**QM-IConf Testing Scenarios**

**Description:**

This document provides testing scenarios for the QM-Config Tools (Tool) that can be used both in release preparation testing and in tool evaluation phase.

Each scenario is described using a following template:

* Goal – what is the user trying to do
* Prerequisite–which test scenario must be executed before so that the actual scenario can be executed successful
* Process – how to do it
* Result – what is the actual result

*Please test the Eclipse-based version first, then at least one of the deployed versions. Note the results of the individual tests for each version / date in an excel sheet and commit the sheet.*

We use Consolas font to denote program elements or commands. Further, we use the notation Menu|Sub-Menu to refer to menu entires. If parts of program elements are variable, we put them into brackets and display them in italics, e.g., Delete *<PipelineName>*.

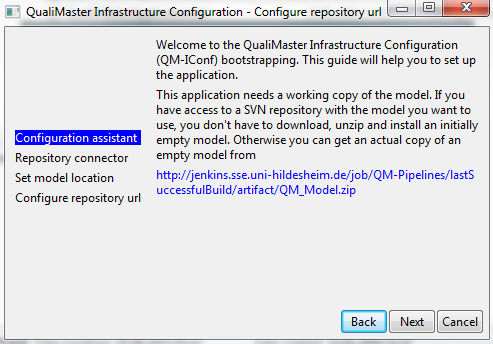
**Scenarios:**

## Log in (initial model download)

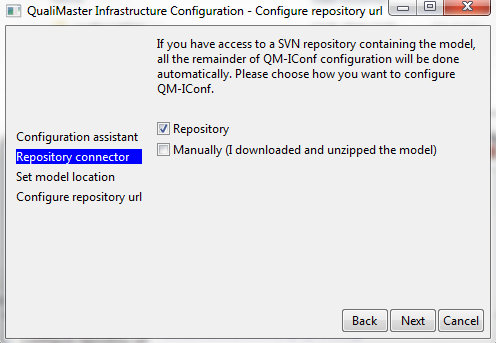
**Goal:** Downloading the model when logging into the Tool for the first time

**Prerequisite:** none

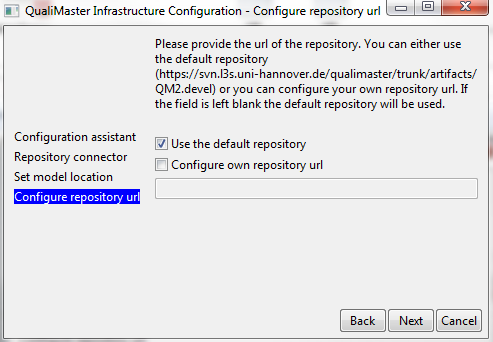
**Process:** Follow the wizard. Press “Next”.



Select “Repository”



Select “Use default repository”



Use your login credentials (L3S)



**Result:** Fresh model is downloaded into your workspace and loaded by the tool. The tool shall open and show the configurable elements tree on the left side of the main window.

## Start using local data (no log in required)

**Goal:** Ability to use the Tool without connecting to the server (offline).

**Prerequisite:** none

**Process:** Log in without using your login credentials and with No login – local data check box ticked.



**Result:** The tool shall open and show the configurable elements tree on the left side of the main window based on the already available model without connecting to the server.

## Saving changes in the local model

**Goal:** Ability to save changes to the local model.

**Prerequisite:** Start using local data (Scenario 2)

**Process:** Modify a value, save and restart the Tool.

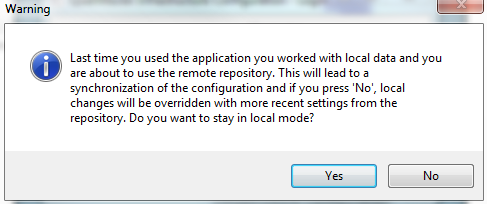
**Result:** Value should be the modified value.

