20th International Congress on Modelling and Simulation, Adelaide, Australia, 1–6 December 2013
www.mssanz.org.au/modsim2013

Integrating scientific workflows with web services for data validation and provenance reporting

T. Smith[a], N.J. Car[a]

[a] *CSIRO Land & Water, Environmental Information Systems*
Email: timothy.smith@csiro.au

Abstract: Scientific workflows aim to assist scientists in making their complex modelling tasks automated and repeatable. They allow scientists to run and re-run experiments, run modelling scenarios or apply complex procedures to datasets. Increasingly these datasets are provided externally as online resources or web processing services. This paper details methods for incorporating Web Services as outputs and inputs of scientific workflows using the linked data API.

The workflow engine Trident is built on .NET Windows Workflow Foundation which allows easy leveraging of a wide range of libraries and programming tools. Trident is a dedicated scientific workflow engine with comprehensive logging, fine-grained component versioning, provenance tracking, the visual representation of workflow status and it is designed to run on multiple machines.

We have created custom distributed computing components for Trident that allow workflows to remotely source and deliver data. This is done via RESTful Web Services using the Linked Data API as well as updating remote metadata registers of their run instances and products outputs. These products were created for and used by a specific project but have been contributed to a wider Trident development program known as Hydrologist's Workbench to allow them to be used in other projects.

In this paper we give an overview of a workflow as well as details of the distributed computing components created and how they may be repurposed for other workflows.

Keywords: *scientific workflow, Trident, provenance, PROMS, DIDS, web services*

1. INTRODUCTION

Scientists and modellers often apply a series of procedures to one or more datasets in the course of running experiments, performing complex modelling tasks, or creating new data sets. Many tools exist to assist in doing so repeatedly – batch file processing, macros, general workflow engines and, more recently, scientific workflow engines. For simplicity all of these tools will be referred to as workflow automators hereafter.

Workflow automators have certain shared desired characteristics, such as ease of use and performance, but when used for scientific and modelling purposes there are additional characteristics that are desired namely reproducibility and provenance. These characteristics typify *scientific* workflow tools (Barga et al, 2010).

Reproducibility is the ability to exactly repeat a workflow that has been executed previously. Given the same inputs and running conditions a deterministic workflow should yield the same results.

Provenance is "…information about entities, activities, and people involved in producing a piece of data or thing, which can be used to form assessments about its quality, reliability or trustworthiness" (Moreau and Missier, 2013). For scientific projects, provenance inspection can be used to confirm results, perform related experiments and highlight the importance of various datasets and the results of their use, such as the flow-on impact of faulty data.

To take advantage of these capabilities the Geofabric project has implemented workflows in the scientific workflow engine Trident – but there has been increasing need to use multiple heterogeneous workflow automators to do so. Initial attempts to surface provenance externally (Lee and Box, 2012) were dependent on very strong links between tools and servers. We hypothesised that it was possible to achieve the required advantages with a series of simple, flexible and easily used web services.

We have created a series of services designed to allow provenance and other data to be captured and maintained across a series of heterogeneous workflow systems. Multiple workflow automators have been adapted to use these services in order to share provenance information and exchange datasets.

The paper introduces three software services; the Data IDentity Service (DIDS), the PROvenance Management System (PROMS) and the Persistent IDentity Service (PID Service). It will then discuss how two workflow automators were adapted to use them. All systems have been covered however there is an emphasis on DIDS. Further information on PROMS is available from https://wiki.csiro.au/display/proms and further information on the PID Service is available from https://www.seegrid.csiro.au/wiki/Siss/PIDService.

2. BACKGROUND

Automated provenance capture and archiving is important as manual capture and maintenance is often performed badly or not at all. The benefit of good provenance information is uncertain, is typically to future users and the cost is entirely to the current user – so there is little incentive for the current user to capture and maintain provenance information. The difficulties in motivating the acquisition and maintenance of good metadata are well known and have been reported in the scientific modelling context (Hartcher, 2009). Lowering the effort cost to the current user can help promote desirable behaviour.

Scientific workflow engines, such as Trident[1], are designed to automatically capture relevant provenance information on executed workflows[2], however such capture is workflow lifecycle centric and misses information and provenance generated before or after workflow execution. Information not captured includes; information on why the workflow was run, information on the input datasets before the workflow first accessed them (including uniquely identifying the datasets), and information on what happened to the output datasets after it was created (was it used as the input to another workflow, moved, renamed etc).

