BioVeL – Biodiversity Virtual e-Laboratory

Deliverable Report

Deliverable: 8.5

Title: BioVeL customisations to Taverna Workbench

Work Package: 8

Author: Alan Williams
Work Package Leader: Carole Goble

Date: March 2015

Capacities Programme of Framework 7: EC e-Infrastructure Programme – e-Science Environments - INFRA-2011-1.2.1

Grant Agreement No:	283359
Project Co-ordinator:	Mr Alex Hardisty
Project Homepage:	http://www.biovel.eu
Duration of Project:	36 months
Start Date:	Sept 2011
End Date:	Dec 2014

Status of this deliverable

Version 2, 23/03/2015

This version of the deliverable replaces that delivered at the end of February 2014. At the request of reviewers, it has been revised to include additional details of changes and extensions made to the Taverna suite of tools. The information is correct as of March 2015.

Publishable summary

The BioVeL project has made extensive use of the Taverna suite of tools for creating and executing workflows. To support this, important extensions have been made to the tools themselves. These include extensions to support interaction with users, components, support for WebDAV file access, provenance bundles, and support for OAuth, JSON and WADL. This deliverable describes the changes made to the Taverna Workbench and related products, and its current and future status. The different extensions are described and categorized as either incorporated within Taverna, released installable plugins, or as "work in progress".

Table of Contents

Table of Figures

1 Scope of this deliverable

This deliverable describes many of the customisations of the Taverna Workbench for Biodiversity that have been made by the BioVeL project. It includes descriptions of extensions that have been incorporated into Taverna 2.5, or which have been released as installable plugins, and extensions that are in ongoing development. The deliverable extends and updates details provided in deliverable 6.5. The support for components is described in Deliverable 8.1.

2 Current status of the work

The Taverna suite of tools, including Taverna Workbench, Command Line Tool, Taverna Server and Taverna Player, is in ongoing development. As a result, the content of this deliverable is correct as of the time of writing. For up-to-date information, readers are requested to visit the Taverna website[1].

3 Products

This section describes the state of Taverna Workbench, Command Line Tool and Server at the close of the BioVeL project (December 2014).

3.1 Taverna 2.5 Biodiversity Workbench

In April 2014, Version 2.5 of the Taverna Workbench and of the Taverna Command Line Tool were released. Due to general user request, and also for the needs of the BioVeL project, these are available in domain-specific variants, including ones for astronomy, digital preservation, and biodiversity.

All variants of Version 2.5 of the Taverna Workbench and Command Line Tool include capabilities developed during the BioVeL project, such as the interaction service (see section 4.1), WebDAV services (see section 4.3) and support for workflow components (see Deliverable 6.5). Version 2.5 has also included fixes for many of the issues raised by the BioVeL project partners and incorporates suggestions from them. Thus, not only are users in the domain of biodiversity science beneficiaries of this work, but also users in other domains benefit too.

Version 2.5 includes support for the export of “run bundles” (see section 4.4) that contain information about a workflow and an individual workflow run, including the input data, the results and the intermediate data. This is an important step towards support for “research objects” that package all essential information needed for replicating workflow experiments.

The biodiversity variant of Taverna Workbench is configured so that it is able to search for Web services described in the Biodiversity Catalogue[2]. It has a biodiversity-relevant set of default services, including Aphia webservices[3] from the World Register of Marine Species[4] (WoRMS) . It excludes some types of services previously included by default: BioMoby, BioMart and Soaplab as these are not normally applicable to users in the biodiversity science domain. If required by a user, these service types can be added into the biodiversity workbench easily. The biodiversity variant has a biodiversity-specific starter pack of useful workflows[5].

[1] http://www.taverna.org.uk
[2] http://www.biodiversitycatalogue.org
[3] http://www.marinespecies.org/aphia.php?p=webservice
[4] http://www.marinespecies.org/
[5] http://www.myexperiment.org/packs/610.html

As of end of January 2015, the Biodiversity Workbench has been downloaded more than 750 times and the Command Line Tool has been downloaded more than 600 times.

3.2 Taverna Server

Taverna Server and the modifications made for BioVeL are described in detail in Deliverable 6.4.

Version 2.5.4 of Taverna Server (the current version at the time of writing) includes support for the interaction service, components and WebDAV, as described in Section 4.

3.3 Apache Taverna

The Apache Software Foundation[6] is a non-profit organization, forming a community of open-source software projects with strong emphasis on openness, collaboration and a consensus-based development process. Well-known Apache projects include the Apache HTTP server[7], Tomcat[8], Maven[9], Jena[10], CXF[11] and OpenOffice[12].

In October 2014, Taverna was accepted as an Apache Incubator Project[13]. This means that the Taverna code base, mailing lists and website are being migrated to the Apache infrastructure, and releases of Taverna 3 onwards (see section 3.4) will in future be named Apache Taverna. The University of Manchester has formally granted a software license for the Taverna code to the Apache foundation. The Apache Taverna website is http://taverna.incubator.apache.org/ . The inclusion of Taverna as an Apache incubator project helps to provide a long-term sustainable context for Taverna development and will, it is hoped, increase its open source presence and usage.

At the end of October 2014, a Taverna Open Development workshop was held with representatives from nine institutions to discuss future development plans for Taverna. A presentation about Taverna's move to Apache is available online[14].

3.4 Taverna 3

The Taverna community, including the myGrid team based in University of Manchester, are working on the development of Taverna 3. Taverna 3 is a re-engineering of Taverna to use OSGi[15]. It will include new versions of the Workbench, Command Line Tool and Server. The tools are able to run Taverna 2 workflows as well a new workflow format, SCUFL2[16].

Taverna 3 supports alternative execution environments, so that users can decide to run a workflow in their Workbench’s local engine or on a remote Taverna 3 Server. The workflow can have several possible profiles, indicating different ways of running the workflow, for example utilising different service locations (see also section 6.4).

Taverna 3 incorporates many of the adaptations made to Taverna 2 for BioVeL. The interaction service and components and support for run bundles will be included as part of the standard Taverna 3.

[6] http://www.apache.org/
[7] http://httpd.apache.org/
[8] http://tomcat.apache.org/
[9] http://maven.apache.org/
[10] http://jena.apache.org/
[11] http://cxf.apache.org/
[12] http://openoffice.apache.org/
[13] http://taverna.incubator.apache.org/
[14] http://dev.mygrid.org.uk/wiki/display/developer/Taverna+as+an+Apache+Incubator+project
[15] http://www.osgi.org/
[16] http://taverna.incubator.apache.org/documentation/scufl2/

Initial versions (only suitable for developers) of Taverna 3 have been downloaded 156 times.

4 Incorporated extensions

This section describes the extensions to Taverna that have been incorporated as part of the standard release of Taverna. These are: support within workflows for interaction with users (section 4.1); component workflows (section 4.2); and support for access to WebDAV file systems (section 4.3). It also describes the saving of workflow runs in a provenance bundle (section 4.4).

4.1 Interaction service

Few workflows are "press a button and wait for the results". Early in the project, consultations with scientists highlighted the need for scientists to be able to interact with a workflow while it is running. Such interactions include supplying specific information input at run-time and receiving notification of events occurring during workflow execution.

When running a workflow within the Taverna Workbench version 2.3 and 2.4, it is possible to interact (supply information or receive notifications) using the beanshell capability to leverage Java Swing interactions. However, this mechanism is not feasible when running an interactive workflow within a Taverna Server. In this case attempts to use Swing will either fail or, worse, open dialogs on the machine where the server is running.

Based upon consultations and suggestions with project partners, in particular with Partner 2 Fraunhofer, Partner 10 UNIMAN created an "Interaction Service" for Taverna. This allows users to interact with a running workflow by use of their web browser. The interaction service has been initially released as a plugin for Taverna 2.4 and is now fully integrated into the release package of Taverna 2.5.

The interaction service is used within BioVeL workflows, for example to confirm that an ecological niche model is suitable for projecting into the future, or to clean up data within OpenRefine. Although the interaction service has been designed and developed to support the needs of BioVeL, it is applicable to many other domains as well.

4.1.1 Implementation of the Interaction service

Within the Taverna 2.5 Workbench, there are two kinds of interaction service that a user can include in a workflow: pre-defined "helper" interactions, and a configurable interaction service template for browser-based interactions. These are both shown in Figure 1.

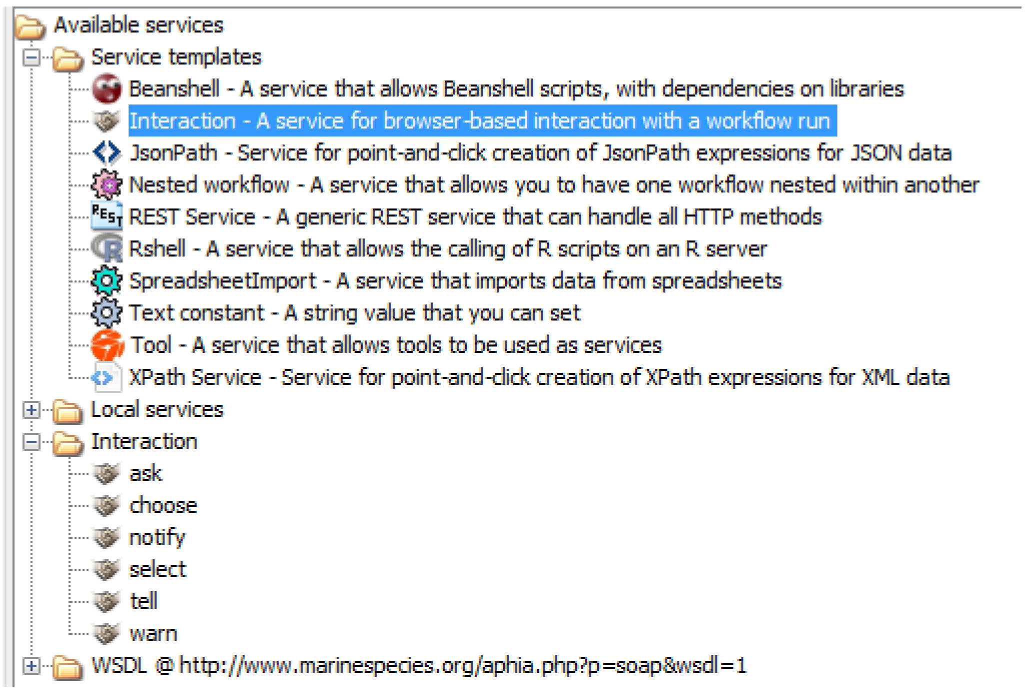

Available services
- Service templates
  - Beanshell - A service that allows Beanshell scripts, with dependencies on libraries
  - Interaction - A service for browser-based interaction with a workflow run
  - JsonPath - Service for point-and-click creation of JsonPath expressions for JSON data
  - Nested 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
- Interaction
  - ask
  - choose
  - notify
  - select
  - tell
  - warn
- WSDL @ http://www.marinespecies.org/aphia.php?p=soap&wsdl=1

Figure 1 Interaction service descriptions

The simple helper interactions are a set of interactions corresponding to the set of UI local services. For example, running the workflow shown in Figure 2 causes the user's browser to show Figure 3.