## Switching between no login (local data) to login(repository connection)

**Goal:** Switching between modes without losing changes or clearing the changes.

**Prerequisite:** Start using local data (Scenario 2)

**Process:** Close the tool (local data mode) and open with logging in (Scenario 1). You will get a message asking if you want to keep the changes.



**Result:** If Yes changes are kept, if No all changes made before shall be reverted.

## Saving by using CTRL+S and Model|Save All

**Goal:** Check if both saving methods work for single and multiple tabs.

**Prerequisite:** Tool is open and ready (Scenario 1 or 2)

**Process:** After modifying some value use CTRL+S to save changes. Repeat with Model|Save All.



**Result:** Ctrl+S saves currently opened tab (single). Model|Save All informs about which tabs where modified and offers to save the changes. Restart QM-IConf to see whether the model was written / read correctly.



## Selecting an Artifact

**Goal:** Select the artifact specification from the user interface rather than entering it directly.

**Prerequisite:** Tool is open and ready (Scenario 1 or 2)

**Process:** Open one of the editors which require an artifact (data source, data sink, algorithm, infrastructure). The actual artifact specification shall be show in the editor field.



Press the Browse... button directly right to the editor field. In case that the Browse... button is greyed out, please check the repository settings in the infrastructure configuration.

Upon first opening the selector, the tool reads the repository structure from the server (online, incremental progress bar while loading). In offline mode, it shall use a cached repository structure, if the selector has been opened before (cache update after 24h). The selector dialog shall open and show the repository structure (please validate at least the top-level structure) and the actual artifact specification from the editor where we started.



Modify the specification either by selecting another artifact from the tree view or by changing the entries manually. Press OK. Cancel shall close the dialog without modification of the actual artifact. Refresh shall refresh the repository tree on request.

**Result:** The selected / entered artifact specification shall now occur in the editor ready for saving.

## Selecting an Artifact (Pipeline Editor)

**Goal:** Select the artifact specification from the user interface of the pipeline properties rather than entering it directly.

**Prerequisite:** Tool is open and ready (Scenario 1 or 2)

**Process:** Open one of the defined pipelines, select the pipeline background, i.e., the pipeline itself and ensure that the properties view is visible. The actual artifact specification shall be show in the editor field.



Click the artifact cell and press the ... button directly right to the editor field (which shall also contain the actual artifact specification). Upon first opening the selector, the tool reads the repository structure from the server (online). In offline mode, it may use a cached repository structure, if the selector has been opened before. The selector dialog (see Scenario 6) shall open and show the repository structure (please validate at least the top-level structure) and the actual artifact specification from the editor where we started. Modify the specification either by selecting another artifact from the tree view or by changing the entries manually. Press OK. Close shall close the dialog without modification of the actual artifact. Refresh shall refresh the repository tree on request.

Store the change to see the update of the model. Change the artifact specification also manually. Restart QM-IConf to see whether the model was written / read correctly.

**Result:** The selected / entered artifact specification shall now occur in the properties field of the pipeline ready for saving (**For now editor is not set dirty, so as workaround make some other change and then save**). After saving, the correct value shall be displayed in a new editor.

## Reverting all changes

**Goal:** To revert all changes made

**Prerequisite:** Tool is open, ready (Scenario 1 or 2) and running in DEMO mode

**Process:** Select Model|Revert all



A message will appear stating that in order to revert all changes the application will be restarted.



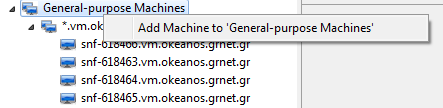
**Result:** If selected Yes application is restarted and all changes reverted. When application is started all tabs are closed. Nothing shall happen if Cancel is selected.

## Adding a new Type, General-purpose Machine, Reconfigurable Hardware Machine, Data Management, Algorithm Family, or Algorithm

**Goal:** Adding new configurable element

**Prerequisite:** Tool is open and ready (Scenario 1 or 2)

**Process:** Select the category items you want to add, right click and select Add *<element description>*



**Result:** A new element appears in the configuration options tree view in the correct grouping and the respective editor is opened.