Missing provenance information means capabilities in reproducibility and provenance inspection – both key advantages of scientific workflow engines – are reduced. Without the ability to identify and access input data sets, guaranteed reproducibility is lost. Without the ability to trace provenance from workflow to workflow many advantages of provenance inspection are lost.

[1] *Project Trident: A Scientific Workflow Workbench* software built by Microsoft Research. See http://tridentworkflow.codeplex.com/.

[2] Within the lifecycle of a workflow Trident in particular has excellent capture of provenance information, storage of provenance (within its registry) and even some ability to surface information such as Data Products through web services. It has issues with registry maintenance, multiple users, links to other workflow tools and data provenance outside the workflow lifecycle.

Other workflow automators typically capture even less provenance information, and seldom have the ability to surface captured provenance information when they do.

Standards such as PROV-DM (Moreau and Missier, 2013) and PROV-O (Lebo et. al., 2013) exist that provide a way to specify provenance information. The first expresses the provenance data model using the Web Ontology Language 2 (OWL2)[3]. PROV-O describes how to specify provenance in RDF[4]. Provenance information expressed in OWL2 or RDF, both of which are Semantic Web tools, can be read by machines and reasoned over using *inference engines*[5] allowing complex questions to be asked of it. In the provenance space, someone may ask questions of workflow automator's provenance data in RDF format like "which processes component version changed between workflow run *m* and workflow run *n*".

3. THE SERVICES

In this section of the paper a trivial workflow example is presented in 3.1, followed by a description of the three services mentioned above; DIDS (3.2), PROMS (3.3) and PID (3.4).

Since the advent of the systems multiple people have learned to use it in under 3 hours. The system has proven robust is under consideration or adoption in three projects, with a potential user base of 50..

3.1. Trivial Workflow System Example

There are three linked workflows.

Workflow 1: takes a file which is a list of numbers, reads it, sums the values and writes the sum as file.

Workflow 2: Takes a file which is a list of numbers, reads it, counts the lines and writes the count as file.

Workflow 3: Reads the sum and count files, divide the sum by the count and writes the result as a file.

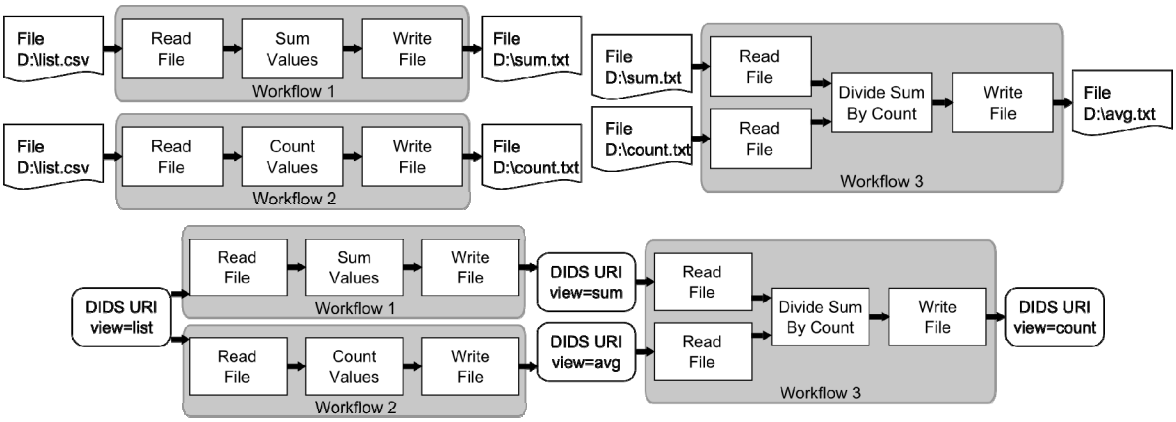

Figure 1: Example workflows. Using files (top) and DIDS (bottom)

3.2. DIDS - Data ID Service

The Data IDentity Service (DIDS) stores data and allocates it a Uniform Resource Identifier (URI) for persistent identification. It also employs Linked Data principles[6] and RESTful web services to provide access to facets of the data.