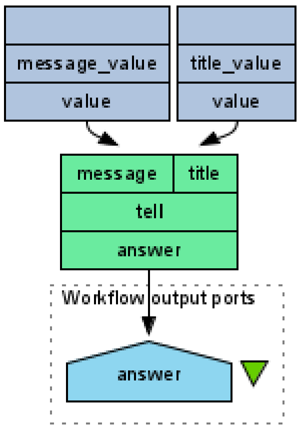
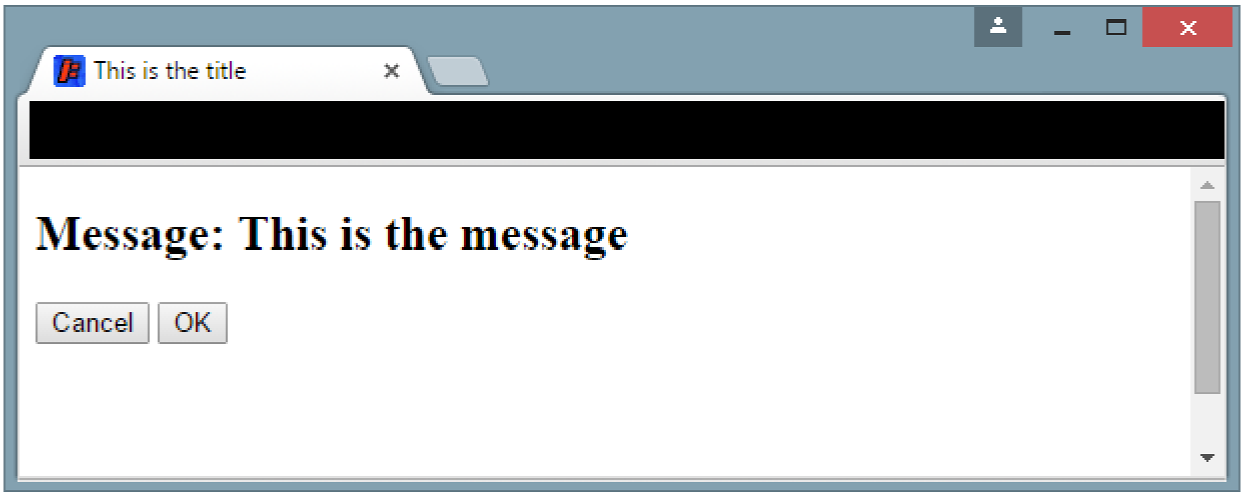


Figure 2 Simple interaction workflow



Message: This is the message

Figure 3 Simple interaction

The workflow waits until the interaction is answered. When the user presses **OK**, the workflow run continues (and in this case finishes). If, **Cancel** is pressed then the interaction fails and the workflow finishes with errors. (One of the helper interactions, **notify**, does not cause the workflow run to pause.)

The interaction service template allows the workflow designer to include interactions that show pages determined by the designer. These pages are commonly more complex than the simple helper interactions just described. Within the BioVeL project, such pages include the showing of OpenRefine interface to users and the presentation of results of ecological niche model testing (AUC curve).

When an interaction service template is added to a workflow, a dialog is presented as shown in Figure 4.

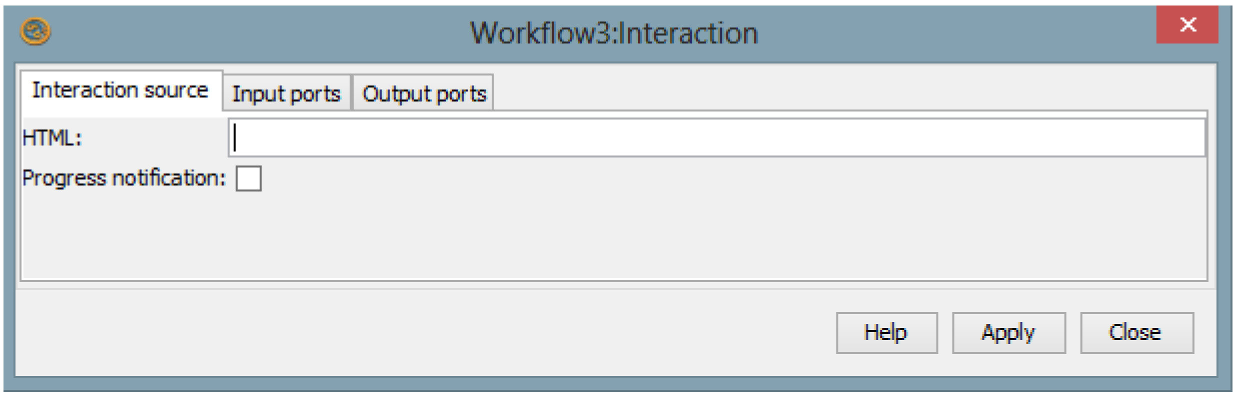


Figure 4 Interaction service template configuration

In the HTML field, the designer enters the URL to the page where the interaction is defined. This page may be, for example, at the same location as a database to which the interaction needs access. In Taverna's implementation, this is called the "presentation page".

The Progress notification tick-box indicates whether the workflow should pause until the interaction is complete. If the interaction just notifies a user of the progress of the workflow run, then the run will not be paused.

The input and output port tabs of the dialog allow the definition of ports of the interaction. During the execution of the workflow, values on the input ports can be used to determine what is shown in the interaction. The interaction is expected to return values for the output ports. (A notification interaction cannot have output ports.)

Within Taverna, the interactions use two important locations:

- A "file server" where it can:
    - Publish container pages for interactions (see below)
    - Place data so that it can be read by interaction pages
    - Hold data returned by interaction pages
- An Atom feed that is used to hold entries that:
    - State that Taverna needs an interaction to be answered or to notify of progress, and
    - State that an interaction has placed its results on the file server.

In the default setup for the Taverna Workbench, both the file server and the feed are hosted within a Jetty[17] web server. The file server is a WebDAV[18] servlet and the feed an Apache Abdera[19] servlet.

The sequence of an interaction is shown in Figure 5.

[17] http://www.eclipse.org/jetty/
[18] http://webdav-servlet.sourceforge.net/
[19] http://abdera.apache.org/

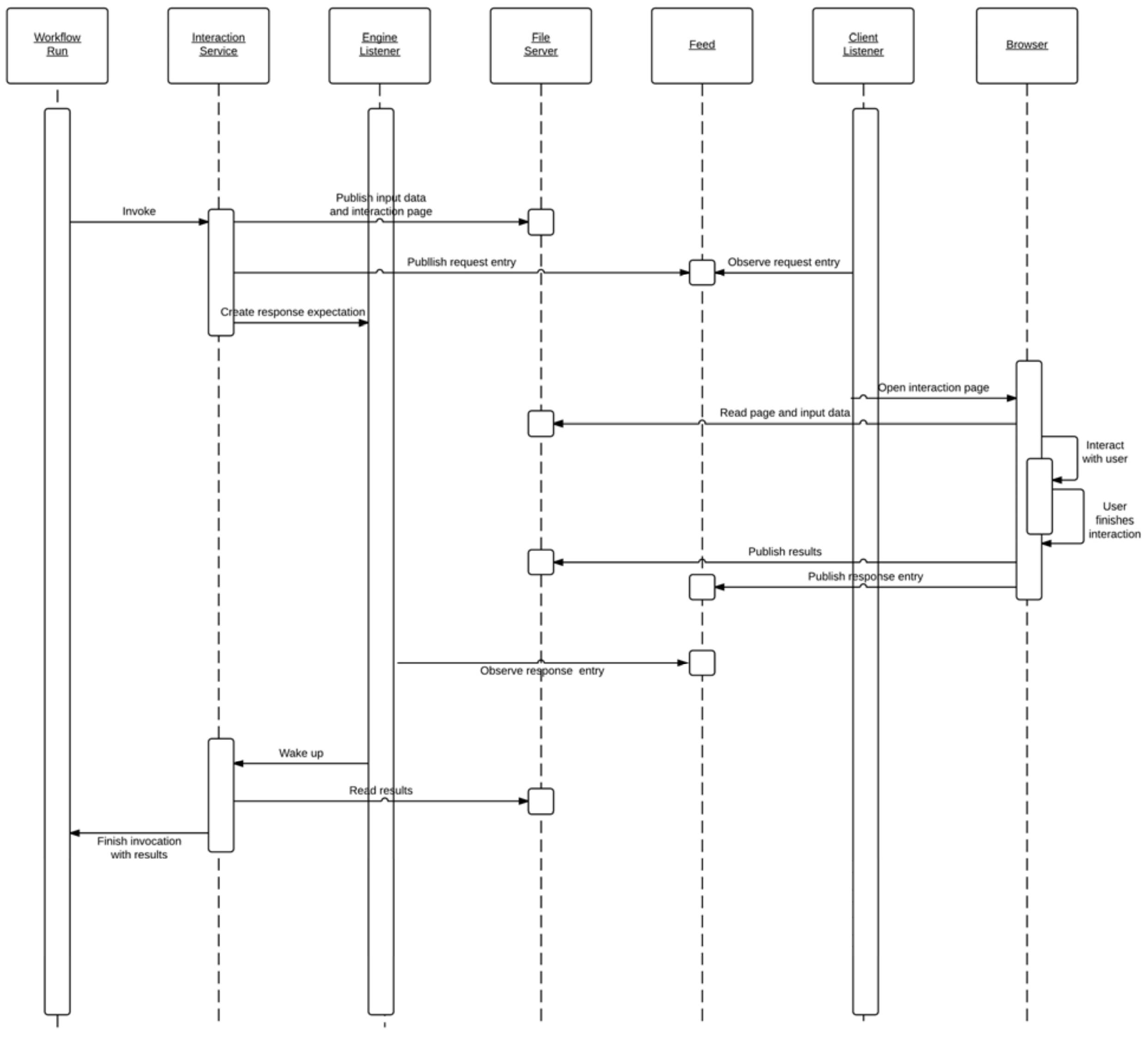


Figure 5 Interaction sequence diagram

When an interaction service is called within a workflow run, the Interaction service invocation:

1. Publishes a JSON document in the file server holding the input values for the interaction;
2. Creates an HTML “interaction page” that is configured to retrieve the input values and allow them to be accessed from the “presentation page”; the interaction page is published on the file server;
3. Publishes an Atom entry onto the Atom feed. The entry contains information such as the location of the interaction page.

In the case of a helper interaction, Taverna also creates the presentation page from an Apache Velocity[20] template. The presentation page is published to the file server. Note that this is not shown in the sequence diagram in Figure 5.

Any subscriber to the Atom feed will receive the interaction entry. In the case of the Taverna Workbench, the Workbench itself has a client listener that listens to the feed and will open the interaction page in the user’s preferred browser.

[20] http://velocity.apache.org/

When the interaction page is opened, Javascript code is used to read the input values from the JSON document and then to display the presentation page in a frame within the interaction page. One of the first actions within the presentation page is normally to request the input values from the interaction page and to configure what is displayed.

When the user has finished interacting, the presentation page returns results to the interaction page. The interaction page then:

1. Writes the results to a JSON document on the file server and
2. Publishes a response entry onto the Atom feed. The entry states what interaction has been responded to and where the results are on the file server.

The strict separation of the interaction and presentation pages is necessary to enable the security of information. Only the interaction pages read and write data to the file server.

