



JISC Final Report

Project Information			
Project Identifier	<i>To be completed by JISC</i>		
Project Title	Web Services Tiered Internet Authorisation		
Project Hashtag			
Start Date	1 January 2010	End Date	31 March 2011
Lead Institution	EDINA		
Project Director	Peter Burnhill		
Project Manager	Fiona Culloch		
Contact email	fiona.culloch@ed.ac.uk		
Partner Institutions	N/A		
Project Web URL	http://edina.ac.uk/projects/wstieria_summary.html		
Programme Name	<i>Access and Identity Management</i>		
Programme Manager	Christopher Brown		

Document Information			
Author(s)	Fiona Culloch		
Project Role(s)	P.I.		
Date	14 April 2011	Filename	
URL	<i>If this report is on your project web site</i>		
Access	This report is for general dissemination		

Document History		
Version	Date	Comments
1	14 March 2011	Initial draft
2	4 April 2011	Draft for review by ED, CH, CLJ, IAY, GG
3	14 April 2011	Incorporate updates for review by programme manager (CB)

Table of Contents

1	ACKNOWLEDGEMENTS	3
2	PROJECT SUMMARY	3
3	MAIN BODY OF REPORT	4
3.1	PROJECT OUTPUTS AND OUTCOMES	4
3.2	HOW DID YOU GO ABOUT ACHIEVING YOUR OUTPUTS / OUTCOMES?	5
3.2.1	<i>Background: The Problem</i>	5
3.2.2	<i>Aims and Objectives</i>	6
3.2.3	<i>Outline of Work Undertaken</i>	7
3.3	WHAT DID YOU LEARN?	10
3.4	IMMEDIATE IMPACT.....	13
3.4.1	<i>Impact on host institution</i>	13
3.4.2	<i>Wider community benefits</i>	13
3.4.3	<i>Change in stakeholder attitudes</i>	13
3.5	FUTURE IMPACT	14
4	CONCLUSIONS.....	15
5	RECOMMENDATIONS.....	16
6	IMPLICATIONS FOR THE FUTURE	16
7	REFERENCES	17
8	APPENDICES	17
9	APPENDIX A	18

1 Acknowledgements

This project was funded by JISC as part of its Access and Identity Management (AIM) Programme.

Considerable assistance was provided during the course of the work by Chad La Joie from the Shibboleth Core Team as a formal consultant and informally by Ian Young and Rod Widdowson from the SDSS Expert Group in Identity Management at EDINA. Chad provided assistance and review throughout. Ian assisted in specifying and setting up the development and test environment for Shibboleth delegation; Rod helped test and review the attempt to apply federated access to WebDAV. Jennie Fletcher from the EDINA geospatial team was particularly helpful in undertaking changes to the Digimap service. Any errors and omissions in this report are, of course, the responsibility of the author.

2 Project Summary

Federated access management systems (e.g., Shibboleth) are usually configured to presume the direct presence of a human user who can interact using a web browser to select their home institution or enter a username and password. However, machine-to-machine *web service* client software may have no support for this login process in its user interface, or no user interface at all, or no access to the end user, or may not be trusted with user credentials. This project has demonstrated two methods by which organisations can provide controlled access to web services from such clients using unmodified, off-the-shelf federated access management software. Using a variant of the first technique, EDINA Digimap now allows licensed UK federation end users to access Web Map Services that present Ordnance Survey data to unmodified desktop Geographic Information System (GIS) programs, as opposed to browsers.

This first method uses an HTTP proxy or “facade” that forwards requests to a protected web service only if the URL presented contains a valid access token. The end user obtains the token by first logging in with a standard browser to a separate web site protected by federated access management in the usual way. The facade URL containing the token is then either statically configured or dynamically copied and pasted by the end user into the actual client application. Accessing the facade requires no user interaction by the client, is compatible with any client that expects a normal server URL, and requires no changes to the web service itself. Moreover, we show how to implement such a facade by relatively simple configuration of an Apache web server rather than by special-purpose software.

The main drawbacks of the facade method are cumbersomeness of the user intervention required, and the implied level of user understanding or training demanded. The second method uses new delegation features in Shibboleth that allow a web application to invoke web services on the user’s behalf with no explicit intervention by the user being required. These web services may themselves invoke other web services, which also have access to user attributes, in so-called “n-tier” scenarios.