Consider the example workflow at the top of Figure 1 for the case where after Workflow 1 (Sum) is run the file D:\list.csv is changed before Workflows 2 (Count) and 3 (Average) are run. How can a user know that Workflow 1 and Workflow 2 were using different files?

The key problem here is that the method used to identify the input data is a file path that points to a storage location (D:\list.csv) which may store different data at different points in time. When file paths are used as

[3] See http://www.w3.org/TR/owl2-overview/ for an overview of OWL2.

[4] See http://en.wikipedia.org/wiki/Resource_Description_Framework.

[5] *Inference* engines are computer programs designed to derive answer from structured knowledge. See http://en.wikipedia.org/wiki/Inference_engine for more details.

[6] See http://code.google.com/p/linked-data-api/ for the Linked Data API

identifiers, the many-to-many ambiguous relationship between identity and data can lead to uncertainty. This makes provenance and reproducibility difficult as the identifier cannot *uniquely* identify data. Using file path as a data identifier is also problematic if the data is to be stored on multiple machines and potentially multiple computing networks. Relative file locations are execution context sensitive and absolute but local file locations are duplicated across machines. Likewise network file locations are subject to arbitrary renaming.

For these reasons, DIDS relies on the URI system of universally, uniquely, resolvable (resource) data identification. While there are several universal, unique resource identification mechanisms that could have been used such as DOI[7], an ISBN[8] variant, pURL[9] and URN[10] URIs have been chosen due to reasoning by the World Wide Web Consortium as quoted in Cox (2011) pp.10. The main principles are that URIs are well understood, widely implemented and have many clients available that can *dereference* (resolve and download) the resources they identify.

DIDS is a service that provides two capabilities; storing data, and providing a URI that uniquely identifies data. For each DIDS URI there is at most only one data item (0..1). For each stored data item there will exist at least one DIDS URI (1..N). See Figure 1.

It is possible a DIDS URI will not have associated data; a DIDS URI may be unassigned, or the data may have been deleted. It is also possible the same data item will be stored within a DIDS instance multiple times, in which case there will be multiple URIs for the same data.

If the example in Section 3.1 stores data in DIDS (instead of files) then when the list data is changed it will be stored with a different identifier as the change cannot be written back to the original DIDS data item (as then the same DIDS URI would point to multiple data items). It is simple to realize that Workflow 1 and Workflow 2 are operating on different data when they have different identifiers for that data.

Linked Data

In the example it is desirable to somehow associate the sum, count and average of the list with the list itself. DIDS makes this possible by partially implementing the LDA-ID (Linked Data API Identifier) specification. A full discussion of LDA-ID or LDA is beyond the scope of this paper but a simple explanation of the capability it provides DIDS is presented below.

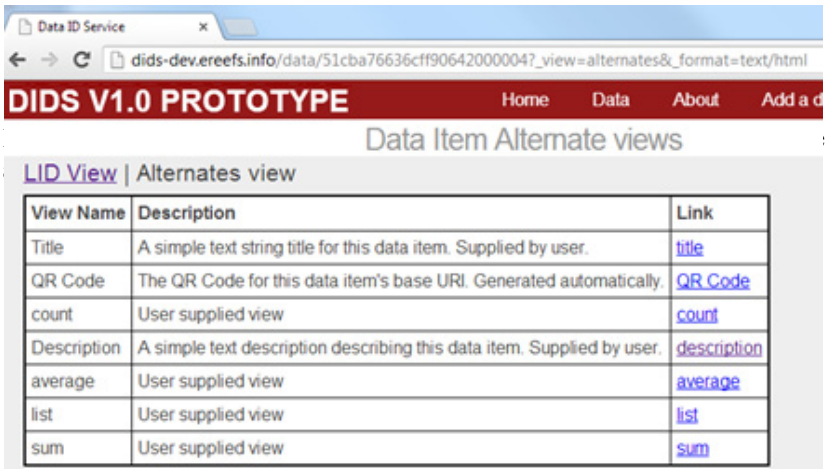

Figure 2: Example DIDS data item

A URI in DIDS represents a conceptual *thing*, ideally a real world existing concept. Let us say that in the example the *thing* is the daily water consumption of a household over a particular time period.

A *view* is a representation of a *thing*. The list (list.csv) is not the daily water consumption; it is a view of that *thing* – a representation. Similarly the average, count and sum are *views* of the same *thing*. See **Figure 2** for an example of what the example would look like in DIDS.