The Atom entries do not directly contain the input and output values. This reduces the size of the ATOM feed entries, but it also adds another layer of security as both the ATOM feed and the WebDAV would need to be compromised in order to give access to the data values.

Taverna has an engine listener that listens to the ATOM feed. (Note this is a separate listener to that used by the Workbench client listener to open web pages in the browser.) The invocation of the interaction service is woken up and its results are read from the JSON document on the file. They can then be passed to other services within the workflow.

Taverna normally enforces the deletion of WebDAV data and ATOM entries once the interaction has finished. (The information is still effectively kept as part of the workflow run's provenance.)

The Interaction service allows requests for interaction coming from a workflow run to be mapped to a particular service invocation within the run. This allows the responses from a "golden" run to be used to drive a data sweep that contains multiple subsequent runs with common input parameters. This is essential for the BioVeL Portal to be able to support data sweeps.

4.2 Components

A description of the use of components within Taverna Workbench is provided in deliverable 8.1. Support for components was initially provided by a plugin to Taverna 2.4 but is now fully integrated in Version 2.5 of the Workbench, Command Line Tool and Server.

4.3 WebDAV Services

"Web Distributed Authoring and Versioning (WebDAV) is an extension of the Hypertext Transfer Protocol (HTTP) that facilitates collaboration between users in editing and managing documents and files stored on World Wide Web servers. "[21]

BioVeL users have identified a need for uploading and sharing data. A test case of phylogenetic workflows was proposed by BioVeL partners and the WebDAV capabilities are now used, for example in the "Bayesian Phylogenetic Inference: Select Model – Submission" workflow[22].

The WebDAV Services are included in Taverna 2.5.

[21] http://en.wikipedia.org/wiki/WebDAV
[22] http://www.myexperiment.org/workflows/3408.html

4.3.1 Implementation of WebDAV Services

Although WebDAV is an extension of HTTP, the current Taverna REST service is not capable of handling WebDAV-specific methods such as COPY. For this reason, Taverna has been extended to support a subset of the WebDAV methods: MKCOL (as the more meaningful CreateDirectory), COPY, MOVE, LOCK, UNLOCK. The PROPFIND and PROPPATCH methods are not currently supported.

Taverna uses the Sardine[23] library to define a set of Local services that can be included in workflows to make WebDAV calls. The additional capabilities of the Sardine library, to check the existence of a resource and to list the contents of a directory, are also included as WebDAV services. The WebDAV services appear in the service panel of Taverna Workbench 2.5 and may be included in Taverna workflows.

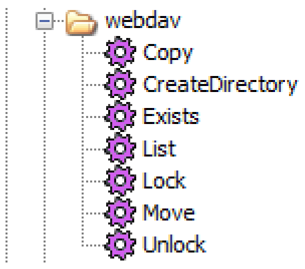


Figure 6 WebDAV service descriptions

Since WebDAV services use HTTP commands, it is common for the WebDAV Services to be used in conjunction with Taverna's REST service. For example, in the phylogenetics component WebDavJST[24], the WebDAV service CreateDirectory is used to create a working space for a run, and then a REST PUT method used to upload a file into the working space.

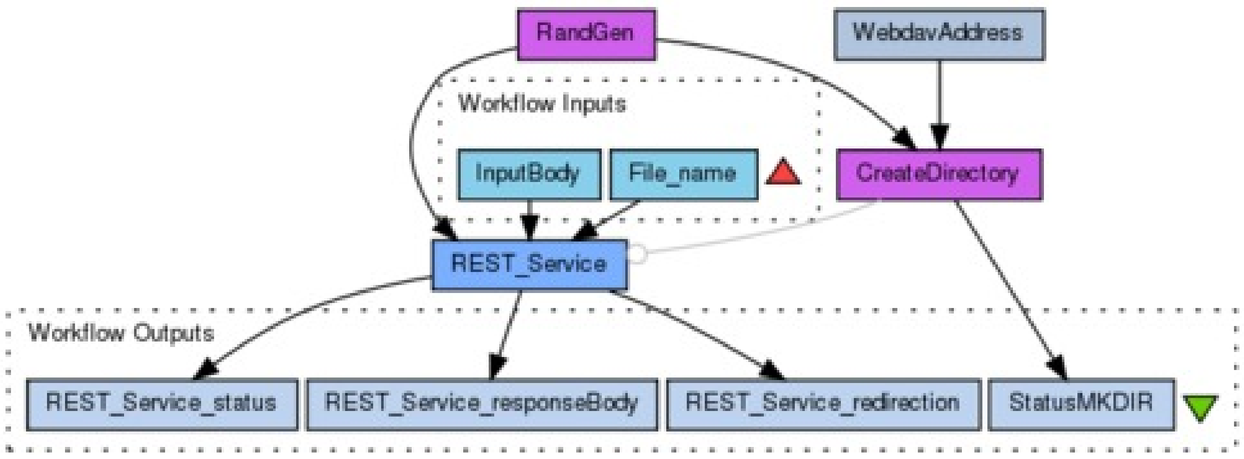


Figure 7 Component using WebDAV and REST

4.3.2 Status of WebDAV Services

A WebDAV Services plugin for Taverna was developed for Taverna 2.4 and its capabilities are included in all variants of Taverna 2.5.

[23] https://github.com/lookfirst/sardine
[24] http://www.myexperiment.org/workflows/3789.html

Since WebDAV is logically an extension of REST, a proposal has been made to extend Taverna's REST service to support WebDAV. It is likely that this will happen as part of a more general re-evaluation of the REST service's capabilities, for example to include OAuth (see 4.4).

4.4 Provenance Bundles

Version 2.5 includes support for the export of "provenance bundles" that contain information about a workflow and an individual workflow run, including the input data, the results and the intermediate data. The provenance bundles are based on the Research Object Bundle specification[25], and comply with the W3C PROV-O[26] provenance specification.

4.4.1 Implementation of Provenance Bundles

Once a run has finished, the provenance can be exported by the "Save provenance bundle" in the "Save all values" dialog.

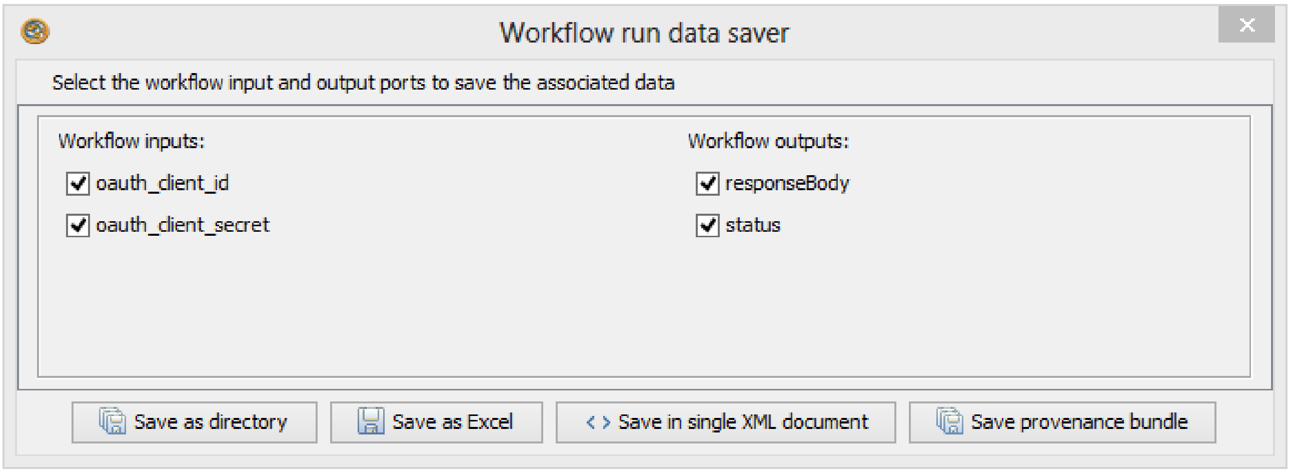


Figure 8 Save provenance bundle dialog

The provenance bundle takes the form of a zip file (Figure 9).

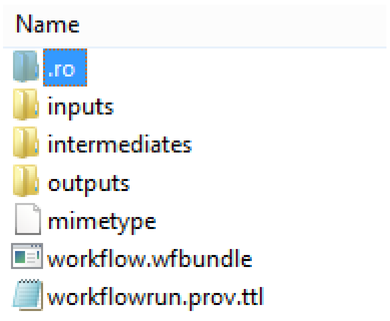


Figure 9 Contents of provenance bundle

The *.ro* folder contains a manifest (Figure 10) of the bundle in JSON format along with files containing Terse RDF Triple Language (Turtle) representations of any annotations on the objects in the workflow that was run.

[25] https://researchobject.github.io/specifications/bundle/
[26] http://www.w3.org/TR/prov-o/

```
{
  "@context" : [ "https://w3id.org/bundle/context" ],
  "id" : "/",
  "manifest" : [ "/.ro/manifest.json" ],
  "createdOn" : "2015-03-16T15:12:02.25Z",
  "createdBy" : [ {
    "uri" : "http://ns.taverna.org.uk/2010/xml/t2flow/raven/org.purl.wf4ever.provtaverna/prov-taverna-
export/2.2.0/org.purl.wf4ever.provtaverna.export.W3ProvenanceExport",
    "name" : "Taverna-PROV plugin"
  } ],
  "aggregates" : [ {
    "file" : "/workflowrun.prov.ttl",
    "folder" : "/",
    "mediatype" : "text/turtle",
    "createdOn" : "2015-03-16T15:11:56.471Z",
    "createdBy" : [ {
      "uri" : "http://ns.taverna.org.uk/2011/software/taverna-biodiversity-2.5.0",
      "name" : "Taverna Workbench Biodiversity 2.5.0"
    }, {
      "uri" : "http://ns.taverna.org.uk/2010/xml/t2flow/raven/org.purl.wf4ever.provtaverna/prov-taverna-
export/2.2.0/org.purl.wf4ever.provtaverna.export.W3ProvenanceExport",
      "name" : "Taverna-PROV plugin"
    } ],
```

Figure 10 Head of provenance bundle manifest

The inputs, intermediates and outputs folders contain the data that was used and produced when the workflow was run. The data can be directly specified, i.e. contained within the bundle, or indirectly i.e. by a URL. The data is not associated with any part of the workflow or any run information.

The *mimetype* file is blank and only exists for compatibility reasons.

The *workflow.wfbundle* folder contains a workflow bundle description of the structure of the workflow that was run. The workflow bundle format is described[27]on the Taverna website. It is a zip file containing an RDF description of the workflow.

workflowrun.prov.ttl, as shown in Figure 11, is a Turtle file that contains information that ties the data to enactments of services within the workflow run and also information about when services were enacted.

