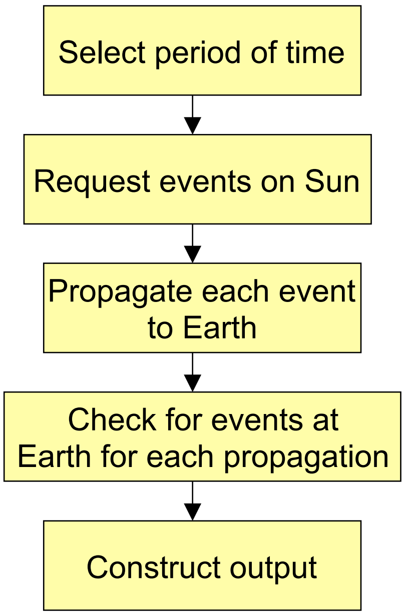
An Introduction to Designing, Executing and Sharing Workflows with Taverna

Anja Le Blanc
Stian Soiland-Reyes
Alan Willams
University of Manchester

Propagate Solar Events to Earth and find Related Events


Select period of time
Request events on Sun
Propagate each event to Earth
Check for events at Earth for each propagation
Construct output

How long is a sensible period?

Step 1 — Restrictions on request?

Step 2 — Which type of propagation?

Step 3 — Which list to use?

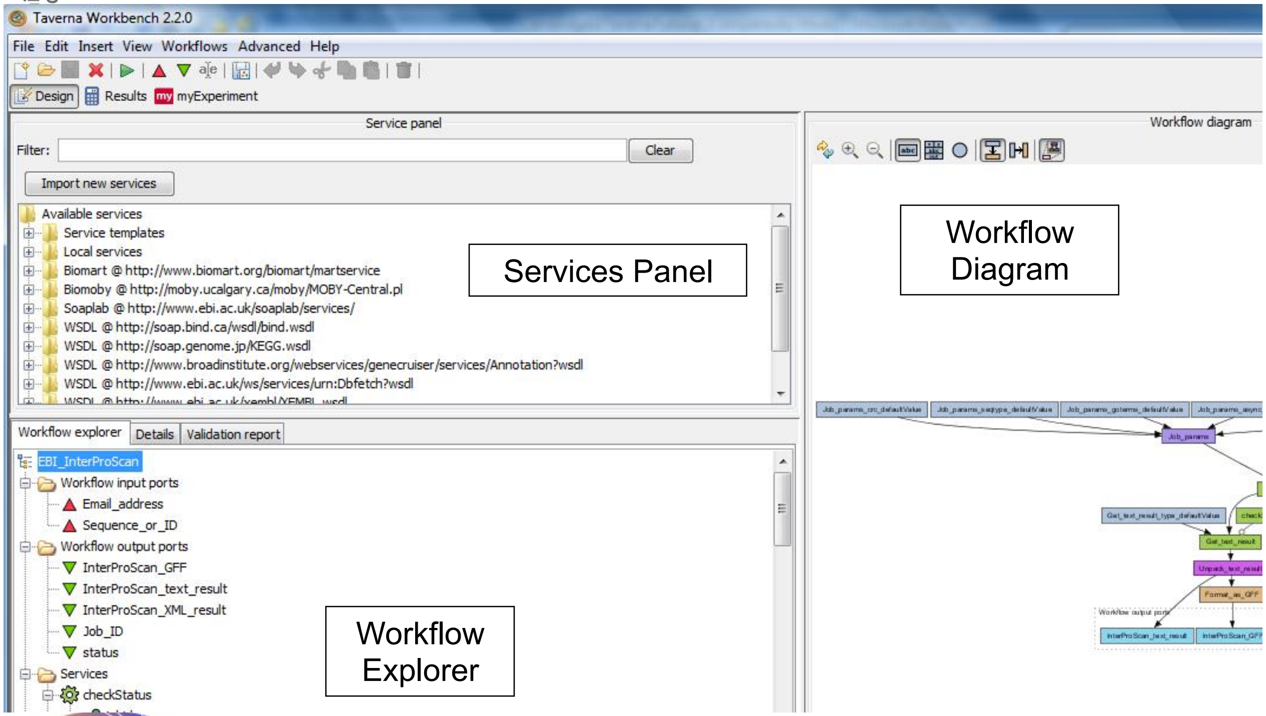
Step 4 — What information do I need and from where?

Exercise 1: Exploring the Workbench

- Taverna can be downloaded from
  http://www.taverna.org.uk/download
  Find the latest **Taverna Workbench** (2.3)
- Download the correct version for your operating system
- Follow the install instructions on the download page
  - Linux users: Check that you have installed Graphviz and Sun/ Oracle Java. Do not use OpenJDK
- Start Taverna

  The following page shows a screenshot of Taverna and the different panels that make up the workbench

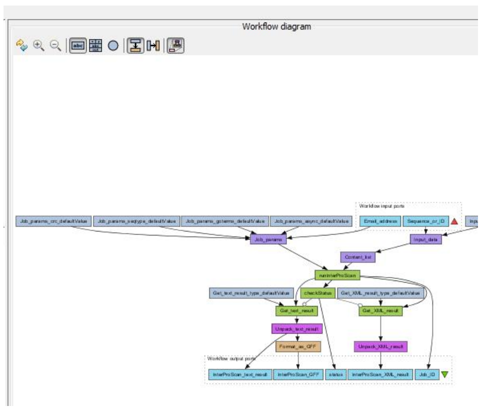
Taverna Workbench



Available services
- Service templates
- Local services
- Biomart @ http://www.biomart.org/biomart/martservice
- Biomoby @ http://moby.ucalgary.ca/moby/MOBY-Central.pl
- Soaplab @ http://www.ebi.ac.uk/soaplab/services/
- WSDL @ http://soap.bind.ca/wsdl/bind.wsdl
- WSDL @ http://soap.genome.jp/KEGG.wsdl
- WSDL @ http://www.broadinstitute.org/webservices/genecruiser/services/Annotation?wsdl
- WSDL @ http://www.ebi.ac.uk/ws/services/urn:Dbfetch?wsdl

EBI_InterProScan
- Workflow input ports
  - Email_address
  - Sequence_or_ID
- Workflow output ports
  - InterProScan_GFF
  - InterProScan_text_result
  - InterProScan_XML_result
  - Job_ID
  - status
- Services
  - checkStatus

1. Workflow Diagram

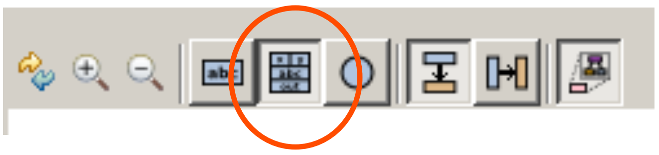


The workflow diagram is the visual representation of the workflow

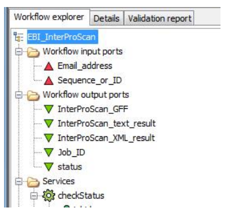
- Shows inputs/outputs, services and control flows
- Allows editing of the workflow by dragging and dropping and connecting services together
- Enables saving of workflow diagrams for publishing and sharing

1. Workflow Diagram

- Switch from ‘Display no service ports’ to Display all service ports’ for all exercises



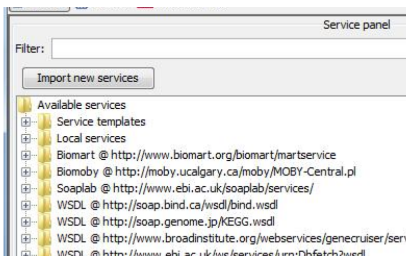
1. Workflow Explorer


Workflow explorer | Details | Validation report

- EBI_InterProScan
  - Workflow input ports
    - Email_address
    - Sequence_or_ID
  - Workflow output ports
    - InterProScan_GFF
    - InterProScan_text_result
    - InterProScan_XML_result
    - Job_ID
    - status
  - Services
    - checkStatus

- The **Workflow Explorer** shows a tree view of the workflow open in the diagram, such as *inputs***,** *outputs* and *services*
- The **Details** tab shows information about the current *selection* in the Diagram and Workflow Explorer, such as *default values*, *descriptions* for service inputs and outputs and where remote services are *located*. It also shows *configuration* details, such as *iteration* and *looping* (more on these later).
- The **validation** tab shows an inspection report of the current workflow. Taverna checks connections, configurations and service availability.

1. Available Services Panel


Service panel

Filter:

Import new services

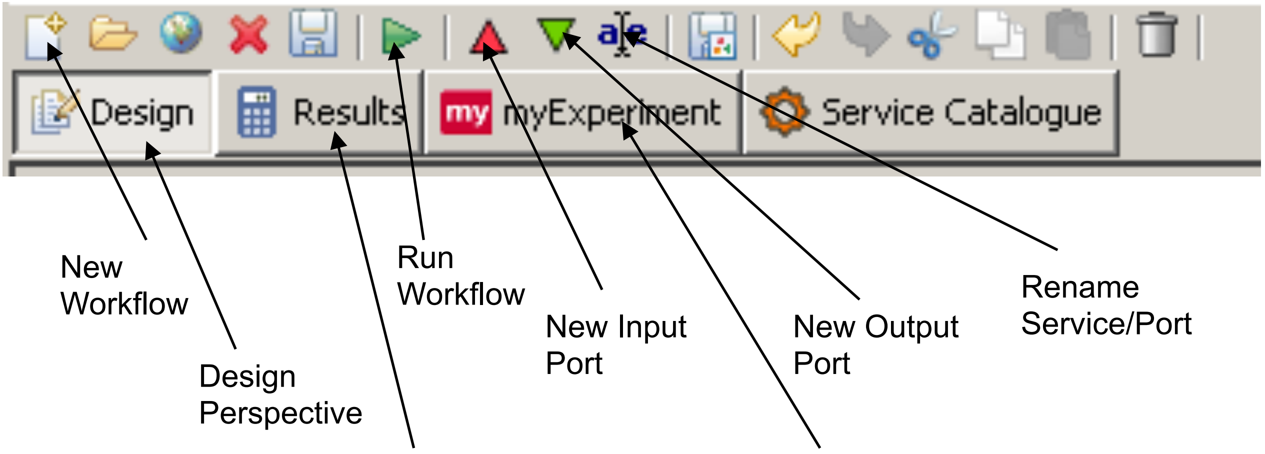
- Available services
  - Service templates
  - Local services
  - Biomart @ http://www.biomart.org/biomart/martservice
  - Biomoby @ http://moby.ucalgary.ca/moby/MOBY-Central.pl
  - Soaplab @ http://www.ebi.ac.uk/soaplab/services/
  - WSDL @ http://soap.bind.ca/wsdl/bind.wsdl
  - WSDL @ http://soap.genome.jp/KEGG.wsdl
  - WSDL @ http://www.broadinstitute.org/webservices/genecruiser/serv
  - WSDL @ http://www.ebi.ac.uk/ws/services/urn:Dbfetch?wsdl

