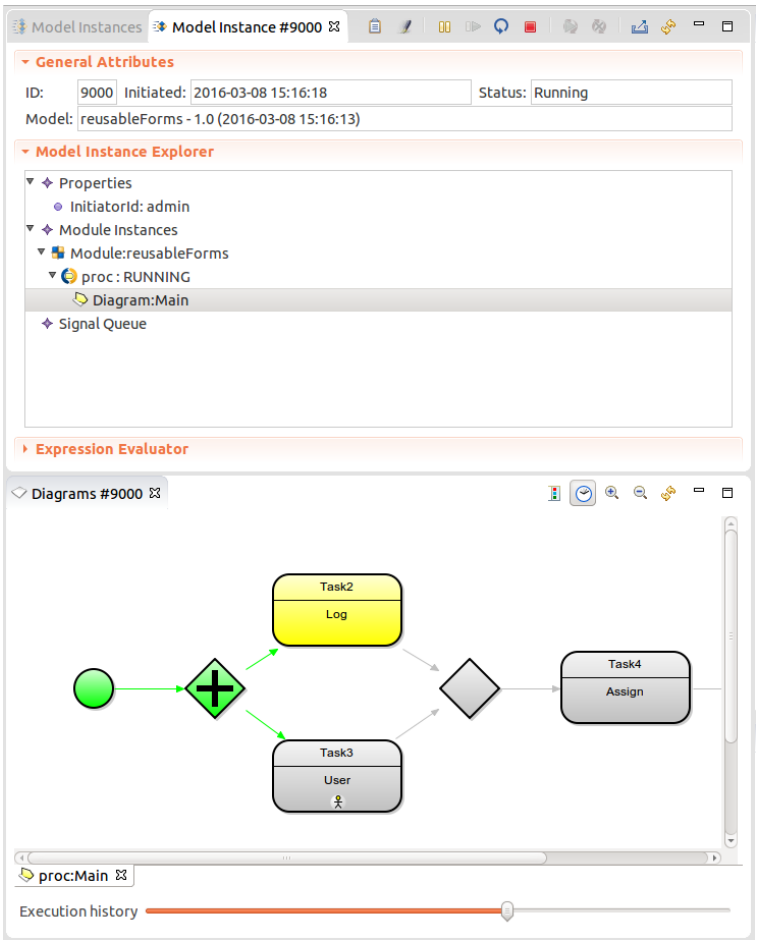


Figure 4.4 Live diagram with execution history slider

You can change the status colors used to indicate the element execution status in live diagrams under Window > Preferences > Process Design Suite > Management > Appearance.

Using the buttons in the toolbar of the model detail, you can in addition do the following:

- Show To-dos generated by the model instance (filter for the respective model instance ID is applied to the To-do List view and the view is focused).
- Show Logs generated by the model instance (filter for the respective model instance ID is applied in the Logs view and the view is focused).
- Suspend and Resume ( ) a model instance.
- Finish a running or suspended model instance.
- Perform actions of `Model update`.
- Export a model instance into an XML.

- Refresh the view content.

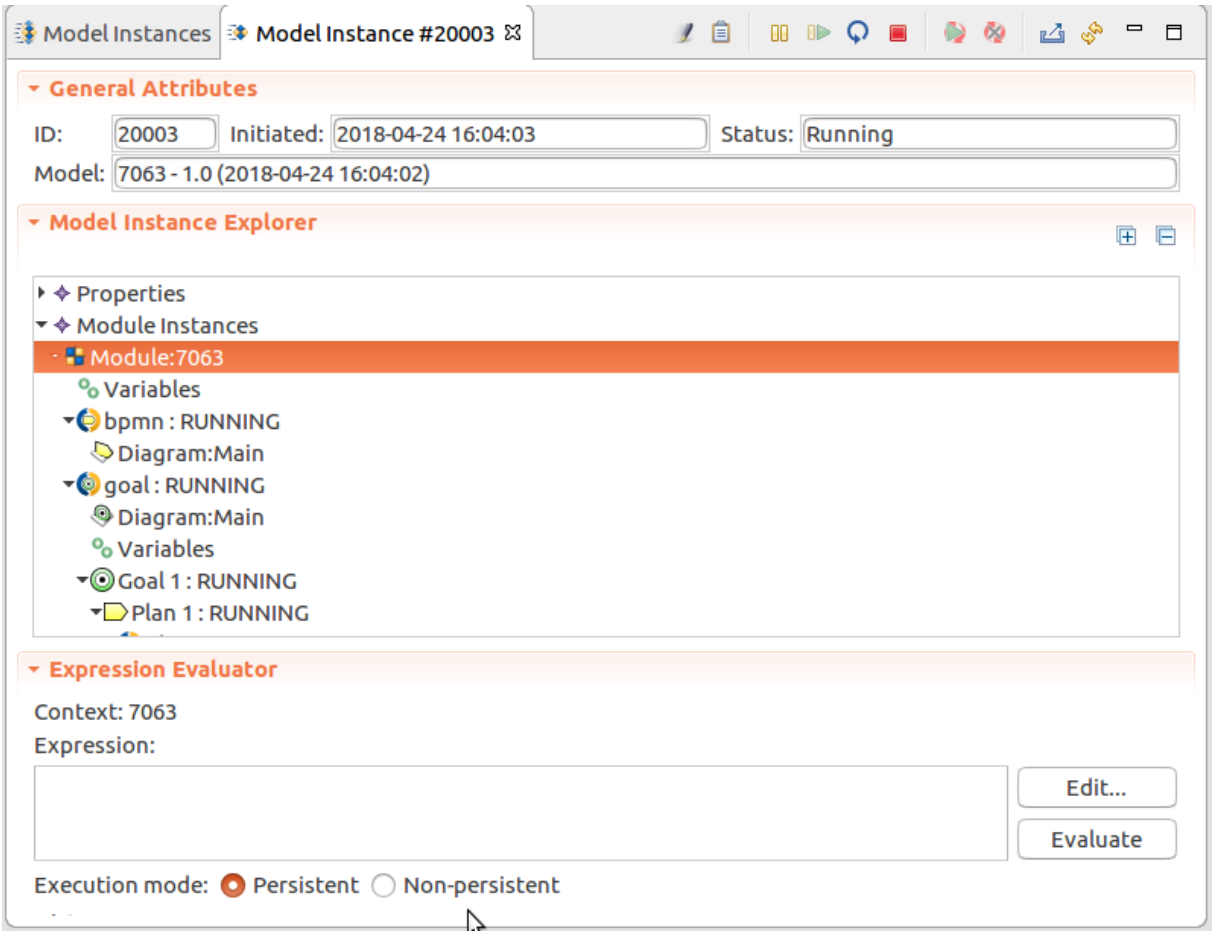


Figure 4.5 Model instance detail view

4.4.2 Displaying a Model Instance Detail from Management Console

To display a model instance detail from Management Console, click the model instance ID on the Model Instance page in the Model Instance ID column.

To display a live diagram with the execution, do the following:

1. On the menu, click Model Instances.
2. On the right, click the ID of the model instance.
3. Go to the *Context* tab.
4. Locate the diagram in the tree and click Diagram in the Action column.

Important: The diagrams in the Management console do not display hyperlink, lane and pool diagram elements.

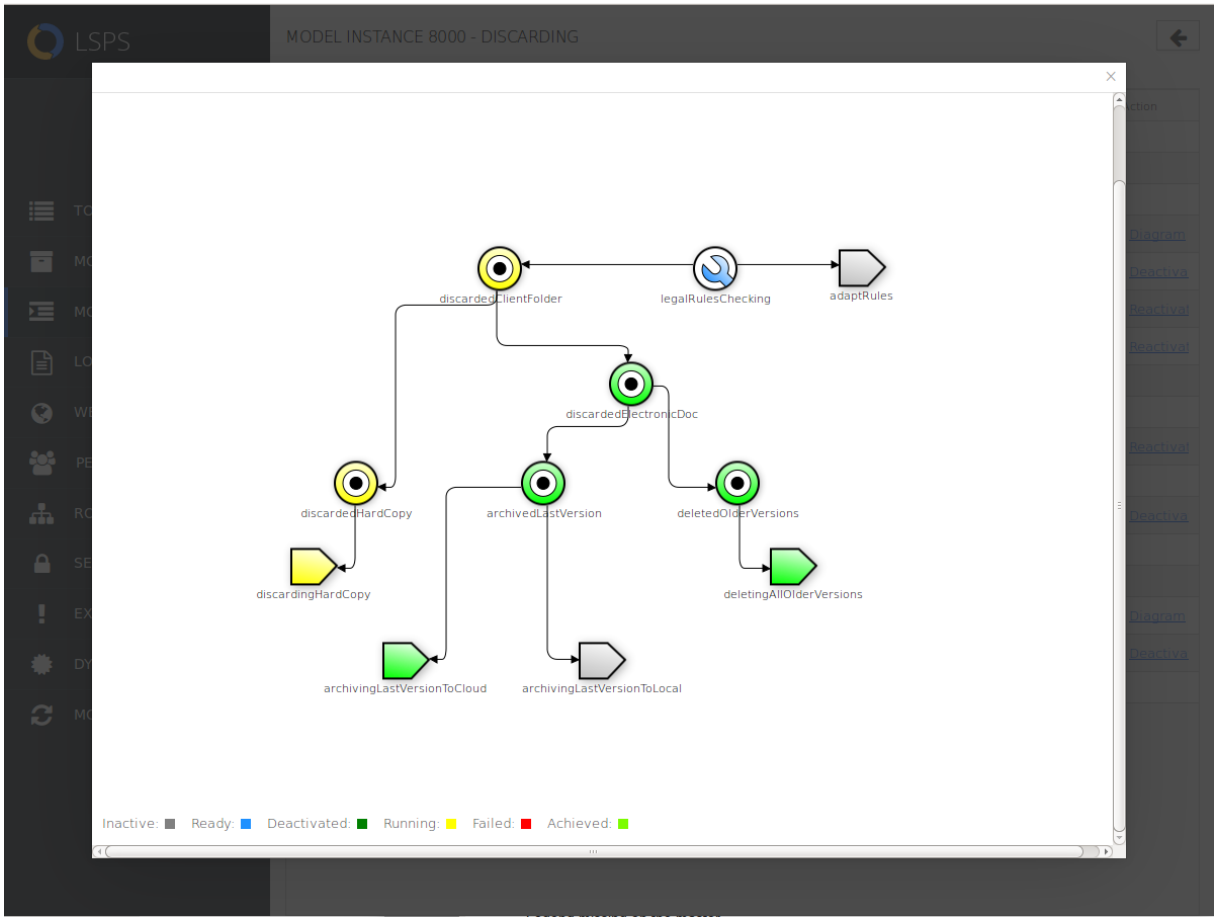


Figure 4.6 Live Goal diagram

4.5 Manipulation of the Context of a Model Instance

The context of model instances can be generally changed in the Expression Evaluator: to change variable values, evaluate an assignment expression, for example, `x:=2`, to send a signal, use `sendSignal()`, for example, evaluate `sendSignal(true, {thisModelInstance()}, "My signal object")`, to deactivate a goal, use `deactivate()`, for example, `deactivate({StockMaintainGoal})`, etc. Refer to the `Standard Library` for further functions.

For some context manipulations, such as, change of variable value, goal activation and deactivation, and signal removal, dedicated management features are provided for convenience.

4.5.1 Expression Evaluation in a Model Instance

4.5.1.1 Evaluating Expressions in a Model Instance from the Management Perspective

To evaluate expressions using the current context values of a Model instance, use the Expression Evaluator in the Model instance detail:

1. Open the Model instance detail view.

2. Expand the Expression Evaluator section.
3. In the tree, select the target context.

The context of the Expression Evaluator is set to the selected context: you can check the current context in the Expression Evaluator view.

The expressions can be executed in persistent or non-persistent mode:

- if in non-persistent mode, the expression is evaluated in the model instance context;
- if in persistent mode, the expression is evaluated and applied on the model instance context and on the database data if Shared Records are involved.

Information on the selected context is displayed in the Expression Evaluator: this is by default the Model instance context. To change the context to a Module, Process, Sub-Process context, select the respective element in the Model instance tree.

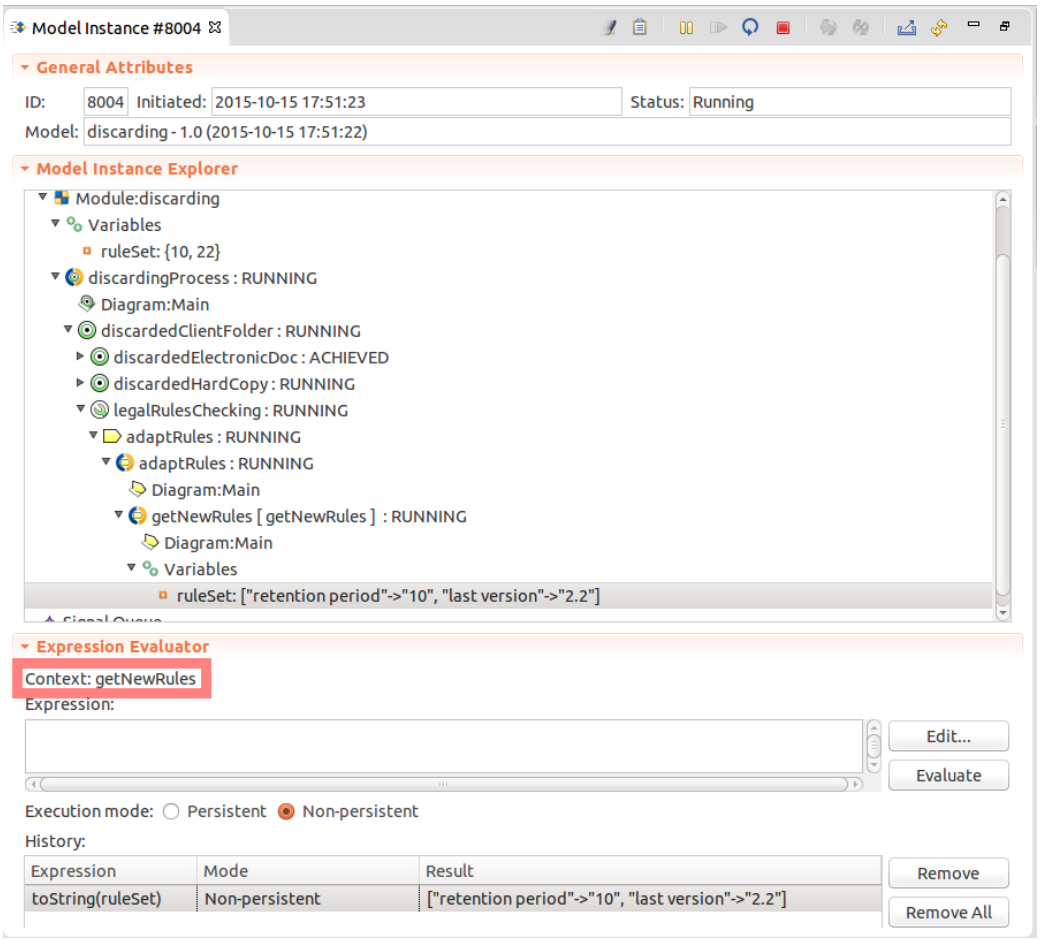


Figure 4.7 Expression Evaluator

4.5.1.2 Evaluating Expressions in a Model Instance from Management Console

To evaluate an expression in the context of a model instance from Management Console:

1. Display the Expression Evaluator tab (display the Model Instances view, click the desired model instance ID, and select the Expression Evaluator tab).
2. Unselect **Persistent** to evaluate the expression without changing of the data in the database.

   If the Persistent flag is selected and the expression changes the value of an entity, for example, a global variable, the change is reflected on the runtime data, that is, the variable takes the newly assigned value.
3. In **Evaluation context**, select the context you want to use.
4. Type the expression in the input field of the evaluator.
5. Click Evaluate.

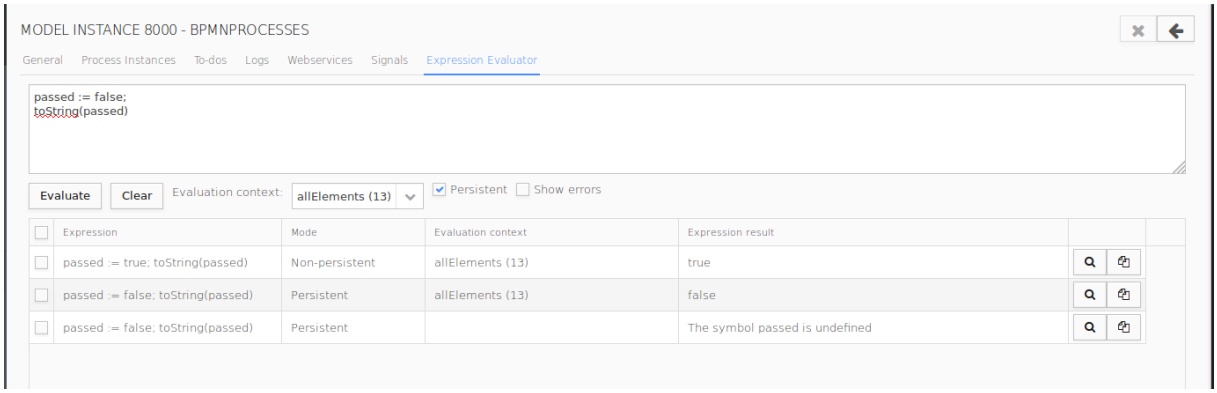


Figure 4.8 Evaluating an expression in different contexts and modes

To show errors, which occurred when attempting to evaluate expressions, select the *Show Errors* check box below the expression input field.

To remove evaluated expressions from the table, select the expressions and click the *Remove* (X) button in the upper-right corner.

4.5.2 Change of Variable Value

To change values of variables during execution, you can either enter an assignments expression, such as, `myVar:="new value"`, in the Expression Evaluator of the model instance or change it in the context tree in the model instance detail.

4.5.2.1 Changing a Variable Value from the Management Perspective

You can change values of variables during execution either from the Expression Evaluator using assignments or from the Model instance detail.

To change the value of a variable in a running model instance, do the following:

1. Open the Model instance detail view.
2. Under Model Instance Explorer, expand Module Instances and the desired Module node (parent namespace of the context variable).

3. Expand the Variables node and double-click the required variable. Variables of a standard library type cannot be edited.
4. In the Update Variable dialog box, select:
   - Value to assign the variable a particular value;

     Note: Depending on the variable type (basic, record, collection, etc.), the Update Variable dialog box shows relevant buttons. For basic values, provide the desired value directly; make sure you follow the Expression Language rules. Collection and record values are edited recursively, that is, if you click Edit, there are gradually "split" in less complex values, until basic values are displayed.
   - Expression and type an expression to be evaluated and used as the variable value. Note the information on the declared type of the variable and the current actual type.

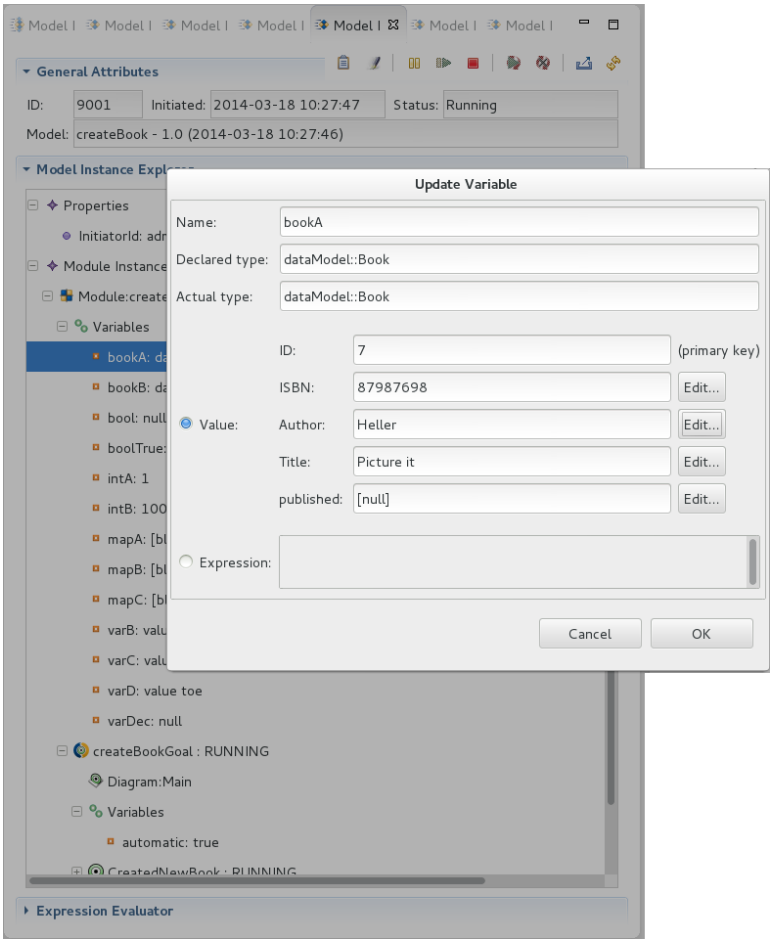


Figure 4.9 Updating variable

4.5.2.2 Changing a Variable Value from Management Console

You can change values of variables during execution either from the Expression Evaluator using assignments or from the Model instance detail.

To change the value of a variable of a running model instance from the Model Instance detail, do the following:

1. On the menu, click Model Instances.
2. On the right, click the ID of the model instance.

3. Go to *Context* tab.
4. Click the variable:
5. For a variable, click Edit in the Action column.

4.5.3 Goal Activation and Deactivation

When a goal is deactivated, it becomes *deactivated* along with all sub-Goals and execution of any plans is halted; activation is the opposite action: a goal that is *deactivated*, *achieved*, or *failed* becomes *ready* and further execution can follow. The mechanism applies to both Maintain and Achieve Goals. Further information is available in the `Modeling Language` guide.