## Configuring a new Type, General-purpose Machine, Reconfigurable Hardware Machine, Data Management, Algorithm Family, or Algorithm

**Goal:** Configure a new configurable element by individual settings.

**Prerequisite:** Tool is open and ready (Scenario 1 or 2), the respective element has been created (Section 9).

**Process:** After creating new elements according to Section 9, directly edit the individual settings, store the settings, close the editor and re-open the editor. Restart QM-IConf to see whether the model was written / read correctly. *Please note that selecting an implementing class requires the capabilities to update the configuration according to the algorithm / component manifest, which is currently not fully integrated and, thus, the respective Browse-Button may be deactivated (or not visible in the Demo mode).*

**Result:** The settings are correctly stored and displayed correctly after re-opening the respective editor / QM-IConf. In case of general-purpose machines, the machine may dependent on its name either appear under an existing grouping if IP/domain name match, a new grouping or in case of simple names at the end of the list. A new algorithm member in a family shall lead to the right grouping in the algorithm section (unused algorithms shall appear at the end of the list)

## Editing a Type, General-purpose Machine, Reconfigurable Hardware Machine, Data Management, Algorithm Family, or Algorithm

**Goal:** Configure an existing configurable element by individual settings.

**Prerequisite:** Tool is open and ready (Scenario 1 or 2), the respective element is available (Section 10).

**Process:** After opening the editor of the respective element, edit the individual settings, store the settings, close the editor and re-open the editor. Restart QM-IConf to see whether the model was written / read correctly. *Please note that selecting an implementing class requires the capabilities to update the configuration according to the algorithm / component manifest, which is currently not fully integrated and, thus, the respective Browse-Button may be deactivated (or not visible in the Demo mode).*

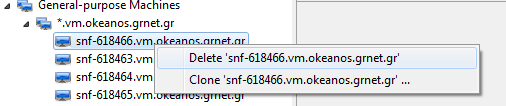
**Result:** The settings are correctly stored and displayed correctly after re-opening the respective editor / QM-IConf. In case of general-purpose machines, the machine may dependent on its name either change its position in the tree and appear under an existing grouping if IP/domain name match, a new grouping or in case of simple names at the end of the list. A changed algorithm member shall change its family assignment in the configuration tree accordingly, i.e., a family shall lead to the right grouping in the algorithm section (unused algorithms shall appear at the end of the list)

## Deleting a Type, General-purpose Machine, Reconfigurable Hardware Machine, Data Management, Algorithm Family, or Algorithm

**Goal:** Deleting a configurable element.

**Prerequisite:** Tool is open and ready (Scenario 1 or 2) and there is an element to be deleted.

**Process:** Select the configurable element you want to delete, right click and select Delete *<element name>.* Re-add an element immediately. Restart QM-IConf to see whether the model was written / read correctly.



**Result:** Configurable element is deleted and, in particular, a new element can be added immediately (Scenario 9). A deleted general-purpose machine shall disappear from its respective group. An empty group may remain in the tree (until restart).

## Cloning a Type, General-purpose Machine, Reconfigurable Hardware Machine, Data Management, Algorithm Family, Algorithm

**Goal:** Making a copy of Machine, Algorithm or Algorithm Family.

**Prerequisite:** Tool is open and ready (Scenario 1 or 2) and there is an element that can be cloned.

**Process:** Select the item you want to copy, right click and select Clone *<element name>*



* General-purpose machines and Reconfigurable hardware machines can be cloned multiple times. In this case the following dialog occurs and asks you for the number of copies / clones. After pressing Cancel nothing shall happen, after pressing OK the desired numbers of clones shall occur. It shall not be possible to enter another value than a positive number.



* The other configurable elements can be cloned only once per executing the clone command, i.e., the created clone shall occur immediately.