Lists services available by default in Taverna

- Local java services
- Command line tools
- WSDL Web Service – secure and public
- RESTful Services
- R scripts (for statistical analyses)
- Beanshell scripts
- Xpath scripts
- Spreadsheet import service

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

Toolbar



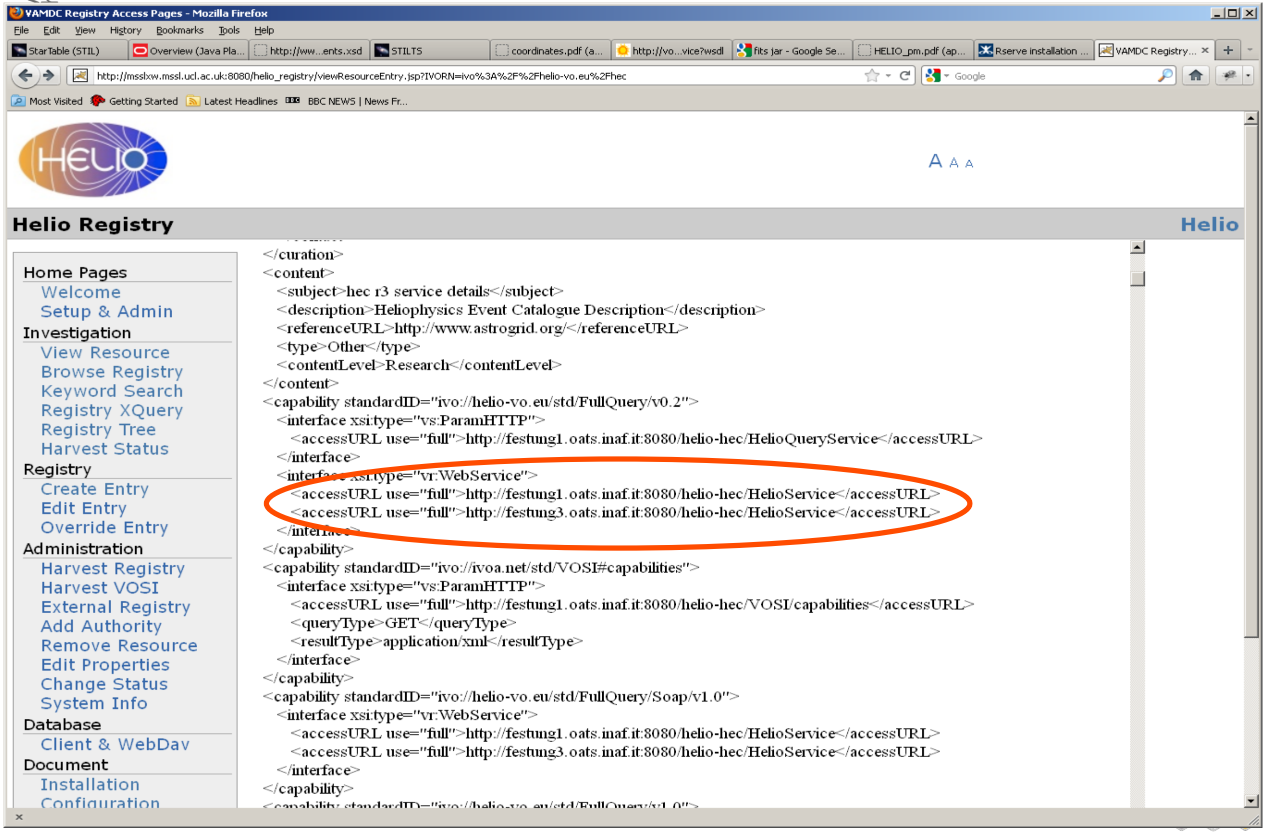
Viewing Results Perspective

myExperiment Perspective

Exercise 2: Changing available Services

- By default Taverna comes with a whole set of Bioinformatics services preloaded
- Remove a WSDL service by right clicking on the service and selecting the entry below ‘*Remove individual service provider*’
- Explore HELIO registry at http://msslkz.mssl.ucl.ac.uk/helio_registry/
- Try to find the URL for one HEC (Helio Event Catalogue) WSDL
  Use the <AccessURL> of the WebService to find details for the service

How to find HELIO services cont …


VAMDC Registry Access Pages - Mozilla Firefox

http://msslxw.mssl.ucl.ac.uk:8080/helio_registry/viewResourceEntry.jsp?IVORN=ivo%3A%2F%2Fhelio-vo.eu%2Fhec

Helio Registry

Home Pages
- Welcome
- Setup & Admin

Investigation
- View Resource
- Browse Registry
- Keyword Search
- Registry XQuery
- Registry Tree
- Harvest Status

Registry
- Create Entry
- Edit Entry
- Override Entry

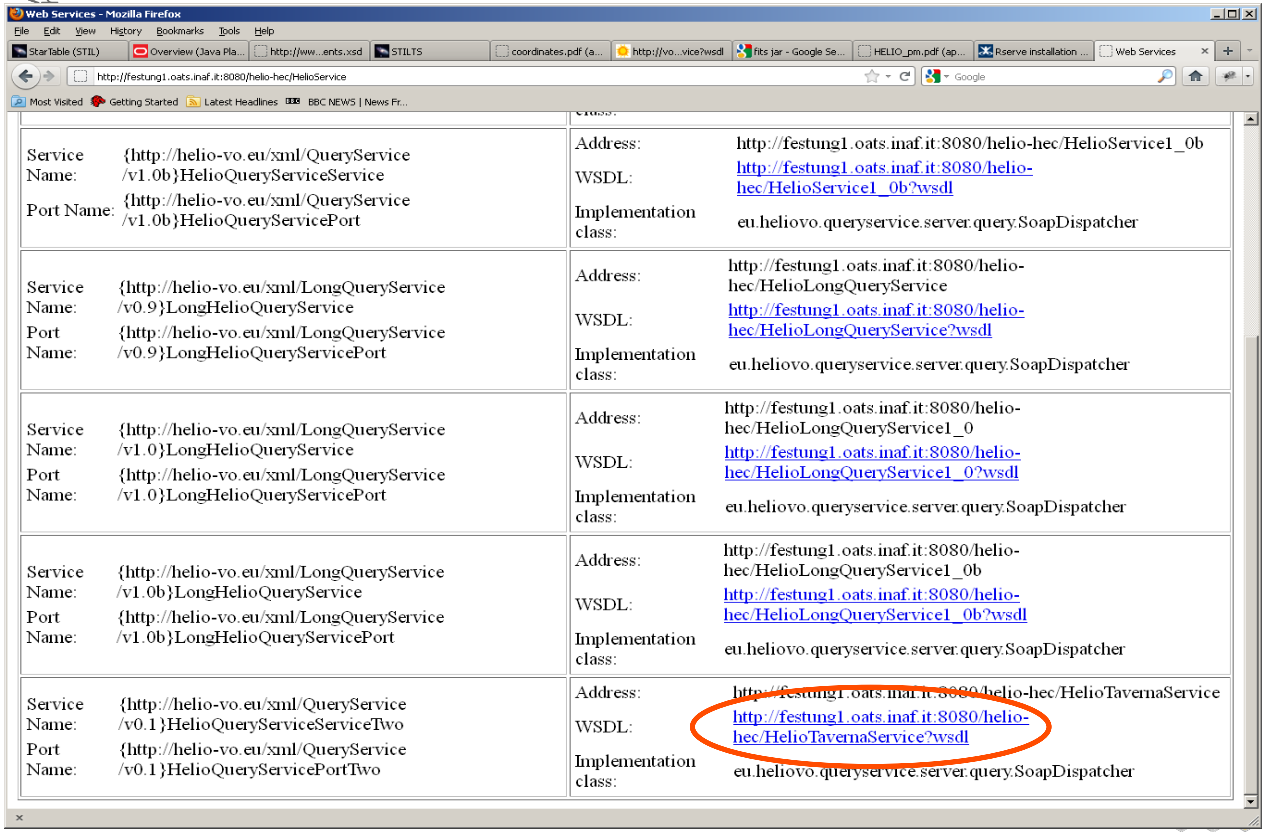
Administration
- Harvest Registry
- Harvest VOSI
- External Registry
- Add Authority
- Remove Resource
- Edit Properties
- Change Status
- System Info

Database
- Client & WebDav

Document
- Installation
- Configuration

```
</curation>
<content>
  <subject>hec r3 service details</subject>
  <description>Heliophysics Event Catalogue Description</description>
  <referenceURL>http://www.astrogrid.org/</referenceURL>
  <type>Other</type>
  <contentLevel>Research</contentLevel>
</content>
<capability standardID="ivo://helio-vo.eu/std/FullQuery/v0.2">
  <interface xsi:type="vs:ParamHTTP">
    <accessURL use="full">http://festung1.oats.inaf.it:8080/helio-hec/HelioQueryService</accessURL>
  </interface>
  <interface xsi:type="vr:WebService">
    <accessURL use="full">http://festung1.oats.inaf.it:8080/helio-hec/HelioService</accessURL>
    <accessURL use="full">http://festung3.oats.inaf.it:8080/helio-hec/HelioService</accessURL>
  </interface>
</capability>
<capability standardID="ivo://ivoa.net/std/VOSI#capabilities">
  <interface xsi:type="vs:ParamHTTP">
    <accessURL use="full">http://festung1.oats.inaf.it:8080/helio-hec/VOSI/capabilities</accessURL>
    <queryType>GET</queryType>
    <resultType>application/xml</resultType>
  </interface>
</capability>
<capability standardID="ivo://helio-vo.eu/std/FullQuery/Soap/v1.0">
  <interface xsi:type="vr:WebService">
    <accessURL use="full">http://festung1.oats.inaf.it:8080/helio-hec/HelioService</accessURL>
    <accessURL use="full">http://festung3.oats.inaf.it:8080/helio-hec/HelioService</accessURL>
  </interface>
</capability>
```