4.5.3.1 Deactivating and Activating Goals from the Management Perspective

Changing execution states of Goals during execution can help you to direct the flow of the execution. Note that you can activate or reactivate only achieved, failed, deactivated or inactive goals, and `deactivate only ready or running goals`.

To deactivate or activate a Goal, do the following:

1. Open a Model instance detail view.
2. Under Model Instance Explorer, expand Module Instances and the desired module node.
3. Expand the respective process, then right-click a goal and select Deactivate or Activate.

Alternatively, you can call `deactivate()` and `activate()` functions from the Expression Evaluator of the model instance.

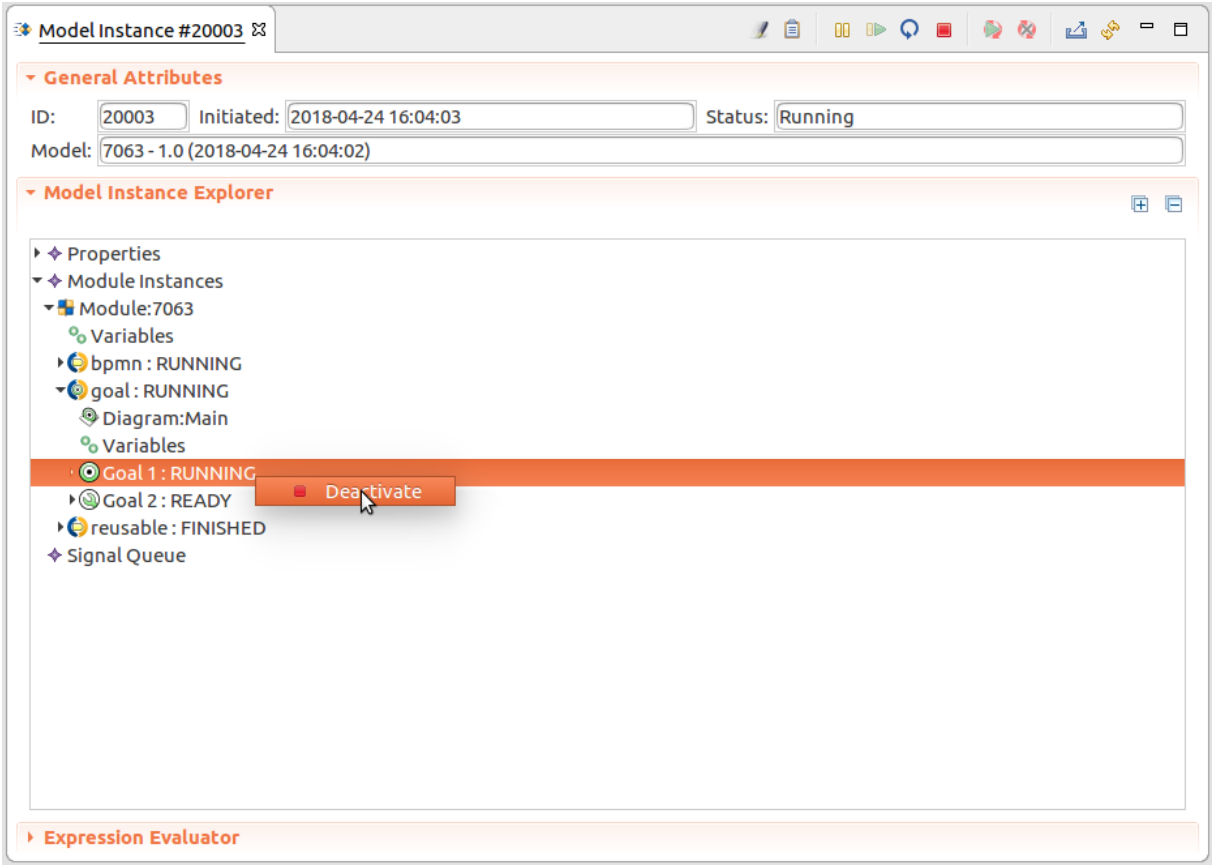


Figure 4.10 Deactivating a Goal

You can activate or deactivate a Goal also from its context menu in a live diagram; mind that disabled goals cannot be deactivated or activated.

4.5.3.2 Deactivating and Activating Goals from Management Console

To deactivate or reactivate a Goal of a running model instance from the Model Instance detail, do the following:

1. On the menu, click Model Instances.
2. On the right, click the ID of the model instance.
3. Go to *Context* tab.
4. In the row with the Goal, click the **Deactivate** or **Reactivate** link in the Action column.

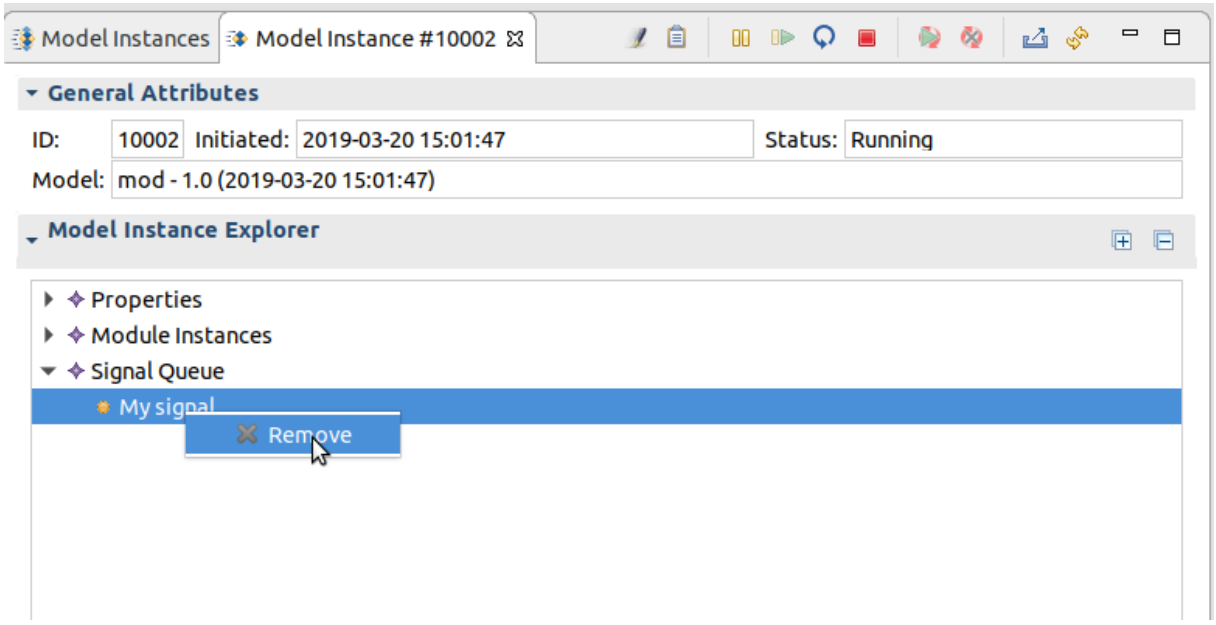
Alternatively, you can call `deactivate()` and `activate()` functions from the Expression Evaluator of the model instance.

4.5.4 Signals Removal

4.5.4.1 Removing Waiting Signals from the Management Perspective

To remove a waiting signal of a model instance from the Management perspective, do the following:

1. Double-click the model instance in the Model Instances view.
2. In the detail view, expand the Signal Queue node.
3. Right-click the signal and select **Remove**.



4.5.4.2 Removing Waiting Signals from Management Console

To remove a waiting signal of a model instance, do the following:

1. Display the Signal tab of the model instance (Model Instances model instance ID Signals).
2. Select the signals you want to remove.
3. Click the Remove (X) button in the upper right corner.

4.6 Export and Import of a Model Instance State

To amend the runtime data of *running* or *suspended* model instances in critical situations, you can export the state to an XML file, modify them in the XML and upload the XML with the new state.

You can modify only the context data of the model instance that are not persisted: if you modify data that is persisted in the database, including model-instance properties, such as, the initiator, the changes are ignored: it is only the context of the model instance that changes.

4.6.1 Exporting and Importing a Model Instance State from the Management Perspective

Important: Importing a corrupted model instance XML can cause fatal failure on the server.

To export and import a model-instance XML, do the following:

1. Open the *Model Instances* view.
2. Select the model instances.
3. Right-click the selection and select **Export Raw Data**
4. Select the target location, where to save the files.
5. Edit the exported XML files. Do not modify data that is persisted in the database, including model-instance properties, such as, the initiator, the changes are ignored.

   If you changed such data, fix the XML and import it again.
6. In the *Model Instances* view, right-click the model instance.
7. Click **Import Raw Data**.
8. Select the modified XML for the model instance. Note that on import, the data persisted in the database remains unchanged in the database, while the context of the model instance changes.

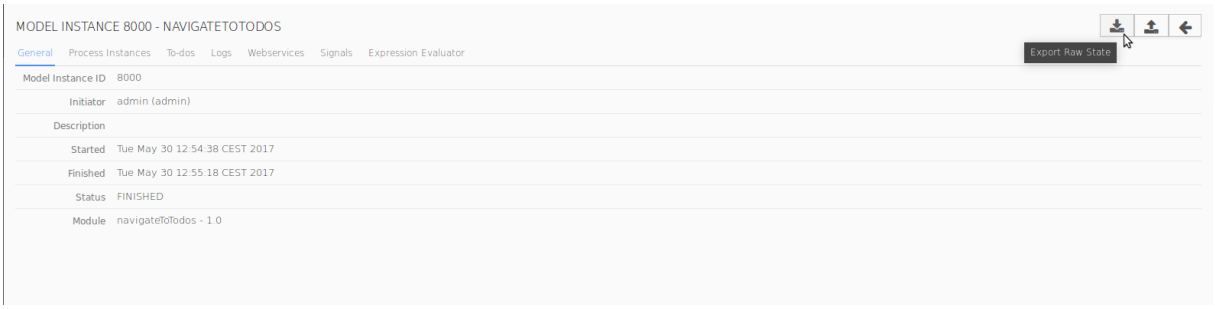
4.6.2 Exporting and Importing a Model Instance State from Management Console

Important: Importing a corrupted model instance XML can cause fatal failure on the server.

To export a model-instance XML and import it with modifications, do the following:

1. Open the Model Instance Detail view: click the ID of the model instance in the Model Instances view.
2. Click the *Export Raw State* button in the upper right corner.
3. Save and edit the export XML file. Do not modify data that is persisted in the database, including model-instance properties, such as, the initiator, the changes are ignored.

   If you changed such data, fix the XML and import it again.
4. Click the *Import Raw State* button in the upper right corner.



4.6.3 Exporting and Importing a Model Instance State from the Command Line

To export a model-instance XML and import its modified version, use the `export` and `import` commands:

export exports a model instance as an XML file;

Long Option	Short Option	Description
`--host*`	-h	host URL
`--username*`	-u	user name
`--password*`	-p	password
`--modelInstance`		model instance ids, for example `8002 9000`
`--modelInstanceFile`		file with model instance IDs
`--location`		output location and file prefix (model instance IDs are appended to the prefix)

∗ required parameters

```
~/lsps-runtime/cli-tools$ cat id.txt
18001
8000
~/lsps-runtime/cli-tools$ java -jar lsps-cli-<VERSION>-full.jar export \
--host http://localhost:8080 --username admin --password admin --modelInstanceFile id.txt
Exporting model instance #18001 to 18001.xml
Exporting model instance #8000 to 8000.xml
Done.
```

4.6.3.1 Importing a Model Instance State from the Command Line

Important: Importing a corrupted model-instance XML can cause fatal failure on the server.

importRaw imports a raw model instance state;

Long Option	Short Option	Description
`--host*`	-h	host URL
`--username*`	-u	user name
`--password*`	-p	password
`--modelInstance*`		model instance ids, for example `8002 9000`
`--file*`		file with model instance

* required parameters

4.7 Model Instance Suspend and Resume

When you suspend a running model instance, it becomes read-only and its execution is halted: All running elements become *suspended* including any to-do which become read-only. Suspended model instances can be resumed. More information on suspend is available in the GO-BPMN Modeling Language Guide.

4.7.1 Suspending and Resuming a Model Instance from the Management Perspective

To suspend a running Model instance or resume a suspended Model instance, do the following:

1. Activate the Model Instances view.
2. Select the instances you wish to suspend/resume.
3. In the view toolbar, click Finish or Resume .

On detailed behavior of Model instances, refer to GO-BPMN Model Language User Guide.

The command is available also in the context menu of individual model instances.

4.7.2 Suspending and Resuming a Model Instance from Management Console

A suspended model instance stops its execution. On resume, the instance continues from where it was suspended.

To suspend or resume a suspended model instance, do the following:

1. Open the Model Instances view.
2. In the displayed table with model instances, select the model instances.
3. In the view toolbar, click the Suspend  or Resume  command.

4.7.3 Suspending and Resuming a Model Instance from the Command Line

4.7.3.1 Suspending a Model Instance from the Command Line

suspend suspends model instances;

Long Option	Short Option	Description
`--host*`	-h	host URL
`--username*`	-u	user name
`--password*`	-p	password
`--modelInstance`		model instance IDs, for example, `8999 8899`
`--modelInstanceFile`		file with model instance IDs to be requested
`--filterModelId`		IDs of the models
`--filterModelName`		model name pattern (supports `?` and * wildcards)
`--filterModelVersion`		model version (supports `?` and * wildcards)
`--filterFrom`		model instance(s) initiated after TIMESTAMP, format: yyyy-MM-dd HH:mm:ss
`--filterTo`		model instance(s) initiated before TIMESTAMP, format: yyyy-MM-dd HH:mm:ss
`--filterProps`		model instance properties defined as `key1 value1 key2 value2`
`--filterExpression`		boolean expression; Only the model instances for which the expression evaluates to true are returned.
`--testFilter`		boolean expression; displays the model instances affected by the command (the command is not executed)

* required parameters

```
java -jar lsps-cli-<VERSION>-full.jar suspend --testFilter --host http://localhost:8080 --username admin
       --password admin --filterProps myProperty "value" --filterFrom "2017-10-31 12:14:00"
```

4.7.3.2 Resuming a Model Instance from the Command Line

resume resumes suspended or updated model instances;

Long Option	Short Option	Description
`--host*`	-h	host URL
`--username*`	-u	user name
`--password*`	-p	password
`--modelInstance`		model instance IDs, for example, `8999 8899`
`--modelInstanceFile`		file with model instance IDs to be resumed
`--filterModelId`		ID of the model
`--filterModelName`		model name pattern (supports `?` and * wildcards)
`--filterModelVersion`		model version (supports `?` and * wildcards)
`--filterFrom`		model instance(s) initiated after TIMESTAMP, format: yyyy-MM-dd HH:mm:ss
`--filterTo`		model instance(s) initiated before TIMESTAMP, format: yyyy-MM-dd HH:mm:ss
`--filterProps`		model instance properties defined as `key1 value1 key2 value2`
`--filterExpression`		boolean expression; Only the model instances for which the expression evaluates to true are returned.
`--testFilter`		boolean expression; displays the model instances affected by the command (the command is not executed)

* required parameters

```
java -jar lsps-cli-<VERSION>-full.jar finish --testFilter --host http://localhost:8080 --username admin
      --password admin --filterProps myProperty "value" --filterFrom "2017-10-31 12:14:00"
```

4.8 Model Instance Finish

When you `finish a model instance`, the running elements stop their execution immediately and the status of the model instance becomes **finished**.

4.8.1 Finishing a Model Instance from the Management Perspective

To finish a running Model instance from the Management perspective, do the following:

1. Open the Model Instances view.
2. Select the instances, you wish to finish.
3. In the view toolbar, click Finish  . The command is available also in the context menu of individual model instances.

4.8.2 Finishing a Model Instance from Management Console

To finish a model instance from Management Console, do the following:

1. Open the Model Instances page.
2. In the displayed table with model instances, locate the particular model instance.
3. In the view toolbar, click the Finish  button.

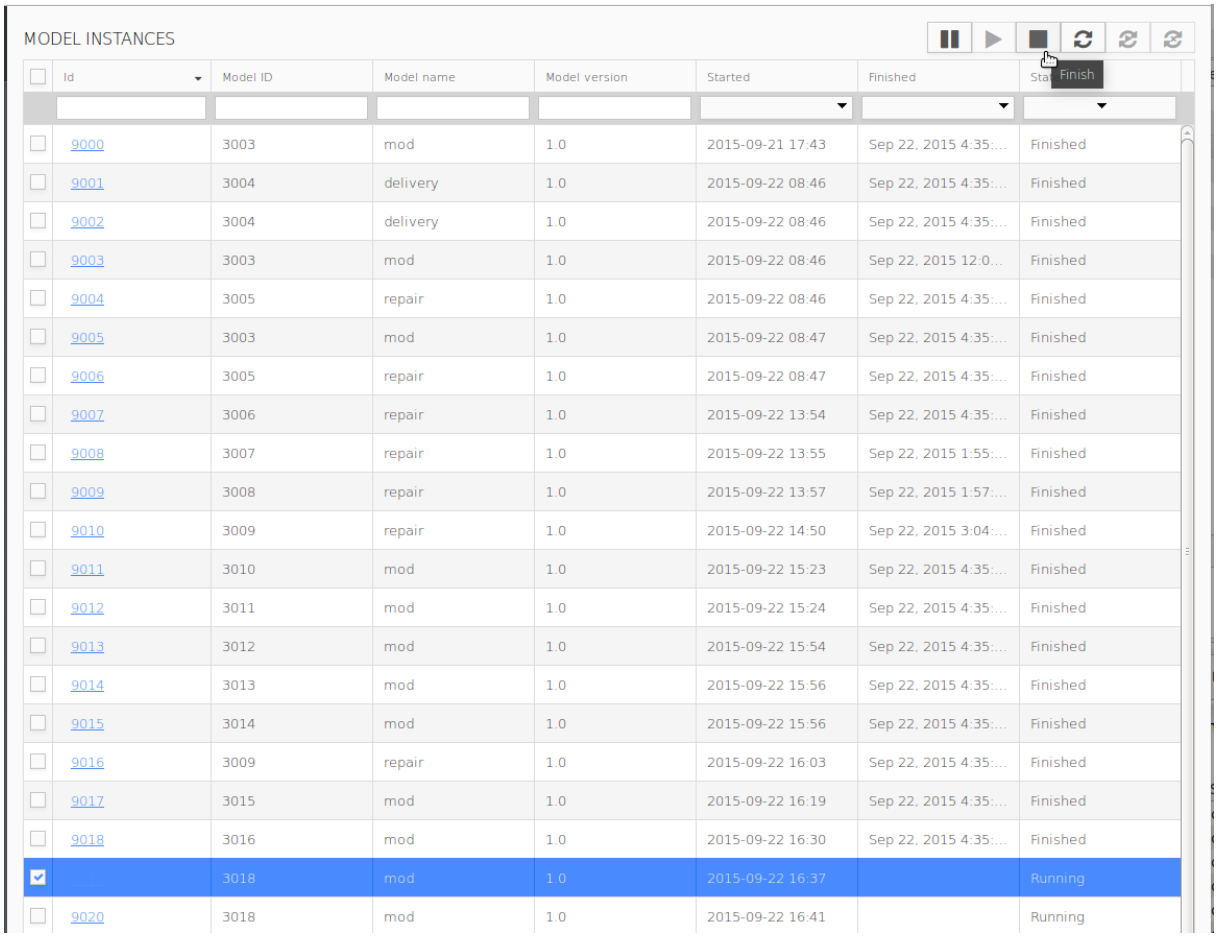


Figure 4.11 Finishing Model Instance

4.8.3 Finishing a Model Instance from the Command Line

finish finishes model instances

Long Option	Short Option	Description
`--host*`	-h	host URL
`--username*`	-u	user name
`--password*`	-p	password
`--modelInstance`		model instance IDs, for example, `7888 7887`
`--modelInstanceFile`		file with model instance IDs to be resumed
`--filterModelId`		ID of the model
`--filterModelName`		model name pattern (supports `?` and `*` wildcards)
`--filterModelVersion`		model version (supports `?` and `*` wildcards)
`--filterFrom`		model instance(s) initiated after TIMESTAMP, format: yyyy-MM-dd HH:mm:ss
`--filterTo`		model instance(s) initiated before TIMESTAMP, format: yyyy-MM-dd HH:mm:ss
`--filterProps`		model instance properties defined as `key1 value1 key2 value2`
`--filterExpression`		boolean expression; Only the model instances for which the expression evaluates to true are returned.

Long Option	Short Option	Description
`--testFilter`		boolean expression; displays the model instances affected by the command (the command is not executed)

∗ required parameters

```
java -jar lsps-cli-3.3.2039-full.jar finish --host http://localhost:8080 --username admin --password admin
      --filterModelName "mod"
```

4.9 Model Instance Restart

When you restart a running model instance, its module instances finish and module instances are created anew within the same model instance: the instances are running under the same model instance which preserves the ID of the model instance. Context of the finished module instances does not impact the context of the new module instances.