**Result:** Selected element is copied (according to the given number in case of General-purpose machines or Reconfigurable hardware machines) with all parameters (except the name identifier). Cloned machines shall occur at the end of the list as ungrouped (as described in Section 10).

## Copying input/output fields

**Goal:** Simplify configuration of Source, Sink, Algorithm or Algorithm Family.

**Prerequisite:** Tool is open and ready (Scenario 1 or 2) and there is an element with data field entries that can be cloned.

**Process:** Create a new element (see above). Right click on the (input/output) fields to copy. A dialog shall open showing potential sources with images in sorted order. Select the item you want to copy from, right click Ok.



**Result:** Selected element fields are copied into the original editor and the editor becomes dirty.

## Adding a pipeline

**Goal:** Adding a new pipeline.

**Prerequisite:** Tool is open and ready (Scenario 1 or 2).

**Process:** Select “Pipelines” in the configuration options tree view, right click and select Add Pipeline to ‘Pipelines’.



**Result:** A new empty pipeline is added and the respective pipeline editor is open.

## Deleting a pipeline

**Goal:** Delete an existing pipeline

**Prerequisite:** Tool is open and ready (Scenario 1 or 2) and there is a pipeline to delete.

**Process:** Select a pipeline you want to delete, right click and select Delete *<pipeline name>.*



**Result:** The selected pipeline is deleted, in particular a new pipeline (Scenario 15) can be added immediately.

## Using the Graphical Pipeline Editor

**Goal:** Test if it is possible to create and edit graphical pipeline elements.

**Prerequisite:** Tool is open and ready (Scenario 1 or 2) and there is a pipeline to modify (the Priority pipeline).

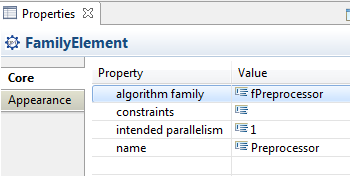
**Process:** Open the graphical pipeline editor for the priority pipeline. In case of the priority pipeline, all elements must have a name and image displayed as shown below.



Try changing the positioning of elements. The pipeline should not break apart.



Try adding a new family element by selecting it in the palette and clicking on the editor. When element appears you will be asked to name the element, if no name is given by you a default name will appear. To edit properties of the element go to the properties tab and enter the shown settings.



Connect the element by flows to the remaining pipelineand name the flows appropriately.

**Result:** Pipeline can be edited, all elements properties are editable. Result is saved. Closing and reopening the editor shall show the same pipeline as before.

## Defining / changing constraints

**Goal:** Define user constraints on runtime variables as boundaries for adaptation on Source, Sink or Pipeline elements.

**Prerequisite:** Tool is open and ready (Scenario 1 or 2) and there is an element with data of those mentioned above.

**Process:** Open the editor of the element (Source or Sink) and open the constraint editor (SLA constraints field). In case of the pipeline editor, this will be in the properties tab with a specific ... button. In case of the other editors, this happens by right-clicking the respective constraint. The constraint editor shall open and you shall be able to define (simple) constraints. The variables available in the actual context shall be shown by their variable name, their display name and help text in the table above the input field. Verify the constraints are taken over, stored, can be modified and deleted. Check the presence of the content assist in the editor.



**Result:** Constraint can be defined, stored, modified, deleted.

## Help texts for configuration entries

**Goal:** Validate if additional help texts are displayed properly.

**Prerequisite:** Tool is open and ready (Scenario 1 or 2).

**Process:** Go through the individual editors and check whether appropriate help messages are shown in the status bar if the mouse is placed for a certain time over the element or (in case of table-based editors only) over its label.

****

**Result:** Appropriate help texts are shown.

## Help texts for configuration sections

**Goal:** Validate if additional help for configuration sections texts are displayed properly.

**Prerequisite:** Tool is open and ready (Scenario 1 or 2).

**Process:** Go through the individual sections of the configuration tree and open the top-level element. Except for the infrastructure settings, for all sections with nested elements there shall be a summary of the configuration settings and their help texts (rather than the EASy configuration editor).