How to find HELIO services

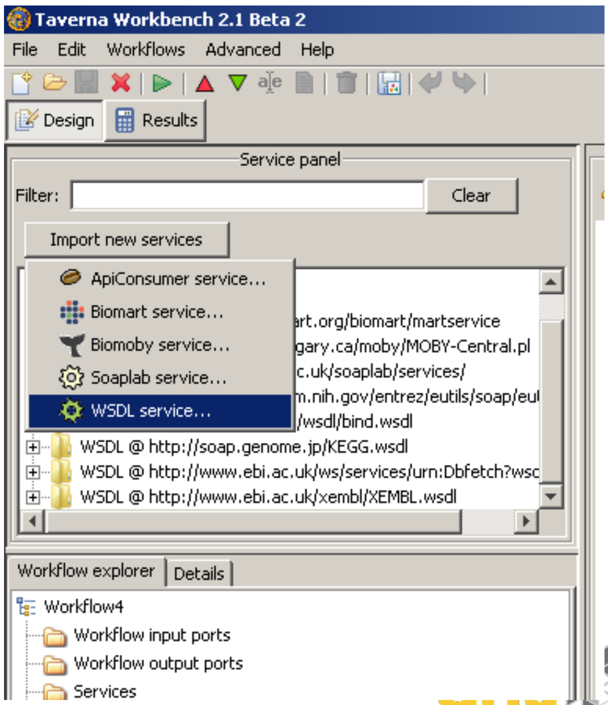

Web Services - Mozilla Firefox

http://festung1.oats.inaf.it:8080/helio-hec/HelioService

Service Name:	{http://helio-vo.eu/xml/QueryService/v1.0b}HelioQueryServiceService	Address:	http://festung1.oats.inaf.it:8080/helio-hec/HelioService1_0b
Port Name:	{http://helio-vo.eu/xml/QueryService/v1.0b}HelioQueryServicePort	WSDL:	http://festung1.oats.inaf.it:8080/helio-hec/HelioService1_0b?wsdl
		Implementation class:	eu.heliovo.queryservice.server.query.SoapDispatcher
Service Name:	{http://helio-vo.eu/xml/LongQueryService/v0.9}LongHelioQueryService	Address:	http://festung1.oats.inaf.it:8080/helio-hec/HelioLongQueryService
Port Name:	{http://helio-vo.eu/xml/LongQueryService/v0.9}LongHelioQueryServicePort	WSDL:	http://festung1.oats.inaf.it:8080/helio-hec/HelioLongQueryService?wsdl
		Implementation class:	eu.heliovo.queryservice.server.query.SoapDispatcher
Service Name:	{http://helio-vo.eu/xml/LongQueryService/v1.0}LongHelioQueryService	Address:	http://festung1.oats.inaf.it:8080/helio-hec/HelioLongQueryService1_0
Port Name:	{http://helio-vo.eu/xml/LongQueryService/v1.0}LongHelioQueryServicePort	WSDL:	http://festung1.oats.inaf.it:8080/helio-hec/HelioLongQueryService1_0?wsdl
		Implementation class:	eu.heliovo.queryservice.server.query.SoapDispatcher
Service Name:	{http://helio-vo.eu/xml/LongQueryService/v1.0b}LongHelioQueryService	Address:	http://festung1.oats.inaf.it:8080/helio-hec/HelioLongQueryService1_0b
Port Name:	{http://helio-vo.eu/xml/LongQueryService/v1.0b}LongHelioQueryServicePort	WSDL:	http://festung1.oats.inaf.it:8080/helio-hec/HelioLongQueryService1_0b?wsdl
		Implementation class:	eu.heliovo.queryservice.server.query.SoapDispatcher
Service Name:	{http://helio-vo.eu/xml/QueryService/v0.1}HelioQueryServiceServiceTwo	Address:	http://festung1.oats.inaf.it:8080/helio-hec/HelioTavernaService
Port Name:	{http://helio-vo.eu/xml/QueryService/v0.1}HelioQueryServicePortTwo	WSDL:	http://festung1.oats.inaf.it:8080/helio-hec/HelioTavernaService?wsdl
		Implementation class:	eu.heliovo.queryservice.server.query.SoapDispatcher

2. Adding New WSDL Services

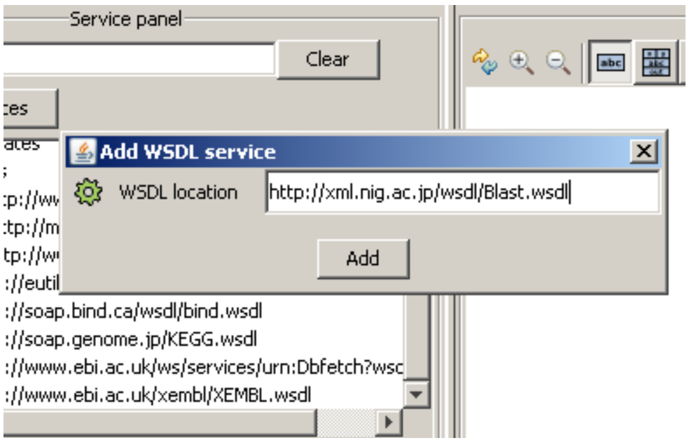
- Go to the *services* panel in Taverna and click "*Import new services*". For each type of service, you are given the option to add a new service
- Select '*WSDL service…*' A window will pop-up asking for a web address



rt.org/biomart/martservice
gary.ca/moby/MOBY-Central.pl
c.uk/soaplab/services/
n.nih.gov/entrez/eutils/soap/eu
/wsdl/bind.wsdl
WSDL @ http://soap.genome.jp/KEGG.wsdl
WSDL @ http://www.ebi.ac.uk/ws/services/urn:Dbfetch?wsc
WSDL @ http://www.ebi.ac.uk/xembl/XEMBL.wsdl

2. Adding New WSDL Services

- Enter the service address you just found
- Scroll down the *Services* list, you will see your new service there
- Have a look at the functions this service provides



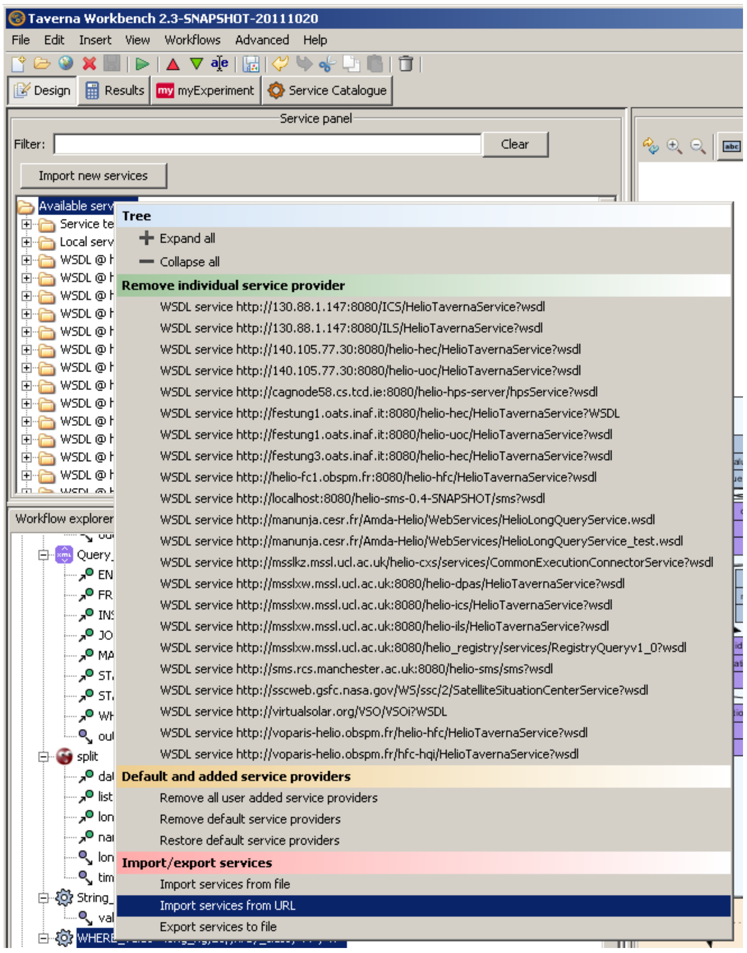
WSDL location http://xml.nig.ac.jp/wsdl/Blast.wsdl

://soap.bind.ca/wsdl/bind.wsdl
://soap.genome.jp/KEGG.wsdl
://www.ebi.ac.uk/ws/services/urn:Dbfetch?wsc
://www.ebi.ac.uk/xembl/XEMBL.wsdl

2. Changing the Service Set

- A Service Set is a list of services of a domain
- It helps adding many related services at one go
- URL of the Service Set: http://www.myexperiment.org/files/687/download/TavernaHelioServices.xml
  (more to MyExperiment later)

- Right click on 'Available Services'
- Select 'Import services from URL'
- Use URL from previous slide
- Replace existing services



Tree

- Expand all
- Collapse all

Remove individual service provider

WSDL service http://130.88.1.147:8080/ICS/HelioTavernaService?wsdl
WSDL service http://130.88.1.147:8080/ILS/HelioTavernaService?wsdl
WSDL service http://140.105.77.30:8080/helio-hec/HelioTavernaService?wsdl
WSDL service http://140.105.77.30:8080/helio-uoc/HelioTavernaService?wsdl
WSDL service http://cagnode58.cs.tcd.ie:8080/helio-hps-server/hpsService?wsdl
WSDL service http://festung1.oats.inaf.it:8080/helio-hec/HelioTavernaService?WSDL
WSDL service http://festung1.oats.inaf.it:8080/helio-uoc/HelioTavernaService?wsdl
WSDL service http://festung3.oats.inaf.it:8080/helio-hec/HelioTavernaService?wsdl
WSDL service http://helio-fc1.obspm.fr:8080/helio-hfc/HelioTavernaService?wsdl
WSDL service http://localhost:8080/helio-sms-0.4-SNAPSHOT/sms?wsdl
WSDL service http://manunja.cesr.fr/Amda-Helio/WebServices/HelioLongQueryService.wsdl
WSDL service http://manunja.cesr.fr/Amda-Helio/WebServices/HelioLongQueryService_test.wsdl
WSDL service http://msslkz.mssl.ucl.ac.uk/helio-cxs/services/CommonExecutionConnectorService?wsdl
WSDL service http://msslxw.mssl.ucl.ac.uk:8080/helio-dpas/HelioTavernaService?wsdl
WSDL service http://msslxw.mssl.ucl.ac.uk:8080/helio-ics/HelioTavernaService?wsdl
WSDL service http://msslxw.mssl.ucl.ac.uk:8080/helio-ils/HelioTavernaService?wsdl
WSDL service http://msslxw.mssl.ucl.ac.uk:8080/helio_registry/services/RegistryQueryv1_0?wsdl
WSDL service http://sms.rcs.manchester.ac.uk:8080/helio-sms/sms?wsdl
WSDL service http://sscweb.gsfc.nasa.gov/WS/ssc/2/SatelliteSituationCenterService?wsdl
WSDL service http://virtualsolar.org/VSO/VSOi?WSDL
WSDL service http://voparis-helio.obspm.fr/helio-hfc/HelioTavernaService?wsdl
WSDL service http://voparis-helio.obspm.fr/hfc-hqi/HelioTavernaService?wsdl

