



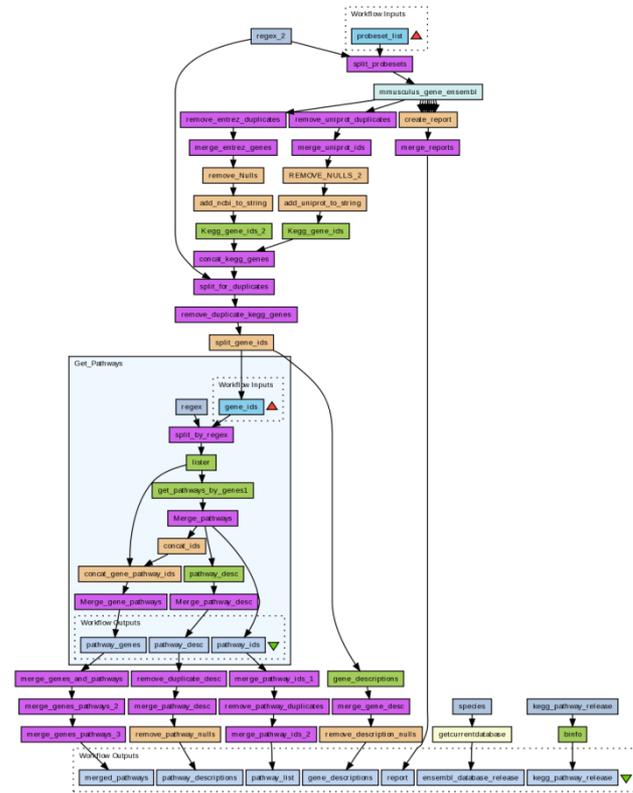
# An Introduction to Designing and Executing Workflows with Taverna

Alan R Williams

University of Manchester

# Workflows

- Sophisticated analysis pipelines
- A set of services to analyse or manage data (either local or remote)
- Data flow through services
- Control of service invocation
- Iteration
- Automation
- Access to intermediate results



## Workflows in Taverna

- This tutorial will give you a basic introduction to designing, and reusing workflows in Taverna and some of its main features.
- Workflows in this practical use small data-sets and are designed to run in a few minutes. In the real world, you would be using larger data sets and workflows would typically run for longer
- Taverna allows you to use different forms of data input and save output data in different formats too – we will look at that in this tutorial as well.

# Taverna Workbench

The screenshot displays the Taverna Workbench Core 2.5.0 interface. The top menu bar includes File, Edit, Insert, View, Workflows, Components, Advanced, and Help. The main workspace is divided into three primary sections:

- Services Panel:** Located on the left, it features a filter and a list of local services such as base64, io, jdbc, list, ncbi, net, and test. A box labeled "Services Panel" is overlaid on this area.
- Workflow Explorer:** Below the services panel, it shows a tree view for the workflow "EBI\_InterproScan\_bro". It details workflow input ports (email, sequence), output ports (status, text, xml), and various services like getTextResult and getXmlResult. A box labeled "Workflow Explorer" is overlaid on this area.
- Workflow Diagram:** The central workspace displays a flowchart of the workflow. It starts with "Workflow input ports" (sequence, email) leading to an "input" node, then "run\_input", "run", and "run\_output". The "run\_output" node branches into "tsv" and "xml" paths. A "Status" sub-diagram shows a loop involving "JobID", "getStatus\_input", "getStatus", "getStatus\_output", and "getStatus\_output\_status". The main flow concludes with "getTextResult" and "getXmlResult" nodes, which lead to "getTextResult\_output" and "getXmlResult\_output", and finally to "Workflow output ports" (status, text, xml). A box labeled "Workflow Diagram" is overlaid on this area.

## Workflow Diagram

- The **workflow diagram** is the visual representation of the workflow, it:
- Shows inputs, outputs, services and data flows
- Allows editing of the workflow by dragging and dropping and connecting services together
- Enables saving of workflow diagrams for publishing and sharing

## Workflow Diagram

Taverna understands difference  
types of services

Type	Description
Beanshell	A user editable scripting operation using Beanshell script
Nested_workflow	A sub-workflow exposed as a single operation
Rshell	A user editable script in R
SpreadsheetImport	An operation for loading spreadsheet data in CSV and Excel format
String_constant	A single constant string
XPath_Service	Extraction of data from XML data

Type	Description
Local_service	A standard Beanshell
wsdl_web_service	A processor accessing a standard SOAP service
REST_Service	A processor accessing a RESTful web service
Tool_service	Calling a local or remote tool/script
Interaction	Interaction in a web browser
Component	Re-usable component workflow