****

**Result:** Appropriate help texts are shown.

## Help documentation

**Goal:** Validate if the help documentation is displayed properly and that all resources are part of the release.

**Prerequisite:** Tool is open and ready (Scenario 1 or 2).

**Process:** Open Help|QM-IConf Documentation.



**Result:** The QM-IConf documentation page opens and the documentation links to the user guide / constraint guide are working and the documents are up-to-date. In the tree, also the EASy-Producer documentation and it’s documents are accessible.

## Validating the model

**Goal:** Validate if the configuration is valid (no constraints not violated).

**Prerequisite:** Tool is open and ready (Scenario 1 or 2).

**Process:** Select Validate|Validate All from the menu



If the model is valid (this shall be the case if all preceding scenarios are completed successfully), the message box Model is valid should appear.



If model is not valid message box Please consult the ‘Problem View’ should appear.



To get the information about failed constraints go to the Problems tab.



In the problem tab you can see the total amount of errors and its descriptions. The respective configuration settings shall be marked in red, as decorator in the usual editors and as shapes in the pipeline editor. Tool tips shall of the markings shall provide a short description of the error. By selecting an error the description also is shown in the bottom.

**Result:** Model is validated, errors, if exist, are found.

## 21.1 Worker Machine has no ports assigned

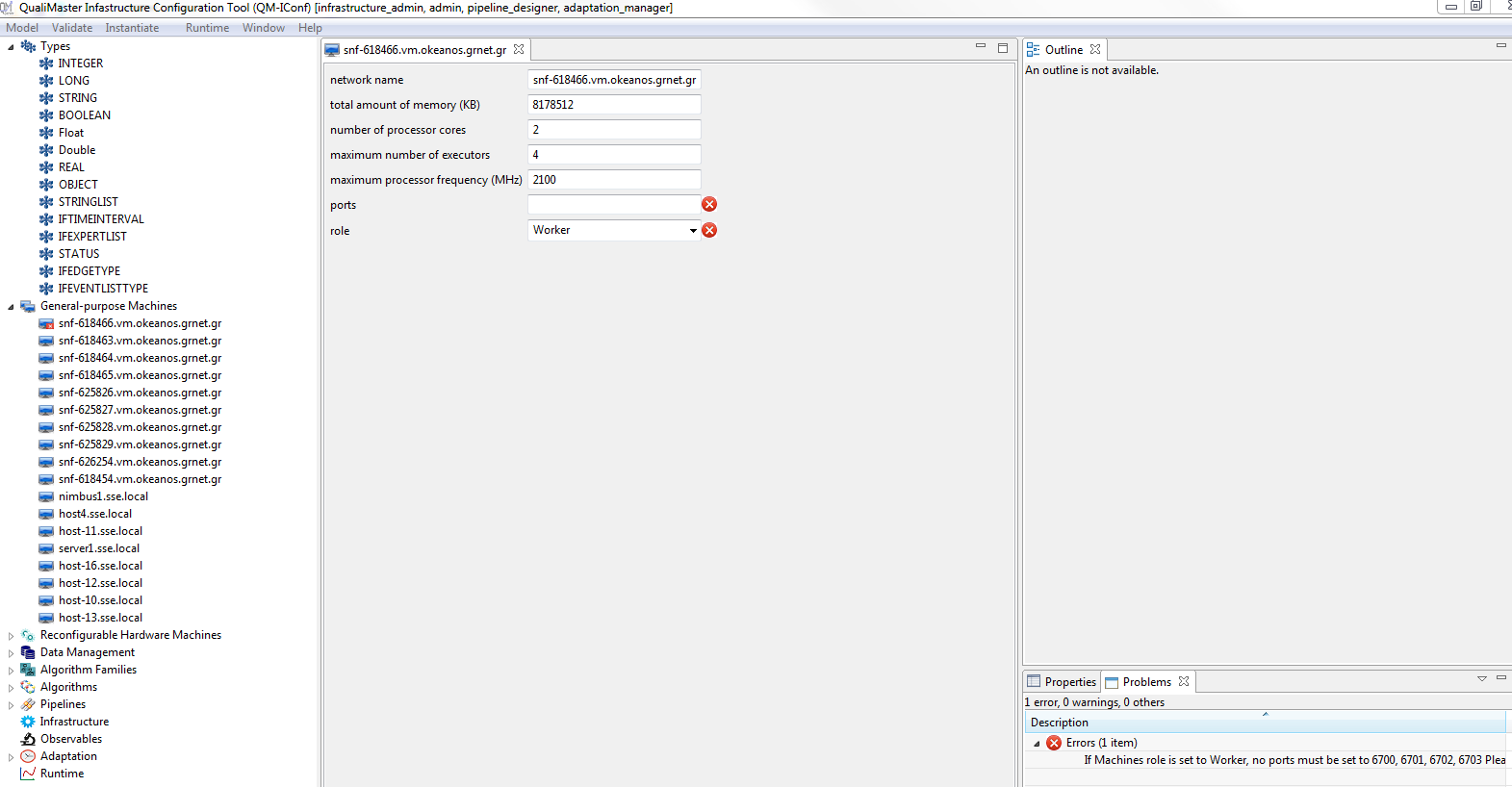
**Goal:** Validate compound constraints.