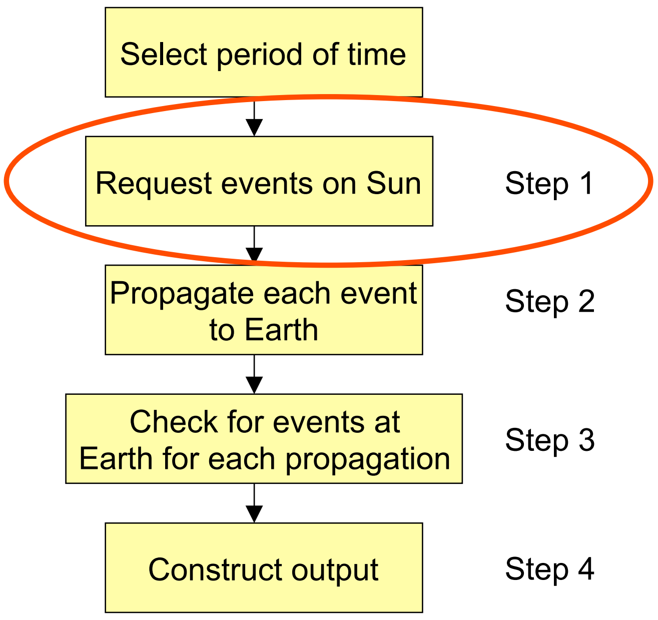
Default and added service providers

Remove all user added service providers
Remove default service providers
Restore default service providers

Import/export services

Import services from file
Import services from URL
Export services to file

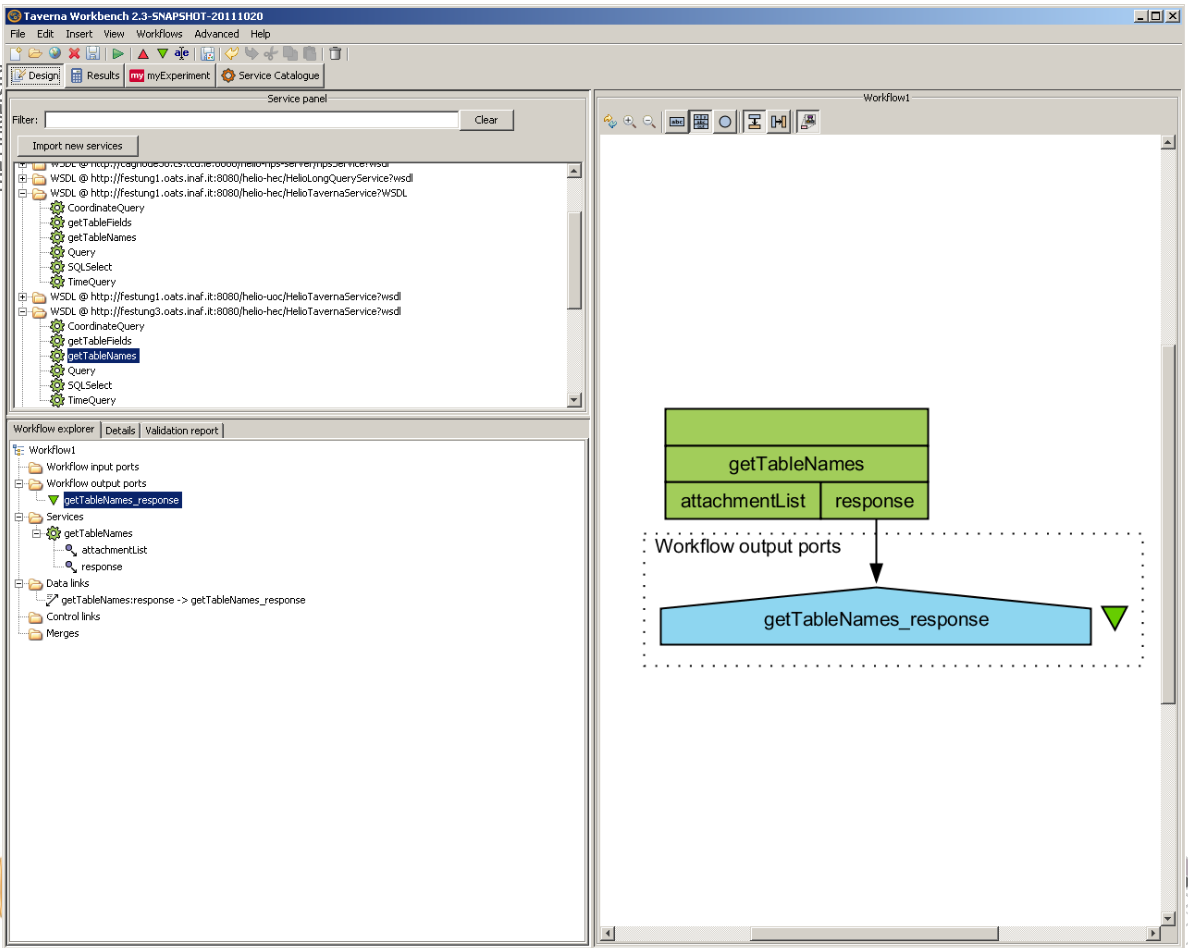
Step 1



Exercise 3: Call a Service

- Locate a HEC service in the Service Panel

  Find information about the call:
- Drag and Drop function ‘*getTabelNames*’ into the Workflow Diagram
- Connect the ‘*response*’ to a new Output port (right click on background of Workflow Diagram and select option)
- Run workflow
- Look at the results and chose the table ‘*goes_sxr_flare*’



WSDL @ http://festung1.oats.inaf.it:8080/helio-hec/HelioLongQueryService?wsdl
WSDL @ http://festung1.oats.inaf.it:8080/helio-hec/HelioTavernaService?WSDL
- CoordinateQuery
- getTableFields
- getTableNames
- Query
- SQLSelect
- TimeQuery

WSDL @ http://festung1.oats.inaf.it:8080/helio-uoc/HelioTavernaService?wsdl
WSDL @ http://festung3.oats.inaf.it:8080/helio-hec/HelioTavernaService?wsdl
- CoordinateQuery
- getTableFields
- getTableNames
- Query
- SQLSelect
- TimeQuery

Workflow explorer | Details | Validation report

Workflow1
- Workflow input ports
- Workflow output ports
  - getTableNames_response
- Services
  - getTableNames
    - attachmentList
    - response
- Data links
  - getTableNames:response -> getTableNames_response
- Control links
- Merges

Exercise 3: Call a Service

- Construct a new workflow (*File -> New workflow*)
- Chose function ‘*getTableFields*’
- Add an ‘*XML Input Splitter*’ (right click on *getTableFields*; select option)

XML Splitter decompose complex types into their components; they do not automatically decompose them to their atoms, so you might require several *XML Splitters*.

- Add a *String constant* (right click on background of Workflow Diagram and select option)

A *String constant* is a simple processer which provides a single string as output, the value of that string does not change.

- Rename String constant (right click on *String_constant* and select option)
- Drag and Drop a data flow from ‘*value*’ of your String constant to ‘*table_name*’ of the XML Input Splitter
- Add a workflow output port
- Run the workflow

Exercise 3: Call a Service

- Construct a new workflow
- Chose function ‘*Query*’
- Add ‘*XML Input Splitter*’
- Create an *String constant* with the value of the table name (as before)
- Connect string constant to the ‘*FROM*’ field
- Create input ports for ‘*STARTTIME*’ and ‘*ENDTIME*’ (Example: *2001-01-20T01:00:00*)
- Create output port for the response
- Run the workflow

Exercise 3: Call a Service

- Examine the output
- ‘*WHERE*’ input expects a string with a PQL WHERE parameter statement
- More info to PQL: http://www.ivoa.net/internal/IVOA/TableAccess/PQL-0.2-20090520.pdf (page 10 for WHERE)
- Modify the workflow by adding a new string constant with a WHERE which filters results to events of the *xray_class M* and *X*

  If you are already familiar with SQL try to use the SQLSelect query instead (unrelated example next slide)

- Save workflow as ‘*Step1*’

SQL example

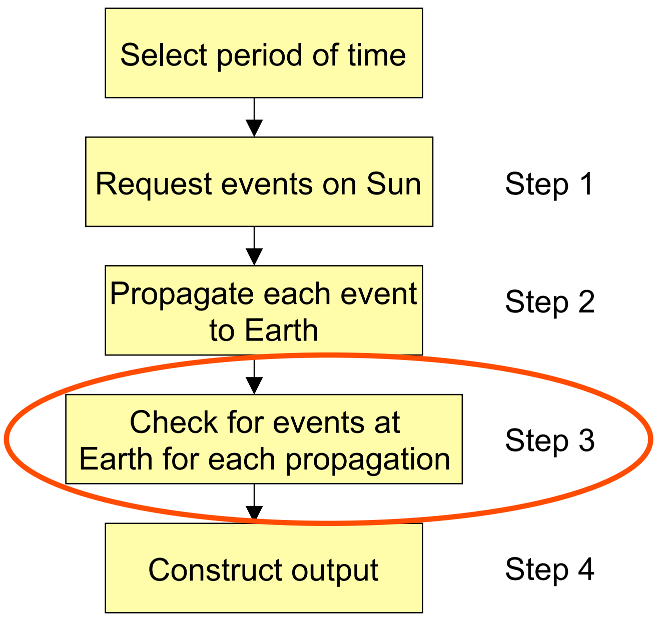
- SQL query:

```
SELECT time_start, time_end, xray_class, optical_class
FROM sgas_event WHERE xray_class IS NOT NULL
ORDER BY time_start DESC  LIMIT 25
```

- SQL SOAP service call:

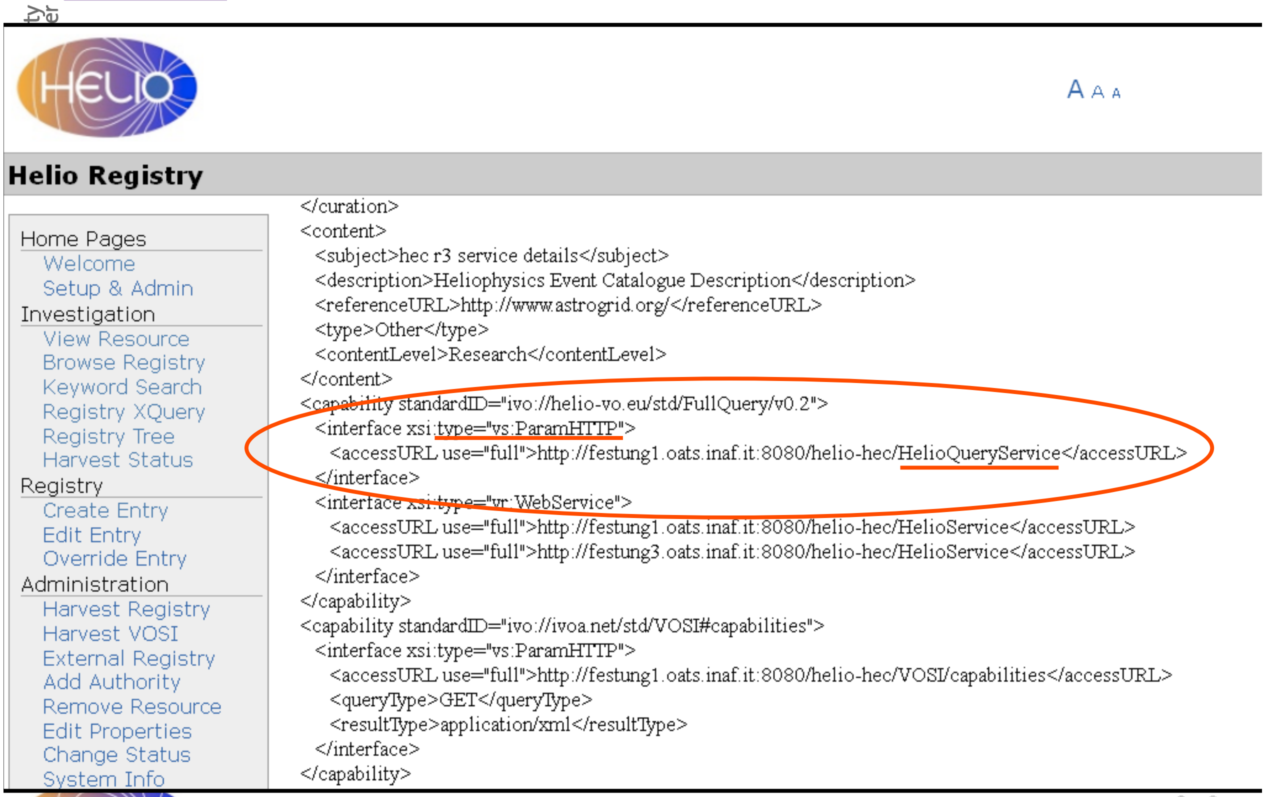
```
WHAT = time_start, time_end, xray_class, optical_class
FROM = sgas_event
WHERE = xray_class IS NOT NULL
ORDER_BY = time_start DESC
LIMIT = 25
```

Step 3



Exercise 4: Call a REST Service

- Use the HEC REST service to request events from goes_proton_event list for a time interval
- Use registry to find URL for a HEC REST service
- Try to write a REST service call in a internet browser first
- *http://[service url]?[parameter=value]{&[parameter=value]}**
  (See next three slides)
- Create a new workflow
- Add a *REST service*
- Allow input values for *STARTTIME* and *ENDTIME*
- HTTP Method: *GET*
- URL Template: copy past URL from web browser
- Input parameters are given by including them in *{brackets}*
  e.g.: …*&STARTINDEX={index}&* …
  produces input port '*index*' of the REST service, which value will be inserted into URL on execution
- Save as '*Step3*'



```
</curation>
<content>
  <subject>hec r3 service details</subject>
  <description>Heliophysics Event Catalogue Description</description>
  <referenceURL>http://www.astrogrid.org/</referenceURL>
  <type>Other</type>
  <contentLevel>Research</contentLevel>
</content>
<capability standardID="ivo://helio-vo.eu/std/FullQuery/v0.2">
  <interface xsi:type="vs:ParamHTTP">
    <accessURL use="full">http://festung1.oats.inaf.it:8080/helio-hec/HelioQueryService</accessURL>
  </interface>
  <interface xsi:type="vr:WebService">
    <accessURL use="full">http://festung1.oats.inaf.it:8080/helio-hec/HelioService</accessURL>
    <accessURL use="full">http://festung3.oats.inaf.it:8080/helio-hec/HelioService</accessURL>
  </interface>
</capability>
<capability standardID="ivo://ivoa.net/std/VOSI#capabilities">
  <interface xsi:type="vs:ParamHTTP">
    <accessURL use="full">http://festung1.oats.inaf.it:8080/helio-hec/VOSI/capabilities</accessURL>
    <queryType>GET</queryType>
    <resultType>application/xml</resultType>
  </interface>
</capability>
```

HELIO Query Interface: known parameters

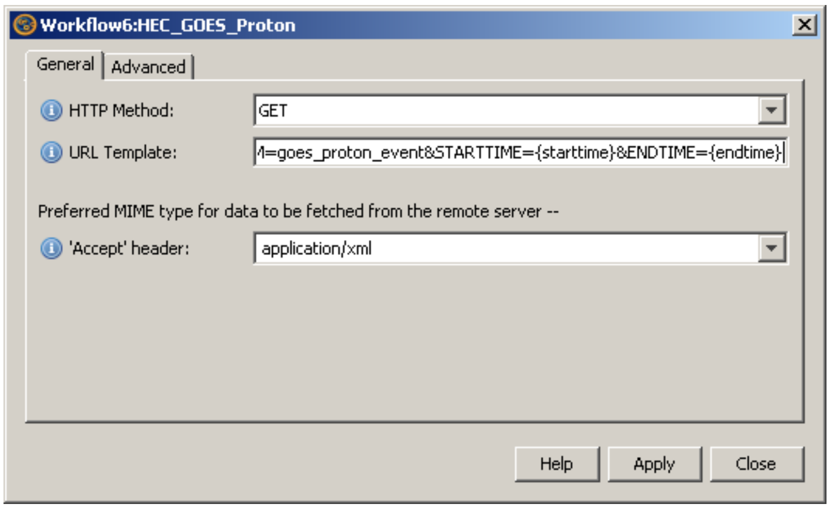
FROM (required)	List name
STARTTIME (required)	Start time of search 2000-03-20T10:23:00
ENDTIME	End time of search 2000-03-20T10:23:00
SELECT	List of fields in the return (default all)
WHERE	PQL where statement
STARTINDEX	Index of the first returned row
MAXRECORDS	Number of maximal returned rows
JOIN	Indication that two tables of the same service need to be joined

HELIO Query Interface: known parameters

SQL like select query

FROM (required)	List name
WHAT	List of fields in the return (default all)
SQLWHERE	SQL where statement
ORDER_BY	In which order the result should be orderd
OFFSET	Index of the first returned row
LIMIT	Number of maximal returned rows

REST Service


Workflow6:HEC_GOES_Proton

General | Advanced

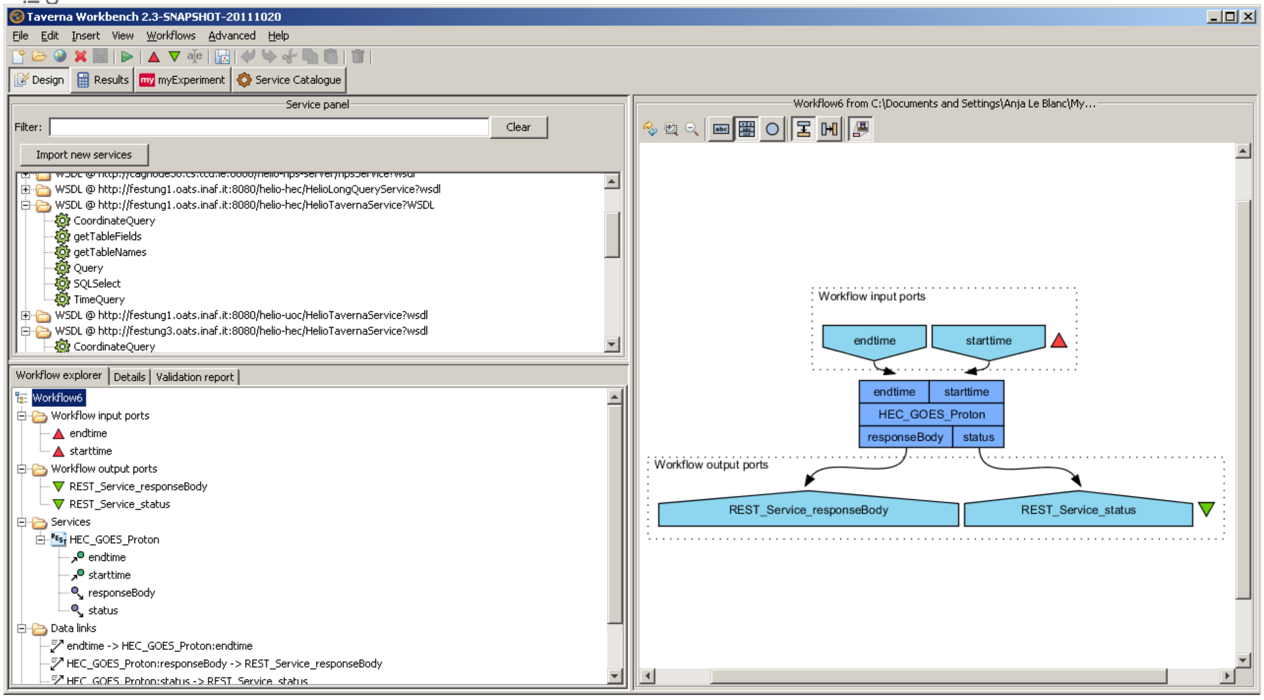
HTTP Method: GET

URL Template: M=goes_proton_event&STARTTIME={starttime}&ENDTIME={endtime}

Preferred MIME type for data to be fetched from the remote server --

'Accept' header: application/xml

REST Service


Taverna Workbench 2.3-SNAPSHOT-20111020

WSDL @ http://festung1.oats.inaf.it:8080/helio-hec/HelioLongQueryService?wsdl
WSDL @ http://festung1.oats.inaf.it:8080/helio-hec/HelioTavernaService?WSDL
- CoordinateQuery
- getTableFields
- getTableNames
- Query
- SQLSelect
- TimeQuery

WSDL @ http://festung1.oats.inaf.it:8080/helio-uoc/HelioTavernaService?wsdl
WSDL @ http://festung3.oats.inaf.it:8080/helio-hec/HelioTavernaService?wsdl
- CoordinateQuery

Workflow6
- Workflow input ports
  - endtime
  - starttime
- Workflow output ports
  - REST_Service_responseBody
  - REST_Service_status
- Services
  - HEC_GOES_Proton
    - endtime
    - starttime
    - responseBody
    - status
- Data links
  - endtime -> HEC_GOES_Proton:endtime
  - HEC_GOES_Proton:responseBody -> REST_Service_responseBody
  - HEC_GOES_Proton:status -> REST_Service_status