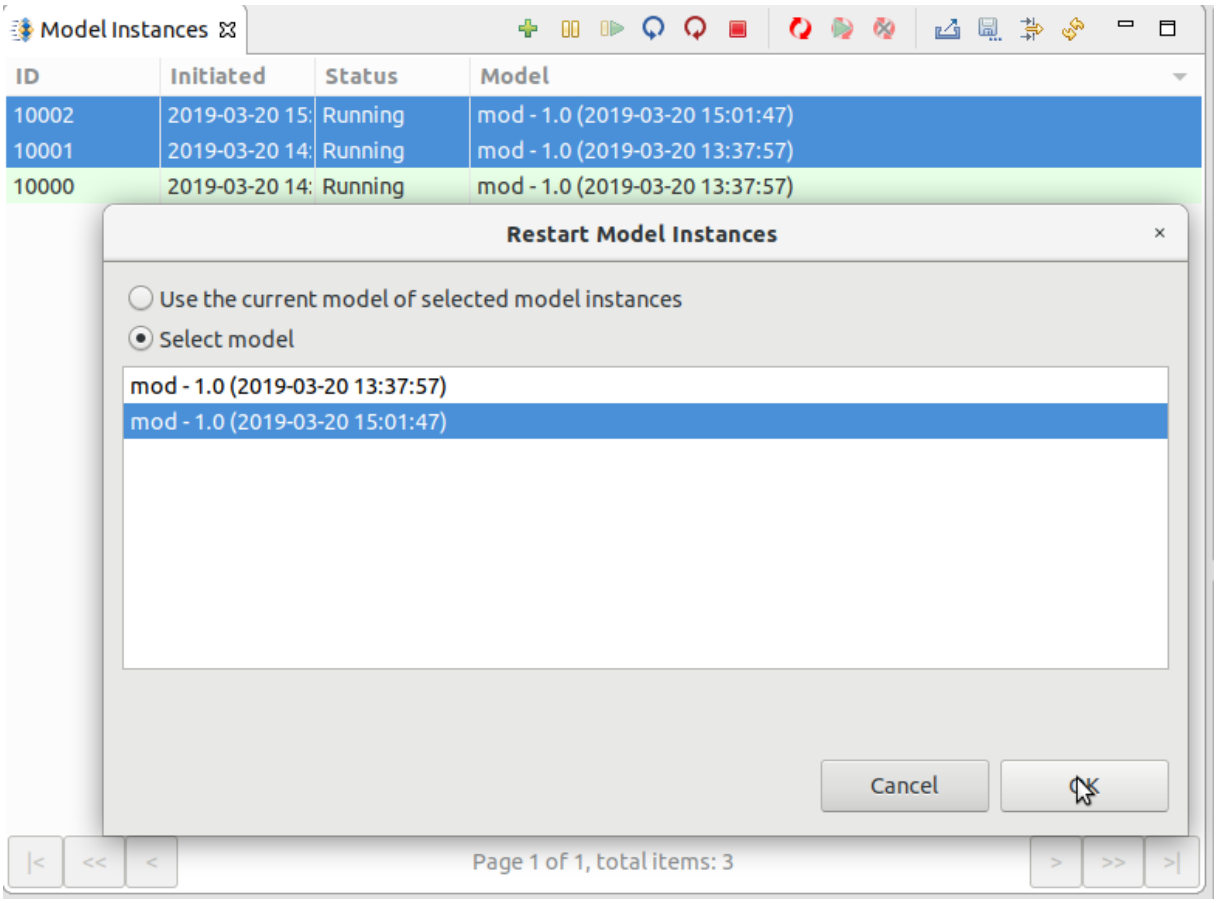
On restart, you can select the model the model instance will use, be it the same or a different one.

> **Important:** The persisted data structures used by the original model and the model used on restart must be compatible; for example, if you remove a column from a database table which was used by the original model and is not used by the model after restart, the restart fails: the column has to be present in the database. To resolve such a situation, use the *Do not change* strategy for update of the database schema and remove the redundant structures from the database after the restart.

4.9.1 Restarting a Model Instance from the Management Perspective

To restart one or multiple model instances from the Management perspective, do the following:

1. On the **Model Instances** view, select the Model instances.
2. Click Restart  in the page toolbar.
3. In the *Restart Model Instances* dialog box, select the Model that should be used by the restarted Model instances.



4. Click OK.

5. Refresh the Model Instances view and check the status of the restarted model instances.

4.9.2 Restarting a Model Instance from Management Console

To restart one or multiple model instances, do the following:

1. On the **Model Instances** page, select the Model instances.

2. Click Restart  in the page toolbar.

3. In the *Restart Model Instance* dialog box, select the Model that should be used by the restarted Model instance.

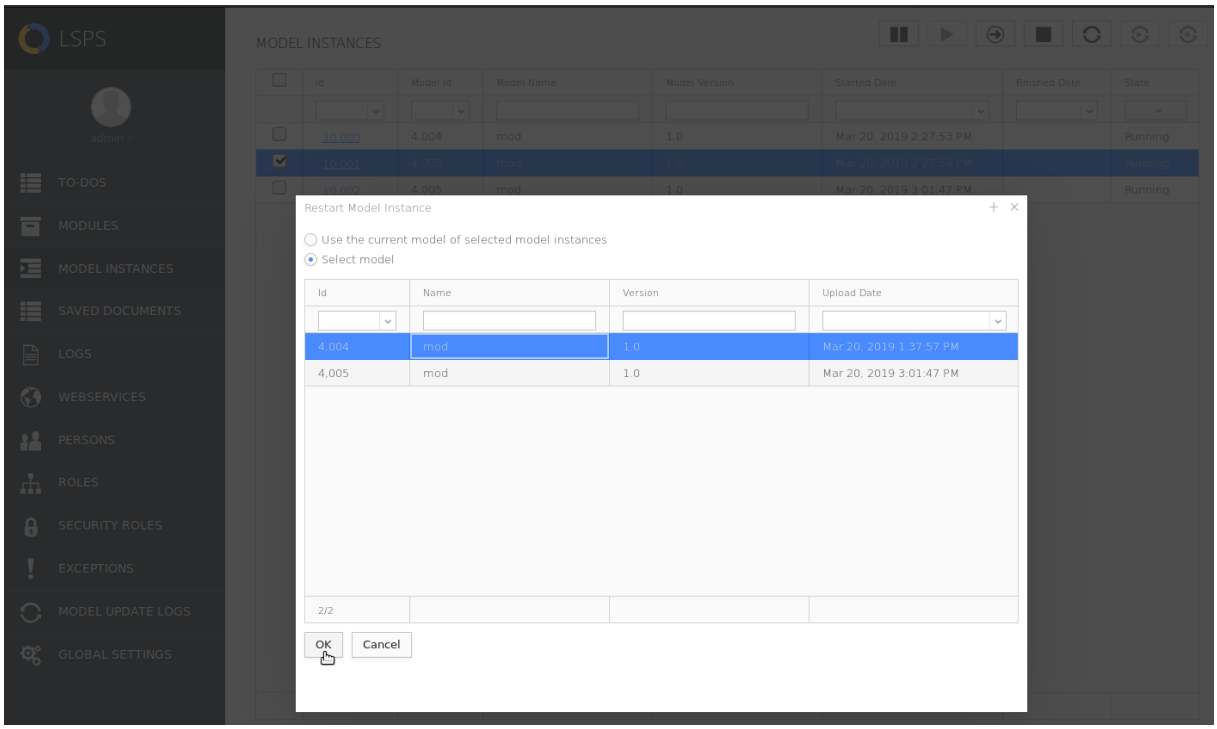


Figure 4.12 Selecting the Model for the restarted Model instance

4. Click OK.

4.9.3 Restarting a Model Instance from the Command Line

restart restarts running or suspended model instances

Long Option	Short Option	Description
`--host*`	-h	host URL
`--username*`	-u	user name
`--password*`	-p	password
`--modelId`		ID of the model used for the restarted model instances (if not specified, each instance is restarted without model change)
`--modelInstance`		model instance IDs, for example, `7848 7687`
`--modelInstanceFile`		file with model instance IDs to be restarted
`--filterModelId`		ID of the model
`--filterModelName`		model name pattern (supports `?` and `*` wildcards)
`--filterModelVersion`		model version (supports `?` and `*` wildcards)
`--filterFrom`		model instance(s) initiated after TIMESTAMP, format: yyyy-MM-dd HH:mm:ss
`--filterTo`		model instance(s) initiated before TIMESTAMP, format: yyyy-MM-dd HH:mm:ss
`--filterProps`		model instance properties defined as "key1 value1 key2 value2"
`--filterExpression`		boolean expression; Only the model instances for which the expression evaluates to true are returned.
`--testFilter`		boolean expression; displays the model instances affected by the command (the command is not executed)

∗ required parameters

```
lsps-runtime/cli-tools$ java -jar lsps-cli-<VERSION>-full.jar restart \
--testFilter --host http://localhost:8080 --username admin --password admin --filterModelName "mod"
Status change candidate: #8000
Status change candidate: #18000
Status change candidate: #18001
Done.
```

4.10 Model Instance Invoke

On invoke, a Model instance attempts to continue its execution: this feature is helpful if a Model instance remains in a state where it is waiting for an event which cannot occur, typically due to manual adjustments (for example, change on timer events or variable values), you can invoke the model instance to make the instance check if it can continue.

4.10.1 Invoking a Model Instance from the Management Perspective

To invoke a model instance, go to Model Instances view, select the model instance and click the Invoke  button.

4.11 Model Instance Update

To change the model used by a running or suspended model instance, use `model update`: During model update, the underlying model is substituted with a new model and updated according the model update configuration file, which is created in PDS.

Before performing model update, make sure you have a model update configuration at hand. Also make sure that the underlying database is in the correct target state. Note that any changes to the database must be backwards compatible.

> **Important:** Model update can result in inconsistent data and corrupt model instances. Make sure to have a backup of your environment before performing model update.

> **Important:** If your changes include incompatible changes to your data type model, you need to suspend any running model instances and migrate your business database before update.

4.11.1 Updating a Model Instance from the Management Perspective

Before updating model instances, make sure the new Model (target) is uploaded. Note that only model instances that are *Running*, *Suspended*, *Updated*, or *Update Aborted* can be updated. During update, the model instance is suspended and becomes read-only. Such a model instance cannot be resumed manually.

When updating Model instances their IDs are preserved and the rules defined in the Model update configuration are applied.

Important: Old and new model must use the same version of Standard Library.

Note that you can perform model update on model instances from the Management perspective of PDS as instructed here, or alternatively from the `Management Console` or `command line with the Command Line`.

To update model instances, do the following:

1. Open the Model Instances view.

2. Click the Update ( ) button in the view toolbar.

3. In the Model Update dialog box, specify the model update configuration and data mapping file to be applied:

   - Click Workspace Resource to select a configuration/data mapping file from the current workspace.
   - Click External File to select a configuration/data mapping file from other location.

4. Click Next.

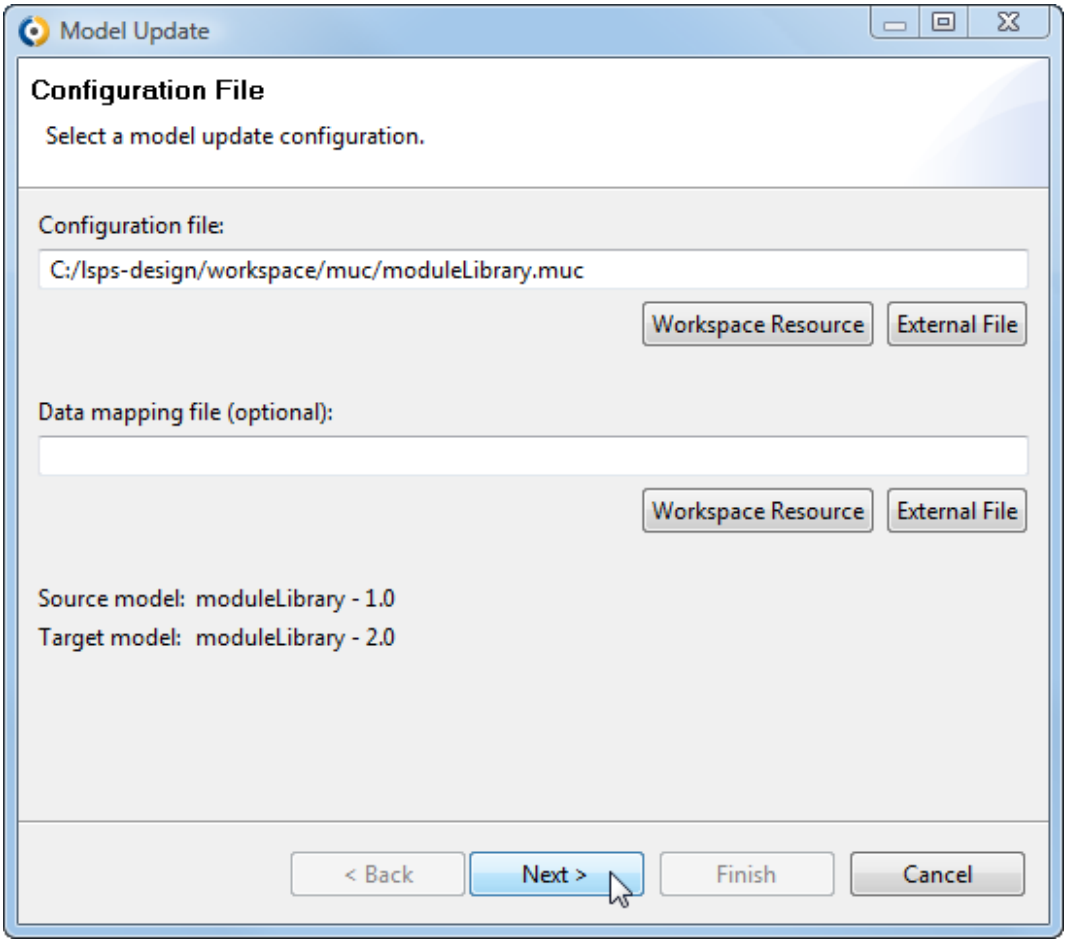


Figure 4.13 Model update dialog

5. On the Filter page, define the filtering criteria for model instances and click **Apply Filter** if applicable.

   You can apply also a previously defined filter. However, status and model criteria are ignored since the model is defined in the Model Update Configuration.

6. In the Model Instances area, select the model instances to update.

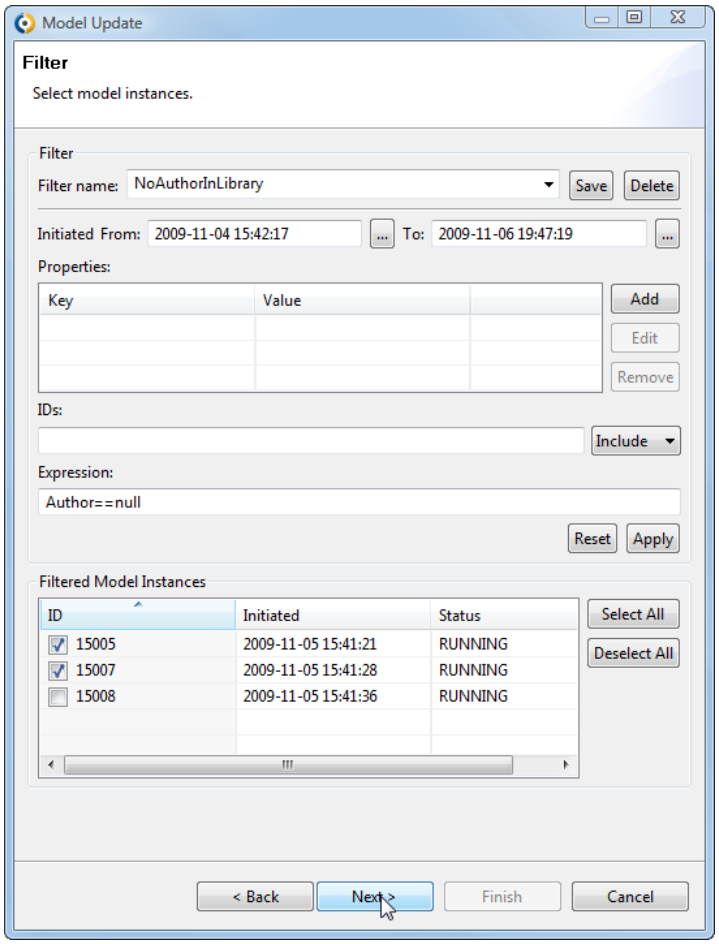


Figure 4.14 Model update dialog box with filtered Model instances

7. Click Next.
8. On the Summary page, chheck the settings. If you plan to perform the model update with the LSPS Command Line, click **Show command line script** and copy the commands for later use.
9. Click Finish.

The selected instances go through the model-update process. Check their detail views and the Model Update Logs view for details. Depending on the model update settings, the update process may be waiting for manual triggering of its transformation or post-processing phase: To resume a model update of a model instance, click the Continue Model Update ( ) button in the Model Instances view in the Management perspective.

After the update has finished, the model instances remain in the status *Updated* and are still not running. To resume the instances, click **Resume** in their context menu.

To resume all model instances returned by the Filter of the view, click **Resume All Updated** in the context menu.

> **Note:** If resuming a large number of instances, make sure the resumed action was applied to all of them. Resume any instances that failed to resume.

4.11.1.1 Aborting Model Update

You can abort model updates from the Management perspective: Click the **Abort Model Update** button ( ) in the view toolbar.

The feature is available in Management Console as well.

4.11.1.2 Model Update Logs

The logs from a model update procedure are available in the Model Update Logs view. The view displays information on source and target models and details on updated model instances (below the list of model updates).

4.11.1.3 Model Update Detail

The Model Update Detail view contains details about the update of a model instance.

To display the view, click the model instance ID in the Model Update Logs view.

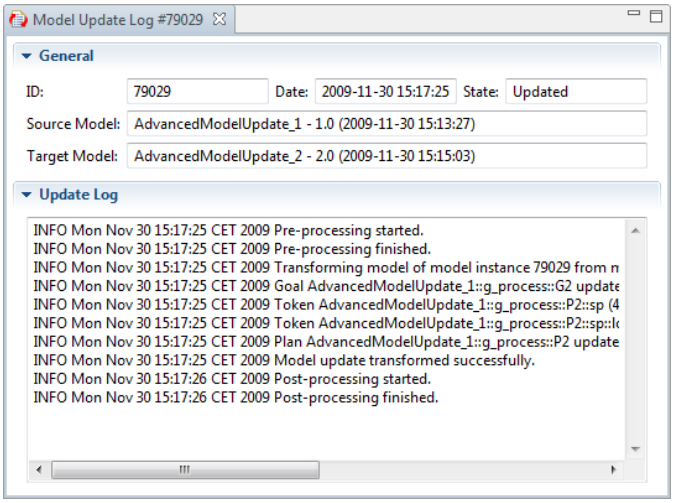


Figure 4.15 Model Update Detail

4.11.1.4 Downloading Model Update Configuration

To download a model update configuration from the LSPS Server, do the following:

1. Open the Model Update Logs view.
2. Select an update log.
3. In the view toolbar, click the Export MUC ( ) button.
4. Define the target location and file name.

4.11.2 Updating a Model Instance from Management Console

To update a model instance, do the following:

1. In the Model Instances view, select the model instance and click the Model Update button.
2. In the dialog box, enter the path to the model update MUC file and the Datamapping file and click Next.
3. Check if the correct instances are selected and click Finish
4. It is recommended to check the model update log information in the Model Update Logs view.

4.11.3 Updating a Model Instance from the Command Line

When calling any of the model update command, the command is executed by default on all model instances of the model according to the update rules defined in the muc file. However you can define the id of a model instance you want to update using the `--modelInstance` parameter. or to perform the command over multiple instances, use the filter parameters:

- `--modelInstanceFile` parameter with a file with the IDs (each ID on a new line)
- filter parameters, `--filterStatus`, `filterFrom` and `filterTo`, `--filterProps`, etc. for multiple model instances based on their properties.

Consider using the `--testFilter` parameter to trigger a dryrun and check the resulting set of model instances.

> **Note:** To acquire the model update command, you can set up a test model update and use the `Show command line script` feature in PDS.

For information on model update, refer to `Modeling`.

Example model update command

```
lsps-runtime/cli-tools$ java -jar lsps-cli-<VERSION>-full.jar modelUpdate \
--host http://localhost:8080 --username admin --password admin --muc "D:/my_workspaces/my.muc"
       --filterStatus RUNNING SUSPENDED UPDATED
```

4.11.3.1 Updating models

modelUpdate updates models instances based on the provided muc file

Long Option	Short Option	Description
`--host*`	-h	host URL
`--username*`	-u	user name
`--password*`	-p	password
`--muc*`		muc file
`--datamapping`		data mapping definition
`--modelInstance`		model instance IDs, for example, `8341 8432`
`--modelInstanceFile`		file with model instance IDs to be updated
`--filterStatus`		only model instances in the defined execution status are updated (RUNNING, SUSPENDED, UPDATED, UPDATE_ABORTED)
`--filterFrom`		model instance(s) initiated after TIMESTAMP, format: yyyy-MM-dd HH:mm:ss
`--filterTo`		model instance(s) initiated before TIMESTAMP, format: yyyy-MM-dd HH:mm:ss
`--filterProps`		model instance properties defined as `key1 value1 key2 value2`
`--filterExpression`		boolean expression; Only the model instances for which the expression evaluates to true are returned.
`--testFilter`		boolean expression; displays the model instances affected by the command (the command is not executed)

∗ required parameters

4.11.3.2 Continuing Model Transformation

continueTransform triggers the transformation phase of update on the model instances;

Long Option	Short Option	Description
`--host*`	-h	host URL
`--username*`	-u	user name
`--password*`	-p	password
`--modelInstance`		model instance IDs, for example, `8346 8436`
`--modelInstanceFile`		file with the model instance
`--filterModelId`		ID of the model
`--filterModelName`		model name pattern (supports `?` and ∗ wildcards)
`--filterModelVersion`		model version (supports `?` and ∗ wildcards)
`--filterFrom`		model instance(s) initiated after TIMESTAMP, format: yyyy-MM-dd HH:mm:ss
`--filterTo`		model instance(s) initiated before TIMESTAMP, format: yyyy-MM-dd HH:mm:ss
`--filterProps`		model instance properties defined as `key1 value1 key2 value2`
`--filterExpression`		boolean expression; Only the model instances for which the expression evaluates to true are returned.
`--testFilter`		boolean expression; displays the model instances affected by the command (the command is not executed)

∗ required parameters

4.11.3.3 Continuing Post Processing of Model Update

continuePostprocess triggers the postprocess phase of model update;

Long Option	Short Option	Description
`--host*`	-h	host URL
`--username*`	-u	user name
`--password*`	-p	password
`--modelInstance`		model instance IDs, for example, `8355 8443`
`--modelInstanceFile`		file with the model instance
`--filterModelId`		ID of the model
`--filterModelName`		model name pattern (supports `?` and ∗ wildcards)
`--filterModelVersion`		model version (supports `?` and ∗ wildcards)
`--filterFrom`		model instance(s) initiated after TIMESTAMP, format: yyyy-MM-dd HH:mm:ss
`--filterTo`		model instance(s) initiated before TIMESTAMP, format: yyyy-MM-dd HH:mm:ss
`--filterProps`		model instance properties defined as `key1 value1 key2 value2`
`--filterExpression`		boolean expression; Only the model instances for which the expression evaluates to true are returned.
`--testFilter`		boolean expression; displays the model instances affected by the command (the command is not executed)

∗ required parameters

4.11.3.4 Aborting Model Update

abortModelUpdate aborts update of model instances;

Long Option	Short Option	Description
`--host*`	-h	host URL
`--username*`	-u	user name
`--password*`	-p	password
`--modelInstance`		model instance IDs, for example, `8308 8529`
`--modelInstanceFile`		file with the model instance
`--filterModelId`		ID of the model
`--filterModelName`		model name pattern (supports `?` and ∗ wildcards)
`--filterModelVersion`		model version (supports `?` and ∗ wildcards)
`--filterFrom`		model instance(s) initiated after TIMESTAMP, format: yyyy-MM-dd HH:mm:ss
`--filterTo`		model instance(s) initiated before TIMESTAMP, format: yyyy-MM-dd HH:mm:ss
`--filterProps`		model instance properties defined as `key1 value1 key2 value2`
`--filterExpression`		boolean expression; Only the model instances for which the expression evaluates to true are returned.
`--testFilter`		boolean expression; displays the model instances affected by the command (the command is not executed)

∗ required parameters

Chapter 5

Person Management

Persons represent users that interact with the system, such as, administrator, or participate in to-do execution, hence have access to the Application User Interface. Such persons require credentials, such as, name, login, etc., and a set of rights which define what they have access to in the application.