## Workflow Diagram

Taverna workflows also have

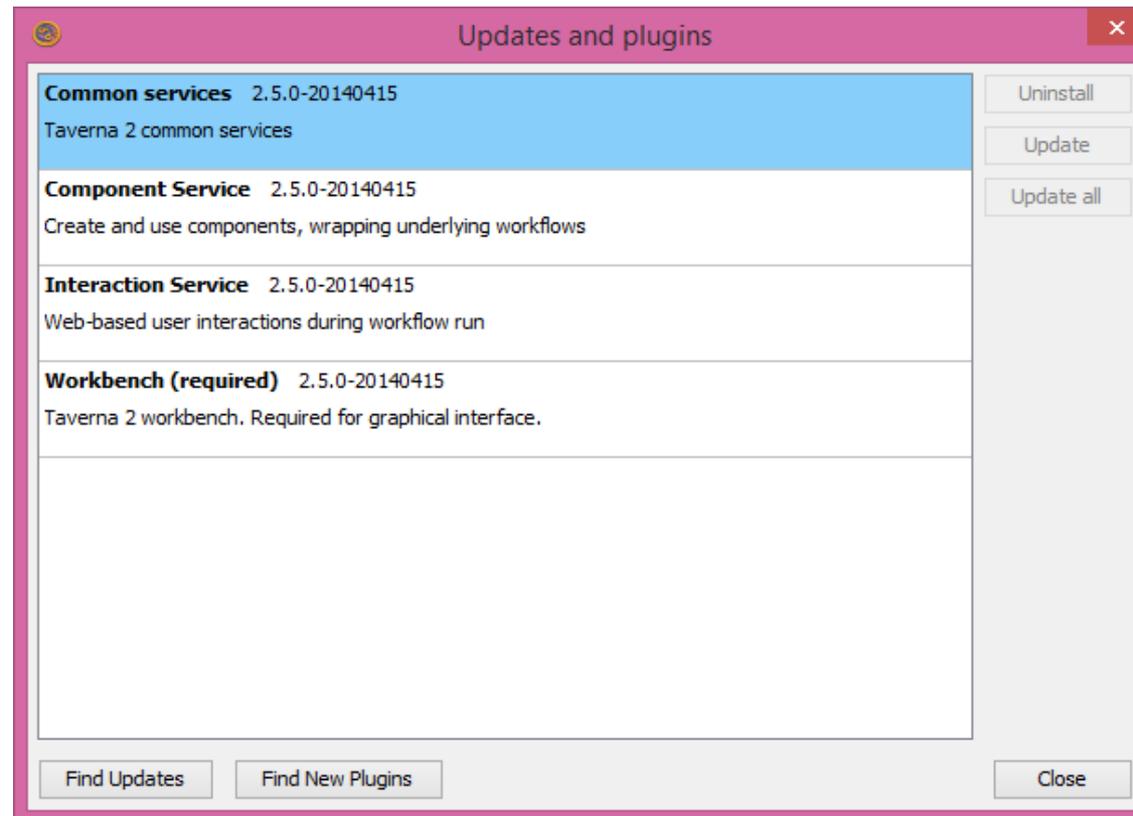
Element	Description
	Input port
	Output port
	"Merge" - make a list out of elements (often sub-lists)
	XML Splitter
	Data flow
	Control flow (run after)

## Workflow Explorer

- The **Workflow Explorer** shows the detailed view of your workflow. It shows default values and descriptions for service inputs and outputs and it shows where remote services are located. It also shows configuration details, such as iteration and looping
- Workflow validation details can also be found here. Before a workflow is run, Taverna checks to see if it is connected correctly and if its services are available.

# Updates and Plugin Installation

- It's a good practice to update Taverna regularly
- Taverna updates are issued on a regular basis
- There is also a number of plugins which are developed for Taverna
- To get the updates and plugins select **Advanced** -> **Updates and plugins**



## Available Services Panel

Lists services available by default in Taverna

- Local services – a standard set of utilities
- WSDL Web Service – secure and public
- RESTful Services
- R Processor services (for statistical analyses)
- Beanshell scripts
- XPath scripts
- Spreadsheet import service
- Interaction service – allows the workflow to “ask” for information in a web browser
- Component – re-usable workflows

The services panel also allows you to add new services from the web or from file systems – there are loads more available!

## Exercise 1: Building a Simple Workflow

- We will start with something easy – we will use an Allen Brain service to retrieve information about the experiments for a gene whose name we will provide
- Go to the [www.biocatalogue.org](http://www.biocatalogue.org) and search for “Allen brain”

BioCatalogue "The Life Science Web Services Registry"

Search:   | [Home](#) | [Services](#) | [Register](#)

[Home](#) »

The BioCatalogue: providing a curat

[Helpful Links](#)

BioCatalogue currently has [2491 services](#)

## Exercise 1: Building a Simple Workflow

- From the results select Allen Brain Atlas Browse

Search query "allen brain" returned 3 items

Services (2)

Service Providers (1)

### Allen Brain Atlas Browse

REST



Browse the Allen Brain Atlas (<http://www.brain-map.org/>). User specifies the model, model id and format for result (json, xml, csv).  
For example, <http://api.brain-map.org/api/v2/data/Chromosome/12.json>

Provider: [api-brain-map-org](#) | Base URL: <http://api.brain-map.org/api/v2/data>

### MouseGeneSearch

SOAP



Data Retrieval

Identifier Retrieval

Image Retrieval

The service allows you to retrieve data (e.g mouse gene) from the Allen Mouse Brain Atlas, an interactive genome-wide image database of gene expression.  
project website: <http://mouse.brain-map.org/welcome.do>

Provider: [Allen Institute for Brain Science](#) | WSDL Location: <http://mouse.brain-map.org/services/GeneSearchService?wsdl>

## Exercise 1: Building a Simple Workflow

- Have a look at the service description

Overview **Rest Services (4)** Examples Monitoring History

**Provider:**  
[api-brain-map-org](#)

**Location:**  
Lowell, United States

**Submitter/Source:**  
 [Michael Cornell](#) **Curator** (2 months ago)

**Base URL:**  
<http://api.brain-map.org/api/v2/data>

**Documentation URL(s):**

<http://help.brain-map.org/display/api/RESTful+Model+Access+%28RMA%29> by [Michael Cornell](#) **Curator** (2 months ago)

[Login to add a documentation URL](#)

**Description(s):**

Browse the Allen Brain Atlas (<http://www.brain-map.org/>). User specifies the model, model id and format for result (json, xml, csv).  
For example, <http://api.brain-map.org/api/v2/data/Chromosome/12.json>

by [Michael Cornell](#) **Curator** (2 months ago)

[Login to add a description](#)

## Exercise 1: Building a Simple Workflow

- Select the Rest Services tab and see how the service can be used
- Click on GET SectionDataSets

Overview Rest Services (4) Examples Monitoring History

What is an endpoint?