Workflow6 from C:\Documents and Settings\Anja Le Blanc\My…

Exploring myExperiment

- Go to http://www.myexperiment.org
- myExperiment is a social networking site for sharing workflows and workflow expertise and experiences
- Browse around the site and see what it contains
- Find everything that has been tagged with '*VOTable*', for example

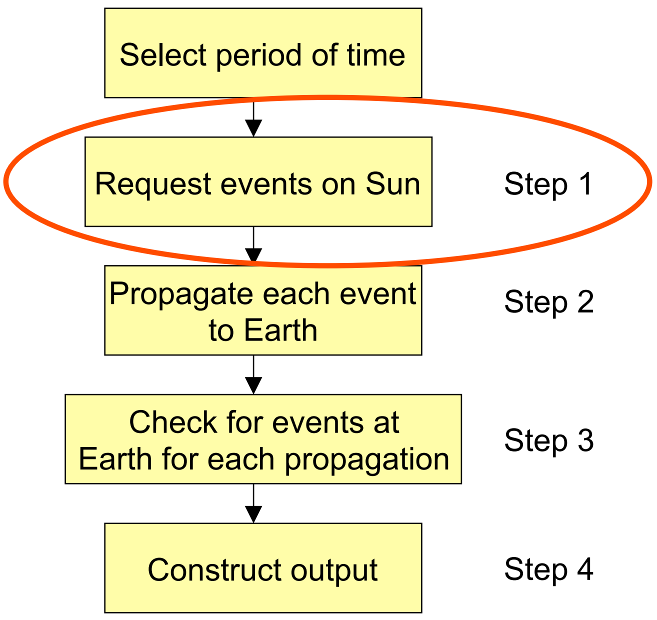
Exploring myExperiment

- There are already several workflows available for HELIO on myExperiment, but some are not public. You must join the *helio* group before you can see them and use them.
- Create yourself an account on myExperiment and join the group called '*helio*' (**NOTE**: you need to join this group to access content for future exercises)
- Once you have been accepted download the file: HELIO Taverna Workshop Material – you will need the content later on
- Find the workflow called '*Extract content of columns from VOTables*' and see what it does by reading its description.

Using Workflows from myExperiment

- You can download and run workflows from the myExperiment website, or you can use myExperiment directly from Taverna
- To use a workflow from myExperiment, you can either:
  - **Download** the workflow as a file, and open with *File->Open*
  - ..or under "*Run this Workflow in the Taverna Workbench*" **copy** the provided **download link**, in Taverna paste this URL into the dialogue of *File->Open workflows from the web*
  - ..or browse to find and open the workflow using the *myExperiment* perspective from within in Taverna

Continue Step 1



Exercise 5: Import nested workflow from myExperiment

- Go back to Taverna and change to workflow ‘*Step1*’
- click on the myExperiment icon at the top of the workbench
- Go to ‘*My Stuff*’ tab and log in (using the same credentials as the web page)
- Find the helio group by using the ‘*search*’ option.
- Look at the shared items and find the workflow called ‘*Extract content of columns from VOTables* ’
- Click on ‘*Import*’ and this workflow will be imported into your current Taverna design window
- Create *String constants* with the values *time_start* and *long_hg* (values needed for propagation)
- Connect both to the Import Port ‘*ColumnNames*’ of the nested workflow
- Connect a new *Output* ports to *ValueLists* and *ColumnNames*
- Run Workflow

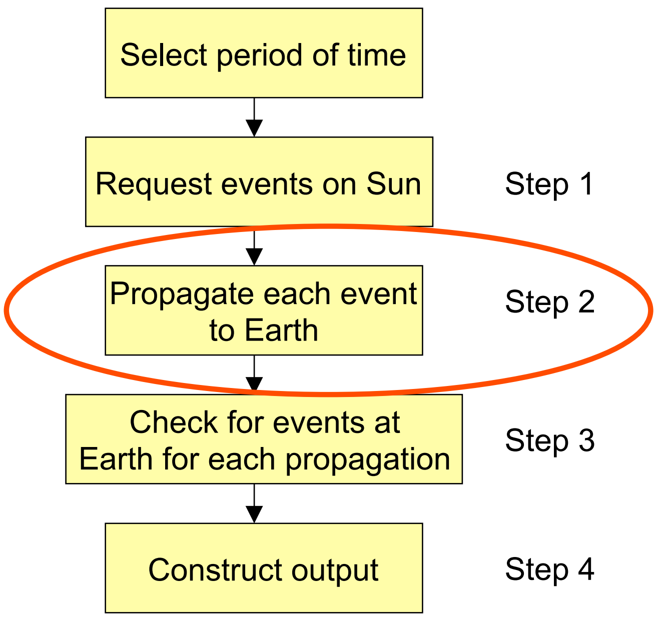
Analysing the results

- In the *Results* tab you can watch the progress of the workflow execution
- Any problems with the execution are shown by a red colour of the service which encountered a problem
- *Inputs* and *Outputs* of a workflow run are shown underneath the Graph
- Examine the two outputs
- Retrieve *intermediate values* by clicking on a service in the Graph
- Intermediate Values help to debug workflows

Improve workflow

- You notice that there are empty entries in the list with the longitude values.
- Step 2 will be the propagation of the event to Earth -> a longitude value is required
- The sun rotates -> only Events with positive longitude values have a chance to cause events at Earth
- Modify the *WHERE* clause to only retrieve Events which have a positive *long_hg* value

Step 2

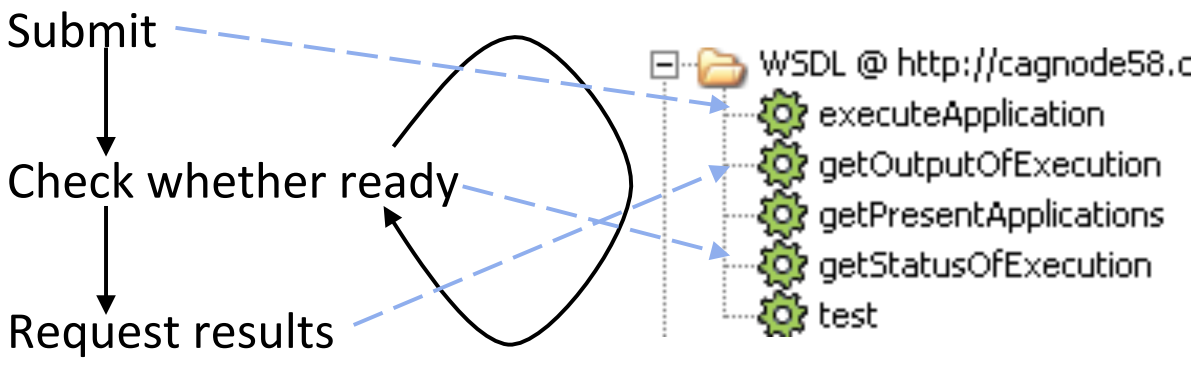


Exercise 6: Call a Asynchronous Service

- Step 2: Propagate event

- The HPS (Processing Service) runs a number of propagation models
- Write a new workflow which informs you about the applications present on the HPS service
  - Find the *helio-hps-server* in the *Services Panel*
  - Which is the appropriate function?
- For more clarity look at the result as an *XML tree*
- Find out:
  - Which propagation is appropriate?
  - What are the required parameters?

Asynchronous SOAP call - HPS



Asynchronous = the execution of a service call may not start immediately as requested; it does not block main flow

Exercise 6a: Submit a Process

- Create a new workflow which performs a stand-alone propagation
- Drag function '*executeApplication*' onto your workflow diagram
- Add XML input splitters to all complex input types (tip you can add XML splitters to XML splitters)
- Make sure you provide the correct number of parameters for the selected propagation method
- Decide which parameter should be provided by the user

Information to the HPS service

- Use for resource intensive calculations
- Works in two modes
  - On the Grid
    - For calculations which require many processors and run for a longer period of time (feature extraction)
    - Requires authentication for execution of code
    - requires a certificate for uploading code
  - On a separate machine
    - Less heavy weight calculations (finish within 30 minutes)
    - Predefined jobs (such as propagation) does not require authentication
    - Requires authentication for execution of code
    - requires a certificate for uploading code

Information to the HPS service

- Input requirements:
  - fastExecution = true
  - numOfParallelJobs = 1
  - Application id = copy from output of getPresentApplications
  - For each parameter: name as defined in getPresentApplications + value
- Output:
  - Job Id - required in consequent steps

Exercise 6b: Check the Status of a Process

- Create a new workflow
- Drag function ‘*getStatusOfExecution*’ into your workflow diagram
- Complete workflow so that it accepts the execution id as an input and produces the status as an output
- Execute workflow with the ID of the previous workflow

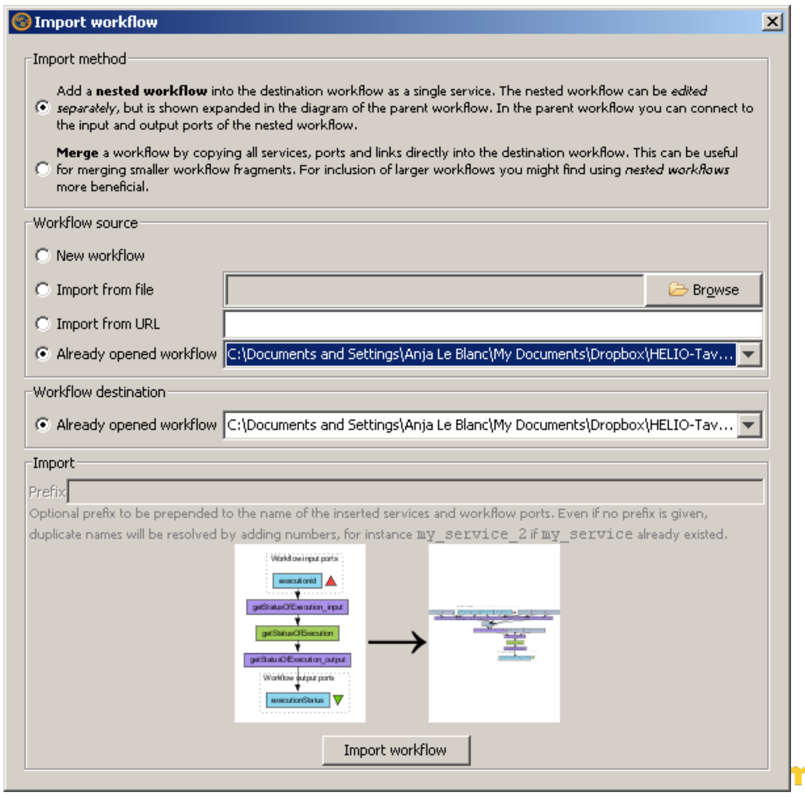
Web Service reliability - Retrys

- Most web service calls will succeed (assuming the service is running), but a small percentage of calls can fail for a variety or reasons
- To avoid that the whole workflow fails because of a single failed web service call we can configure a service to retry failed executions
- In the workflow, click on the ‘*getStatusOfExecution*’ service and go to the ‘*Details*’ tab
- In the ‘*Advanced*’ section, select Retry - ‘*Configure*’ and change the number of retries to 3 in the pop-up window
- Save workflow as ‘Step2b’

Exercise 6b: Check the Status of a Process

- Next we want to combine the workflow created in Exercise 6a with the one of 6b
- Change the workflow to the one created in Exercise 6a
- Insert the workflow created in 6b (Step2b) as a nested workflow
- Connect the executionId with the input port of the nested workflow
- Create a new output port for the status

Nested workflow



Import workflow

Import method

[✓ Add a **nested workflow** into the destination workflow as a single service. The nested workflow can be *edited separately*, but is shown expanded in the diagram of the parent workflow. In the parent workflow you can connect to the input and output ports of the nested workflow.]