LSPS users are represented by *persons* with personal details and access rights to the application, which include security and modeled roles. Optionally they can define their substitutes, that is, persons who can act on their behalf (stand-ins) when the substitution is activated for the given person.

5.1 Person List

5.1.1 Listing Persons in the Management Perspective

List of available Persons is displayed in the Persons view. By double-clicking a person entry in the Persons view, you can display the person detail view with the person's data and settings.

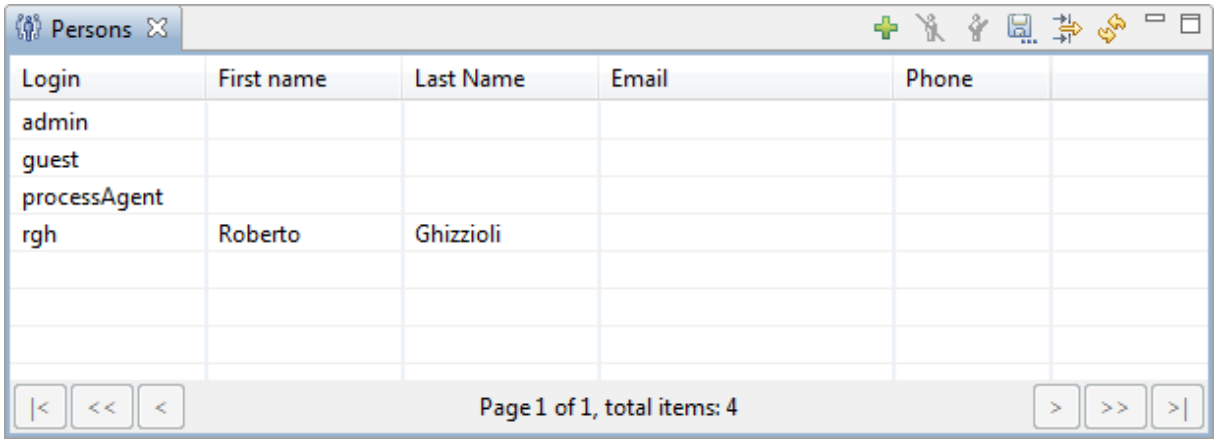


Figure 5.1 Persons view

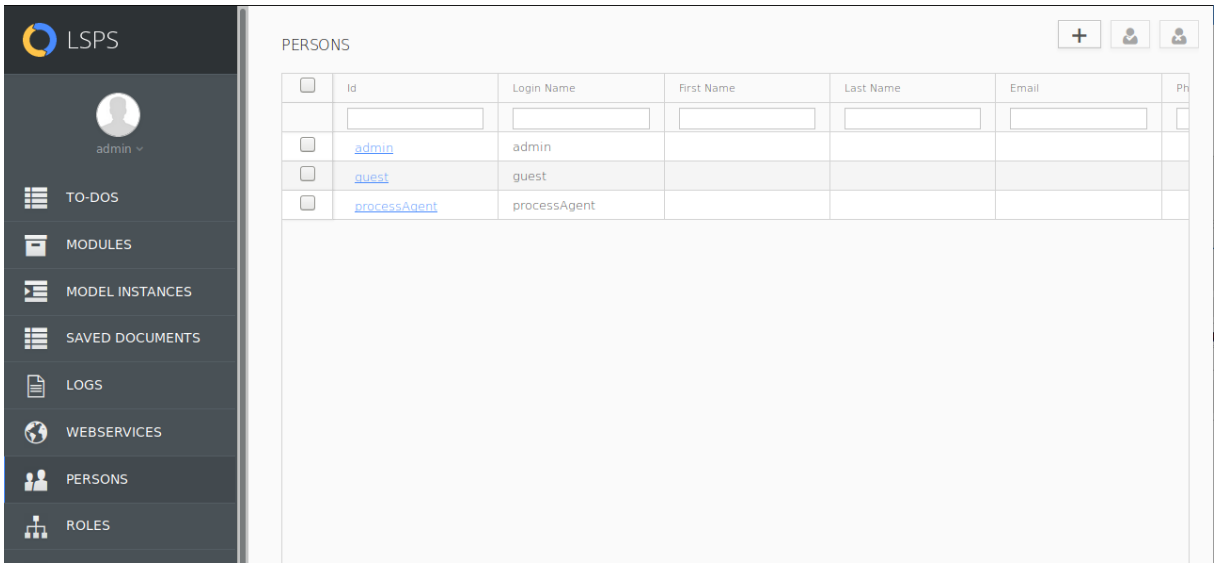
5.1.1.1 Exporting Persons List

To export the person data into an csv file, do the following:

1. Open the **Persons** view.
2. Filter the data so that only the entries you want to export are displayed.
3. Click the **Export Displayed Content to CSV** button  .

5.1.2 Listing Persons in Management Console

You can view and manage all the details on the *Persons* page.



5.1.2.1 Filtering Persons

The Persons page is displayed.

To apply a filter, do the following:

1. In the filtering row of the table, define the filtering criteria.

   Filtering is case-sensitive and supports wildcards (*, ?); for example, filtering `Task?` `*` returns results containing the word `Task` followed by any character, a space, and any subsequent characters.

2. Press Enter.

5.2 Person Creating

5.2.1 Creating a Person in the Management Perspective

To create a new person, do the following:

1. Open the **Persons** view.
2. Click the **Add** button in the view menu.
3. Enter the data in the displayed person detail.
4. Click the **Save** button .

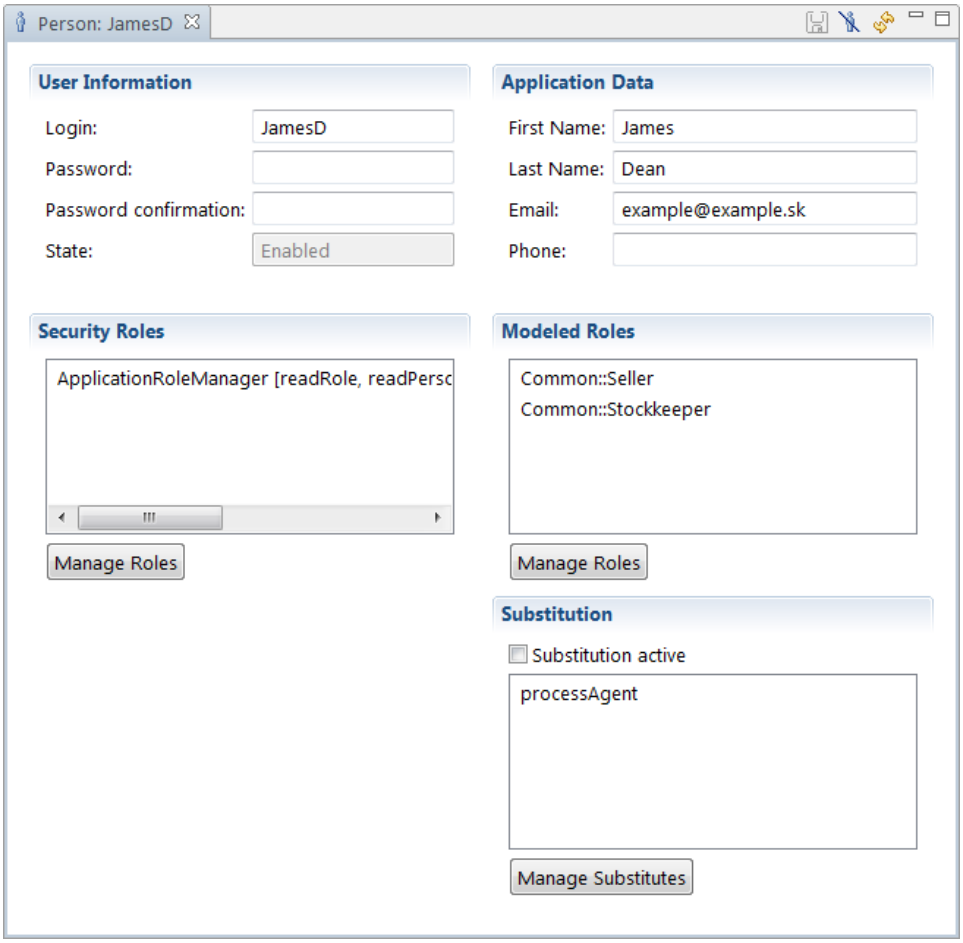


Figure 5.2 Person detail view

5.2.2 Creating a Person in Management Console

To create a new person, do the following:

1. Open the Persons view.

2. Click the Add  button in the view toolbar.
3. In the new view, define user details.
4. Optionally, define substitution settings: select Substitution active and select the substitutes.
5. Select the modeled roles you wish to assign the person.
   Assign a person at least one security role.
6. Click Submit  in the view toolbar.

5.3 Person Editing

5.3.1 Editing a Person in the Management Perspective

To edit a person's details in the Management perspective, do the following:

1. Open the Persons view: go to *Window* > *Show View* > *Persons*.
2. Double-click the login of the person.
3. In the view with person detail, edit the user information.
4. Define substitution settings.
5. Manage the roles of the person.
6. Click the Save  in the view toolbar.

5.3.2 Editing a Person in Management Console

To edit a person's details in Management Console, do the following:

1. Open the Persons page.
2. Click the login of the person.
3. In person detail, click Edit  in the upper-right corner.
4. Edit the displayed user information.
5. Optionally, define substitution settings.
6. Select the roles to be assigned to the person.
7. Click Submit  in the view toolbar.

5.3.3 Change Person's Password from the Command Line

updatePerson updates person's password

The server must be restarted to apply any password changes.

Long Option	Short Option	Description
`--host*`	-h	host URL
`--username*`	-u	user name
`--password*`	-p	password
`--new-password*`	-np	new password

* required parameters

```
java -jar lsps-cli-<VERSION>-full.jar updatePerson -h http://localhost:8080 -u admin -p admin -np admin2
```

5.4 Person Disabling

Once a person is created, it is not possible to remove them. Instead of removing a person from the system, you can disable them: A person that was disabled, cannot access the LSPS Server.

When persons are disabled, the following applies:

- Any assigned security and modeled roles, and substitutes, if defined, are permanently removed.
- Any locked to-dos are unlocked and released, and assigned to another person with the respective roles. Note that if no such person is available, the to-do becomes orphaned.

A disabled person can be recovered (enabled). However, none of the removed data is retrieved.

> **Note:** The *admin* and *processAgent* cannot be disabled.

5.4.1 Disabling and Enabling Persons in the Management Perspective

Information about person state is shown in the respective person detail view under the User Information area. In the Persons view, disabled persons are shown highlighted in red.

To enable or disable a person, do the following:

1. Open the Persons view.
2. Select the person.
3. In the view toolbar, click Disable ( ) or Enable ( ).

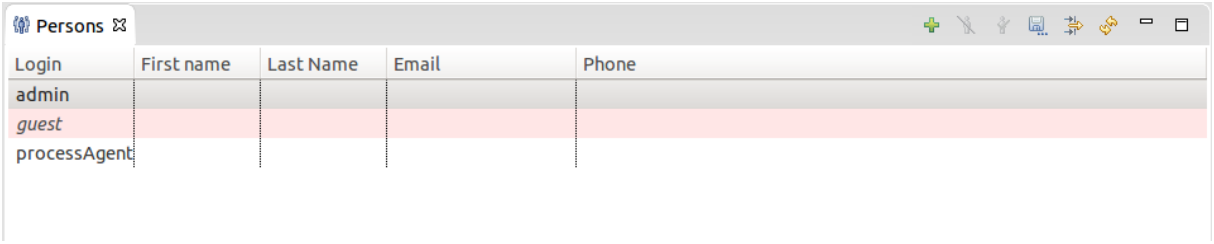


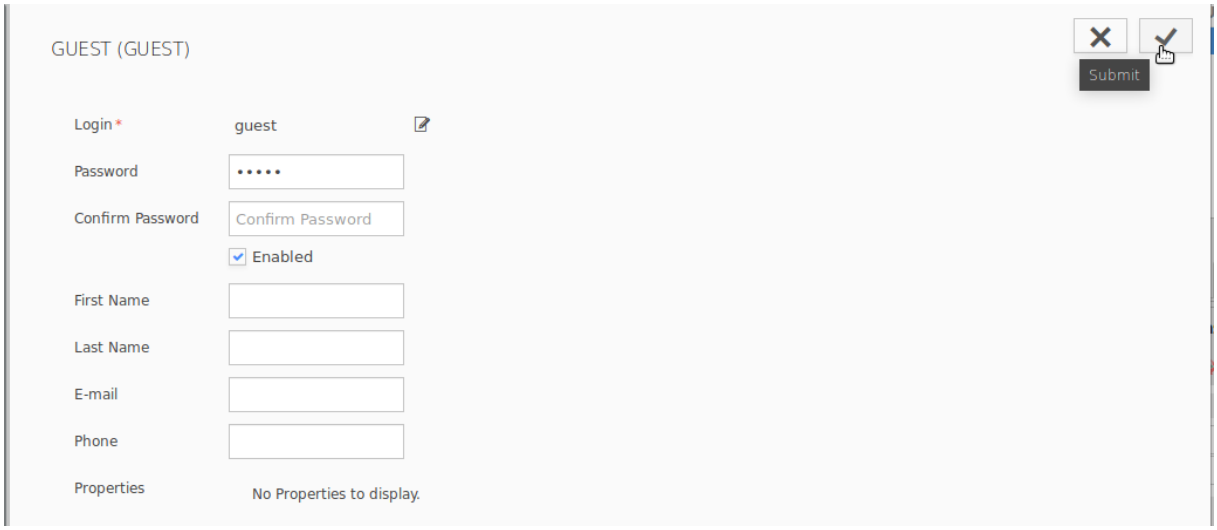
Figure 5.3 Persons view with the guest person disabled

5.4.2 Disabling and Enabling Persons in Management Console

To enable or disable a person, do the following:

1. Open the Persons page.
2. Click the id of the person.
3. Click the Edit button in the upper-right corner.
4. Select or unselect the Enabled option.

Note: The *admin* and *processAgent* cannot be disabled.



5.5 Person Substitution

Important: The built-in substitute mechanism is provided as a base for custom substitution mechanism.

The substitution mechanism serves to define stand-ins for persons: when activated any unlocked to-dos assigned to the user are assigned also to all their substitutes. The to-dos are displayed in the to-do lists of both the users and their substitutes.

Note: If a substitute lacks a modeled role required by the to-do, they cannot see the content of a to-do.

When substitution is deactivated, all to-dos listed in the substitute's to-do list which were originally allocated to the substituted person and are not locked by the substitute are removed from the to-do list of the substitute and only displayed in the to-do list of the person.

Substitution is transitive: If a substitute has activated their substitution, the substitution also applies the to-dos assigned as a result of the previous substitution activation.

When substitution is deactivated, any unlocked to-dos listed in the substitute's to-do list are removed; locked to-dos remain in the to-do list.

5.5.1 Defining Substitutes in the Management Perspective

To define a person's substitutes, open the Person Detail view and in the Substitution section, click **Manage Substitutes**. In the pop-up define the substitutes.

Check if the Substitution is activated for the person.

5.5.2 Activating and Deactivating Substitution in the Management Perspective

The Substitution feature can be applied from the front-end application as well as from PDS. To activate or deactivate substitution on a person from PDS, open the Person Detail view and select or unselect the **Substitution active** option.

5.5.3 Defining Substitutes in Management Console

To define substitutes of a person, do the following:

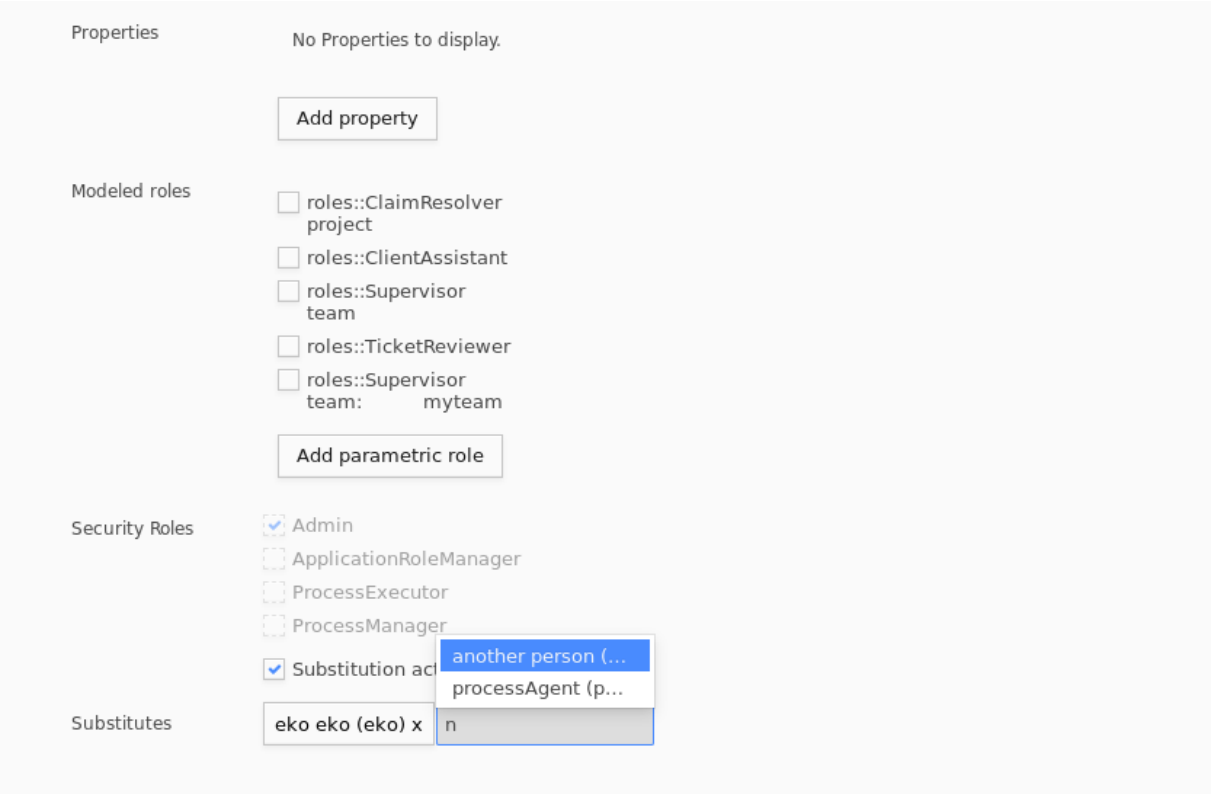
1. Click Persons in the navigation bar.
2. In the persons table, click the person‘s login.
3. On the person detail page, click the Edit  button.
4. Select the Modeled roles to assign a button.
   Editable person detail appears.
5. Select *Substitution active*.
6. On the Select Substitutes for Person page, type the search criteria into the search field and click **Search**. You can use wildcards to display persons: To display all persons, type $*$.
7. Select a substitute in the text box and click Add.
8. Activate substitution if required: select Substitution active.
9. Click OK.

5.5.4 Activating and Deactivating Substitution in Management Console

You have defined substitutes for the particular person.

To activate or deactivate substitution, do the following:

1. Open the respective person detail (Persons USER_ID)
2. Select or unselect *Substitution active*.
3. When enabling substitution, define the substitutes in the token field below.
4. Click **Submit**  .


Properties
No Properties to display.
Add property
Modeled roles
roles::ClaimResolver
project
roles::ClientAssistant
roles::Supervisor
team
roles::TicketReviewer
roles::Supervisor
team: myteam
Add parametric role
Security Roles
Admin
ApplicationRoleManager
ProcessExecutor
ProcessManager
Substitution act
another person (...
processAgent (p...
Substitutes
eko eko (eko) x
n

Chapter 6

Role Management

Modeled roles represent groups of persons that have the same responsibilities in a process. They are defined as part of a GO-BPMN Module. Once uploaded, they can be assigned to persons.

A typical usage of Roles is represented by To-Do performers: you design an organization model with a role. Then you create a process with a User Task. The User Task defines its Performer parameter as a role of organization model, for example, `Admin()` or `ClaimOwner`. The generated to-do is allocated only to persons with the runtime role `Admin`. Note that if you add a role to a person later during execution, the to-do is allocated to this person as well.