Quick Browse | GET SectionDataSets | GET SectionImages | Download Images | GET /{Model}/ {ModelId}. {resultFormat} |

**GET SectionDataSets** | [GET /query.xml?criteria=model::SectionDataSet,rma::criteria,genes%5Bacronym\\$eq%27{gene}%27%5D&num\\_rows=2](#)

Part of Service: [Allen Brain Atlas Browse](#)

Part of Endpoint Group: none

Template: [http://api.brain-map.org/api/v2/data/query.xml?criteria=model::SectionDataSet,rma::criteria,genes%5Bacronym\\$eq%27{gene}%27%5D&num\\_rows=2](#)

No description(s) yet

Tags on this endpoint: none

**GET SectionImages** | [GET /query.xml?criteria=model::SectionImage,rma::criteria,%5Bdata\\_set\\_id\\$eq{data\\_set}%5D&num\\_rows=2](#)

Part of Service: [Allen Brain Atlas Browse](#)

Part of Endpoint Group: none

Template: [http://api.brain-map.org/api/v2/data/query.xml?criteria=model::SectionImage,rma::criteria,%5Bdata\\_set\\_id\\$eq{data\\_set}%5D&num\\_rows=2](#)

No description(s) yet

Tags on this endpoint: none

**Download Images** | [GET /section\\_image\\_download/{image}](#)

Part of Service: [Allen Brain Atlas Browse](#)

Part of Endpoint Group: none

Template: [http://api.brain-map.org/api/v2/data/section\\_image\\_download/{image}](#)

No description(s) yet

Tags on this endpoint: none

## Exercise 1: Building a Simple Workflow

- This service returns the results for a gene name that we will provide, and limits the number of results to 2
- We can copy the Template  
[http://api.brain-map.org/api/v2/data/query.xml?criteria=model::SectionDataSet,rma::criteria,genes%5Bacronym\\$eq%27{gene}%27%5D&num\\_rows=2](http://api.brain-map.org/api/v2/data/query.xml?criteria=model::SectionDataSet,rma::criteria,genes%5Bacronym$eq%27{gene}%27%5D&num_rows=2)
- In Taverna Workbench go to the **Services Panel** in the **Design** view
- From the **Available Services** select **Service Templates** and **REST**
- Right-click on it and select **Add to workflow** (see the next slide)

## Exercise 1: Building a Simple Workflow

Import new services

Available services

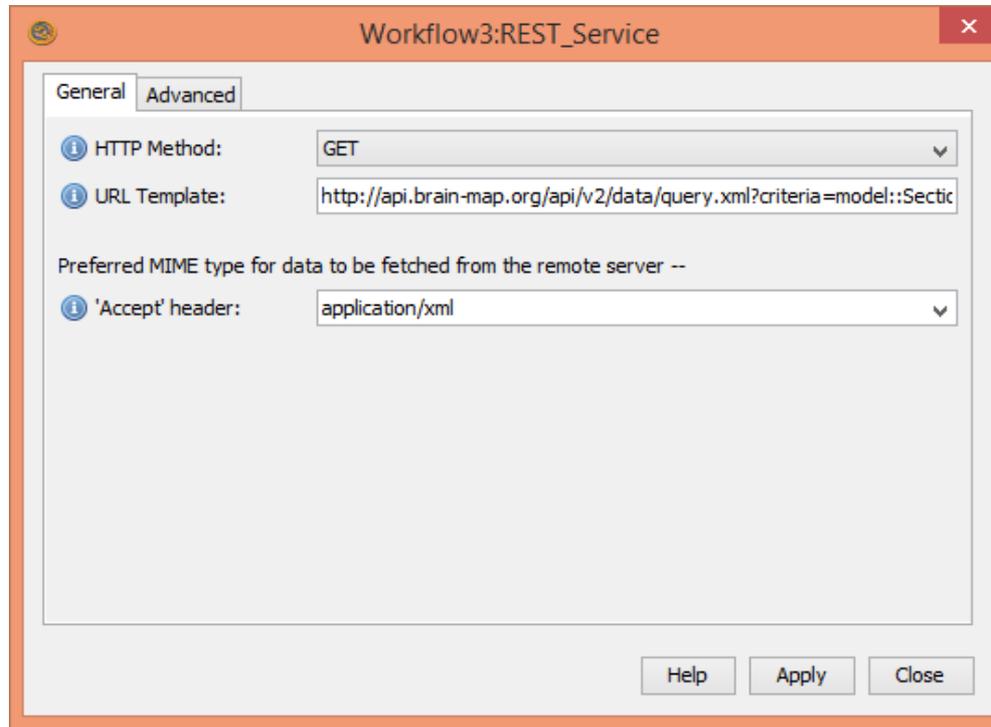
- Service templates
  - Beanshell - A service that allows Beanshell scripts, with dependencies on libraries
  - Interaction
  - Nested workflow - A service that allows you to have one workflow nested within another
  - REST Service - A generic REST service that can handle all HTTP methods**
  - Rshell - A service that allows the calling of R scripts on an R server
  - SpreadsheetImport - A service that imports data from spreadsheets
  - Text constant - A string value that you can set
  - Tool - A service that allows tools to be used as services
  - XPath Service - Service for point-and-click creation of XPath expressions for XML data
- Local services

REST Service

- Add to workflow
- Add to workflow with name...

Workflow explorer | Details | Validation report

## Exercise 1: Building a Simple Workflow

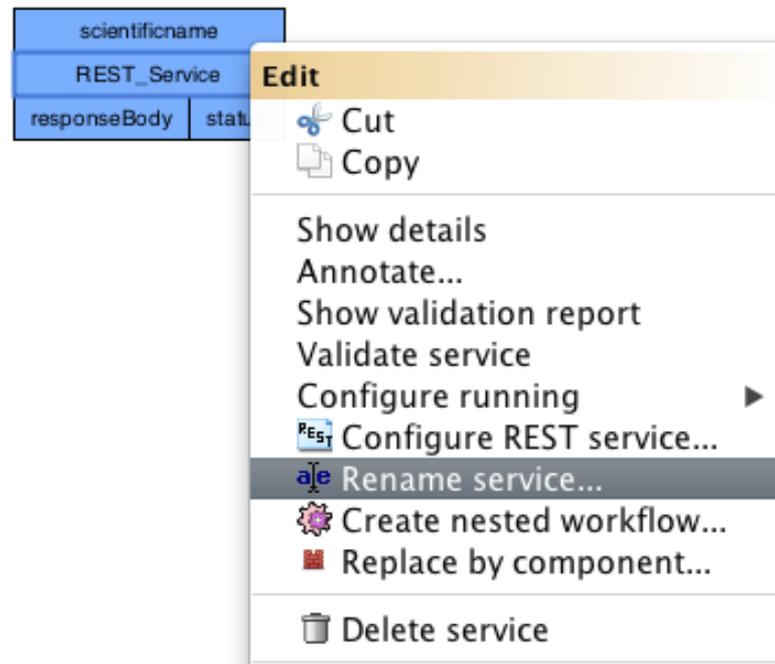


- Enter the template you copied into the URL template field:

[http://api.brain-map.org/api/v2/data/query.xml?criteria=model::SectionDataSet,rma::criteria,genes%5Bacronym\\$eq%27{gene}%27%5D&num\\_rows=2](http://api.brain-map.org/api/v2/data/query.xml?criteria=model::SectionDataSet,rma::criteria,genes%5Bacronym$eq%27{gene}%27%5D&num_rows=2)

## Exercise 1: Building a Simple Workflow

- Let's change the name of the service to:  
*AllenExperimentsForGene*



## Exercise 1: Building a Simple Workflow

- At the top of the workflow diagram panel, change the view to show all ports by clicking on the icon shown below

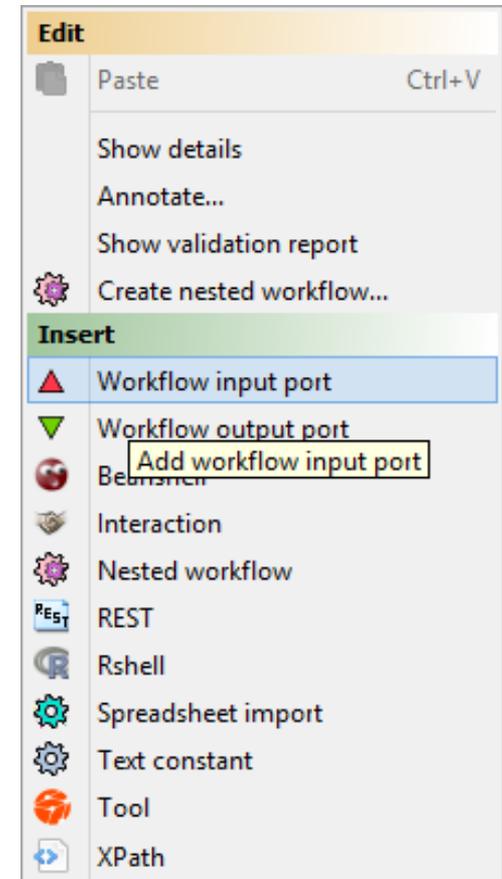


Show all ports icon

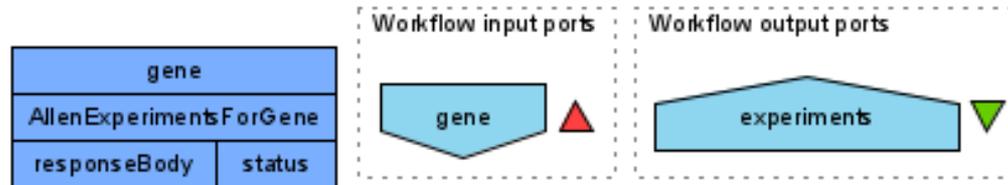
- This view allows you to see any data input/output or parameter value options for your chosen service

## Exercise 1: Building a Simple Workflow

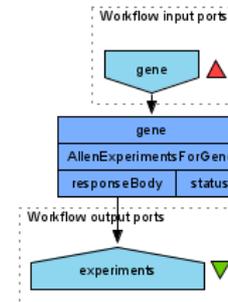
- In a blank space in the **workflow diagram**, right-click and select **Workflow input port** from the **Insert** section
- Type in a name for this input (e.g. *gene*) and click **OK**
- Do the same to create a new workflow output. Call this output *experiments*



## Exercise 1: Building a Simple Workflow

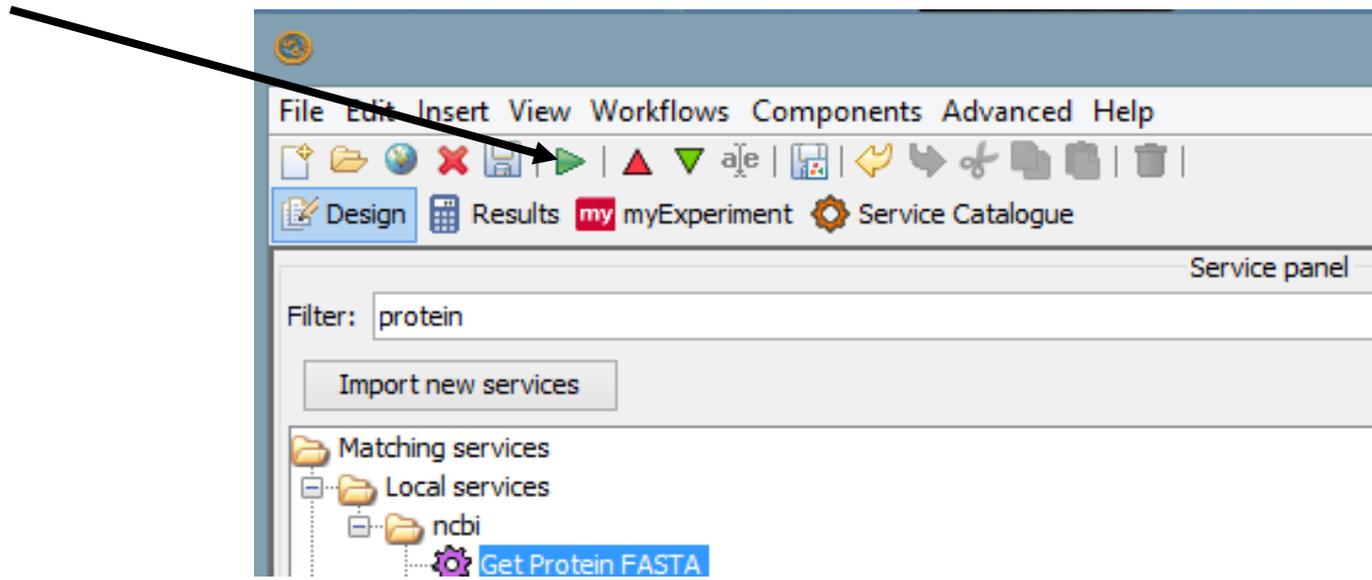


- Connect the input and output ports
- Your workflow should look like this



## Exercise 1: Building a Simple Workflow

- Run the workflow by selecting **File -> Run workflow**, or by clicking on the play button at the top of the workbench



## Exercise 1: Building a Simple Workflow

- You'll get a pop up window where you can enter the data for the workflow.