All *things*, in the DIDS data model, have certain default, mandatory or automatically generated *views* as well as any number of optional, user supplied *views*. All *things* have a title *view*. All *things* have a description *view (*which defaults to the title). All *things* have an auto-generated Quick Response Code and an "alternates" *view,* which is a list of all other views. When storing a data item in DIDS, a user must supply a title, may

[7] http://en.wikipedia.org/wiki/Digital_object_identifier

[8] http://en.wikipedia.org/wiki/International_Standard_Book_Number

[9] http://en.wikipedia.org/wiki/Persistent_uniform_resource_locator

[10] http://en.wikipedia.org/wiki/Uniform_resource_name

supply a description, and may supply any number of other views named as they choose. The DIDS data model is given in Figure 3.

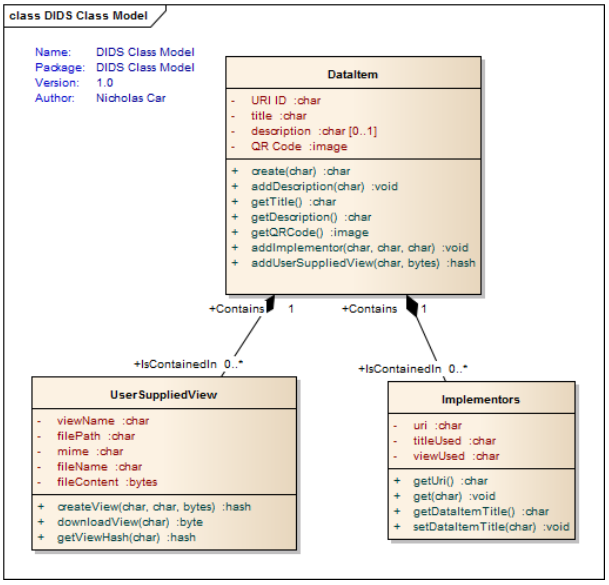

Figure 3: The DIDS data model

In addition to views it is also possible to specify a *format*. A *view* may have multiple formats, but all must be translations of each other. *Formats* should follow the Multipurpose Internet Mail Extension (MIME) standard. – an existing, well-known, well-supported taxonomy of content types. In the example the list might be available as both text/csv and application/vnd.ms-excel.

Software components

DIDS is implemented as a linked set of RESTful web services. REST was used because a key driver was that it be as easy as possible to create software component DIDS interfaces. REST is considered much easier to implement than WSDL style web services, especially when using older or lower level languages. Further only a subset of HTTP commands were used (GET and POST were used, PUT and DELETE are not).

There are two critical software components that must be implemented for a workflow automator to interact with DIDS; **Upload a view to DIDS**, and **Download a view from DIDS**. These software components have been created in multiple languages (C#, Python, VB) and used in Trident and web based workflows.

3.3. PROMS – Provenance Management Service

PROvenance Management System (PROMS): stores provenance information using the PROV-O standard (Lebo et. al., 2013) in a database and makes it available via RESTful web services. It stores various levels of representational data on workflows or other processing systems that have been created ranging from very simplistic (a workflow was run) through to very detailed (almost complete details from Trident's automatic provenance reporting module). Details of exactly how and why data is stored are not covered herein but area available in detail on the PROMS wiki, which contains both documentation and "Starter Kits" available for download for prospective users (The PROMS Team, 2013).

A critical feature of PROMS is that it is designed such that a provenance report can reference a URI and a particular view for data items it contains. This means that software components can be written that can indirectly reference a data item in DIDS data knowing only the PROMS report ID (URI) and the title of the view that was used to upload or download the data.

Using the example from Section 3.1, it is possible for Workflow 3, rather than specifying "D:\sum.txt" and "D:\count.txt" to ask for "Sum" and "Count" from provenance reports on Workflow 1 & 2.

Software components

There are two critical software components that must be implemented for an automated workflow automator to interact with PROMS; **Report provenance to PROMS**, and **Get DIDS URI from PROMS**. These software components have been created in multiple languages (C#, Python) and used in Trident and we-based workflows.

3.4. PID Service – Persistent Identity service.

Persistent IDentity service (PID Service): provides URIs that act as aliases for other URIs. Section 3.4. PID is an existing product that is part of the Spatial Information Services Stack[11] (SISS).