[27] http://taverna.incubator.apache.org/documentation/scufl2/taverna_bundle

```
<http://ns.taverna.org.uk/2011/run/51a7818b-2315-4b84-bd60-7baef7a37db9/process/92373553-dbf1-45ea-83d4-
b97615eddcdc/>
        rdf:type                    wfprov:ProcessRun ;
        prov:qualifiedUsage         _:b11 ;
        rdfs:label                  "Processor execution Get_Home_Timeline"@en ;
        wfprov:usedInput            <http://ns.taverna.org.uk/2011/data/51a7818b-2315-4b84-bd60-
7baef7a37db9/ref/44a0c4a0-53ac-48c3-beed-027ed1ef5b24> ;
        prov:qualifiedEnd           _:b3 ;
        prov:qualifiedUsage         _:b12 ;
        wfprov:usedInput            <http://ns.taverna.org.uk/2011/data/51a7818b-2315-4b84-bd60-
7baef7a37db9/ref/9b9a173c-4fe9-4405-b668-dea47811ea5b> ;
        wfprov:usedInput            <http://ns.taverna.org.uk/2011/data/51a7818b-2315-4b84-bd60-
7baef7a37db9/ref/5219626d-af45-4070-84e8-ee93d48f4d09> ;
        prov:wasAssociatedWith      <#taverna-engine> ;
        prov:used                   <http://ns.taverna.org.uk/2011/data/51a7818b-2315-4b84-bd60-
7baef7a37db9/ref/d539fa96-ad0a-43ec-8ddd-f7663f84e2cb> ;
        wfprov:usedInput            <http://ns.taverna.org.uk/2011/data/51a7818b-2315-4b84-bd60-
7baef7a37db9/ref/d539fa96-ad0a-43ec-8ddd-f7663f84e2cb> ;
        prov:qualifiedUsage         _:b13 ;
        prov:used                   <http://ns.taverna.org.uk/2011/data/51a7818b-2315-4b84-bd60-
7baef7a37db9/ref/5219626d-af45-4070-84e8-ee93d48f4d09> ;
        wfprov:describedByProcess   <http://ns.taverna.org.uk/2010/workflowBundle/ddde0008-6cf6-4286-a6a3-
2f364f0d3ce8/workflow/Workflow1/processor/Get_Home_Timeline/> ;
        prov:endedAtTime            "2015-03-16T14:18:16.913Z"^^xsd:dateTime ;
        prov:used                   <http://ns.taverna.org.uk/2011/data/51a7818b-2315-4b84-bd60-
7baef7a37db9/ref/44a0c4a0-53ac-48c3-beed-027ed1ef5b24> ;
        wfprov:wasPartOfWorkflowRun  <http://ns.taverna.org.uk/2011/run/51a7818b-2315-4b84-bd60-7baef7a37db9/>
;
        prov:qualifiedUsage         _:b14 ;
        prov:startedAtTime          "2015-03-16T14:18:16.329Z"^^xsd:dateTime ;
        prov:qualifiedStart         _:b2 ;
        wfprov:wasEnactedBy         <#taverna-engine> ;
        prov:qualifiedAssociation   _:b15 ;
        prov:used                   <http://ns.taverna.org.uk/2011/data/51a7818b-2315-4b84-bd60-
7baef7a37db9/ref/9b9a173c-4fe9-4405-b668-dea47811ea5b> .
```

Figure 11 Section of provenance bundle

4.4.2 Status of Provenance Bundles

The export of provenance bundles is integrated into all variants of Taverna 2.5. The use of workflow and provenance bundles will be continued and extended in Taverna 3.

5 Released plugins

This section describes extensions to Taverna that have reached sufficient maturity to be released as plugins that can be installed. These include support of OAuth (see section 5.1), extraction of data from JSON files (section 5.2), and the reading of sets of REST services from a WADL file (section 5.3).

5.1 OAuth Services

"OAuth[28] is an open standard[29] for authorization. OAuth provides a method for clients to access server resources on behalf of a resource owner (such as a different client or an end-user). It also provides a

[28] http://oauth.net/

[29] https://tools.ietf.org/html/rfc6749

process for end-users to authorize third-party access to their server resources without sharing their credentials (typically, a username and password pair), using user-agent redirections."[30]

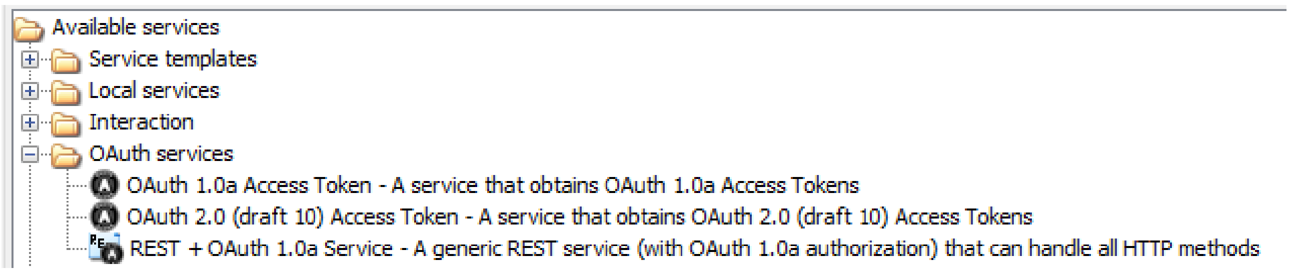
The use of OAuth for BioVel has been investigated with the authentication being done at Marine Ecological GenomiX[31] portal (megx) developed by the Max Planck Institute for Marine Microbiology in Bremen.

The OAuth plugin for Taverna Workbench enables the use of OAuth-protected REST services. It currently supports versions 1.0a[32] and 2.0 (draft 10)[33] of the OAuth protocol.

5.1.1 Implementation of the OAuth plugin

When the OAuth plugin is installed, three new services appear in the Taverna Workbench's service panel (Figure 12):

- Two services for obtaining access tokens using the different OAuth versions;
- A modified REST service that can be configured for OAuth 1.0a is also included in the Workbench.



```
Available services
 ⊞ Service templates
 ⊞ Local services
 ⊞ Interaction
 ⊟ OAuth services
     OAuth 1.0a Access Token - A service that obtains OAuth 1.0a Access Tokens
     OAuth 2.0 (draft 10) Access Token - A service that obtains OAuth 2.0 (draft 10) Access Tokens
     REST + OAuth 1.0a Service - A generic REST service (with OAuth 1.0a authorization) that can handle all HTTP methods
```

Figure 12 OAuth service descriptions

There is currently no modified REST service for OAuth 2.0 and, although its tokens can be used by complex configuration of a REST service, it is not considered further here.

Before you are able to access an OAuth protected resource, you must register an "application" with the authorization provider in order to allow the resource owner to determine whether the application will conform to their acceptable use policies. For example, a Twitter application can be registered at https://dev.twitter.com/apps/. An OAuth application will have a "client id" and a "client secret" key.

30 http://en.wikipedia.org/wiki/OAuth
31 http://mb3is.megx.net/
32 http://oauth.net/core/1.0a/
33 http://tools.ietf.org/html/draft-ietf-oauth-v2-10

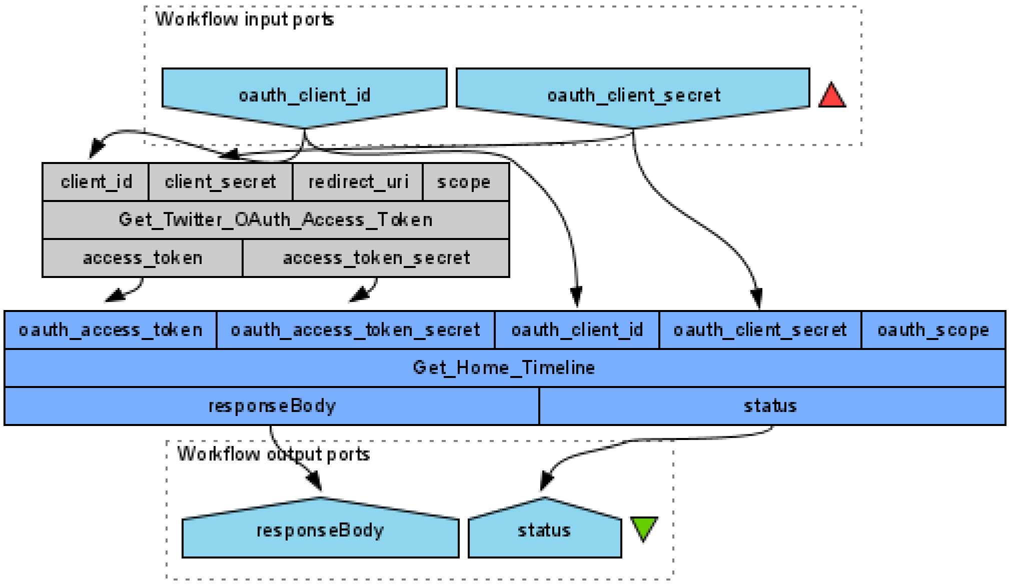
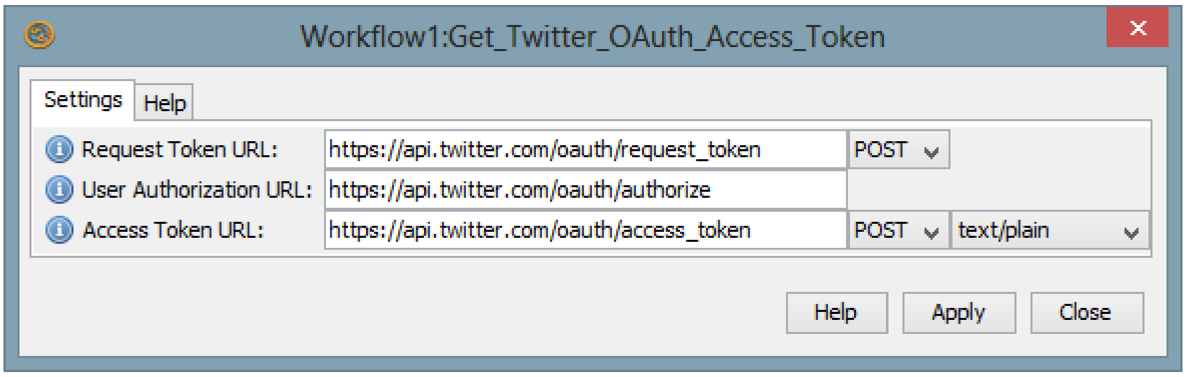


Figure 13 OAuth workflow

In the workflow shown in Figure 13, the first service "Get_Twitter_OAuth_Access_token" uses the id and secret key of the application to obtain a verification code. That service is configured with the locations within the Twitter OAuth server (Figure 14).


Workflow1:Get_Twitter_OAuth_Access_Token

Settings | Help

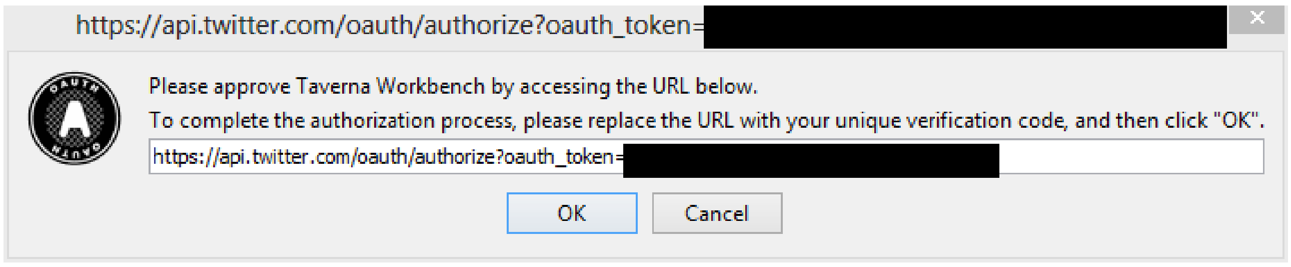
Request Token URL:	https://api.twitter.com/oauth/request_token	POST	
User Authorization URL:	https://api.twitter.com/oauth/authorize		
Access Token URL:	https://api.twitter.com/oauth/access_token	POST	text/plain

Figure 14 Configuration of OAuth service

To obtain the code, a user normally has to authorize the application for their data and this workflow run. The verification code can then be used by Taverna to obtain a use-specific access token and secret key.