The screenshot shows a window titled "Input values for 'Workflow48'". On the left, there are three panels: "Diagram" showing a workflow graph with nodes like "Workflow input port", "Workflow task", and "Workflow output port"; "Workflow description" with "No description"; and "Workflow author" with "No author". The main area on the right is for input configuration. It has a "sciName" field at the top. Below it are three text areas: "Port description" (containing "No port description"), "Example value" (containing "No example value"), and a large empty area. Above this large area are buttons: "Delete", "Set value", "Set file location...", and "Set URL ...". A yellow tooltip "Set the input value" is over the "Set value" button. At the bottom, there are buttons for "Load previous values", "Save values", "Use examples", "Run workflow", and "Cancel".

## Exercise 1: Building a Simple Workflow

- Click **Set value**
- Enter *Adora2a* (note no newline)
- At the bottom of the window **Run workflow**

The screenshot shows a workflow editor window with a toolbar at the top containing 'Delete', 'Set value', 'Set file location...', and 'Set URL ...'. The main area is split into two panes: the left pane shows a text input field with 'Adora2a' entered and highlighted, and the right pane shows the value 'Adora2a'. Below the panes, there are fields for 'Data type:' and 'Character Set:'. A status bar at the bottom of the window displays the message 'Added new value. Edit value on right.' and a toolbar with 'Load previous values', 'Save values', 'Help', 'Use examples', 'Run workflow', and a close button.

# Exercise 1: Building a Simple Workflow

The screenshot shows the Taverna Workbench Core 2.5.0 interface. The main window displays a workflow diagram with the following components:

- Workflow input ports:** A box labeled "gene" with a red triangle icon.
- Workflow process:** A box labeled "AllenExperimentsForGene" with a small "1" in the top right corner.
- Workflow output ports:** A box labeled "experiments" with a green triangle icon.

Below the diagram, the workflow status is shown as "Finished". At the bottom, the "Workflow results" panel is open, showing a tree view with "gene" and "experiments". The "experiments" node is expanded, and the "Value 1" node is selected. The results are displayed in a text area:

```
<Response success='true' start_row='0' num_rows='2' total_rows='15'><section-data-sets>
<section-data-set>
<blue-channel nil='true'/>
<delegate>true</delegate>
<expression>true</expression>
<failed>false</failed>
<failed-facet>734881840</failed-facet>
<green-channel nil='true'/>
<id>100074949</id>
<name nil='true'/>
<plane-of-section-id>2</plane-of-section-id>
<qc-date>2009-09-24T10:35:35Z</qc-date>
</section-data-set>
</section-data-sets>
</Response>
```

- You should see the workflow running
- Once the workflow finished running
- Click on *experiments* and **Value** to see the results

# Exercise 1: Building a Simple Workflow

- Let's save the workflow now as "experiments\_for\_gene"

The screenshot shows the Taverna Workbench Core 2.5.0 interface. The main window displays a workflow diagram for 'Workflow3'. The workflow consists of a 'gene' input port connected to an 'AllenExperimentsForGene' service, which has two output ports: 'responseBody' and 'status'. These two outputs are connected to an 'experiments' output port.

The 'File' menu is open, and the 'Save workflow as...' option (F6) is highlighted. The 'Service panel' on the right shows the details for the 'AllenExperimentsForGene' service, including its description: 'All scripts, with dependencies on libraries interaction with a workflow run you to have one workflow nested within another that can handle all HTTP methods'. It also lists supported methods: 'GET', 'POST', 'PUT', and 'DELETE'. The 'Data links' section shows the connection between the service's 'responseBody' and the 'experiments' output port.

## Service ports

- ❑ Most of the time, you don't need to connect all ports. Some are optional and some already have default values set.
- ❑ Service documentation should tell you this. You can use the BioCatalogue to find documentation and user descriptions
- ❑ Change the orientation of the port names to fit them on the screen more easily by clicking on the icon shown below



change orientation

## Service ports

- ❑ Most of the time, you don't need to connect all ports. Some are optional and some already have default values set.
- ❑ Service documentation should tell you this. You can use the BioCatalogue to find documentation and user descriptions
- ❑ You can change the orientation of the port names to fit them on the screen more easily by clicking on the icon shown below