Merge a workflow by copying all services, ports and links directly into the destination workflow. This can be useful for merging smaller workflow fragments. For inclusion of larger workflows you might find using *nested workflows* more beneficial.

Workflow source

New workflow

Import from file — Browse

Import from URL

[✓ Already opened workflow] C:\Documents and Settings\Anja Le Blanc\My Documents\Dropbox\HELIO-Tav...

Workflow destination

[✓ Already opened workflow] C:\Documents and Settings\Anja Le Blanc\My Documents\Dropbox\HELIO-Tav...

Import

Prefix

Optional prefix to be prepended to the name of the inserted services and workflow ports. Even if no prefix is given, duplicate names will be resolved by adding numbers, for instance `my_service_2` if `my_service` already existed.

Import workflow

Looping

The default behaviour in a workflow is to call each service only once for each item of data – so what if your job has not finished when ‘*getStatusOfExecution*’ workflow asks?

- Run the workflow using the example input value provided in the workflow
- Almost every time, the workflow will finish with the status ‘*Running*’

Looping

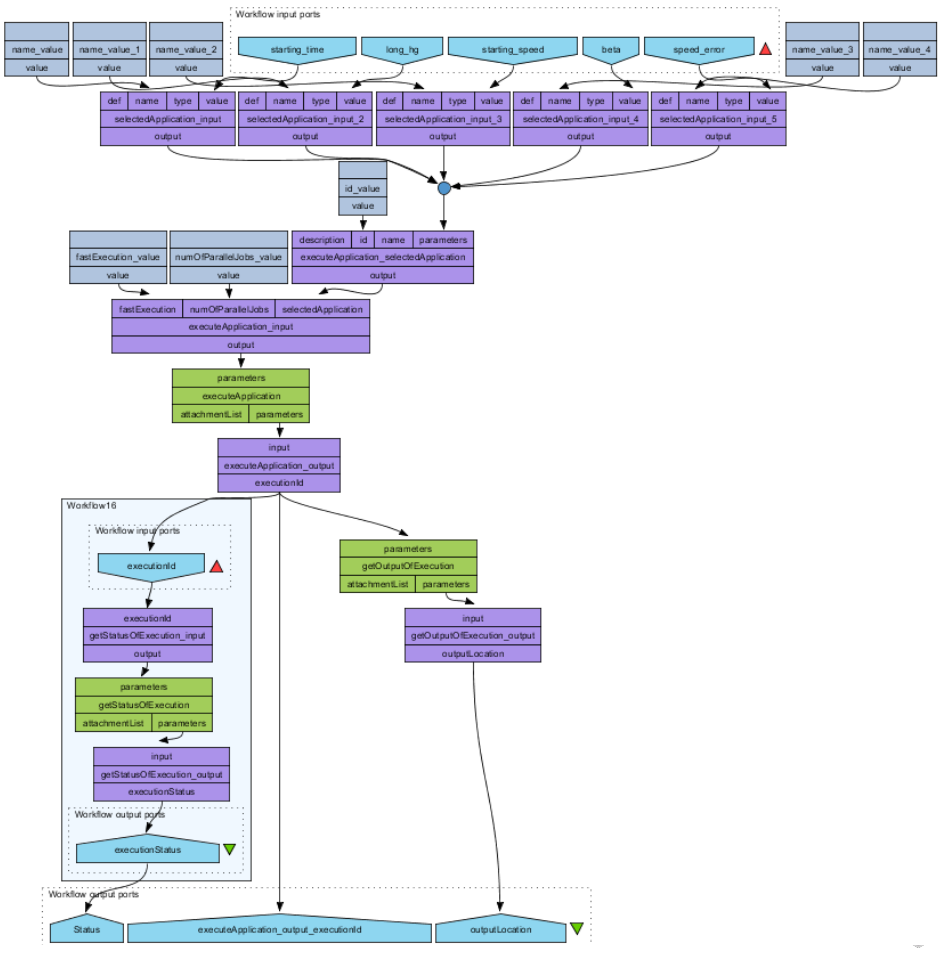
This is where looping is useful. Taverna can keep running the '*getStatusOfExecution*' service ***until*** it reports that the job is done.

- Select the nested workflow and click on the '*Details*' tab in the workflow explorer
- Select '*Advanced*' and click on '*Add looping*'
- *executionStatus* can have three different values
  - *Running* (continue looping)
  - *Completed* (stop looping)
  - *Error* (stop looping)
- Use the drop-down boxes in the looping window to set '*executionStatus* ' to a suitable stop condition
- Add a delay of one second between calls

More help: http://www.mygrid.org.uk/dev/wiki/display/taverna/Loops

Exercise 6c: Get Output of the Execution

- Continue with the same workflow
- Add '*getOutputOfExecution*' to your workflow
- Add Input and Output splitters to the function call
- '*getOutputOfExecution*' requires the executionId as an Input – connect it to the output of '*executeApplication*'
- Create a new Output port for '*getOutputOfExecution*' service call
- Run workflow – what happens?



Workflow input ports
name_value
value
name_value_1
value
name_value_2
value
starting_time
long_hg
starting_speed
beta
speed_error
name_value_3
value
name_value_4
value
def | name | type | value
selectedApplication_input
output
def | name | type | value
selectedApplication_input_2
output
def | name | type | value
selectedApplication_input_3
output
def | name | type | value
selectedApplication_input_4
output
def | name | type | value
selectedApplication_input_5
output
id_value
value
description | id | name | parameters
executeApplication_selectedApplication
output
fastExecution_value
value
numOfParallelJobs_value
value
fastExecution | numOfParallelJobs | selectedApplication
executeApplication_input
output
parameters
executeApplication
attachmentList | parameters
input
executeApplication_output
executionId
Workflow16
Workflow input ports
executionId
executionId
getStatusOfExecution_input
output
parameters
getStatusOfExecution
attachmentList | parameters
input
getStatusOfExecution_output
executionStatus
Workflow output ports
executionStatus
parameters
getOutputOfExecution
attachmentList | parameters
input
getOutputOfExecution_output
outputLocation
Workflow output ports
Status
executeApplication_output_executionId
outputLocation

Run after

Execution of ‘*getOutputOfExecution*’ does not wait until the ‘*getStatusOfExecution*’ has announced that processing has finished

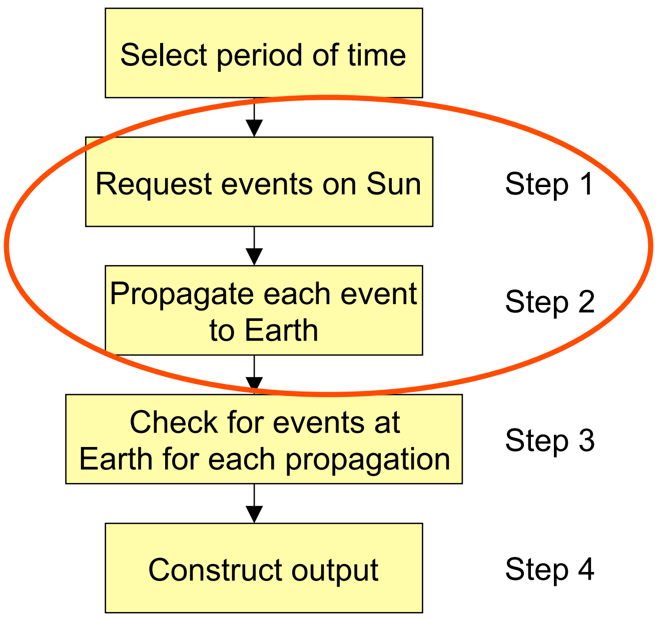
- You can adjust the running order to make ‘*getOutputOfExecution*’ wait until a different part of the workflow has finished
- Right click on ‘*getOutputOfExecution*’ and select ‘*Run after*’ and choose the nested Workflow as process to wait for
- Run workflow again

More help: http://www.mygrid.org.uk/dev/wiki/display/taverna/Control+links

Modify output

- Propagation returns link to web page, but we require a VOTable with the results.
- VOTable is located in the same directory, called sep_pm.votoable (for solar energetic particle propagation)
- Add the service ‘*Concatenate two strings*’ from the ‘*Local Services*’ -> ‘*text*’ service menu
- Connect ‘*outputLocation*’ to ‘*string1*’ and create a string constant with the value ‘/sep_pm.votable’ and connect this to ‘*string2*’
- Use ‘*Get Web Page from URL*’ (‘*Local Services*’ -> ‘*net*’) to download the VOTable
- Create a new Output port for the result
- Run the workflow

Combine Step 1 and Step 2



Exercise 7: Combine Step1 and Step2

- Open workflow 'Step1'
- Add nested workflow
- Decide which Inputs into the propagation should be made by the workflow user and which should be static inputs
- Create new output ports for the propagation results
- Create the extra input ports (and constants)
- Problem: How to connect the values from the extraction to the propagation

Shims

- In Taverna, any 'helper service' that is only involved in reformatting is called a **shim.**
- We will use shims throughout the workflow where service formats are incompatible, or where we want to convert and view something in a different format.
- Many shims are local scripts that Taverna provides by default.
- Others you need to write yourself in Beanshells

Exercise 7: Build a Shim in a Beanshell

Shim to transform the output of ‘*Extract content of columns from VOTables*’ nested workflow to required input for propagation nested workflow

- Insert a new Beanshell
- Click on the tab ‘*Input ports*’
- Add new Ports for ValueLists, ColumnNames, time_start string, and long_hg string

Depth of Inputs:

  - Depth 0 = single value
  - Depth 1 = list of values
  - Depth 2 = list of lists of values
  - Depth 3 = list of lists of lists of values

- Consider which depth should be associated with each input (e.g. look at the result of a previous execution)

Exercise 7: Build a Shim

- Click on the tab ‘*Output ports*’
- Add output ports for *time_start* and *longitude*
- Consider which depths should be associated with each output (same schema as for inputs)
- Click on the tab ‘*Script*’

The script’s language is Java; the names for variables of inputs and outputs are known; all single values are considered to be Strings (might need translating into different types)

- Write and test the Shim, if you are unfamiliar with Java import workflow ‘*shim_split_list*’ (use option *Merge*) and connect all inputs and outputs (part of file HELIO Taverna Workshop Material)

Iteration

Taverna can automatically iterate over sets of data. I.e. When you provide a list as input to a service which expects only a single value.