A role is defined by the module name and its own name, *not* the module version. As a result, an organization element can define multiple parameters, which can originate from different module versions with the given role.

- If another version of the same module with an updated organization definition is uploaded, the role is identified as the same role as the one created by the previous module version.
- If another version of the same module contains the same role with a new parameter name, the parameter name is added to the already-existing parameter names of the modeled role.

More information on roles and organization units is available in the GO-BPMN Modeling Language Specification.

6.1 Parametric Roles

Parametric roles are modeled roles with one or more parameters defined. A runtime role created based on a parametric role can define a value of such a parameter.

If a runtime role has no parameter value, the role is considered superior to a role with a parameter value; if a runtime role has a parameter value, the runtime role is considered inferior to the runtime role without a parameter value.

> **Example:** Consider a parametric modeled role, Developer, with a parameter, Language.
>
> There are three persons in the system: Javan, Cee, and Deve. Each has the Developer runtime role but with a different Language value: Javan has the Developer role with the parameter Language set to Java, Cee with the parameter Language set to C, and Deve with no parameter value.
>
> When a process dispatches a to-do with the initial performers set to the role Developer with parameter Java, the to-do will appear in to-do lists of Javan and Deve – the person with the role with no parameter value receives to-dos assigned to the role regardless of the required parameter value. However, since Cee has the parameter set to C, he will not receive the to-do.

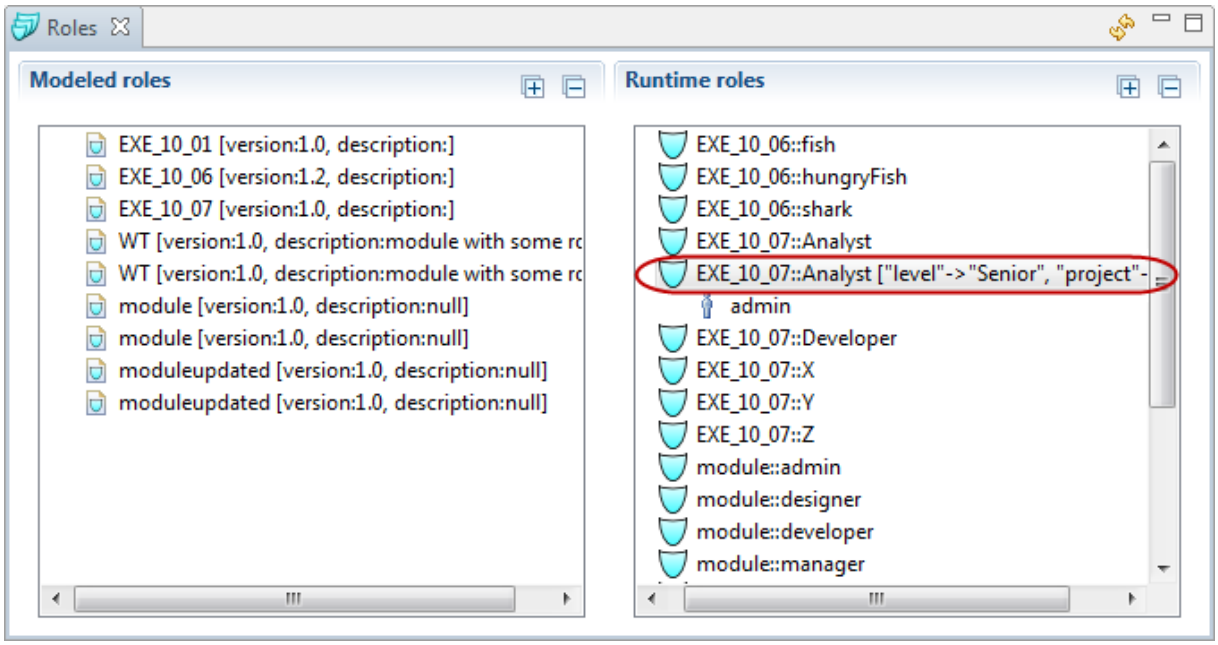


Figure 6.1 Roles view with a parameter runtime role outlined

6.2 Runtime Roles Management

6.2.1 Managing Runtime Roles from the Management Perspective

To assign a runtime role to one or more persons, do the following:

1. Open the Roles view.
2. Under Runtime Roles, right-click the respective role and select Manage Persons. If the modeled role was not assigned to any persons yet, it is not available. Go to the respective person detail and assign the person at least one modeled role.
3. In the Filter box of the Persons dialog box, type the person's name or its part. To display all available persons, type ∗ (asterisk).
4. Select a person in the box below Filter and click the right two-headed arrow to add the person to the Resulting selection.
5. Click OK.

6.2.1.1 Assigning Parametric Roles from the Management Perspective

To define a parameter value of a modeled parametric role:

1. Open the respective person detail view.
2. Under Modeled Roles, click Manage Roles.
3. In the Resulting selection area, select the parametric role.

4. Edit parameter values and click Add Role.

The added parameter value is shown in the Resulting Selection.

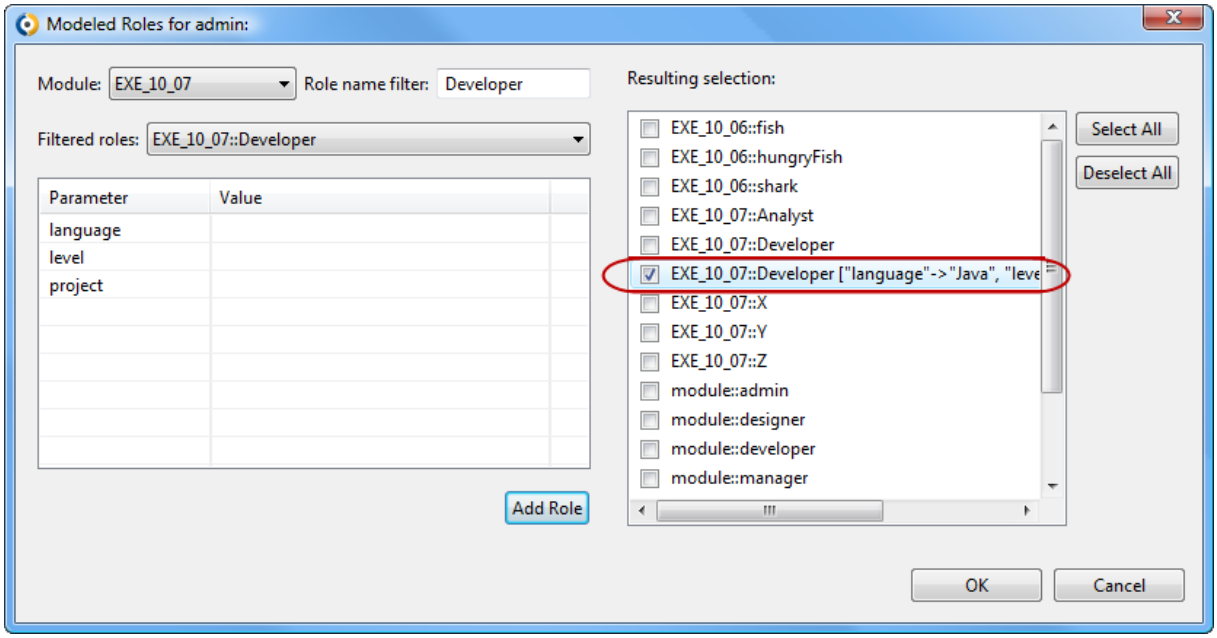


Figure 6.2 A parametric role with a parameter

6.2.2 Assigning a Role to multiple Persons from Management Console

To assign a role to multiple persons, do the following:

1. Open the Roles view and select on of the tabs with roles.
2. Click the role name.
3. In the displayed role detail, click Edit  .
4. Select the persons and click Submit  button.

Figure 6.3 Assigning a runtime role to multiple persons

6.2.3 Assigning a Role to a Person from Management Console

To assign a person a role, do the following:

1. Open the Persons view.
2. Click the person id.
3. Click the Edit  button.
4. Select the Modeled roles to assign as runtime roles to the person.

   To assign the modeled roles with a parameter specified, click the *Add parametric role* button and enter the details in the popup.
5. Click the Submit  button.

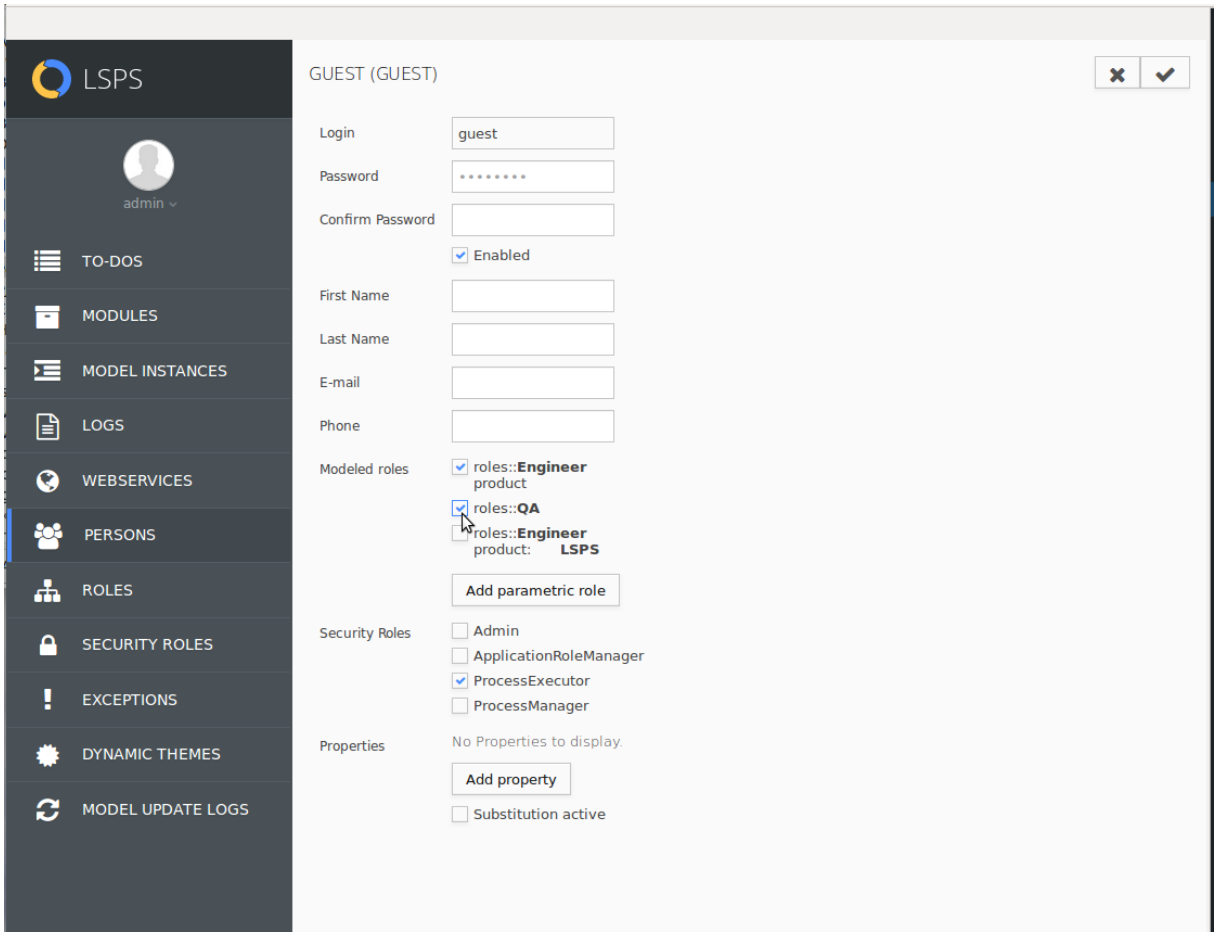


Figure 6.4 Assigning multiple roles to a person

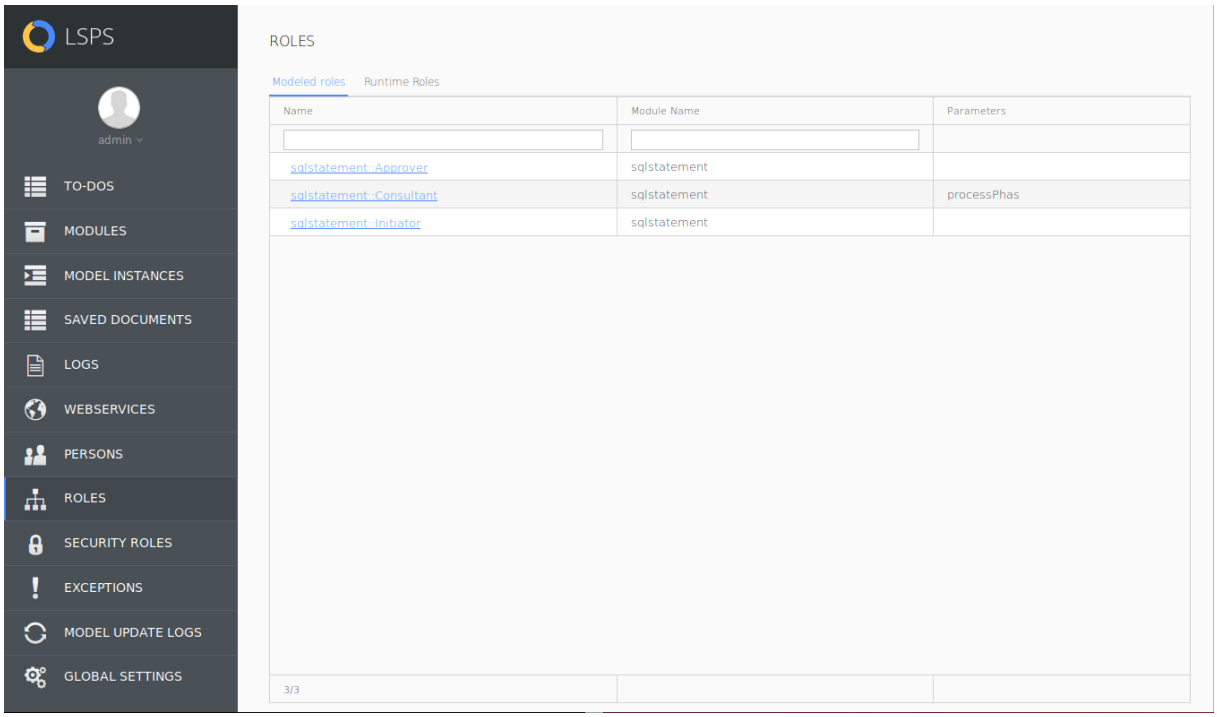


Figure 6.5 Runtime Roles tab

Chapter 7

Security Role Management

Security roles serves to control access of users to management and application features, such as, running new model instances, changing model instance context or updating passwords. Access to each feature is governed by a security right. A person can access a management feature only if they have a security role, which has the security right. Otherwise, the respective feature is inaccessible to the user regardless of whether they use the Management perspective of PDS, Management Console, or the Command Line.

The following predefined security roles are available:

- Admin: unrestricted access to management features The security role is read-only and cannot be deleted.
- ApplicationRoleManager: management of persons, and security and model roles
- ProcessExecutor: no access to management features; only to the Application User Interface
- ProcessManager: management of model instances

When you delete a security role, its assignments are removed from the users as well.

> **Important:** If a person does not have a security right for an action, the result is absence of the respective GUI components in the Application User Interface; for example, if a person does not have a security role with the right `Todo:Read_Own`, the To-do List navigation item will not be displayed in the application when the person is signed in.

7.1 Security Role Management

You can create, modify, and delete security roles.

7.1.1 Creating and Editing a Security Role from the Management Perspective

To create a security role, do the following:

1. Open the Security Management view.
2. In the Security Management view, click Add ( ) or Edit ( ) in the view toolbar.
3. In the Role Name text box of the Security Role Definition dialog box, type the security role name.
4. Select the security rights to assign to the role, and click OK.

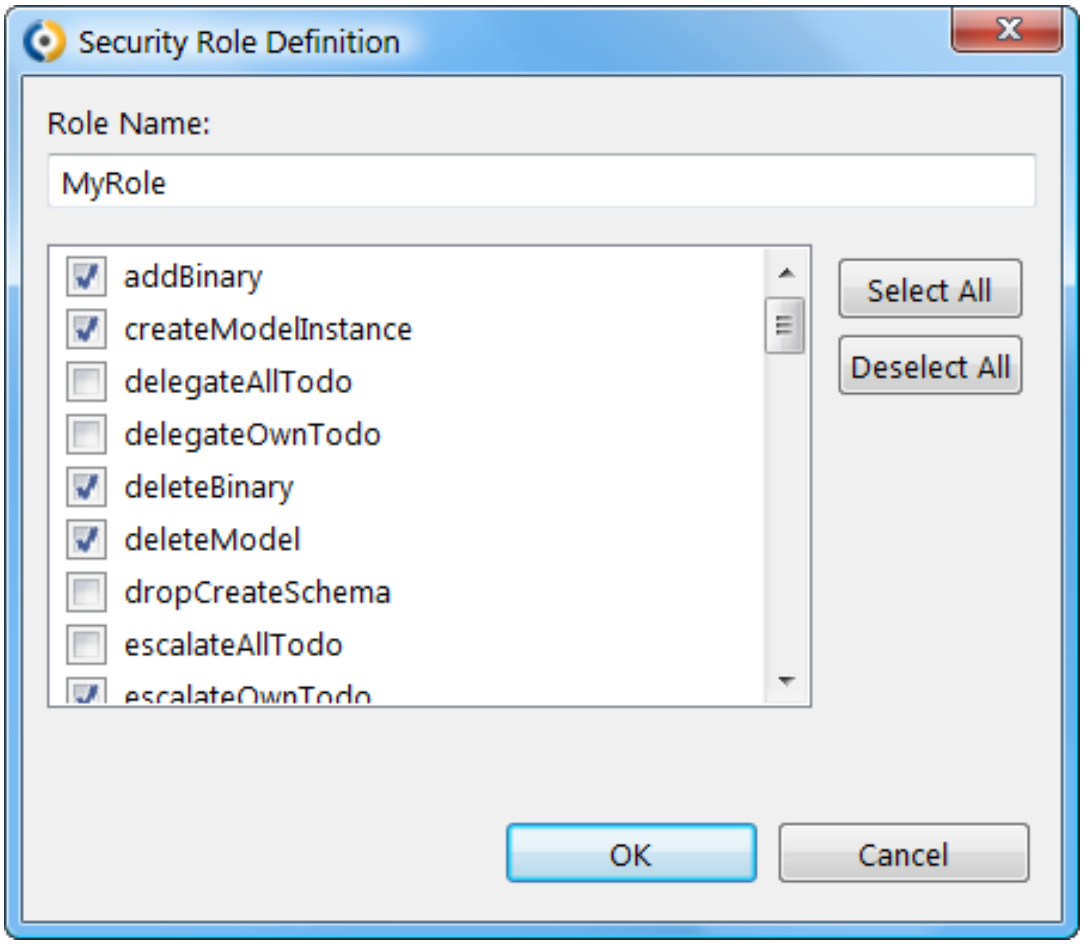


Figure 7.1 Security role definition dialog

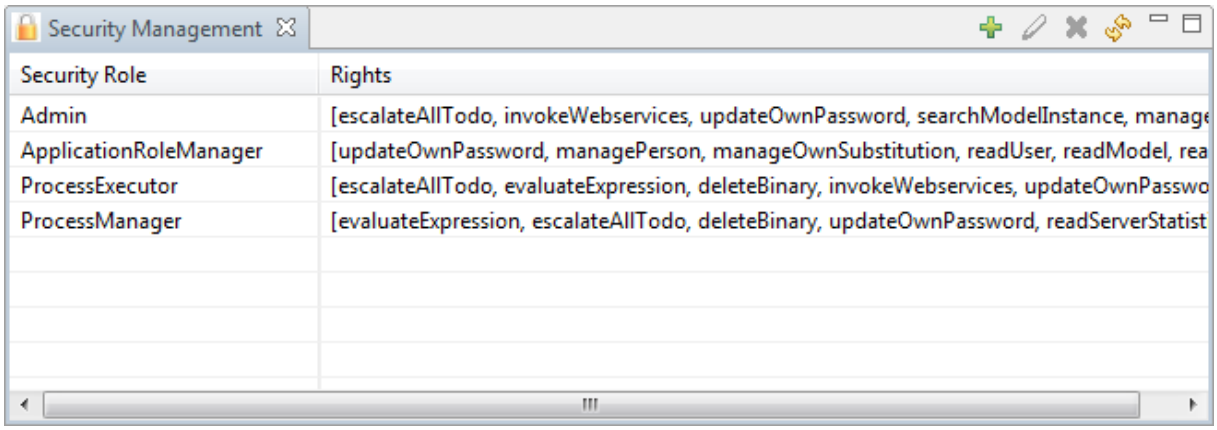


Figure 7.2 Security Management view

7.1.2 Creating and Editing a Security Role from Management Console

To create or edit a security role, do the following:

1. Open the *Security Roles* view.
2. Click **Add Role** or select a role a click Edit .
3. In the Role name text field, enter the new security role name.
4. Select the security rights for the person.
5. Click the **Submit** button.

7.1.3 Deleting a Security Role from the Management Perspective

To delete a security role, do the following from the Management perspective:

1. Open the *Security Management* view.
2. In the *Security Management* view, select the security role.
3. Click **Delete** ( ) in the view toolbar.

7.1.4 Deleting a Security Role from Management Console

To delete a security role, do the following:

1. Open the *Security Roles* view.
2. Select the security roles.
3. Click **Remove** .

7.2 Security Role Assignment

7.2.1 Assigning a Security Role from the Management Perspective

To add or remove person's security roles, do the following:

1. Display the respective person detail view.
2. Under Security Roles, click Manage Roles.
3. Select the security rights to assign to the role, and click OK.
4. Save the changes.