change orientation

## Adding a Workflow Description

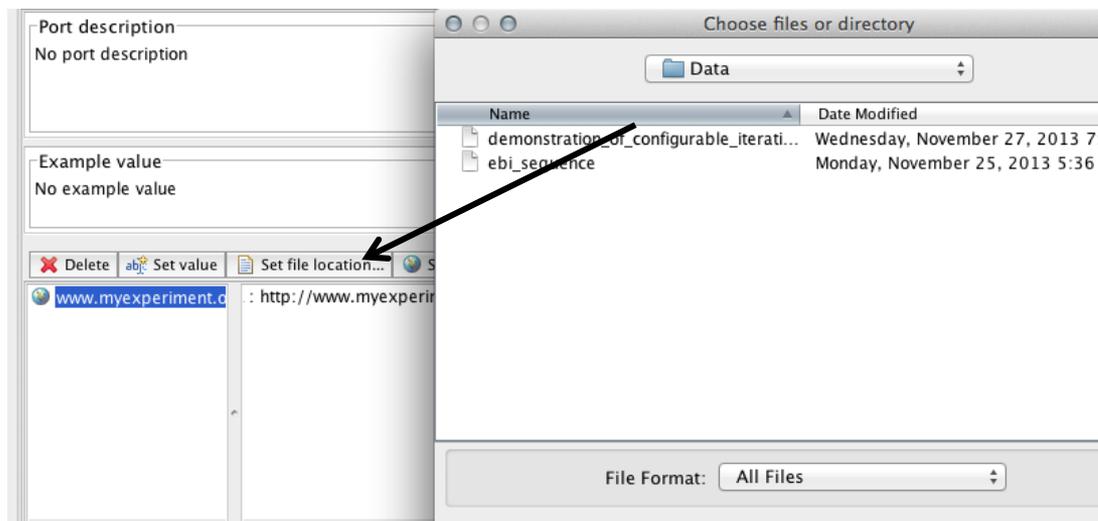
- In the **Design** view, right-click on a blank part of the workflow diagram and select **Annotate**
- Add some details about the workflow e.g. who is the author, what the workflow does
- You can also add examples and descriptions for the workflow inputs by selecting them and selecting **Annotate**
- Add an example for the gene *Adora2a*
- Save the workflow by going to “File -> save workflow”
- Run the workflow again and look at the results

## Additional Exercise 1: Xpath Service

- This exercise is optional.
- Our workflow returns the result in the XML format.
- Taverna provides a service which helps to process XML data – **XPath service**
- Go to myExperiment and find *XPath service Tutorial*
- Using this tutorial try adding the **XPath Service** to the workflow to process the XML results

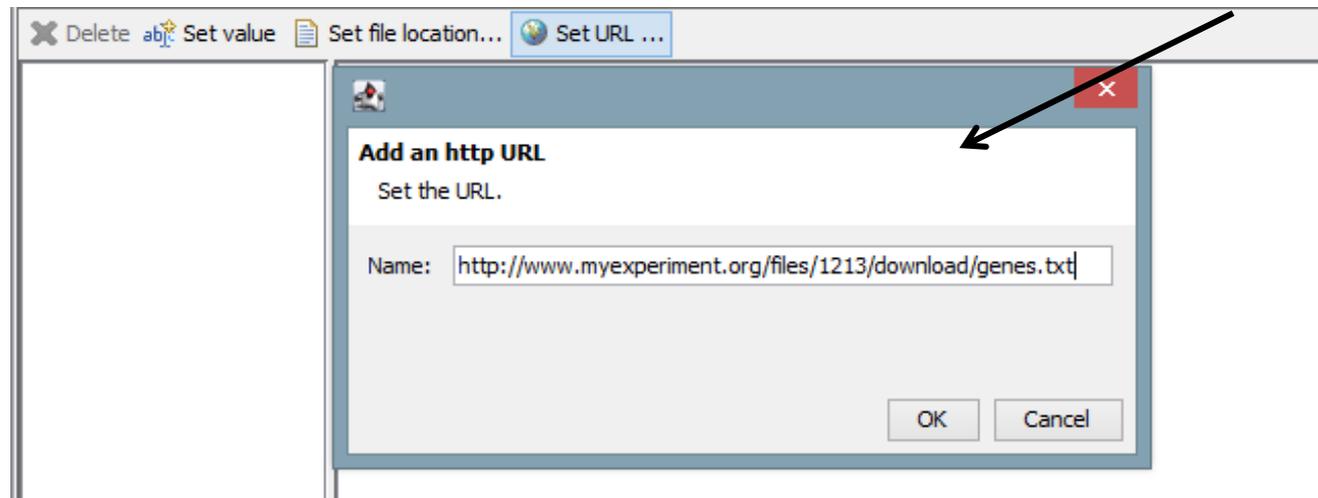
## Exercise 2: Entering data from different sources

- We can add input data into the workflow not only manually but also from a file. Go to myExperiment pack and download the file *Genes for VPH-Share workshop*
- Click **Run workflow** again but instead of selecting Set value select **Set file location** and navigate to where you saved the file



## Exercise 2: Entering data from different sources

- Instead of downloading the file we can point the workflow to the file's URL (if we know it). Let's run the workflow again but this time select **Set URL** and paste in <http://www.myexperiment.org/files/1213/download/genes.txt>



## Exercise 2: Entering data from different sources

- So far we used simple text files but it is also possible to use spreadsheets as sources of input data. In order to do that we will need to add a Spreadsheet tool to our workflow.
- From the myExperiment group download the file *genes.xls* open it on your machine and see what it contains (the list of the genes is in cells B2 to B5)
- From the **Service Templates** select **SpreadsheetImport** right-click on it and add it to the workflow

## Exercise 2: Entering data from different sources

The screenshot shows a software interface with a top toolbar containing various icons for file operations and workflow management. Below the toolbar is a navigation bar with tabs for 'Design', 'Results', 'my myExperiment', 'XworX BIFI Perspective', and 'Service Catalogue'. The main area is titled 'Service panel' and features a search filter with a 'Clear' button and an 'Import new services' button. A list of 'Available services' is shown under a 'Service templates' folder, including:

- Beanshell - A service that allows Beanshell scripts, with dependencies on libraries
- Interaction
- Nested workflow - A service that allows you to have one workflow nested within another
- REST Service - A generic REST service that can handle all HTTP methods
- Rshell - A service that allows the calling of R scripts on an R server
- SpreadsheetImport - A service that imports data from spreadsheets**
- Text constant - A string value that you can set
- Tool - A service that allows tools to be used as services

## Exercise 2: Entering data from different sources

- In the pop up window set the correct range for columns and rows (untick the box “all rows”)