**Prerequisite:** Tool is open and ready (Scenario 1 or 2).

**Process:** Select snf-618466.vm.okeanos.grnet.gr from General-purpose Machines. Change the role from “Manager” to “Worker”. Save and validate the model.

**Result:** 1 errors is found

1. If Machines role is set to Worker, no ports must be set to 6700, 6701, 6702, 6703. Role and ports fields in snf-618466.vm.okeanos.grnet.gr are marked by a red decorator. Same machine is marked in the View as well



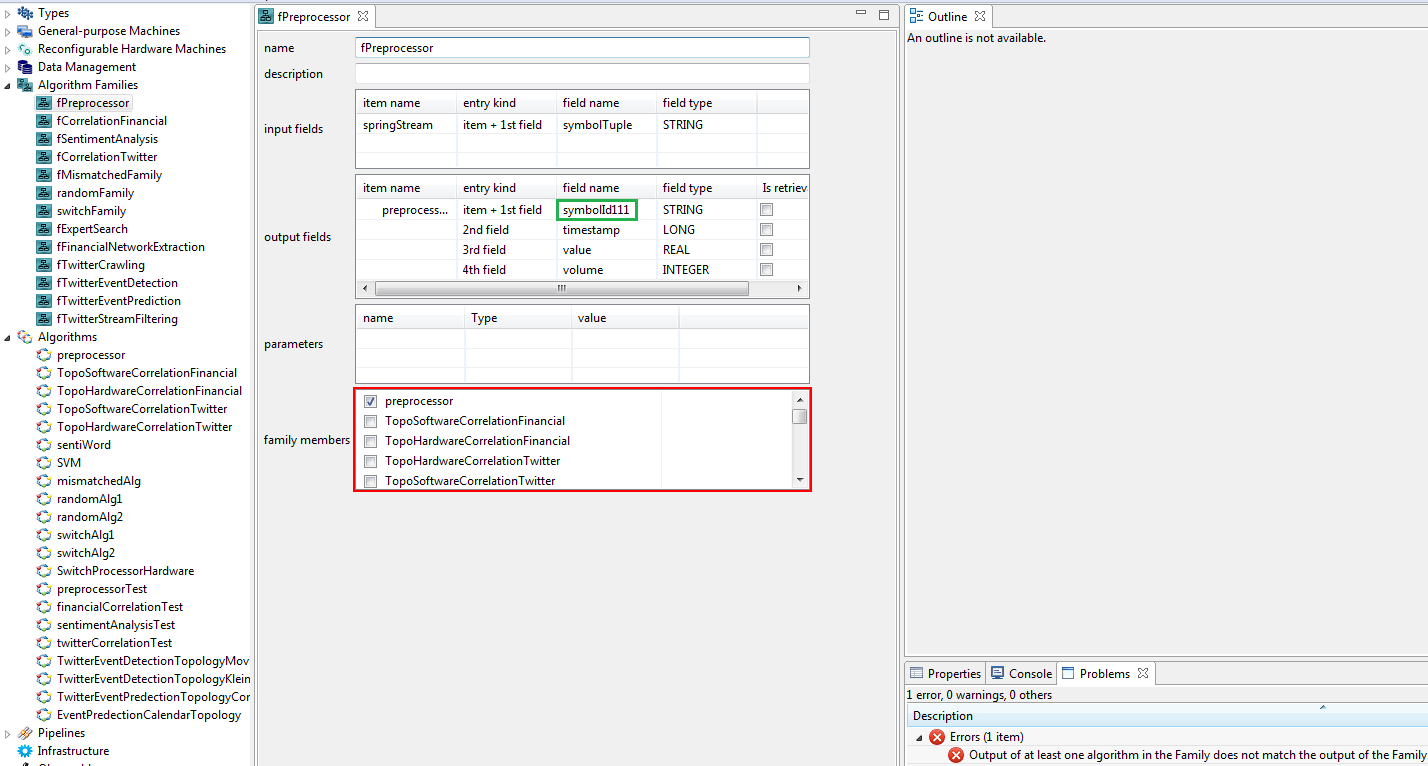
## 21.2 Input/output mismatch in Algorithm Families and Algorithms