PID Service is a redirect service that maps PID IDs (also URIs) to URIs provided by other services such as PROMS or DIDS. The PID Service is a development on the Apache Web Server's *mod_redirect* module that allows aliases for web pages to be created and to be resolved to the original. Requests to the PID Service

[11] See http://siss.auscope.org

result in response codes and actions outlined in the Hypertext Transfer Protocol standard (Internet Society, 1999). For every PID URI there may be one (and only one) matching DIDS URI or PROMS URI (not both). For every PROMS or DIDS URI, there may be any number of PIDs.

This creates the following advantages:

1. It is possible to use much more compact and aesthetically pleasing URIs. (e.g. http://www.pid.x.au/example_household_uses rather than http://x.y.gov.au/data/12345678901234567890/?_view=list&_format=csv)
2. DIDS and PROMS URIs can be changed (so long as *only* PID URIs are used, even internally). e.g. if the URI http://x.y.gov.au/data/12345678901234567890/?_view=title is moved to domain http://x.z.gov.au/data/12345678901234567890/?_view=title the URI http://www.pid.x.au/example_household_uses can simply be remapped.
3. There are things that may exist in two contexts, leaving them mapped to multiple servers. In the example the household's water use over the year might exist in two contexts; in one context it is one of millions of recorded household water usages, but is then later used as experimental evidence for the reduction in water usage of households across Australia using a new water recycling system. The two contexts can have a different PID URIs, but still both map to the same DIDS Uri.
4. The PID Service, DIDS and PROMS all support URI-based IDs, making them familiar and dereferencable by both humans (with web browsers) and software.

Software components
As the PID Service is a software service, no additional software components are needed to use it.

4. GEOFABRIC WORKFLOW B – USAGE EXAMPLE

The Geofabric project (Bureau of Meteorology, 2013) uses geoprocessing workflows to implement the approach outlined in Section 3 in order to ensure that reproducibility is possible and that provenance is captured. These geoprocessing workflows are implemented using two systems; Trident and a web based workflow automator designed to support manual human workflows.

Workflow B, detailed in Figure 4, is one of the geoprocessing Trident workflows and consists of three geoprocessing activities and four support activities. The support activities are; Download Data (download a view from DIDS), Upload Data (upload a view to DIDS), Provenance Reporter (report provenance to PROMS) and Email Notifier - not related to DIDS or PROMS - which emails a user on completion.

Workflow B is part of a series of heterogeneous workflow automators communicating via DIDS and PROMS. Part of the data downloaded by Workflow B is created by a different workflow, Workflow H, which is not a Trident workflow. Workflow H runs before Workflow B, and has therefore uploaded data to DIDS and its provenance to PROMS.

The effort overhead required to utilize PROMS, DIDS and the PID Service beyond that of simply implementing the geoprocessing tasks is thus only the time taken to include and configure these three support activities in the workflow. This is straightforward due to their applicability to any Trident workflow. The computational overhead for their use is negligible compared to the workflow's geoprocessing. Due to both local caching (workflow activity) and remote caching (PROMS/DIDS), network bandwidth usage is roughly equivalent to any system that commits important results to a non-local archive.

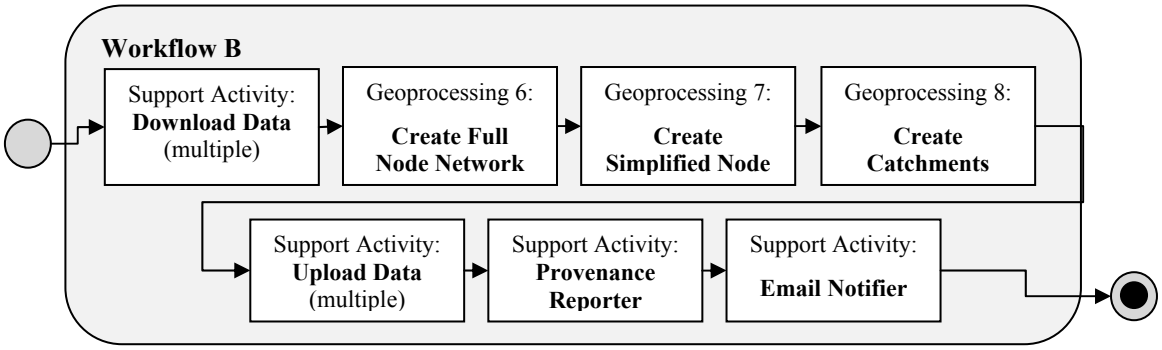

Figure 4: The conceptual layout of the Geofabric Project's "Workflow B" – a geoprocessing workflow – that utilises components that communicate to a provenance architecture via Web Services

5. CONCLUSION

This paper has presented two new applications, DIDS and PROMS created using simple and flexible web services, that together act as an external data and provenance store. Using them it is possible to preserve the reproducibility and provenance capture capabilities of the scientific workflow engine Trident.