When the workflow is run, the authorization happens in three stages:

1. A dialog is shown (Figure 15) asking for the verification key. At the same time, a page is opened in your web browser asking you to authorize the application (Figure 16).

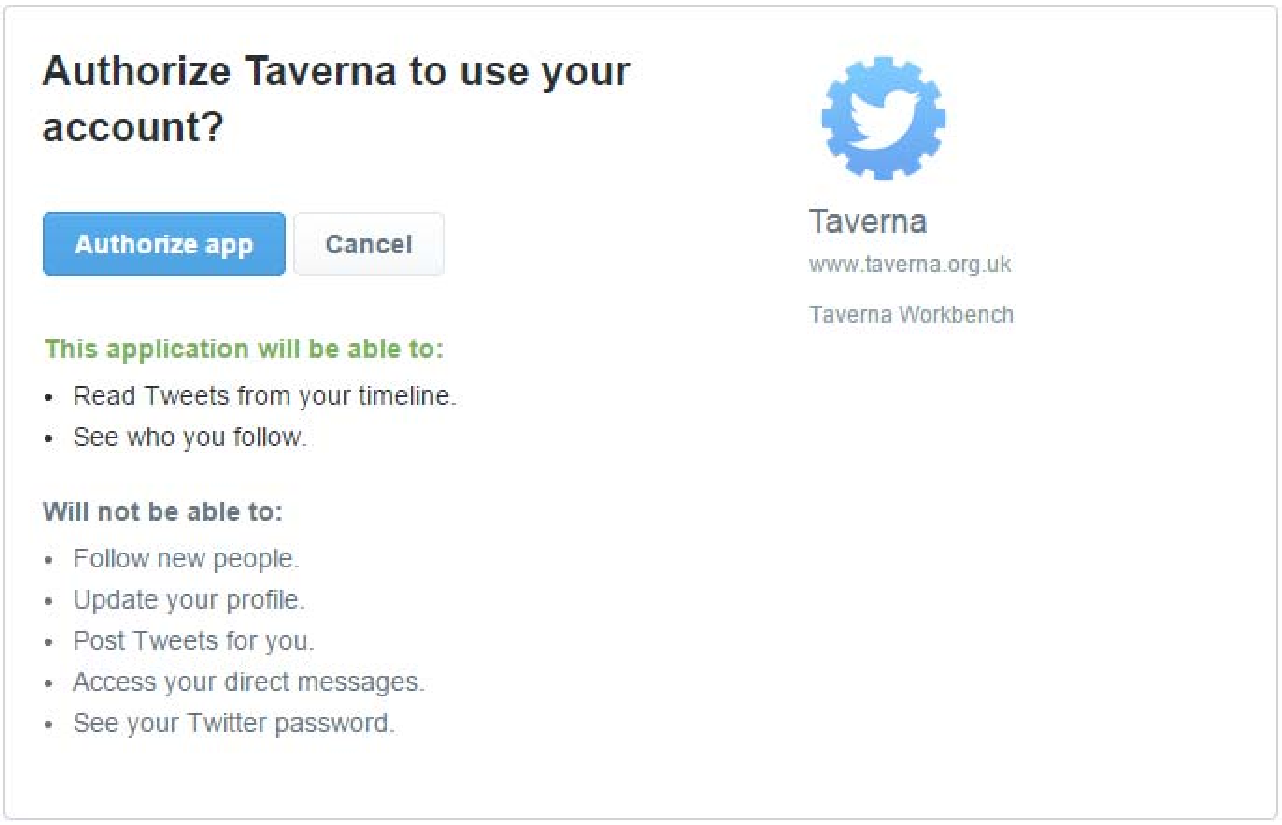


https://api.twitter.com/oauth/authorize?oauth_token=[redacted]

Please approve Taverna Workbench by accessing the URL below.
To complete the authorization process, please replace the URL with your unique verification code, and then click "OK".

https://api.twitter.com/oauth/authorize?oauth_token=[redacted]

Figure 15 OAuth verification key dialog



Authorize Taverna to use your account?

Taverna
www.taverna.org.uk
Taverna Workbench

This application will be able to:

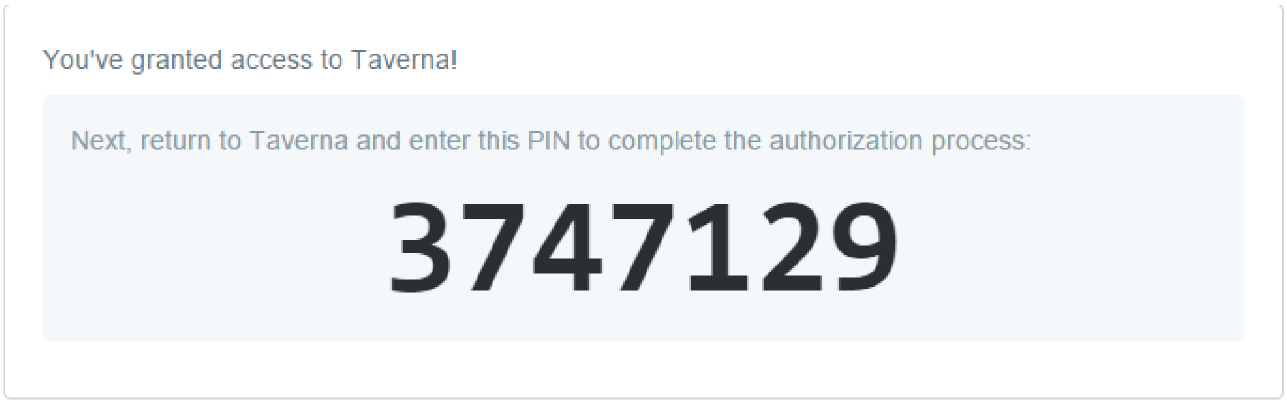
- Read Tweets from your timeline.
- See who you follow.

Will not be able to:

- Follow new people.
- Update your profile.
- Post Tweets for you.
- Access your direct messages.
- See your Twitter password.

Figure 16 OAuth application authorization page

2. Once the user authorizes the application, this may involve them logging in to the authorization server, the server will show a verification code (Figure 17).



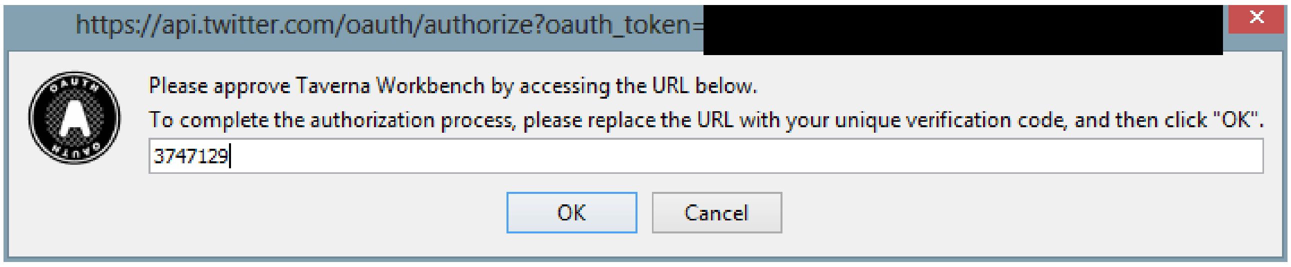
You've granted access to Taverna!

Next, return to Taverna and enter this PIN to complete the authorization process:

3747129

Figure 17 OAuth verification code

3. The verification code can then be entered into the dialog opened in stage 1, and OK'd (Figure 18).



https://api.twitter.com/oauth/authorize?oauth_token=[redacted]

Please approve Taverna Workbench by accessing the URL below.
To complete the authorization process, please replace the URL with your unique verification code, and then click "OK".

3747129

Figure 18 Completed OAuth verification key dialog

The two tokens and two keys are then used as input parameters to the REST service call that is then able to retrieve data specific to the user.

5.1.2 Status of the OAuth plugin

The OAuth plugin has been released as a prototype Taverna plugin for Taverna 2.4 and has been successfully tested for securing access to services authorized by the megx portal. An updated version of the OAuth extension to work with Taverna 2.5 was released in August 2014.

The authorization for the OAuth service in Taverna Workbench currently uses Beanshell-based dialogs. This prevents it working in the context of a Taverna Server. Future work will enhance the OAuth service to enable authorization via a web browser. This is made possible by the subsumption and consequent securing of the interaction infrastructure within the Taverna Server.

The OAuth plugin differs from other security mechanisms in Taverna as the access provision (visiting the authorizing site) is separated from the calling of the secured service. It is likely that authorization by OAuth will be included within a revised REST service as part of future Taverna 3 releases.

5.2 JSON Support

JavaScript Object Notation[34] (JSON) is a text-based standard for data interchange. It is becoming common for web services, in particular REST services, to consume and produce JSON data. JSON is used by general resources such as Google, Facebook and Twitter. It is also increasingly used by biodiversity-related services, including GBIF's ECAT name finder for example[35].

[34] http://json.org/
[35] http://tools.gbif.org/nameparser/api.do

Taverna 2.5 “out of the box” supports the sending and receiving of JSON data. However, it has no explicit facilities for the creation of JSON documents, nor for the the extraction of data from a JSON document. A prototype JSON Path[36] extension was developed for Taverna, similar to its existing XPath capabilities. This extension simplifies the extraction of data from a JSON document. The creation of JSON documents can be achieved using the template support (see section 6.2).

5.2.1 Implementation of JSONPath service

The JSONPath service is available as a “Service template” within Taverna’s Service panel (Figure 19). Being a Service template means that it is highly configurable and not tied to specific locations or data.