7.2.2 Assigning a Security Role from Management Console

To assign a person a security role, do the following:

1. On the Persons page, click person‘s login.
2. In the person detail view, click the Edit  button.
   Editable person detail appears.
3. In the Security Roles area, select the security roles of the person.
4. Click Submit  .

Chapter 8

To-Do Management

A *to-do* represents a piece of work that an end user needs to accomplish before the execution can continue: it is generated by the `User tasks`, a type of task provided in the Standard Library.

When the execution flow of a process instance enters the User task, the task generates a to-do (To make a User task generate multiple to-dos, use the `looping` of the task). The to-do has the status *Alive* and can be seen by a set of users, called assignees. At this point, the to-do assignees are a set of persons based on the Performers property of the User task referred to as *initial performers*.

If the Performers property returns a set of `Roles or Organization Units` and the administrator or the system re-assigns the Roles or Organization Units, the performers of the to-do changes accordingly.

The set of performers can change throughout the life of the todo by other mechanisms, such as escalation, rejection, etc.

The to-do is displayed in the To-Do lists of the *Application User Interface* of the performers. The moment one of the performers opens the to-do, it becomes *locked*: the to-do is no longer available to other assignees. The performer accomplishes the to-do, which becomes *accomplished* and the execution of the underlying task continues. The to-do becomes accomplished when the performer `submits the to-do`.

However, the performer might release the to-do by delegating, reset, rejecting, cancel rejection, or activate substitution: as a result of these actions, the assignees of the to-do change.

> **Important:** The to-do management mechanisms are provided as a showcase for custom to-do management features and under no circumstances to be considered production-ready.

Under some circumstances, such actions can leave a to-do to with no assignees. Such *orphaned* to-dos require the attention of an administrator: in Management Console, the administrator can search specifically for such to-dos.

8.1 To-Do List

8.1.1 Listing To-Do in the Management Perspective

To manage to-dos in the Management Perspective, use the *To-Do List* view.

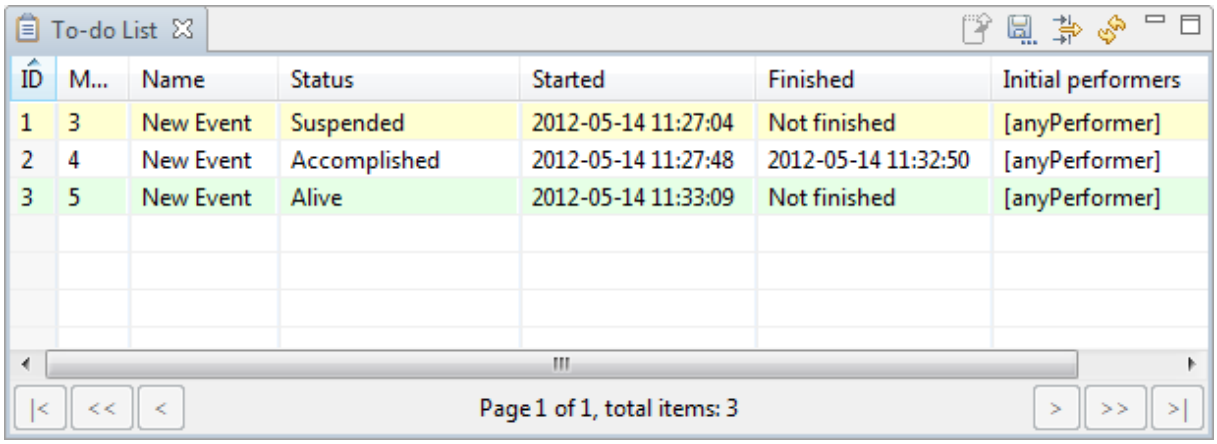
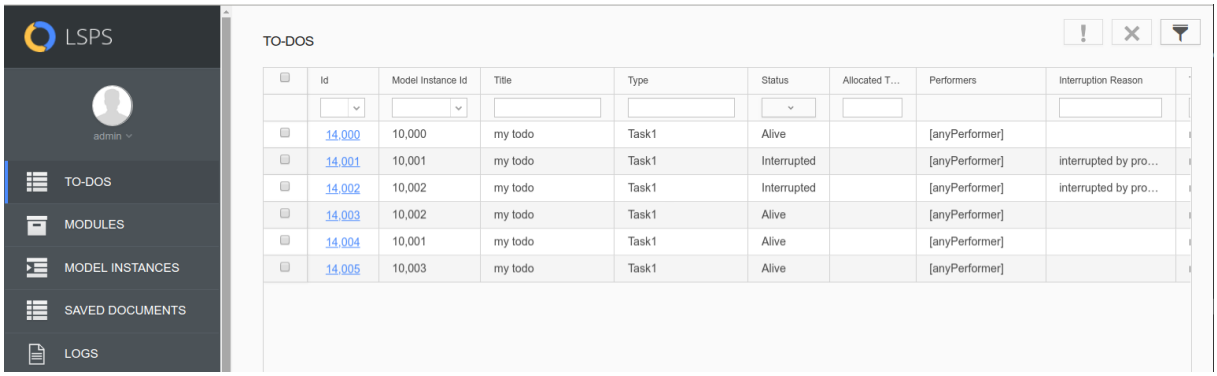


Figure 8.1 To-Do List view

8.1.2 Listing To-Do in Management Console

To manage to-dos in Management Console, go to the *To-Dos* page.



8.2 To-Do Detail

A To-Do Detail view contains detailed information about a particular to-do and allows you to perform further actions available for the to-do, such as delegation, undo delegation, and cancel rejection.

8.2.1 Opening To-Do Details in the Management Perspective

To open a to-do detail view in the Management perspective, double-click a to-do entry in the *To-do List* view.

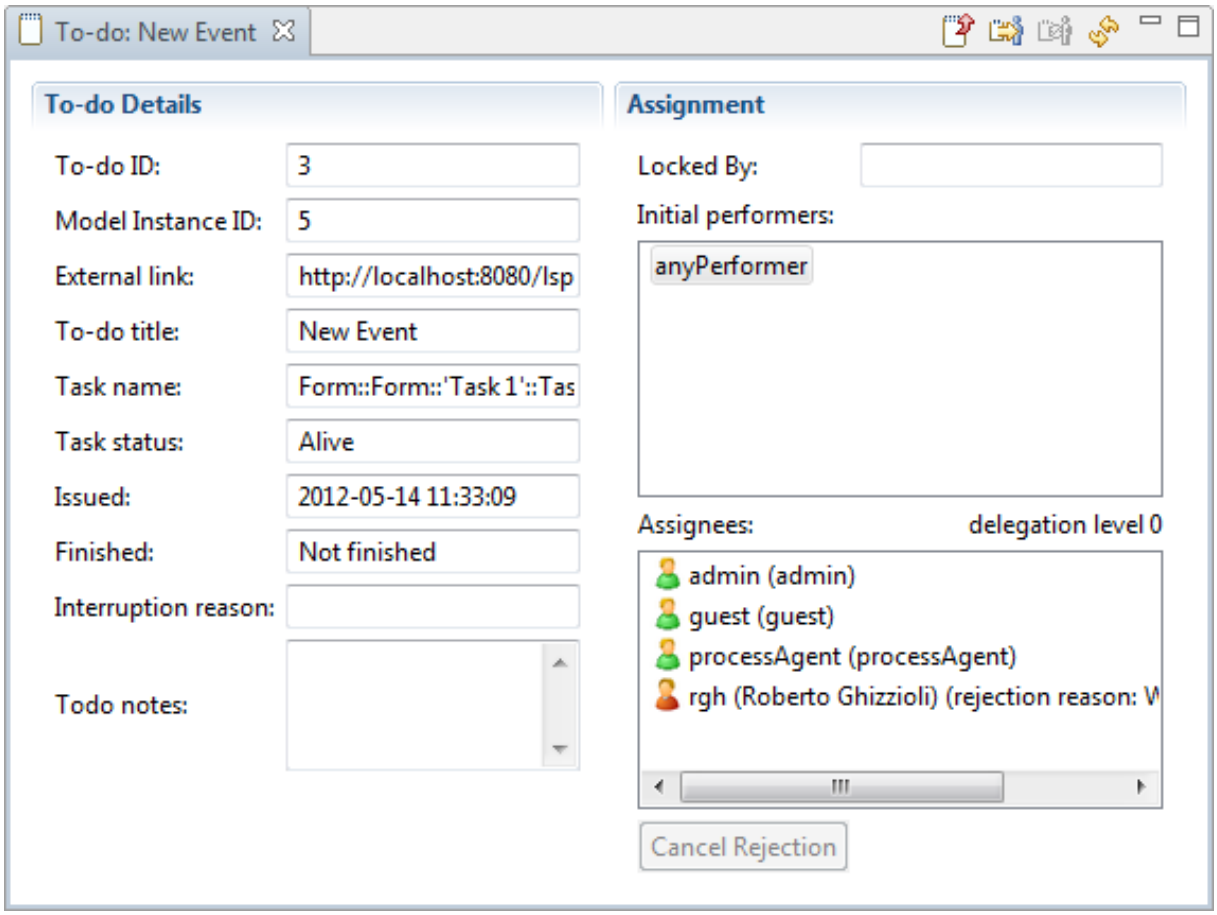
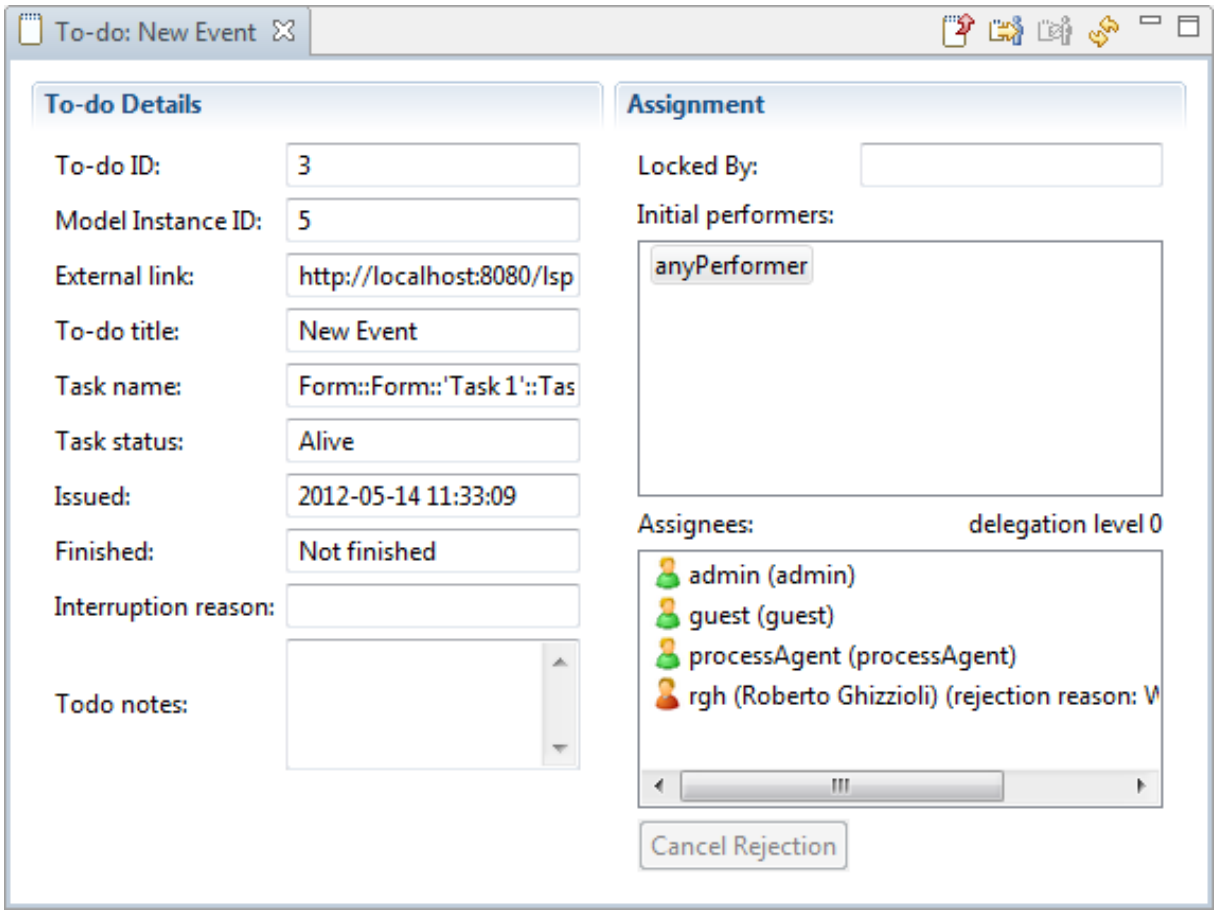


Figure 8.2 To-Do Detail view of a rejected To-Do

8.2.2 Opening To-Do Details in Management Console

To open a to-do detail view, double-click the to-do id on the *To-do List* page.



8.3 Delegation

Important: The built-in delegation mechanism is provided as a base for custom mechanism.

Delegation enables users to assign an *alive* to-do to other users, for example, to ask for assistance or simply because they know better. When a user delegates a to-do, locked or unlocked, the to-do disappears from the to-do lists of all current assignees and appears in the to-do list of the selected users–delegates.

A delegate can delegate the to-do further–delegation can take place on several levels.

8.3.1 Delegating To-Dos in the Management Perspective

To delegate a to-do from the Management perspective, In the toolbar of the To-Do Detail view, click the **Delegate** ( ) button and define the delegates.

To undo delegation, click the Undo Delegate button in the toolbar of the respective To-Do Detail view.

8.3.2 Delegating To-Dos in Management Console

To delegate a live to-do to another user, do the following:

1. Activate the respective to-do detail (To-dos To-do ID).
   Only live to-dos can be delegated.
2. Click the Delegate button.
3. Filter out the relevant users using the filter in the column of the table header.
   Wildcard usage is supported; to display all possible substitutes, type ∗.
4. Select delegates from the table and click the Delegate button.

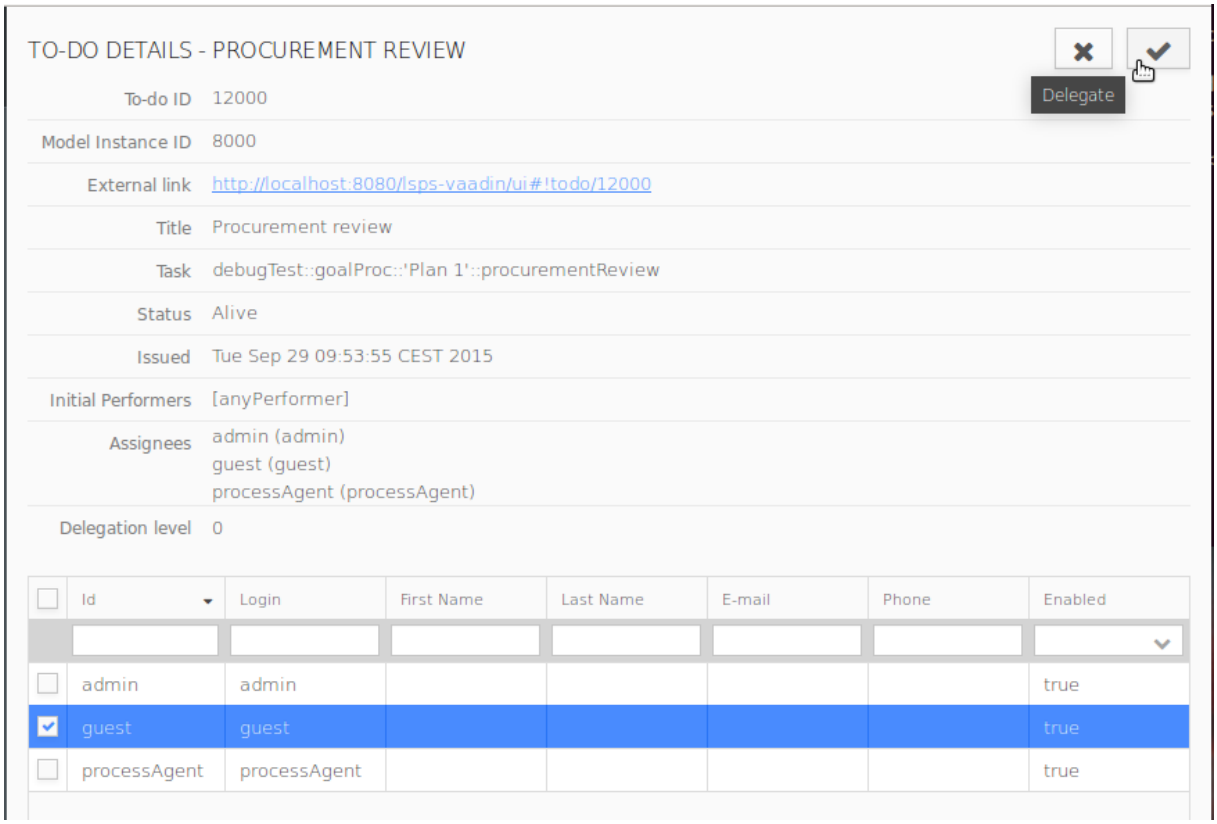


Figure 8.3 Selecting delegates

5. Click OK.

Delegation can be undone so the to-do is assigned to the previous users.

8.3.3 Undoing Delegation in Management Console

Delegation can be undone only for live to-dos.

To undo delegation of a to-do, do the following:

1. Open the to-do detail (To-dos To-do ID).
2. In the view toolbar, click the Undo Delegation  button.

Delegation is undone and the delegation level decreases. Repeat the process to undo delegation on several levels.

8.4 Escalation

To-do escalation provides a mechanism that sends a special type of Signal, to the server. The underlying Model is expected to catch the signal and process it as appropriate. The signal can be caught and processed by a catch event. If escalation handling and catching of a particular signal is not handled within the model instance, escalation takes **no effect**. The unconsumed Signals remain in the signal queue of the model instance.

> **Important:** The escalation mechanism described below is deprecated. Use the `escalation mechanism` instead; for example, call the `throwEscalation()` function on a button component and process the escalation with Catch Escalation elements.

8.4.1 Sending Escalation Signal in the Management Perspective

Escalation sends an escalation Signal. The underlying Model is expected to catch and process it as appropriate with a Signal Catch Intermediate Event or Signal Start Event.

To send such an escalation signal from a to-do from the Management perspective, click the Escalate () button in the toolbar of the To-Do Detail view or select the to-dos in the To-Do List and click Escalate.

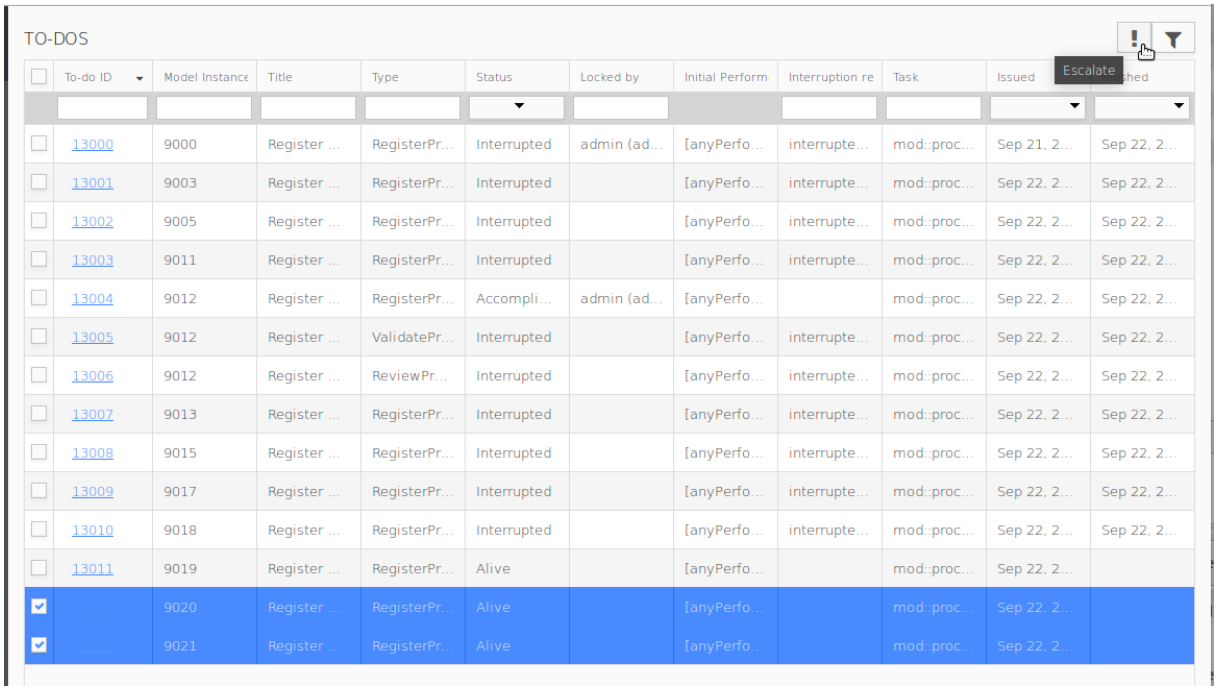
Note that this escalation mechanism is **not related** to the `Escalation of the GO-BPMN Modeling Language`.

8.4.2 Sending Escalation Signal in the Management console

To escalate one or several to-dos, do the following:

1. Open the To-dos view.
2. Select the to-dos you want to escalate.

   Only live to-dos can be escalated.
3. Click Escalate  .
4. Type escalation reason and click Escalate.