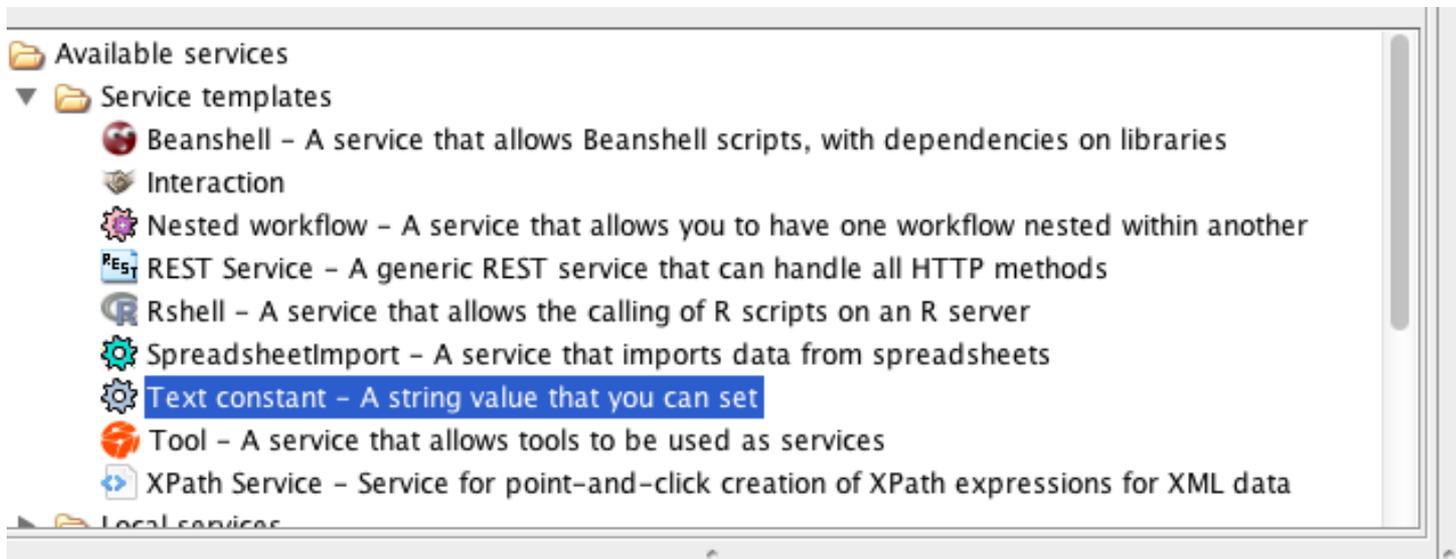
The image shows a 'Spreadsheet Import Configuration' dialog box overlaid on a workflow diagram. The dialog box has the following settings:

- Columns:** From B1 to B1
- Rows:** From 2 to 5,  All rows,  Exclude header row
- Ignore blank rows
- Empty cells:**  Use an empty string,  Use this value: [text box],  Generate an error value

The workflow diagram shows a 'gene' input port connected to a 'gene' activity. The activity has two output ports: 'responseBody' and 'status'. The 'responseBody' output port is connected to an 'experiments' output port. A 'Spreadsheet' icon is also visible in the workflow.

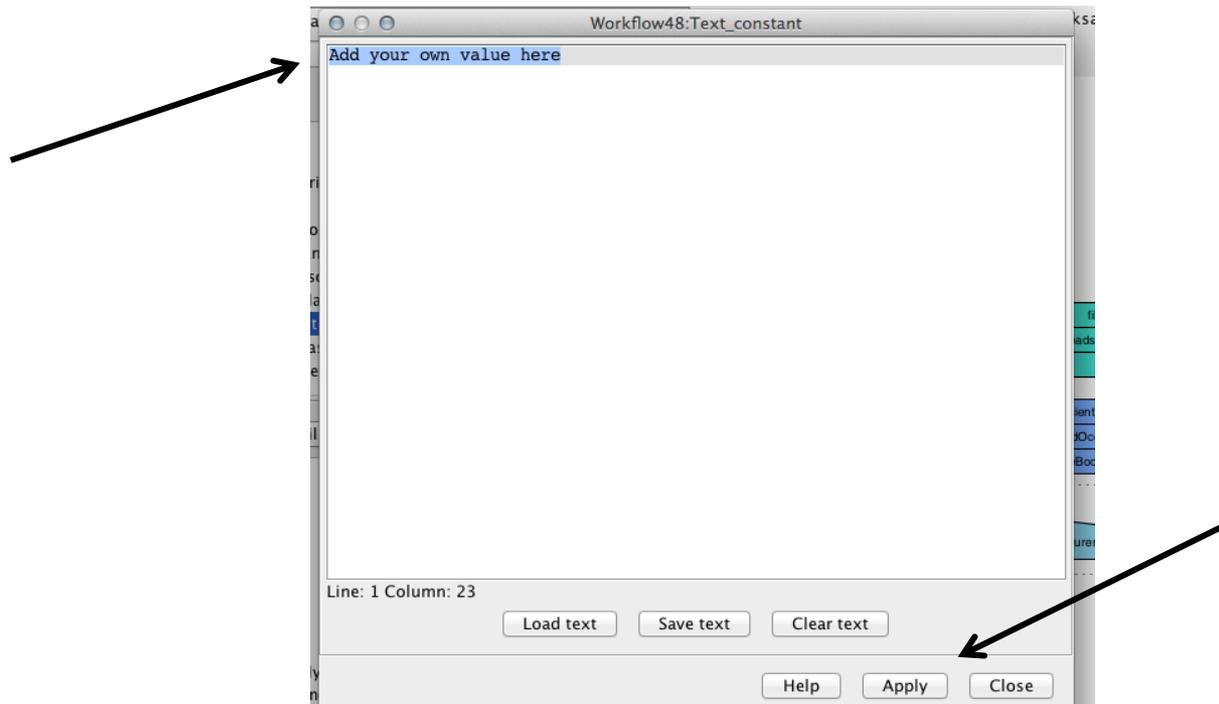
## Exercise 2: Entering data from different sources

- We need to delete the input port for the workflow (right click on it and select **Delete**)
- The Spreadsheet tool expects as an input the URL (or path) to the file. The best way to feed in that URL/path is to add a service called **Text constant**