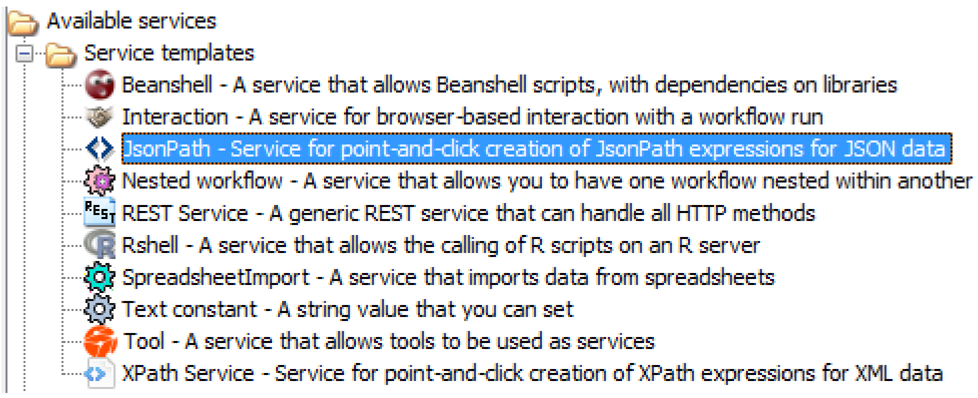
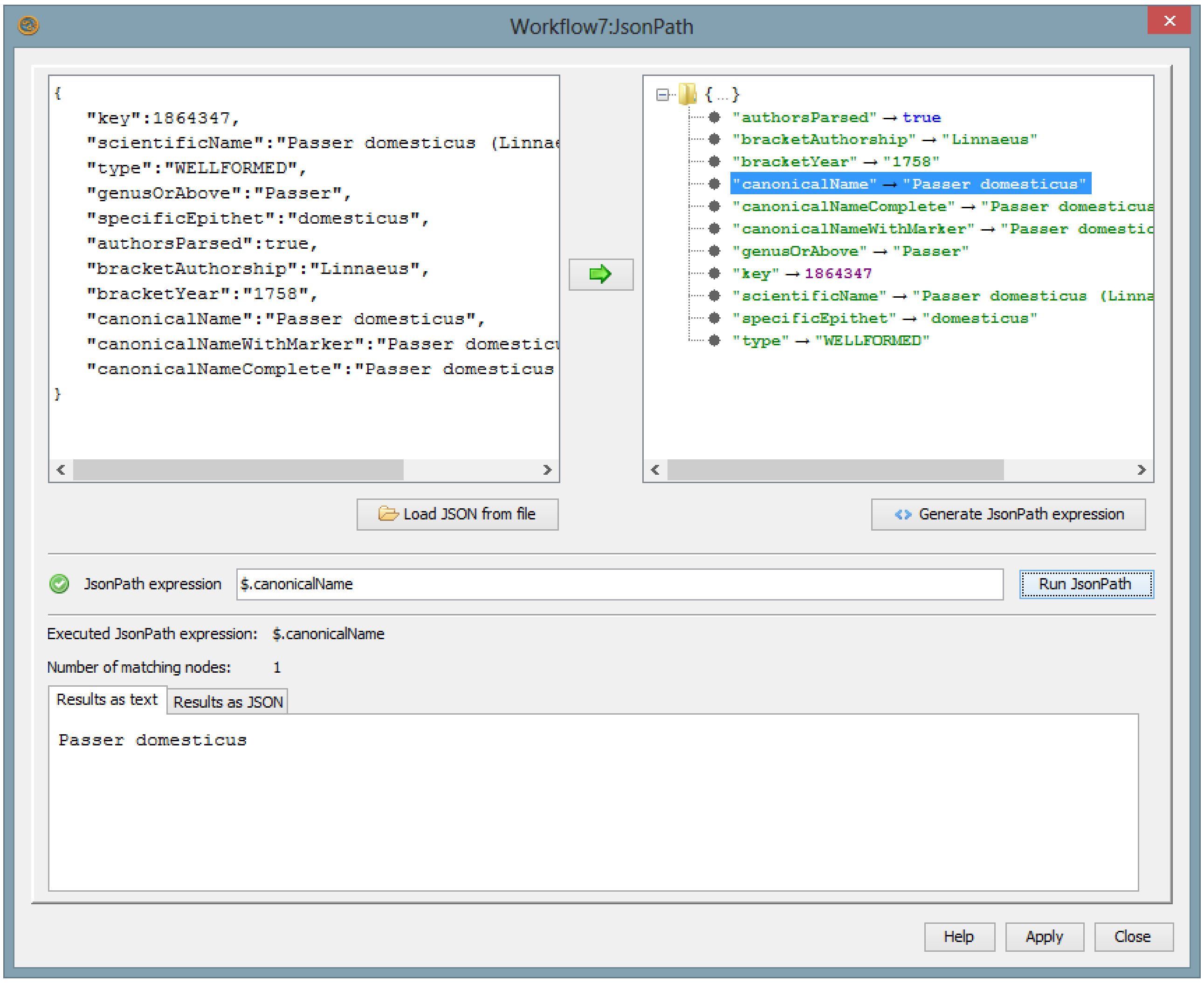

Available services
Service templates
Beanshell - A service that allows Beanshell scripts, with dependencies on libraries
Interaction - A service for browser-based interaction with a workflow run
JsonPath - Service for point-and-click creation of JsonPath expressions for JSON data
Nested workflow - A 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

Figure 19 JSONPath Service template

When a JSONPath Service is included within a workflow, then the user is able to configure the service. The JSONPath configuration works on an example JSON document that is then parsed. The user is able to select a part (or parts) of the document that will be extracted by the service. In Figure 20 the example document is the result of previously retrieving information about the species *Passer domesticus* from GBIF with GBIF ID: 5231190.

[36] http://goessner.net/articles/JsonPath/



```
{
   "key":1864347,
   "scientificName":"Passer domesticus (Linnae
   "type":"WELLFORMED",
   "genusOrAbove":"Passer",
   "specificEpithet":"domesticus",
   "authorsParsed":true,
   "bracketAuthorship":"Linnaeus",
   "bracketYear":"1758",
   "canonicalName":"Passer domesticus",
   "canonicalNameWithMarker":"Passer domesticu
   "canonicalNameComplete":"Passer domesticus
}
```

```
{...}
  "authorsParsed" → true
  "bracketAuthorship" → "Linnaeus"
  "bracketYear" → "1758"
  "canonicalName" → "Passer domesticus"
  "canonicalNameComplete" → "Passer domesticus
  "canonicalNameWithMarker" → "Passer domestic
  "genusOrAbove" → "Passer"
  "key" → 1864347
  "scientificName" → "Passer domesticus (Linna
  "specificEpithet" → "domesticus"
  "type" → "WELLFORMED"
```

JsonPath expression: $.canonicalName

Executed JsonPath expression: $.canonicalName

Number of matching nodes: 1

```
Passer domesticus
```

Figure 20 JSONPath service configuration

Within a workflow, JSONPath services are normally used together with REST services. In the workflow shown in Figure 21, the REST service retrieves information about the species specified by the input id. The JSONPath (configured as in Figure 20) extracts the *canonicalName*.

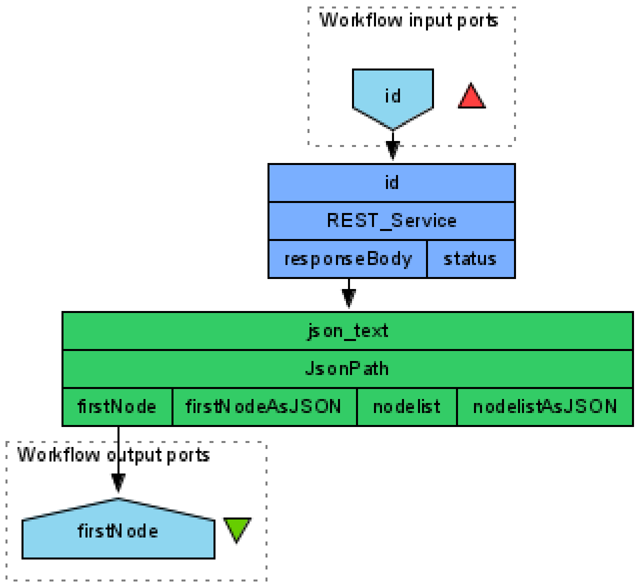


Figure 21 JSONPath workflow

When the workflow is run with a different id, say "5231170", the data for that species is retrieved from GBIF and the JSONPath executed on the GBIF data. As a result, the firstNode output port receives the value "*Emberizoides ypiranganus*".

5.2.2 Status of JSONPath service

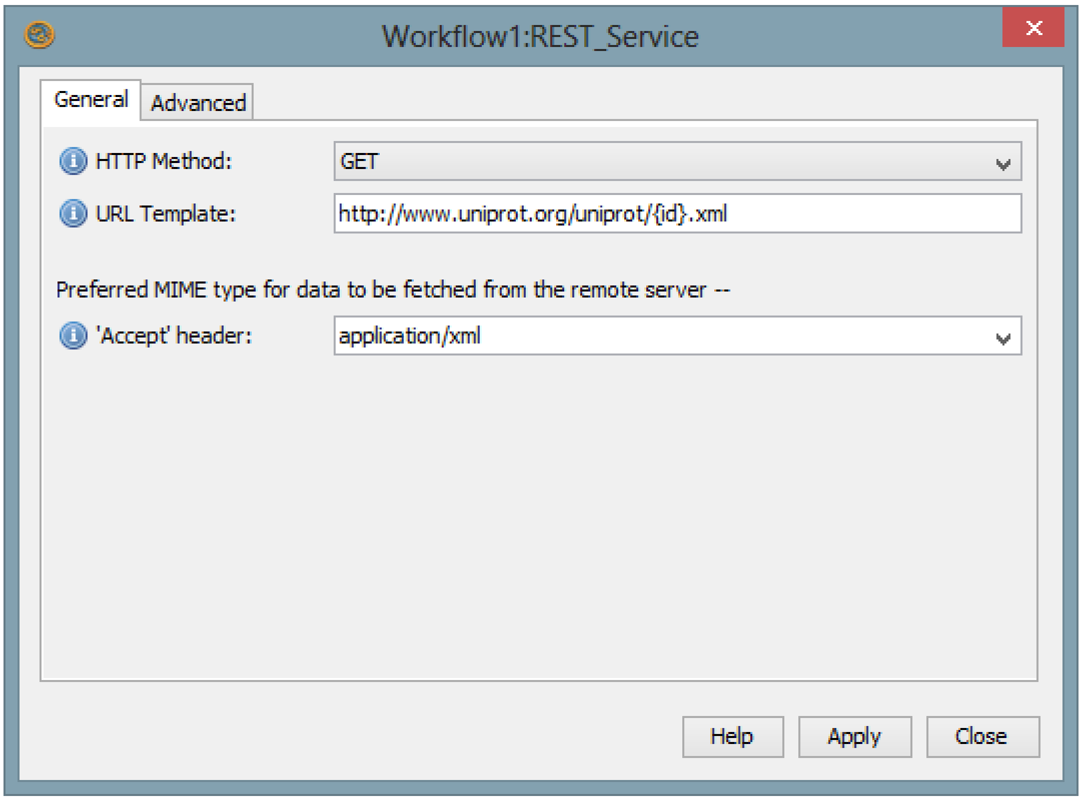
The JSONPath plugin for Taverna was released in July 2014. It is available as a standard prototype plugin for Taverna 2.5. A description of how to install a plugin is given in the Taverna User Manual[37].

5.3 WADL Support

It is becoming more common to use HTTP REST calls to expose tools and data on the internet. There are multiple reasons for this, including the ease with which REST services can be tested in a web browser.

Taverna Workbench supports the design of workflows that call REST services. However, in the standard release package, the REST services must be defined completely each time by configuration of a REST Template (see Figure 22). There is no way to read in the set of REST services that a service provider exposes.