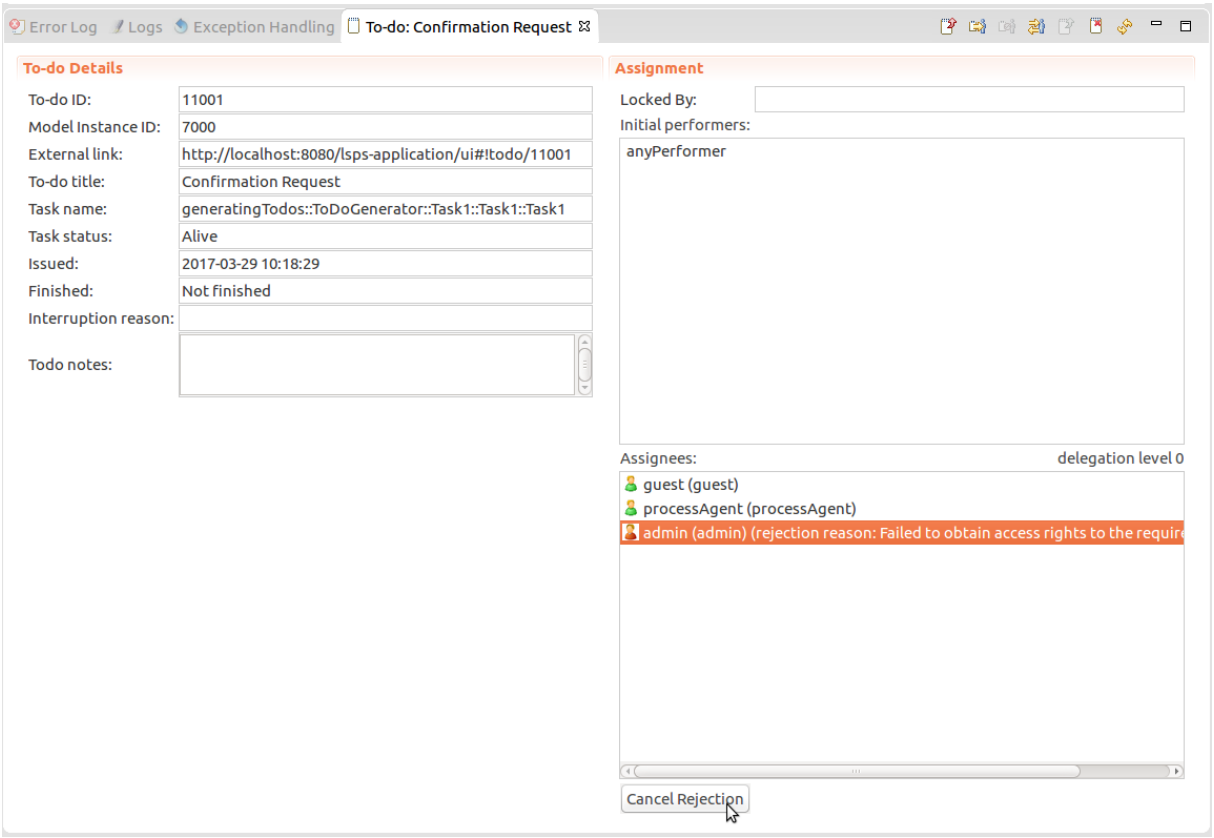


8.5 Rejection

Rejection allows a front-end user who is an assignee of a to-do to exclude herself from the assignees list and have the to-do removed from their to-do list. You can cancel the rejection, so if the to-do is not locked by other users, it will reappear in their to-do list.

8.5.1 Cancelling Rejection in the Management Perspective

To cancel rejection, open the to-do detail view and, in the Assignees box, located the assignee, who has rejected the to-do. The rejection reason provided by the assignee is shown next to their icon and name. Select the assignee and click the ∗∗Cancel Rejection∗ button below. it will reappear in their to-do list.



8.6 Resetting

To-do reset ( ) erases the data in a saved to-do while the to-do remains locked by the given user. This feature is useful if the data used by the saved to-do have changed; typically on model update. Note that the reset to-do remains locked by the user.

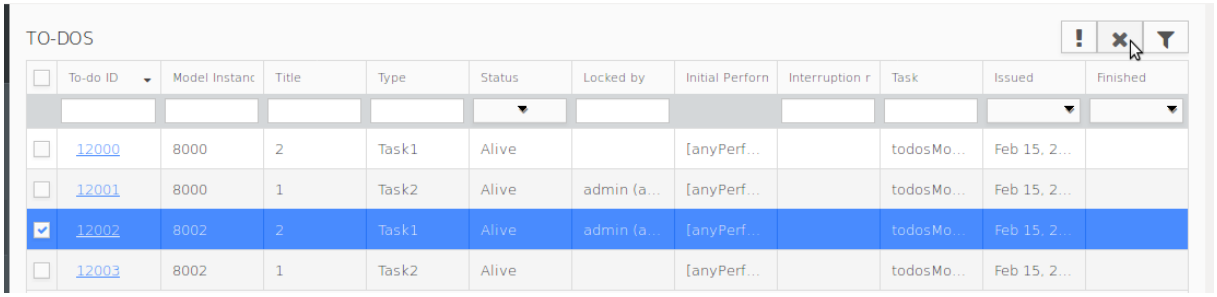
8.6.1 Resetting a To-Do in the Management Perspective

To reset a to-do and lose any save to-do data, click the to-do reset button at the top of the to-do detail view.

8.6.2 Resetting a To-Do in Management Console

To reset a live to-do and lose any saved to-do data, do the following:

1. Open the To-Dos page.
2. Select the to-dos in the list: make sure the to-do is alive.
3. Click the Reset button.



8.7 Reassignment

Important: The built-in reassignment mechanism is provided as a base for a custom mechanism and is by no means production-ready.

Reassignment enables an administrator to change the set of initial performers of a to-do. It changes the current set of performers of an Alive to-do to the new set of initial performers. The to-do disappears from the to-do lists of the current assignees and appears in the to-do lists of the new assignees. The information on original assignees are lost.

Important: Reassignment cannot be undone.

8.7.1 Reassigning a To-Do in the Management Perspective

Click the to-do reassign button at the top of the to-do detail and define the roles and persons who should become the new initial performers.

8.7.2 Reassigning a To-Do in Management Console

Reassignment enables an administrator to assign an Alive to-do to other users, who become the initial performers.

Note: Reassignment cannot be undone.

To reassign a to-do, do the following:

1. Display the to-dos detail (go to To-do List view and click a to-do ID).
2. Click the Reassign button in the view toolbar.
3. Select Persons to reassign the to-do to users or Roles to reassign the to-do to users with a runtime role.
4. In the table below, select the persons or roles.
5. In the view toolbar, click the confirmation button.

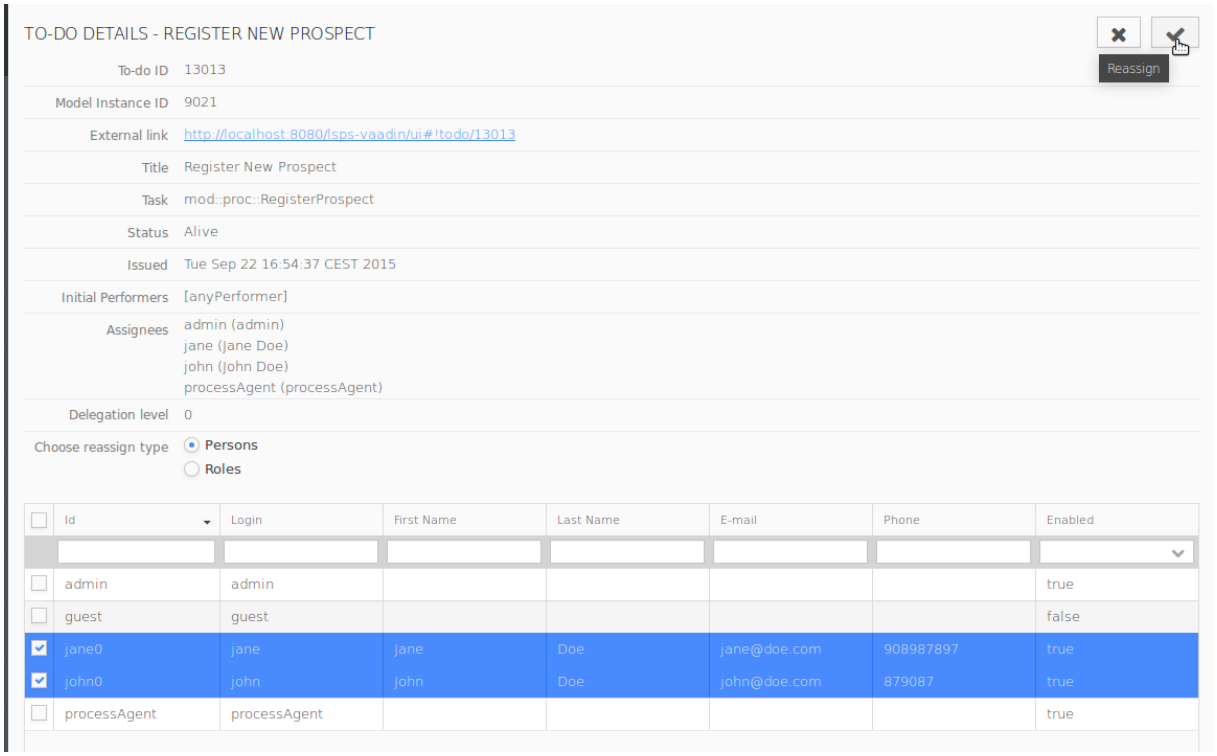


Figure 8.4 Reassigning To-Do

8.8 Export a Saved Todo and Import a Saved To-Do

If you want to modify the data of a saved to-do that are not persisted, you can do so directly in the raw XML of the model instance: Export the to-do into the raw XML, modify the data, and import the new XML back to the server.

If you modify XML data that is persisted, the persisted data remain unchanged.

Only users with the `Todo:Write_All` security right have access to this feature.

8.8.1 Exporting and Importing a Saved To-Do in the Management Perspective

To export or import the model instance XML with the to-do data, do the following:

1. Open the *To-Do List* view.
2. Find the *saved* to-do.
3. Right-click the to-do and select *Export To-Do State* or *Import To-Do State* respectively.

Important: Importing a corrupt XML can result in corrupt to-do state and in turn failure of model instance execution.

8.8.2 Exporting and Importing a Saved To-Do in Management Console

To export or import a to-do state, do the following:

1. Open the To-Dos page.
2. Click the to-do.
3. On the To-Do detail page, click the *Export To-Do State* or *Import To-Do State* respectively.

   Important: Importing a corrupt to-do state can cause the system to fail to work with the to-do.

8.8.3 Exporting and Importing a Saved Todo from the Command Line

To export the state of a saved To-Do as an XML file run the **savedTodoExport** command.

Long Option	Short Option	Description
`--host*`	-h	host URL
`--username*`	-u	user name
`--password*`	-p	password
`--todoId*`		to-do id
`--location`		output location prefix (todo id will be appended)

∗ required parameters

```
lsps-runtime/cli-tools$ java -jar lsps-cli-3.3.2039-full.jar savedTodoExport --host http://localhost:8080
       --username admin --password admin --todoId 25000 --location myTodo
Exporting todo state #25000 to myTodo25000.xml
Done.
```

To import the state of a saved To-Do into an existing To-Do, run the **savedTodoImport** command.

Long Option	Short Option	Description
`--host*`	-h	host URL
`--username*`	-u	user name
`--password*`	-p	password
`--todoId*`		todo ID
`--file*`		file with to-do state

∗ required parameters

```
lsps-runtime/cli-tools$ java -jar lsps-cli-3.3.2039-full.jar savedTodoImport --host http://localhost:8080
       --username admin --password admin --todoId 29000 --file myTodo29000.xml
```

8.9 Orphaned To-Dos

An orphaned to-do is a to-do, which cannot be seen by any front-end application user, possibly as a result of escalation, delegation, etc., for example, when a to-do was rejected by all assigned persons, or delegated to persons with insufficient security rights.

8.9.1 Searching for Orphaned To-Dos in the Management Perspective

You can search for such to-dos from the To-Do List view: the view toolbar, click Filter , and select *Orphaned to-dos only*.

8.9.2 Searching for Orphaned To-Dos in Management Console

To display orphaned To-dos in the table of the To-Do view, click the Filter button in the view toolbar; in the displayed dialog, select the Orphaned to-dos option and confirm.

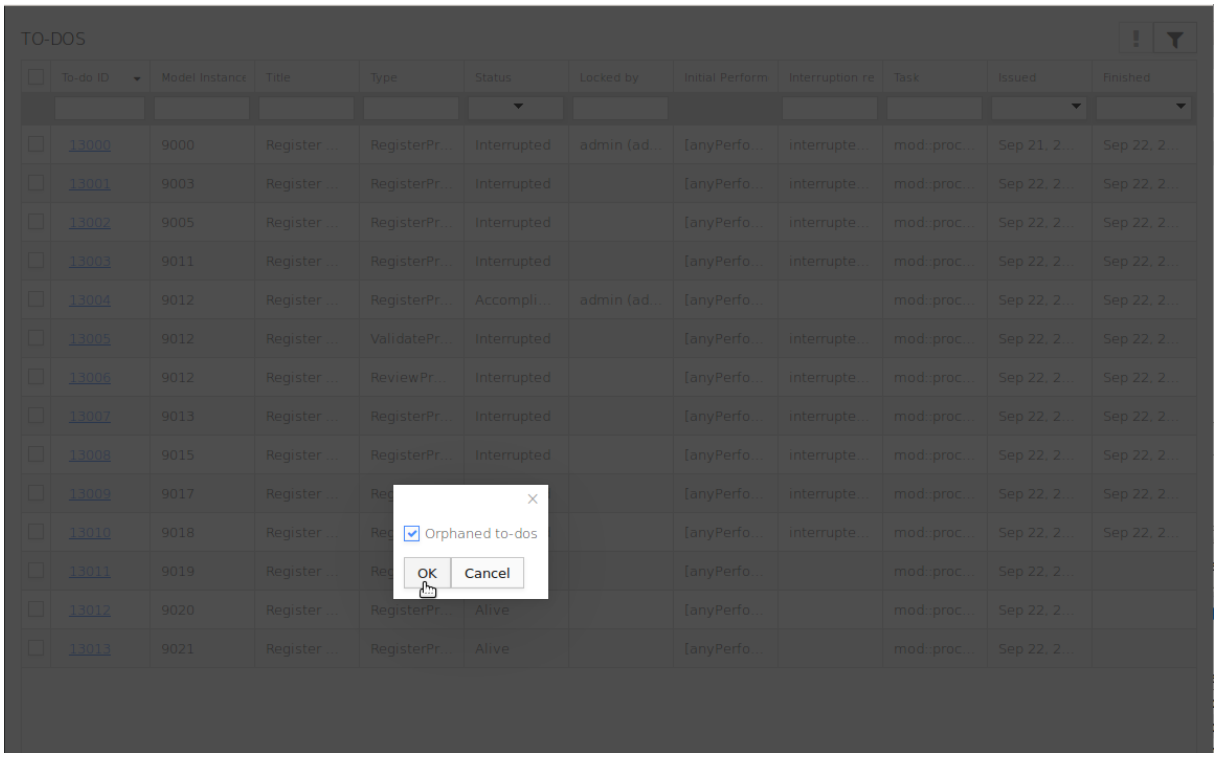


Figure 8.5 Activating orphan to-dos filtering

Chapter 9

Document Management

Documents are instances of document definitions. These are uploaded as part of modules. They serve to interact with the user via pages with business data that are not dependent on a model instance.

For more information, refer to the `Modeling guide`.

9.1 Document List

The users can save the documents they are working on. Saved documents can be exported in the XML format and imported back as well. Note that the features are subject to the security right `Document:Write`.

9.1.1 Listing Saved Document from the Management Perspective

The documents users have saved and are hence probably working on are available in the *Saved Documents* view.

9.1.2 Listing Saved Document from Management Console

The documents users have saved and are hence probably working on are available on the *Saved Documents* page.

9.2 Document Export

9.2.1 Exporting a Saved Document from the Management Perspective

To export a saved document to XML, do the following:

1. Open the *Saved Documents* view.
2. Right-click the document and select *Export Saved Document State*.

You can edit the downloaded XML and then import it back into the document.

9.2.2 Exporting a Saved Document from Management Console

To export a saved document to XML, do the following:

1. Open the Saved Documents view.
2. Click the saved document.
3. Click the *Export Saved Document State* button.

You can edit the downloaded XML and then import it back into the document.

9.2.3 Exporting a Saved Document from the Command Line

savedDocumentExport exports a saved document

Long Option	Short Option	Description
`--host*`	-h	host URL
`--username*`	-u	user name
`--password*`	-p	password

* required parameters

```
lsps-runtime/cli-tools$ java -jar lsps-cli-3.3.2039-full.jar savedDocumentExport --host
       http://localhost:8080 --username admin --password admin --savedDocumentId 33000
```

9.3 Document Import

The users can save the documents they are working on. Saved documents can be exported in the XML format: you can adapt the XML and imported it back. This allows you to manipulate the runtime data of a saved document out of the management tools. Note that the feature is subject to the security right `Document:Write`.

Important: Importing a corrupt document can cause the system to fail to work with the document.

9.3.1 Importing a Saved Document in the Management Perspective

To import a saved document XML, do the following:

1. Open the *Saved Documents* view.
2. Right-click the document and select *Import Saved Document State*.

9.3.2 Importing a Saved Document in Management Console

To import modified data of a saved document back to the server, do the following:

1. Open the *Saved Documents* view.
2. Click the *Import Saved Document State* button.

9.3.3 Importing a Saved Document from the Command Line

savedDocumentImport imports a saved document; the document must already exist in the database

Long Option	Short Option	Description
--host*	-h	host URL
--username*	-u	user name
--password*	-p	password
--savedDocumentId*		saved document id
--file*		file with saved document state

* required parameters

```
java -jar lsps-cli-3.3.2039-full.jar savedDocumentExport --host http://localhost:8080 --username admin
       --password admin --savedDocumentId 33000
```

Chapter 10

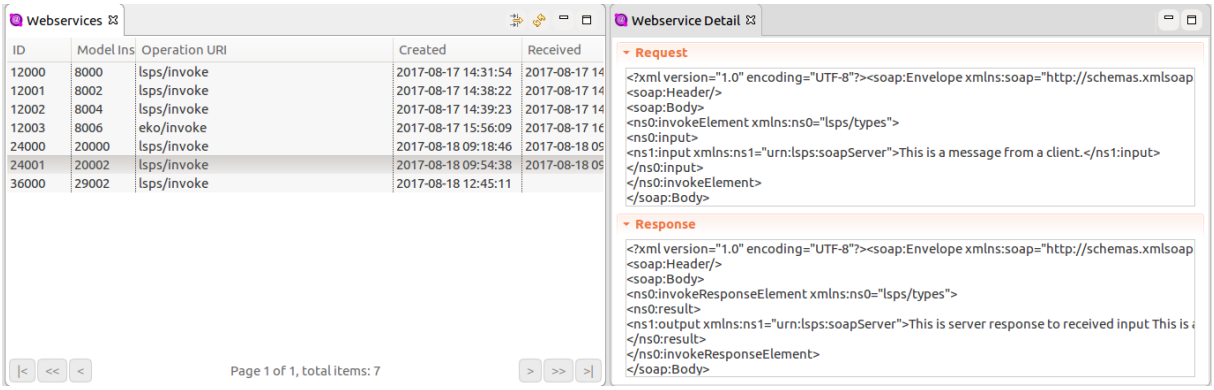
Web Service Management

10.1 Managing Web Services in the Management Perspective

You can check the status of calls to your **web-service server Processes** wait points in the *Web Services view*. The view contains a list of active and processed wait points waiting for a Web service request from a client.

If no model instance handled a Web service request, the wait point is highlighted in red.

Double-clicking a wait point entry, opens the respective **Webservice Detail** with the request and response on the wait point. Note that the request and response are available only if the underlying Task has the logXMLMessage parameter set to true since excessive request and response data could cause a significant database growth.



10.2 Filtering Wait Points in Management Console

To apply a filter, do the following:

1. Open the Webservices view.
2. In the filtering row of the table, define the filtering criteria.

   Filtering is case-sensitive and supports wildcards (*, ?); for example, filtering `Task?  *` returns results containing the word `Task` followed by any character, a space, and any subsequent characters.
3. Press Enter.

Chapter 11

Log Management

Logs are dedicated LSPS Application logs: they are created by the `Log` tasks and `log()` functions, which are part of the Standard Library and stored in system database table LSPS_LOGS.

> **Note:** Availability of logs does not depend on the CREATE_PROCESS_LOG database setting: this setting is related to the availability of module runtime data, which is displayed, for example, in the Model Instance views.

11.1 Managing Logs in the Management Perspective

You can display application log entries in the *Logs* view with the following details:

- Model instance ID: identifier of the model instance, which executed the Log task that generated the log message
- Timestamp: date and time when the log message was created
- Level: severity of the log message (info, debug, or error)
- Description: log message content

11.2 Managing Logs in Management Console

You can display application log entries on the *Logs* page.

Chapter 12

Exceptions

The LSPS Server returns an exception when a component of a model instance cannot perform the required action, for example, if data is missing, or entry data does not correspond with the expected data (a model throws an exception with the `error()` call, refer to `Exception Handling`).

When a model instance fails with an Exception, the Execution Engine rolls back the transaction that caused the exception.

Such error on model instances is recorded in the Exceptions log. For further information on transactions in model instance, refer to `Transactions in Model Instances`).

As part of the management features, you can resend inputs to such model instances.

12.1 Managing Exception from the Management Perspective

The list of such exceptions that occurred in model instances is displayed in the **Exception Handling** view with their details:

- ID: unique exception identifier
- Component: component which generated the exception
- Creation Time: time when the exception occurred
- Resend: time when the resend request was executed
- Exception: exception message

In the view toolbar in the upper-right corner, the buttons for the following actions are available:

- Remove (): removes the selected exception entry.
- Resend (): resends the request.
- Export to File (): saves one or multiple exceptions in a text file.
- Filter (): filters the content of the displayed list.
- Refresh (): refreshes the view content.

You can copy multiple rows to the clipboard with Ctrl + c.