**Goal:** Validate link constraint between pipeline model and algorithms.

**Prerequisite:** Tool is open and ready (Scenario 1 or 2).

**Process:** Select fPreprocessor from Algorithm Families. In the output field change the field name from “symbolId” to ”symbolId111” (marked with green). Save and validate the model.

**Result:** 1 error is found - Output of at least one algorithm in the Family does not match the output of the Family. Algorithm field is marked with red (now by a decorator rather than a rectangle). Name in one of the fields does not match in Algorithm Family and assigned algorithm.





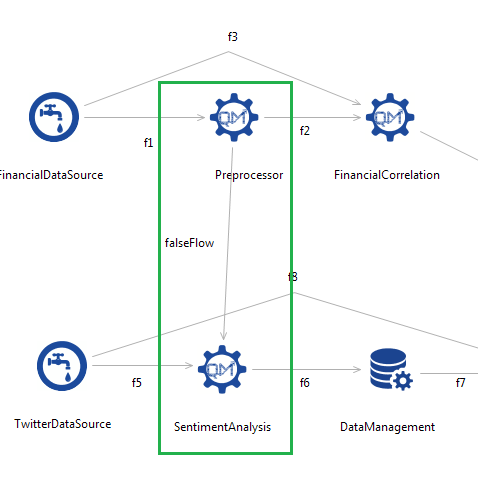
## 21.3 Input/output mismatch by an additional Flow

**Goal:** Validate pipeline constraint by one additional element.

**Prerequisite:** Tool is open and ready (Scenario 1 or 2).

**Process:** Select PriorityPip from Pipelines. In the pipeline designer add a Flow between Preprocessor and SentimentAnalysis family elements (marked below in green). Save and validate the model.

**Result:** 1 error is found - Input and output types should be the same. The corresponding elements (SentimentAnalysis, falseFlow, pre-processor, f2, FinancialCorrelation) shall be marked by red shapes in the pipeline editor.