[37] http://dev.mygrid.org.uk/wiki/display/tav250/Installing+plugins



HTTP Method: GET

URL Template: http://www.uniprot.org/uniprot/{id}.xml

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

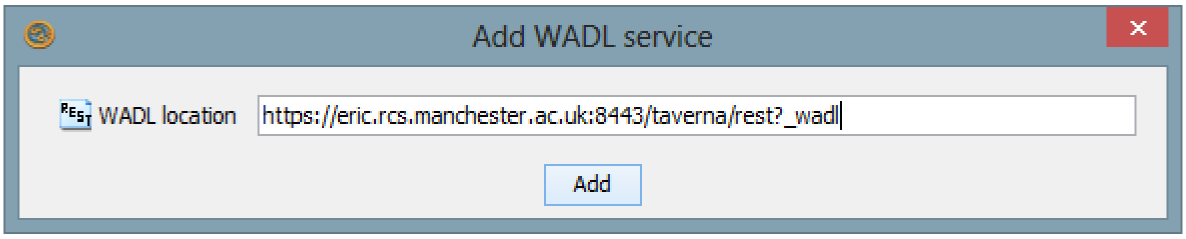
'Accept' header: application/xml

Figure 22 REST Template configuration

REST services can be described in a WADL file[38]. Moreover, WADL can be produced by Apache CXF[39] (one of the commonest mechanisms used to expose Java methods). A WADL file specifies the endpoints, the HTTP operations and the parameters for a REST API. WADL support within Taverna simplifies the creation of REST services. A prototype extension has been developed to read a WADL document and to inform Taverna Workbench of the REST services described within it. The extension was released in June 2014.

5.3.1 Implementation of WADL support

The set of REST services described in a WADL document can be added to Taverna Workbench's service panel (Figure 23).



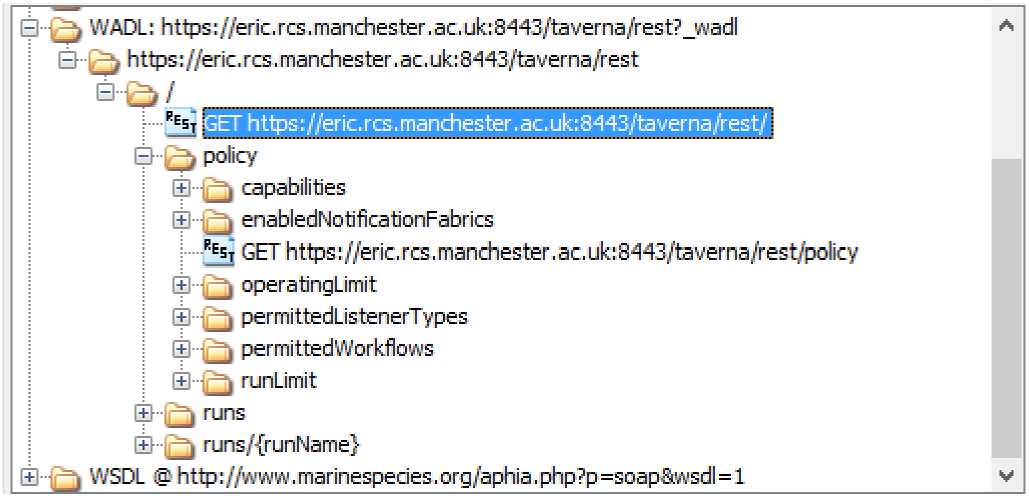
WADL location https://eric.rcs.manchester.ac.uk:8443/taverna/rest?_wadl

Figure 23 Add WADL services dialog

The set of services appear in the service panel, following the structure of the exposed resources.

Internally, Taverna is using Java libraries from GlassFish's wadl2java[40]. wadl2java by default generates Java classes from a WADL file. Taverna overrides the generation methods and instead generates service descriptions (Figure 24).

[38] https://wadl.java.net/
[39] http://cxf.apache.org/
[40] https://wadl.java.net/wadl2java.html



```
WADL: https://eric.rcs.manchester.ac.uk:8443/taverna/rest?_wadl
    https://eric.rcs.manchester.ac.uk:8443/taverna/rest
        /
            GET https://eric.rcs.manchester.ac.uk:8443/taverna/rest/
            policy
                capabilities
                enabledNotificationFabrics
                GET https://eric.rcs.manchester.ac.uk:8443/taverna/rest/policy
                operatingLimit
                permittedListenerTypes
                permittedWorkflows
                runLimit
            runs
            runs/{runName}
WSDL @ http://www.marinespecies.org/aphia.php?p=soap&wsdl=1
```

Figure 24 WADL service descriptions

In Figure 24, the service descriptions correspond to the REST interface to a Taverna Server instance. The service descriptions may be added into a Taverna workflow where they appear as REST services (e.g., Figure 25).

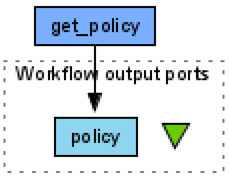


Figure 25 Simple workflow with WADL service

When this workflow is run, it retrieves the policy of the Taverna Server.

5.3.2 Status of WADL support

The WADL plugin was released in June 2014 and has been shown to work with various WADL providers including the Semantic Web service Registry[41] (BioSWR).

The configuration of the REST service is derived from the WADL service description that specifies, for example, the format of the response. The current version of the extension allows all possible changes to be made to the REST service, including changes that are incompatible with the source WADL service description. Future versions of the plugin will constraint the configurability to conform to the WADL description.

6 Work in progress

This section describes several prototype extensions to Taverna that have been developed during the BioVeL project. They are all experimental implementations and have not yet reached sufficient maturity to have been released to general users as plugins. They are likely to be taken up as further developments with Apache Taverna. They include support for tabular data (section 6.1), extended capability to call scripts and

[41] http://inb.bsc.es/BioSWR/

also to create data from templates (section 6.2), and also the calling of OGC Web Processing Services (section 6.3).

6.1 Table Support

The Astronomy variant of Taverna 2.5 includes table reading, manipulation and extraction services. These are based on the STILTS[42] library, however the services are (because of their target domain) tailored to work with tables in the astronomy-specific VO (Virtual Observatory) Table format[43]. Work is ongoing to make these services more generic and to allow their inclusion within other variants of Taverna, such as that for biodiversity.

6.1.1 Status of Table support

The "table activity" is under development with the code shared on Github[44]. It is possible to install the developing plugin into Taverna 2.5 Workbench and create and run workflows. One such workflow is shown in Figure 26

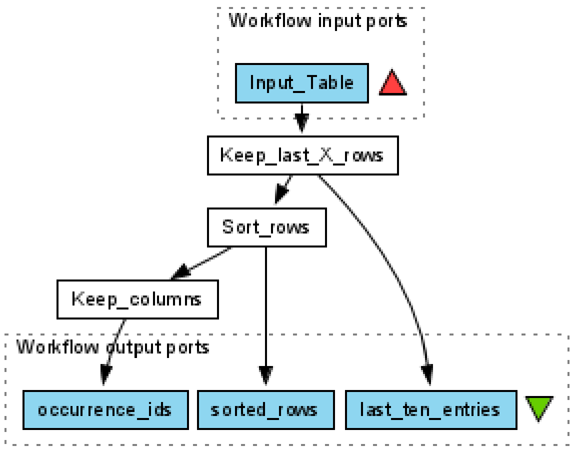


Figure 26 Table services workflow

The workflow takes a csv file, keeps the last ten rows, sorts the rows based upon the contents of the first column and then keeps only the first column. All of the services are configurable, so that the workflow developer is able to specify, for example, the number of rows to keep.

It was originally intended that support for tables based upon the STILTS library would be included within Taverna 3. However, the license of STILTS (GNU General Public License) makes this problematic. Two alternative approaches being considered are:

- To keep support for tables as an external plugin that is not part of Apache Taverna
- To use Apache Taverna compatible libraries such as Apache Spark[45] and its DataFrame capabilities[46].

[42] http://www.star.bris.ac.uk/~mbt/stilts/
[43] http://www.ivoa.net/documents/VOTable/
[44] https://github.com/taverna-extras/table-activity
[45] https://spark.apache.org/
[46] https://spark.apache.org/docs/latest/sql-programming-guide.html#dataframes

6.2 Template and Script service

An extension to Taverna is being developed that allows the execution of scripts written in languages for which there is a Java ScriptEngine. These languages include Jython/Python, Javascript, Scala and JRuby. In addition, by use of the ScriptEngine for the templating language Velocity[47], the extension allows the creation of JSON, XML or textual documents from Velocity templates.

6.2.1 Implementation of the Template and Script service

The code for the Template and Script service is shared on GitHub[48].

When Taverna Workbench is started up, the possible types of script handling are determined from the Java *ScriptEngineManager*. For each script handler, *ScriptEngine*, a service description is added to Taverna's service panel as shown in Figure 27 Script service descriptions.

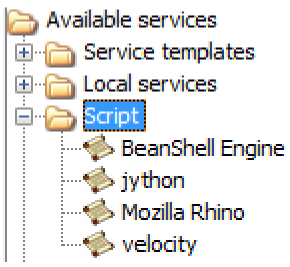
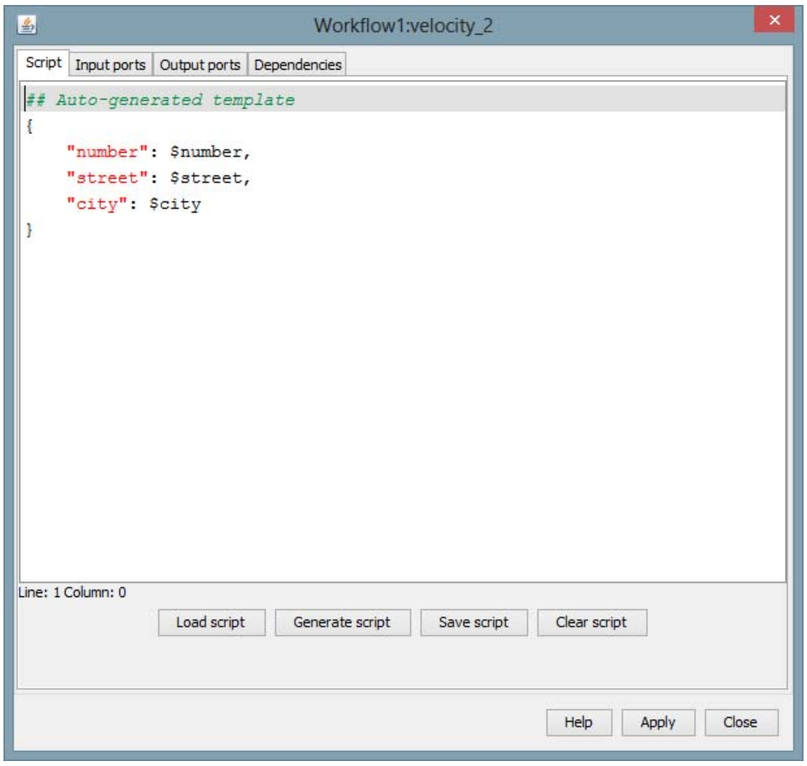


Figure 27 Script service descriptions

When a Script service description is added to a workflow, a dialog is shown. In a similar manner to Taverna's Beanshell service[49], the user is able to specify the input and output ports of the service, any dependencies and also the script that is to be run. The language of the script varies according to the specific service type e.g., for Mozilla Rhino it is JavaScript.