Further, it is possible for workflow automators, without the capacity to persistently store data and capture provenance, to acquire those capabilities through the creation and use of four simple software components that interact with PROMS and DIDS. Where all workflow automators in a series use these services, it is possible to preserve individual workflow's provenance, the entire series' provenance chain and data lineage across heterogeneous workflow tools.

An existing tool, the PID service, can increase the usability and maintainability of DIDS and PROMS.

DIDS and PROMS have proven easy to learn and understand, and robust enough for wider adoption. Over ten users have been trained in under three hours to use the services through a web interface and in scientific workflows. A non-programmer has written DIDS and PROMS software components. DIDS and PROMS are in trial or adoption for multiple projects and it is anticipated should have over fifty users within a year.

Users have requested additional functionality that is not yet implemented but is likely to be developed once use cases are articulated. These include:

Secure and/or private DIDS servers. Some projects require data to be kept with a certain level of confidentiality. This can be implemented through standard webserver authentication.

Links from DIDS items to PROMS items that used them. This allows fast provenance tree traversal - meaning it will be possible to trace from PROMS to DIDS and back again, and therefore to query which workflows have used the data (and how). For example this could allow discovery of which processes relied (directly or indirectly) on faulty data. This can be implemented through allowing a reporting system to upload a notification of the use of a DIDS data item to that item by way of adding an *implementor* item (a URI and title). The DIDS data model shown in Figure 3 already shows a class for *Implementors*.

Web interface query tools. Feedback shows large DIDS and PROMS repositories can be difficult to navigate with the web interface. Query/Filter tools would do a lot to help this. The tools can be implemented through web pages using SPARQL queries on the underlying repository.

REFERENCES

Barga, R., Simmhan, Y., Withana, E.C., Sahoo, S. and Jackson, J. (2010) Provenance for Scientific Workflows: Towards Reproducible Research. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering. Online at http://ceng.usc.edu/~simmhan/pubs/barga-deb-2010.pdf.

Cox, S.J.D., (2011) Identifiers for Water Features: Requirements and Current Best Practices. In Water for a Healthy Country Flagship Report Series ISSN: 1835-095X.

Lee, B. and Box, P. (2012), Generation of data product provenance information from HWB. CSIRO Water for a Healthy Country Flagship, Australia

Hartcher, M.G. and Lemon, D. (2009), Developing data audit trails for the CSIRO Sustainable Yields projects. In Anderssen, R.S., R.D. Braddock and L.T.H. Newham (eds) 18th World IMACS Congress and MODSIM09, pp. 2377. ISBN: 978-0-9758400-7-8. http://www.mssanz.org.au/modsim09/J4/hartcher.pdf

Lebo, T., Sahoo, S. and McGuinness, D. eds. (2013) PROV-O: The PROV Ontology. W3C Recommendation. From http://www.w3.org/TR/2013/REC-prov-o-20130430/ retrieved 30 April 2013.

Moreau, L. and Missier, P. eds. (2013) PROV-DM: The PROV Data Model. W3C Recommendation. Web page retrieved on 30 April 2013 from http://www.w3.org/TR/2013/REC-prov-dm-20130430/.

The Bureau of Meteorology (2013) Australian Hydrological Geospatial Fabric (Geofabric). Project description web page retrieved on 7 July 2013 from http://www.bom.gov.au/water/geofabric/.

The Internet Society (1999), Hypertext Transfer Protocol – HTTP1.1. The World Wide Web Consortium (W3C), Web page retrieved on 30 April 2013 from http://www.w3.org/Protocols/rfc2616/rfc2616.html.

The PROMS Team (2013) The Provenance Management System. Wiki web page, online at https://wiki.csiro.au/display/proms. Accessed 7 July 2013. CSIRO Land & Water.