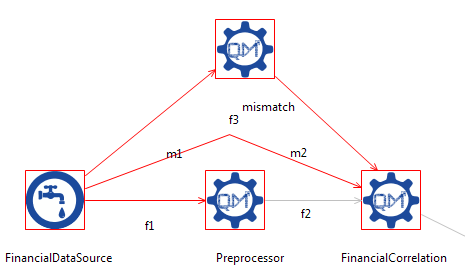
## 21.3 Input/output mismatch by an additional element and two flows

**Goal:** Validate pipeline constraint.

**Prerequisite:** Tool is open and ready (Scenario 1 or 2).

**Process:** Select “PriorityPip” from Pipelines. In the pipeline designer add a Family Element “mismatch” with family “fMismatchedFamily”. Create a Flow between “FinancialDataSource” (name “m1”) and “mismatch” as well as “mismatch” and “FinancialCorrelation” (name “m2”). Save and validate the model.

**Result:** 1 error is found - Input and output types should be the same. The corresponding elements shall be marked by red shapes in the pipeline editor.

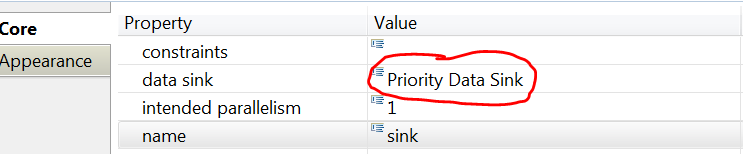


## 21.3 Input/output mismatch with Data Management Element

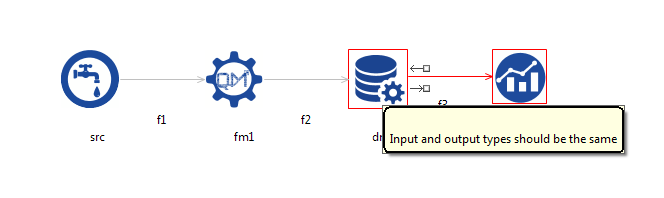
**Goal:** Validate pipeline constraint on transparent pipeline element.

**Prerequisite:** Tool is open and ready (Scenario 1 or 2).

**Process:** Select “testDMPip” from Pipelines. In the pipeline designer change the data sink from “Random Sink” to the mismatching “Priority Data Sink”. Save and validate the model.



**Result:** 1 error is found - Input and output types should be the same. The corresponding elements shall be marked by red shapes in the pipeline editor.

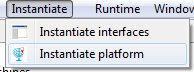


## Instantiating

**Goal:** Instantiate pipelines to the local drive.

**Prerequisite:** Tool is open and ready (Scenario 1 or 2). Model is valid.

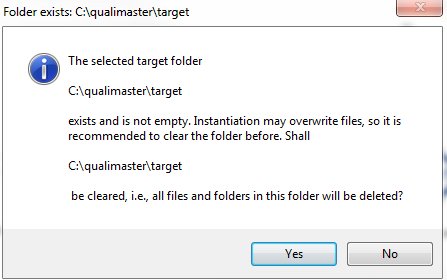
**Process:** Select Instantiate|Instantiate platform in the menu bar



Browse folders dialog will appear. Select the target folder.



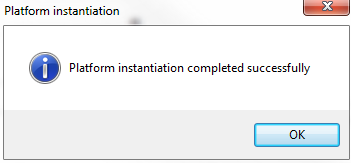
If selected folder is not empty a message asking if you would liketo clean the folder before will appear. To clean the folder press “Yes”.



You can observe the progress in the Console tab. It will also notify if the build was successful or not.



**Result:** Except for installation / operating system problems, the final result of the instantiation shall be BUILD SUCCESS and appropriate message will popup

.

## Help

**Goal:** Help should be displayable.

**Prerequisite:** Tool is open and ready (Scenario 1 or 2).

**Process:** Select in menu “Help” -> “QM-IConf Documentation”. A new window should open, containing a HTML help with two links. Click on both links and check whether the included PDFs are displayed.

**Result:** The HTML help as well as the two PDFs should be displayed.

## Open topics

The following topics deserve own testing scenarios in the future:

* Several scenarios causing different validation errors -> unmatchedFamily.
* Change pipeline, instantiate – do we get the right thing?
* Create more complex input/output fields.
* Enter / manipulate parameters.
* Observables
* Adaptation settings
* Display of monitoring data