[47] http://velocity.apache.org/
[48] https://github.com/taverna/taverna-script-activity.git and https://github.com/taverna/taverna-script-activity-ui.git
[49] http://dev.mygrid.org.uk/wiki/display/tav250/Beanshell



```
## Auto-generated template
{
    "number": $number,
    "street": $street,
    "city": $city
}
```

Figure 28 JSON template service

In the case of a velocity script description, there is the option to automatically generate a JSON document based on the input ports (as shown in Figure 28 JSON template service).

One difference from a beanshell is that it is possible to state that the value that will arrive at an input during execution is in JSON or XML format and therefore should not be treated as just a piece of text. For example, given the input **"Hello",** this value will either be echoed as **"Hello"** or as "\"**Hello**\""

6.2.2 Status of the Template and Script service

The script service has been tested with several *ScriptEngines* including JavaScript, Velocity, Jython, Scala and Beanshell *ScriptEngine*. It has not yet been released as a prototype plugin, pending further work to improve the user-friendliness of the configuration dialog.

It is intended that, in future versions of Taverna 3, the script service will be combined with Taverna's current Beanshell service. Taverna also has the ability to call R scripts that are run on a local or remote RServe[50]. The script extension is unlikely to be combined with the execution of R scripts because of the lack of a suitable R ScriptEngine. (Rscript[51] cannot be used because of its GPL license.)

6.3 WPS Services

The Data e-Infrastructure Initiative for Fisheries Management and Conservation of Marine Living Resources (iMarine) project[52] has a large number of applications[53] allocated to categories or (following the terminology of the underlying gCube[54] software) ”cubes”. Many of these applications are relevant to biodiversity research; in particular those in BioICube include:

[50] https://rforge.net/Rserve/
[51] http://rforge.net/rscript
[52] http://www.i-marine.eu/
[53] http://uripreview.i-marine.eu/82ed9639-c5c8-4f88-a579-c74614da2784.pdf
[54] http://www.gcube-system.org/

- Discovering and accessing species occurrence and taxonomic data;
- Species occurrence datasets processing;
- Building a rich array of species distribution models;
- Retrieving species distribution maps;
- Comparing two taxonomic checklists in DarwinCore-Archive format; and,
- Taxonomic data matching.

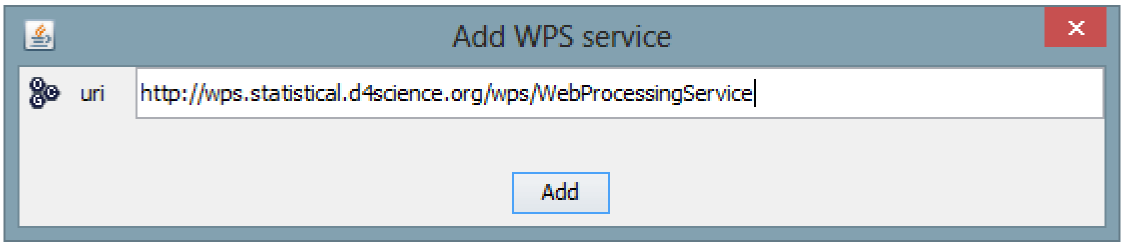
The iMarine project are making these applications available as Open Geospatial Consortium[55] (OGC) Web Processing Services[56] (WPS). WPS defines a standard mechanism for describing geospatial processing services and their inputs and outputs. During April - June 2014 a prototype WPS extension has been developed for Taverna in collaboration with the iMarine project. This extension allows calling of WPS services within a Taverna workflow. This work is useful, not just for calling of WPS services, but as an example of how to call OGC services in general.

Providing access to iMarine services not only enhances the capabilities of the BioVeL infrastructure and the Taverna Workbench for Biodiversity, but diminishes data and service silos within biodiversity research as well.

6.3.1 Implementation of WPS service

The Taverna WPS extension uses the 52north[57] WPS client library[58] to interact with the WPS server, both for determining the set of available Web Processing Services and also for their invocation within a workflow. This 52north library uses the OpenGIS Web Processing Service (WPS) Interface Standard.

The set of Web Processing Services available from a WPS server can be added to Taverna Workbench's service panel (Figure 29).

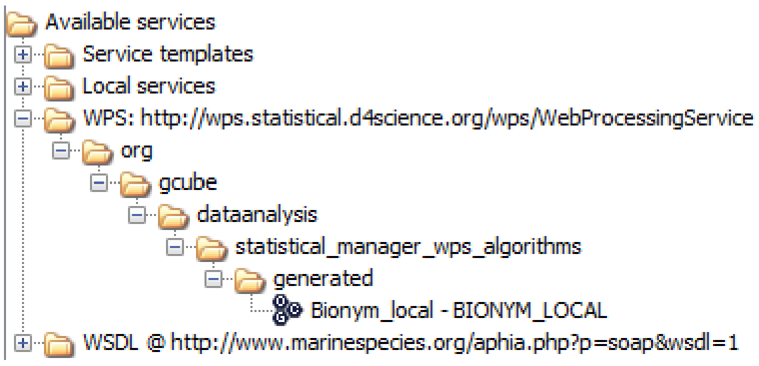


http://wps.statistical.d4science.org/wps/WebProcessingService

Figure 29 Add Web Processing Services dialog

The set of services appears in the service panel. In this example there is only one WPS as shown in Figure 30.

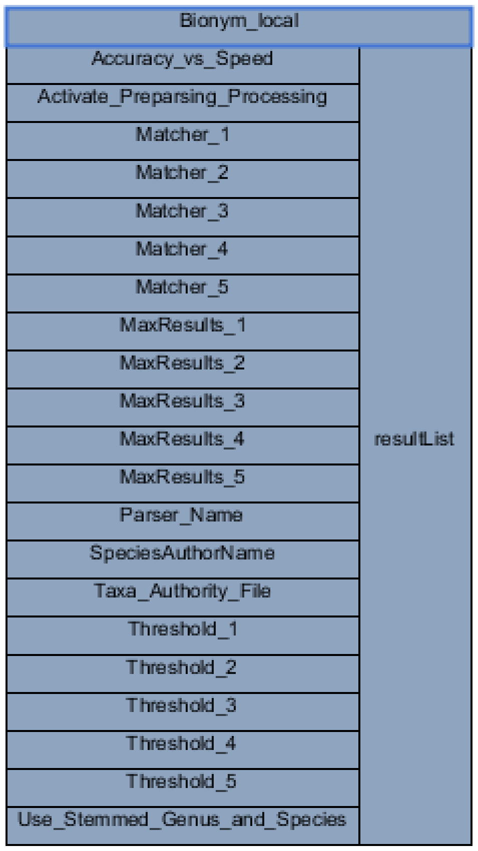
[55] http://www.opengeospatial.org/standards/wps
[56] http://www.opengeospatial.org/standards/wps
[57] http://52north.org/
[58] http://52north.org/downloads/category/15-wps



```
Available services
  Service templates
  Local services
  WPS: http://wps.statistical.d4science.org/wps/WebProcessingService
    org
      gcube
        dataanalysis
          statistical_manager_wps_algorithms
            generated
              Bionym_local - BIONYM_LOCAL
  WSDL @ http://www.marinespecies.org/aphia.php?p=soap&wsdl=1
```

Figure 30 Web Processing Service descriptions

The WPS descriptions can be added into a workflow where their ports are available for connection, as shown in Figure 31.



Bionym_local	
Accuracy_vs_Speed	resultList
Activate_Preparsing_Processing	
Matcher_1	
Matcher_2	
Matcher_3	
Matcher_4	
Matcher_5	
MaxResults_1	
MaxResults_2	
MaxResults_3	
MaxResults_4	
MaxResults_5	
Parser_Name	
SpeciesAuthorName	
Taxa_Authority_File	
Threshold_1	
Threshold_2	
Threshold_3	
Threshold_4	
Threshold_5	
Use_Stemmed_Genus_and_Species	

Figure 31 Web Processing Service in Taverna

The WPS can be connected up to other services and input and output ports, as shown in Figure 32

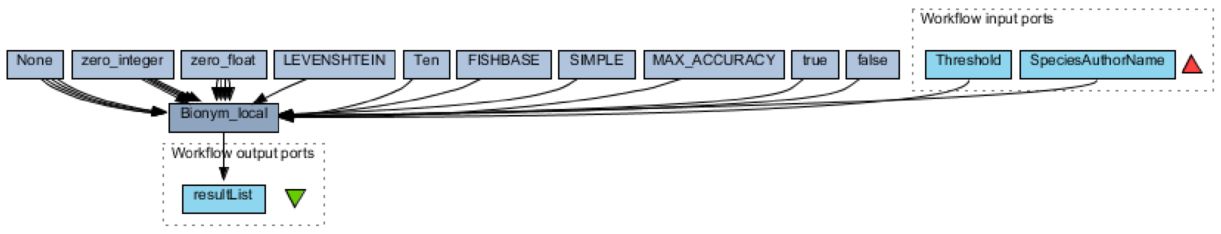


Figure 32 Simple Web Processing Service workflow

When the workflow is run, the input values for the WPS are collated into a document and used to invoke the WPS. When the WPS has finished, the results are extracted from the document returned from the WPS.

6.3.2 Status of WPS service

The WPS capability is still a prototype and has not been released as a plugin. It has been tested with an iMarine WPS server and also with one hosted at Plymouth Marine Labs.

There is continuing interest in the use of Taverna to orchestrate the calling of Web Processing Services and to integrate them with other kinds of web service. It is likely that this work will proceed as part of Taverna 3.

6.4 Service location profiles

The definition of a Taverna workflow includes the network location (endpoint) of services to be run. For example, R services specify the URL of the R server and WSDL services include that of the WSDL endpoint. However, when deploying workflows in different environments, it is useful to be able to switch service locations. For example, a population modelling workflow may use a remote R server (that has special installed packages) when running from a workbench, but when run from the BioVeL portal use an equivalent co-located R server for speed and efficiency reasons.

The BioVeL requirements provide excellent use-cases for the improvements in service location handling being done as part of Taverna 3, specifically in the use of different “location profiles” for a single workflow.

In the meantime, a Taverna 2-based solution has been developed that uses a workflow[59] to change the locations within a family of components. myExperiment access, the component family, and the changed locations are all chosen using the interaction service (see Interaction service, section 4.1).

This workflow takes care of direct and indirect location dependencies:

- A direct location dependency is where a component contains a service whose location has changed.
- An indirect location dependency is where a component uses another component that has direct or indirect location dependencies.

The whole family of components must be updated together or any workflow that uses the component family could end up using a mixture of location settings, with unpredictable consequences.

The current “Component Family Rebaser” workflow[59] works for a family of components. However, it is intended that in the near future it will be generalized to change locations in any workflow.

[59] http://www.myexperiment.org/workflows/4602.html

7 Future work

As described in section 3, Taverna is developed by an Open Source community, now part of the Apache Software Foundation. It is unlikely that there will be a new release of the Taverna 2 suite of tools, although patches may be released if necessary. Taverna 3 is now being developed under the Apache banner and will be released as Apache Taverna.

The incorporated extensions described in this document (section 4) will be included within the initial release of Apache Taverna 3. The capabilities of the released plugins (section 5) will either be included in the initial release of Apache Taverna 3 or in subsequent releases. The work in progress described in section 6 has widespread support in the Apache Taverna community and, although it may require significant further development, is also likely to be included in later releases of Apache Taverna.