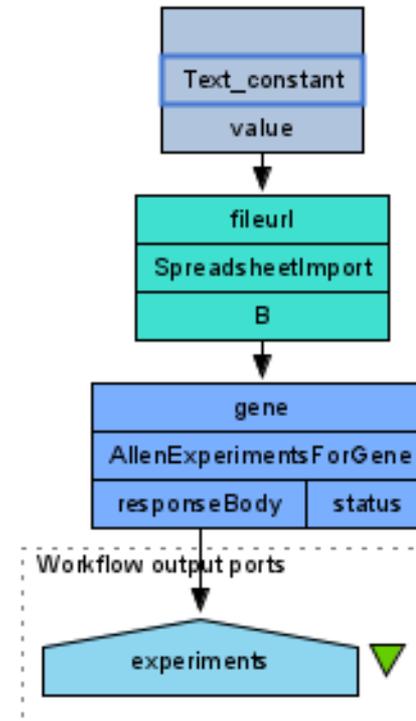
## Exercise 2: Entering data from different sources

- Where it says **Add your own value here** enter:  
*<http://www.myexperiment.org/files/1214/download/genes.xls>*  
(or if you prefer the full path to your local file), then **Apply** and **Close**



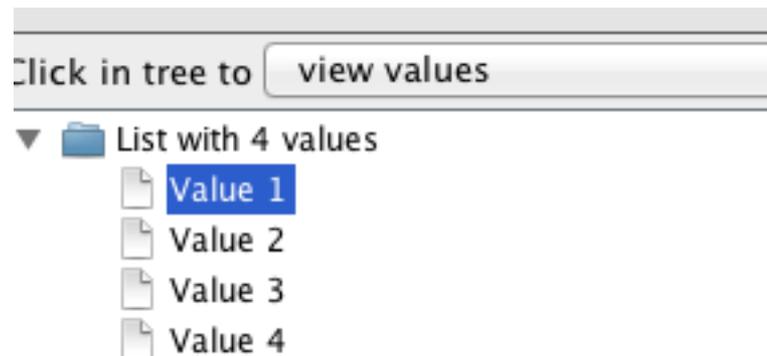
## Exercise 2: Entering data from different sources

- Connect the *Text constant* with the *SpreadsheetImport* and the *SpreadsheetImport* with the input to the *AllenExperimentsForGene* service



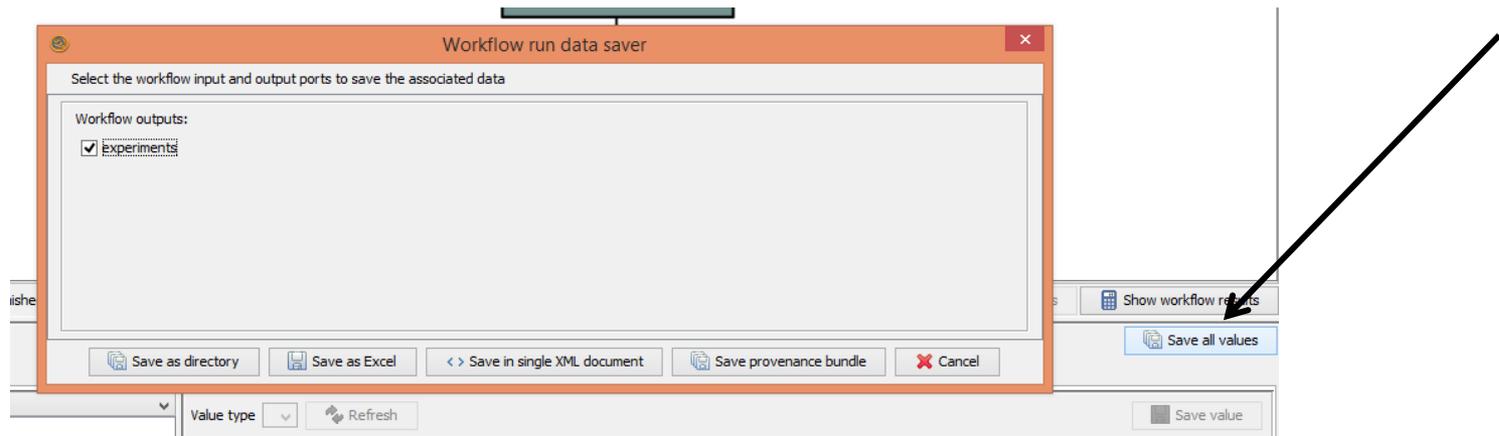
## Exercise 2: Entering data from different sources

- When we run the workflow, we can see that there are four values for the results (as there were 4 gene names that we read from the spreadsheet).
- Taverna implicitly iterated over these 4 input values and processed them.



## Exercise 3: Saving workflow results

- Taverna allows you to save results in different formats and also allows you to save intermediate workflow results (which is very useful when you run a large workflow)
- You can save all result values:



- Taverna allows you to save values in a variety of formats

## Exercise 3: Saving workflow results

- You can also save each single value separately:

The screenshot shows a web interface for viewing workflow results. On the left, a tree view shows a folder 'List with 4 values' containing four items: 'Value 1', 'Value 2', 'Value 3', and 'Value 4'. An arrow points to 'Value 1'. The main area displays the content of 'Value 1' as XML. Above the XML is a 'Value type' dropdown set to 'Text', a 'Refresh' button, a 'Wrap text' checkbox, and a 'Save value' button. An arrow points to the 'Save value' button. The XML content is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="http://data.gbif.org/ws/rest/occurrence/stylesheet"?>
<gbif:gbifResponse xsi:schemaLocation="http://portal.gbif.org/ws/response/gbif http://data.gbif.org/ws/rest/occurrence/
<gbif:header>
  <gbif:help>http://data.gbif.org/ws/rest/occurrence/help
</gbif:help>
... ..
```

- In order to save intermediate values, in the results tab select the part of the workflow which you want to save the values for, then in the results window you should see these values and you will be able to save them

## Summary

- You can now
  - Create workflows from a variety of services
  - Connect services together
  - Run workflows
  - Use input values from different sources
  - Save workflows
  - Save results