When 2 sets of iterated data are combined, however, Taverna needs extra information about how they should be combined. You can have:

- **A cross product** – combining every item from list 1 with every item from list 2 - ***all against all***
- **A dot product** – only combining item 1 from list 1 with item 1 from list 2, and so on – ***line against line***
- **A combination of the two** – as the user defines it

Excurse: Iteration

Find and load the workflow '*Demonstration of configurable iteration*' from myExperiment

- Read the workflow metadata to find out what the workflow does (by looking at the '*Details*')
- Select the '*ColourAnimals*' service and select the '*Details*' in the workflow explorer and 'configure list handling'
- Click on '*dot product*' in the pop-up window. This allows you to switch to cross product

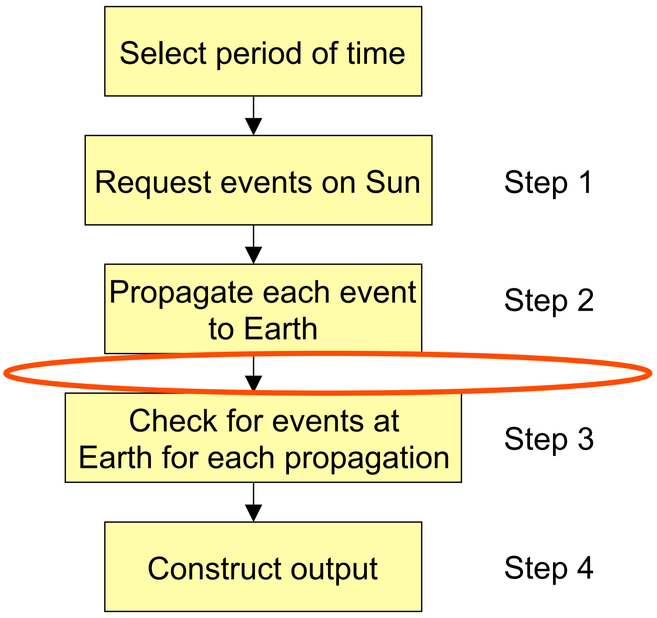
Iteration

- Run the workflow twice – once with ‘*dot product*’ and once with ‘*cross product*’.
- Save the first results so you can compare them – what is the difference? What does it mean to specify dot or cross product?

Exercise 7: Iteration

- Return to your workflow
- Think about what iteration is required to call the propagation nested workflow the correct number of times
- Click on the nested workflow and open the *list handling* in the *Details* tab
- By default everything input is iterated via *Cross product* with each other
- By clicking on ‘*Cross product*’ other options are enabled in the menu
- You can drag and drop input ports to the required iteration strategy
- Run workflow and test that the iteration is working

Prepare for Step 3



Next Steps

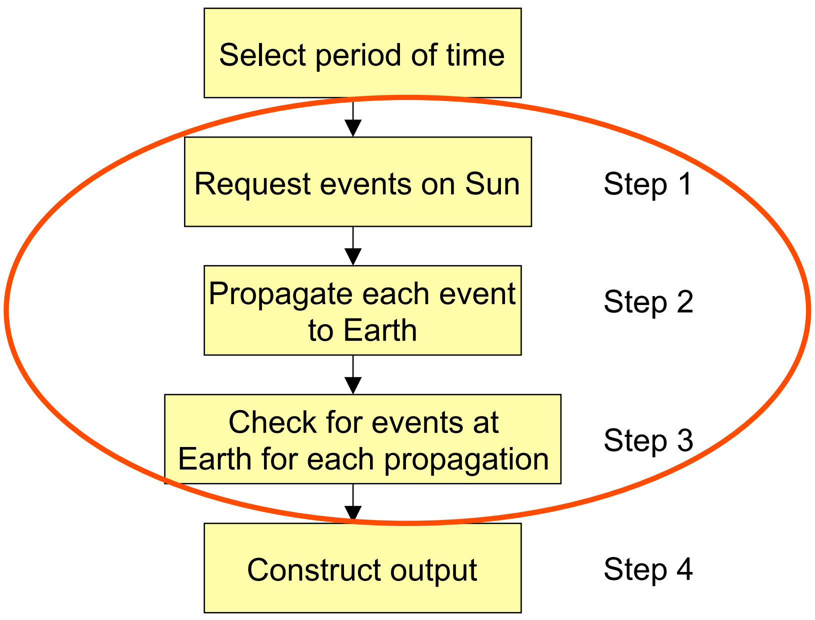
- Propagation produces a VOTable with information to all planets
- We need to filter the results which indicate that the event has hit Earth
- Propagation model give estimated time of arrival (ETA)
- We have to define which **time interval** we want to use to check for events

More Shims needed …

Next Steps

- Writing all shims takes too long
- Short cut: include (merge) workflow '*shim_earth_filter*' (part of file HELIO Taverna Workshop Material)
- Connect the VOTable output of the propagation nested workflow with the port '*propagation*' of the beanshell '*FindEarthHits*'
- Read beanshells and find out what they are doing

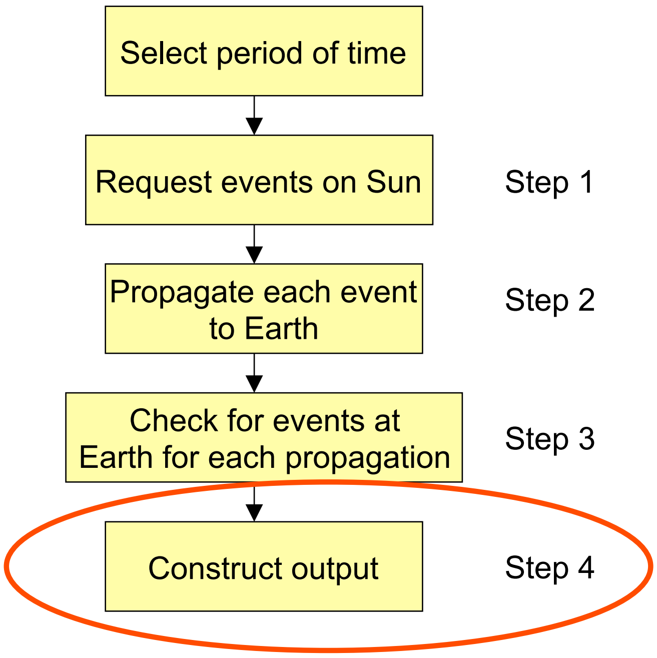
Combine Steps 1+2 and 3



Find events at Earth

- Import the workflow with the REST HEC call '*Step3*' into your workflow
- Connect the inputs and create a new Output port
- Run workflow again

Step 4



Create Output

- Last step in your workflow should be to create a output which combines the data in a way that the workflow user can interpret the results
- When constructing the final output consider which ‘*parameters*’ should be included
- Original question was:

  Which flare events on the Sun can be connected to proton events at Earth

Create Output

- Include (merge) workflow ‘*shim_create_output*’ (part of file HELIO Taverna Workshop Material) and connect all inputs
  - Flare = flare HEC VOTable (step1)
  - Proton = list of proton HEC VOTables (step 3)
  - Hec_flare_label = name of flare list in HEC
  - Hec_proton_lable = name of proton list in HEC
  - Org_id = ids of events which hit Earth (output of Beanshell ‘FindEarthHits’)
  - Sw_speed = solar wind speed (as assumed in propagation)
  - Sw_error = solar wind speed error (as assumed in propagation)
- Create final Output port
- Run workflow

Annotations

- Annotating your workflow is important
  - Helps you when you revisit your workflow some time later
  - Helps others to understand what the workflow is doing
  - Shows examples of what is expected as input
- Supported annotations
  - On the workflow
    - Author
    - Title
    - Description
  - On Input and Output ports
    - Description
    - Example
  - Services/Operators
    - Description

Exersice 8: Annotate your workflow

- Annotate the workflow
  - Click in the workflow diagram at the background
  - Click on the ‘*Details*’ tab in the workflow explorer
  - Enter all details
- Annotate all Input ports
  - Click in the workflow diagram on a Input port
  - Click on the ‘*Details*’ tab in the workflow explorer
  - Enter the details (working example values are very important – to check whether workflow is still working, provides information for runs on Taverna Server, …)

Exersice 8: Annotate your workflow

- Annotate all Output ports
  - Click in the workflow diagram on a Output port
  - Click on the ‘*Details*’ tab in the workflow explorer
  - Enter the details

Example Inputs

- Time_start: 2001-01-01T00:00:00
- Time_end: 2002-01-01T00:00:00
- Solar_wind_speed: 400
- Solar_wind_error: 110
- Percentage of speed of light: 0.9

External Tools

- The purpose is to be able to call tools which are either local on the machine or remote reachable (ssh), but which are not web services
  - Reuse scripts as part of a workflow (i.e. IDL)
  - Write shims in a language you know and include them in your workflow (i.e. python)
  - Call program with special functions (i.e. topcat)
- Likely to make your workflow less portable/ reusable by other people

Exercise 9: Call command line tools

- Try to call the tool you are used to use with VOTables

Example: Topcat

  - Look up the command line arguments
  - Try out how you would call Topcat from the command prompt
  - Drag and Drop the ‘*Tool*’ service from the Service Panel ‘*Service templates*’ onto the workflow diagram
  - Copy past the string you used to start up the tool into the ‘Command’ tab
  - Add input files in the ‘*File inputs*’ tab the name you define as the Taverna port name has to be used in the command line as well to be past to the command

    (e.g. java -jar C:\Users\ag\Dropbox\HELIO\topcat-full.jar VOTable)

Additional Suggestions

If you have reached this point and you have still time to the end of today, you've done better than expected! ☺

Additional Suggestions:

- Try to replace one of the HEC calls with a call to the HEK (Documentation: http://vso.stanford.edu/hekwiki/ApplicationProgrammingInterface)
- Try to build a workflow which calls the context service (Tip: Protocol used: UWS http://www.ivoa.net/Documents/UWS/)