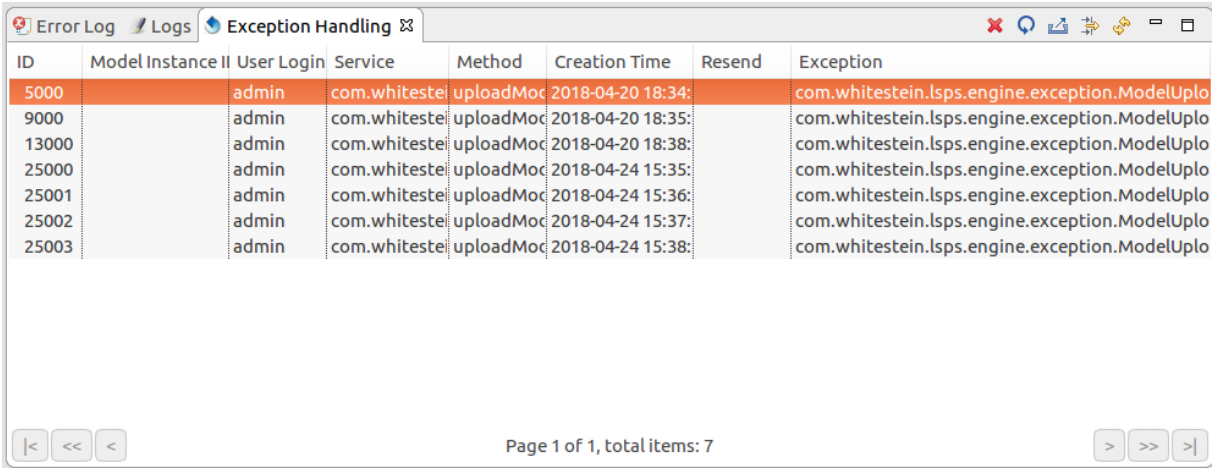


Figure 12.1 Exception Handling view

Double-clicking a table entry displays the respective exception detail view with detailed information about the selected exception and its complete stacktrace. From the detail view you can export the exception to a file: click the Export to File ( ) button in the view toolbar, to do so.

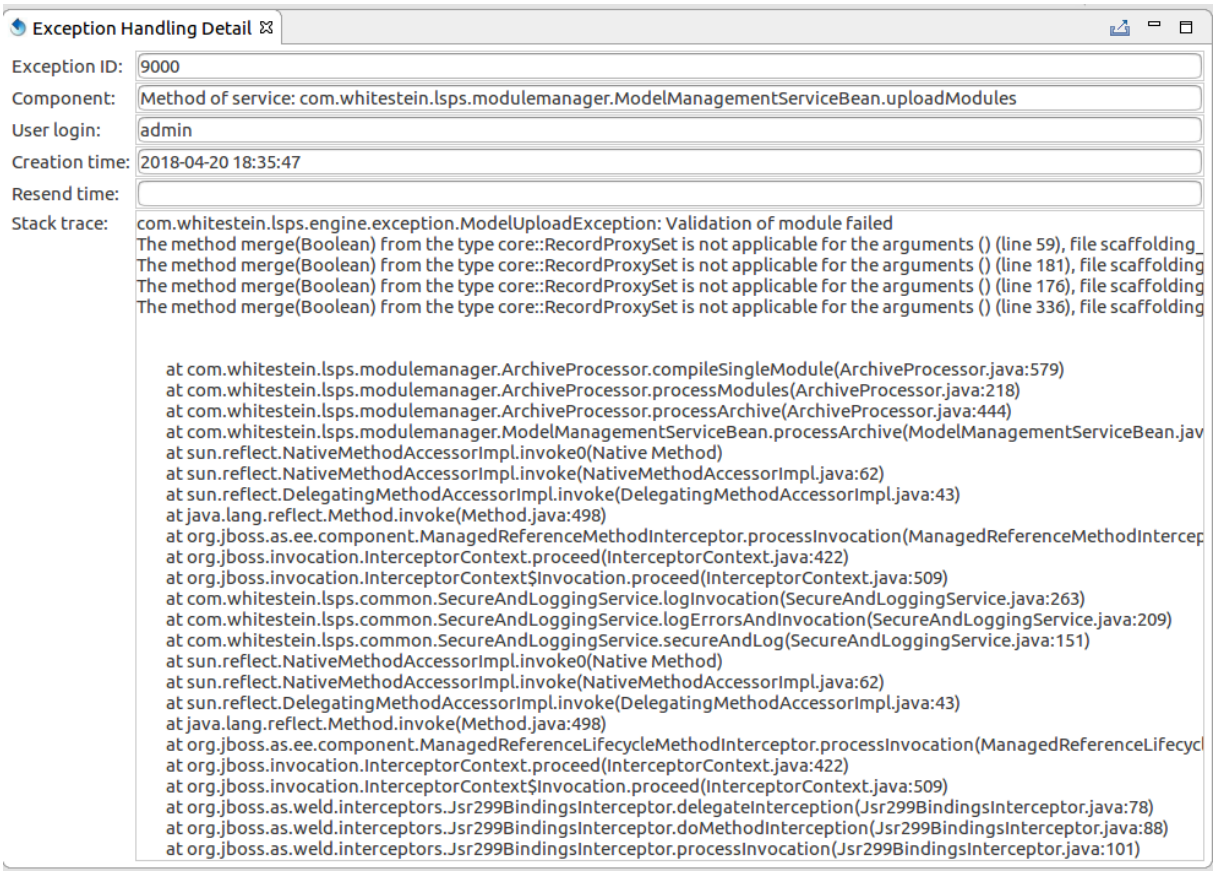


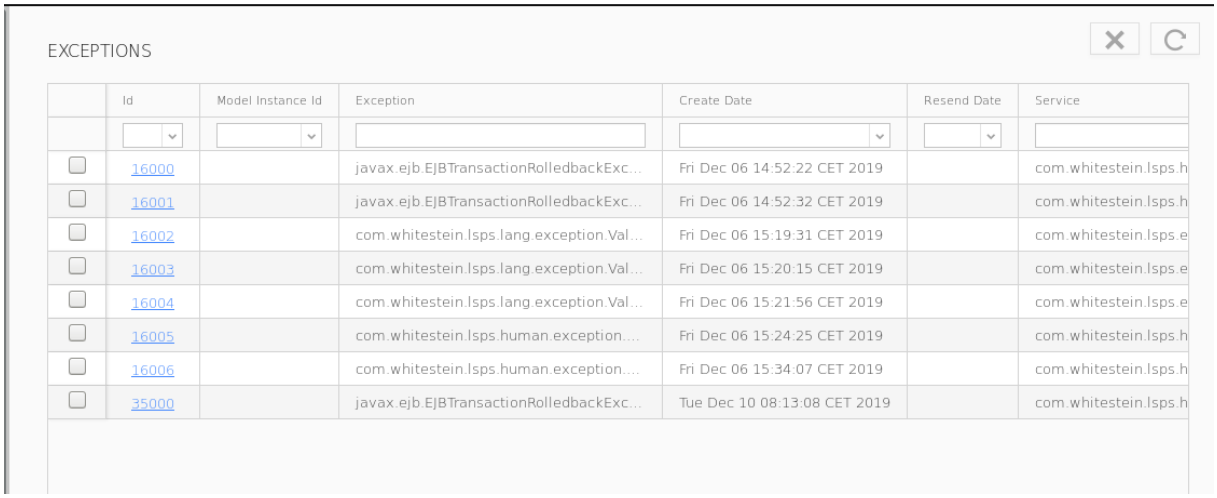
Figure 12.2 Exception detail

12.2 Managing Exception from Management Console

Exception that occurred in model instances are recorded on the Exceptions page.

From Management Console, you can do the following over Exceptions:

- display their stacktraces
- remove exceptions from the log
- resend input to the transaction



12.2.1 Displaying Exception Stacktrace

To display an exception stack trace, click the exception id on the Exception page. To display the full stacktrace, open PDS, connect to your server and inspect the exception in the Exception view of the Management perspective.

12.2.2 Removing Exceptions

To remove an exception entry, in the Exceptions view, select the exception and click the Remove button.

12.2.3 Resending Inputs on Model Instances with an Exception

Resend attempts to execute again a transaction that previously failed: on resend the system resends the inputs which caused the exception to the model instance. Before using this function you will typically fix the underlying data of the model instance or database data, or make sure that an external service was fixed, etc.

To resend the data, do the following:

1. Select the exceptions in the table.
2. Click the Resend button in the upper right corner.

Chapter 13

Server Management

You can configure the server and database via the exposed settings from the LSPS management tools as well as via MBeans properties, for example, com.whitestein.lsps.Settings -> Attributes -> confirmModelUpload or with a database insert into the LSPS_SETTINGS table, for example:

```
INSERT INTO LSPS_SETTINGS(ID, VALUE) VALUES ('CONFIRM_MODEL_UPLOAD', 'true');
```

Note that changes of some settings require a server restart: refer to Deployment Guide for further details.

13.1 Server Settings List

13.1.1 List Database Settings in the Management Perspective

The server settings of the current connection are available in the *Server Settings* view.

If a setting is not listed, click the **Add** button to create it. If you delete a setting it is removed from the database.

Note that changes of some settings require a server restart: refer to Deployment Guide for further details.

13.1.2 List Database Settings in Management Console

The server settings of the current connection are available on the *Global Settings* page.

13.1.3 List Database Settings from the Command Line

listSettings lists the engine settings stored in the database

Long Option	Short Option	Description
--host*	-h	host URL
--username*	-u	user name
--password*	-p	password

∗ required parameters

13.2 Settings Update

13.2.1 Updating a Database Setting

updateSetting updates the engine setting with the value

Long Option	Short Option	Description
`--host*`	-h	host URL
`--username*`	-u	user name
`--password*`	-p	password
`--name`	-n	setting name
`--value`	-v	setting value

∗ required parameters

```
lsps-runtime/cli-tools$ java -jar lsps-cli-3.3.2039-full.jar updateSetting \
--host http://localhost:8080 --username admin --password admin --name SSO_ENABLED --value true
```

13.3 Addition of Settings

13.3.1 Adding a Database Setting in the Management Perspective

To create a new server setting, in the *Server Settings* view, click the **Add** button.

13.3.2 Adding a Database Setting

addSetting adds a new setting with the defined value

Long Option	Short Option	Description
`--host*`	-h	host URL
`--username*`	-u	user name
`--password*`	-p	password
`--name`	-n	setting name
`--value`	-v	setting value

∗ required parameters

13.4 Deletion of Settings

13.4.1 Deleting a Database Setting in the Management Perspective

To delete a server setting, in the *Server Settings* view, click the **Remove** button.

13.4.2 Deleting a Database Setting from the Command Line

deleteSetting removes a setting

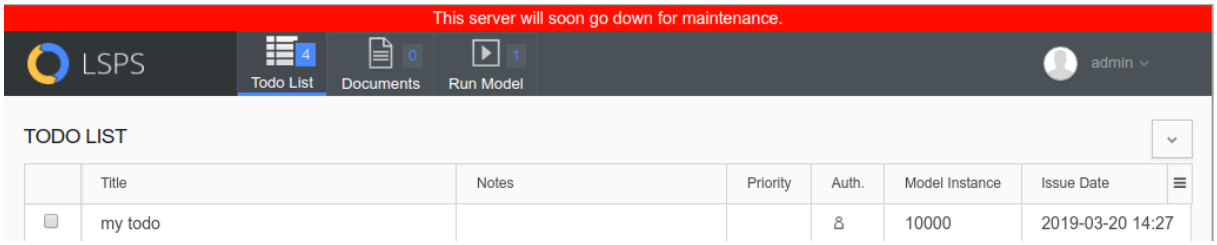
Long Option	Short Option	Description
`--host*`	-h	host URL
`--username*`	-u	user name
`--password*`	-p	password
`--name`	-n	setting name

∗ required parameters

```
lsps-runtime/cli-tools$ java -jar lsps-cli-3.3.2039-full.jar deleteSetting \
--host http://localhost:8080 --username admin --password admin --name SSO_ENABLED
Setting  SSO_ENABLED has been deleted.
```

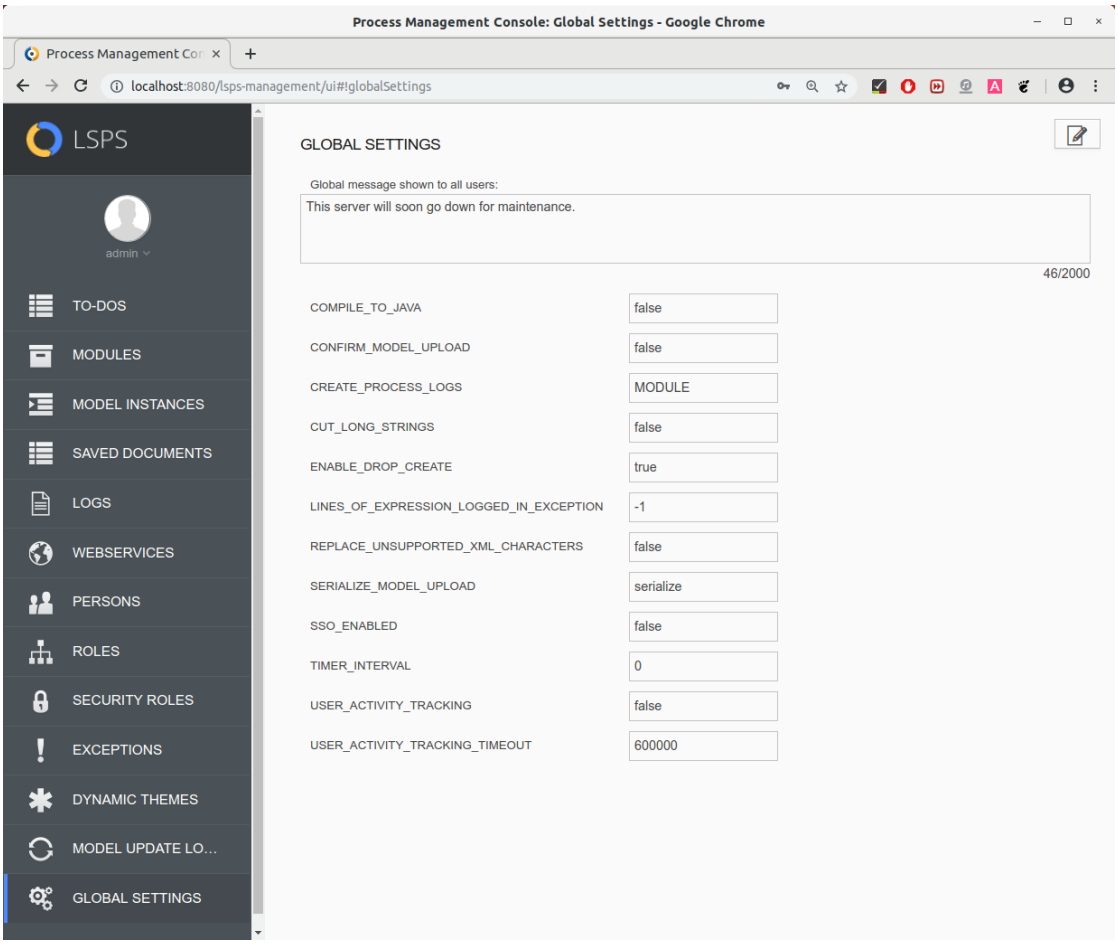
13.5 Global Message

The global message is displayed in the *Application User Interface* to all users.



13.5.1 Setting Global Message in Management Console

To set the global message from Management Console, go to the *General Settings* view, click the Edit  button, set the message, and click the Save button 


Process Management Console: Global Settings - Google Chrome
Process Management Cor
localhost:8080/lsps-management/ui#!globalSettings
LSPS
admin
TO-DOS
MODULES
MODEL INSTANCES
SAVED DOCUMENTS
LOGS
WEBSERVICES
PERSONS
ROLES
SECURITY ROLES
EXCEPTIONS
DYNAMIC THEMES
MODEL UPDATE LO...
GLOBAL SETTINGS
GLOBAL SETTINGS
Global message shown to all users:
This server will soon go down for maintenance.
46/2000
COMPILE_TO_JAVA false
CONFIRM_MODEL_UPLOAD false
CREATE_PROCESS_LOGS MODULE
CUT_LONG_STRINGS false
ENABLE_DROP_CREATE true
LINES_OF_EXPRESSION_LOGGED_IN_EXCEPTION -1
REPLACE_UNSUPPORTED_XML_CHARACTERS false
SERIALIZE_MODEL_UPLOAD serialize
SSO_ENABLED false
TIMER_INTERVAL 0
USER_ACTIVITY_TRACKING false
USER_ACTIVITY_TRACKING_TIMEOUT 600000

Chapter 14

Reference

14.1 Security Rights

A security role given to a person allows the person to perform actions based on the assigned security rights.

The following security rights are available:

14.1.1 Binary Security Rights

Binary rights govern the operations on binaries stored directly to database.

Security right	Description
Binary:Add	adding binary resources; allows the user to upload files directly to database (this is done from the ui::Upload components with *Upload to memory* set to `false`)
Binary:Delete	removing binary resources
Binary:Read	reading binary resources

14.1.2 Debugger Security Rights

Security right	Description
Debugger:Manage	managing debugger–setting, activating, deactivating the debug mode, and adding, updating and removing breakpoints

14.1.3 Document Security Rights

Document rights govern the access to the Documents uploaded as part of modules.

Security right	Description
Document:Read	reading documents
Document:Submit	editing and submitting documents
Document:Write	import document state

14.1.4 Exception Security Rights

Exception rights restrict the access to exceptions in model instance execution.

Security right	Description
Exception:Read	retrieving rollback information
Exception:Remove	removing rollback info
Exception:Resend	resending task data from the Exception Handling view

14.1.5 Expression Evaluation Security Rights

Security right	Description
Expression:Evaluate	evaluating expressions in runtime (in the Expression Evaluator)

14.1.6 Form Preview Security Rights

Form preview right serve to restrict access to `Form preview`

Mind that this feature should be `disabled completely on other than development servers`.

Security right	Description
Form:Preview	running form previews (note that you need the Model:Manage right to be able to preview forms)

14.1.7 Goal State Security Rights

Security right	Description
GoalState:Update	changing states of goals

14.1.8 Log Security Rights

Security right	Description
Log:Read	access to logs created by the Log task and log() function

14.1.9 Management Security Rights

Security right	Description
Management:Login	access to the management tools, the Management Console and Management perspective

14.1.10 Model Security Rights

Security right	Description
Model:Delete	deleting model instances

Security right	Description
Model:Manage	adding and uploading models to the repository and exporting uploaded models to archive files
Model:Read	finding, reading, and updating models and GO-BPMN modules in the repository and reading resources (files) for a given model

14.1.11 Model Instance Security Rights

Security right	Description
ModelInstance:Create	creating model instances
ModelInstance:Notify	notifying model instance from a Web Service
ModelInstance:Read	retrieving model instance related information
ModelInstance:Suspend	suspending running model instances
ModelInstance:Terminate	finishes running model instances
ModelInstance:Update_Model	changing the model for running model instances
ModelInstance:Write	import XML model instance state

14.1.12 Person Security Rights

Security right	Description
Person:Change_Own_Password	changing own password (The person must not have `Person:Manage` either for the restriction to apply.)
Person:Manage	managing general information and associations to modeled roles of persons
Person:Read	retrieving person related information

14.1.13 Report Security Rights

Security right	Description
Report:Dashboard_Management	enables the definition of common dashboard tabs
Report:Read_All	reading all available reports (this right overrides the security setting defined for the report)
Report:Read_Own	reading reports, which the user is entitled to

14.1.14 Role Security Rights

Security right	Description
Role:Manage	managing role
Role:Read	reading roles (read-only access)

14.1.15 Upload Schema Security Rights

Security right	Description
Schema:DropCreate	availability of drop-create strategy for business objects
Schema:Update	updating strategy of business objects
Schema:Validate	validating schema of business objects

14.1.16 Security Role Security Rights

Security right	Description
SecurityRole:Manage	adding, renaming, and removing security roles to/from the persistent storage and assigning rights to security roles and acquiring a set of all rights
SecurityRole:Read	retrieving security roles and associated users

14.1.17 Settings Security Rights

Security right	Description
Settings:Manage	access to the Settings of Management Console and Application User Interface

14.1.18 Signal Security Rights

Security right	Description
Signal:Remove	removing a signal from the model instance queue
Signal:Send	sending signals to model instances

14.1.19 Todo Security Rights

Security right	Description
Todo:Delegate_All	delegating any to-dos
Todo:Delegate_Own	delegating to-dos of the particular person
Todo:Escalate_All	escalating any to-dos
Todo:Escalate_Own	escalating to-dos assigned to the particular user
Todo:Process	getting, submitting and canceling a to-do
Todo:Read	acquiring the list of to-dos meeting the given criteria (disables access to the Todo List)
Todo:Read_All	reading all available to-dos (this right overrides the security setting defined for the to-do)
Todo:Read_Assignees	displaying to-do assignees (in to-do details)
Todo:Read_Own	reading to-dos, which the user is entitled to
Todo:Reassign	reassigning to-dos
Todo:Reject	rejecting to-dos
Todo:Undo_Reject	cancelling to-do rejection
Todo:Write_All	import XML to-do state

14.1.20 Context Security Rights

Security right	Description
Variables:Update	changing values of variables of the given context

14.1.21 Web Service Security Rights

Web Service rights serve to restrict access to data about the web services provided by the modules.

Security right	Description
Webservice:Invoke	platform specific right
Webservice:Read	monitoring of current modeled Web Services

14.1.22 Monitoring Model-Instance Starting

Security right	Description
AppRestart:Read	read information about model instance starting (access to CLI-commands `appRestart↩InfoExport`, `appRestartInfo`
AppRestart:Clear	clearing the model-instance start data by calling `clearApplicationRestart↩Data()`

14.1.23 Technical and Deprecated Security Rights

Security right	Description
Lock:Manage	acquiring or releasing data locks
Testing:All	internal security right; Do not assign this right to any users.