3 Main Body of Report

3.1 Project Outputs and Outcomes

Output / Outcome Type (e.g. report, publication, software, knowledge built)	Brief Description and URLs (where applicable)
Interoperation of web services with federated access management demonstrated in a production service	EDINA Digimap now allows access from non-browser GIS desktop software to Web Map Services containing licensed Ordnance Survey data for users authorised by federated access management, using a variant of the facade method described below. Previously the web services feature was only available to a small, closed group of trial users. http://edina.ac.uk/digimap/
Technical Note 1 <i>Federated Access to an HTTP Web Service Using Apache</i>	Describes how to configure the Apache web server to act as a facade allowing access to a web service only for users who are authorised based on user attributes provided by federated access management, with examples using the UK federation. http://edina.ac.uk/projects/wstieria/files/TN01-facade.pdf
Technical Note 2 <i>An Investigation of Federated Access Management for WebDAV</i>	Explains how far it was possible to apply the method described in Technical Note 1 above to enable federated access to web-hosted file directories using the WebDAV protocol. Although a number of obstacles were overcome, the specifics of the WebDAV protocol made the result less than completely successful. The exercise did clarify the domain of applicability of the method, which depends on a simple HTTP request/response web service that does not use HTTP headers to carry application-level data (unlike WebDAV). http://edina.ac.uk/projects/wstieria/files/TN01-webdav.pdf
Invited presentation to security working group of Open Geospatial Consortium (OGC) 72 nd Tech. Committee Meeting, Frascati, Italy: <i>Practical Approaches to Web Services Authentication</i>	Describes why interoperation of federated access management with web services is problematic and outlines two solutions: the facade method (described in more detail in Technical Note 1 above) and the new delegation features recently added to Shibboleth. http://edina.ac.uk/projects/wstieria/files/2010-03-09_OGC_Frascati.pdf
Raised awareness of feasibility of facade method within the OGC community by participating in live webinar	Blog post: http://wstieria.blogspot.com/2010/12/wstieria-project-recently-participated.html OGC press release describing webinar: http://edina.ac.uk/projects/wstieria/files/PR-webinar.pdf
Presentation at JISC FAM10 conference, Cardiff: <i>The WSTIERIA Project – A Web of Services</i>	Describes the status of the project as of October 2010: http://edina.ac.uk/projects/wstieria/files/2010-10-06_FAM10.pdf
Set up test environment for new Shibboleth delegation software	Environment consisted of four machines (2 physical, 2 virtual), one running a Shibboleth IdP with delegation plug-in, the second running a Java development environment (Eclipse, Maven), the third running a Shibboleth SP protecting a Java web application (JSP) that uses the JASIG delegation library to invoke a protected web service on the fourth machine, which runs the a simple test web service and a Shibboleth SP.

Knowledge gained	How to deploy and configure Shibboleth delegation as above.
Live demonstration of above setup to JISC programme manager	User logs in to the JSP web application (normal Shibboleth flow); web application invokes test web service on user's behalf (delegation). The web service displays the user's attributes as if they were logged in directly, but in fact they are coming via the delegation library within the JSP web application.
Knowledge gained	Before undertaking this work it was not known whether the JASIG delegation library, which was developed for use with uPortal, would also work outside a supporting uPortal framework. We now know that the library can be successfully used by a plain Java calling application (such as a JSP) in place of the expected uPortal portal, provided that a few additional lines of code are added in the caller, to initialise some context that uPortal would normally provide.
Knowledge gained	The delegation plug-in for the identity provider had become incompatible with the latest version of the IdP software. This was reported to the developers (and will be resolved long-term when the plug-in functionality is rolled in to the main IdP code).
Knowledge gained	The current JASIG delegation library implementation does not appear to be compatible with self-signed IdP certificates. The UK federation has recently switched from recommending commercial certificates for IdPs to recommending long-life, self-signed certificates, so this is potentially problematic. The issue has been reported to the developers of the library and a partial work-round is available.
Project web site	http://edina.ac.uk/projects/wstieria_summary.html
Project blog	http://wstieria.blogspot.com/
Final Report	This document

3.2 How did you go about achieving your outputs / outcomes?

3.2.1 Background: The Problem

Significant numbers of web services have been and continue to be developed by the community. We use the term *web services* here specifically for machine-to-machine interaction, typically XML-based, as opposed to plain web *sites* that produce HTML for direct consumption by a human being with a web browser.

A great deal of effort has been expended in deploying federated access management systems for user authentication to HE/FE web sites. This is the purpose of the UK federation and similar federations overseas.

The communication mechanisms (SAML bindings) normally used in federated access management rely on a web browser to pass messages between a Service Provider (SP), commonly a resource publisher web site, and an Identity Provider (IdP), usually an educational institution. Behind the scenes, messages are forwarded through the user's browser using redirections, cookies and HTTP POST invoked by JavaScript, as shown in Figure 1(a). The use of encrypted SSL channels is mandatory in the UK federation.

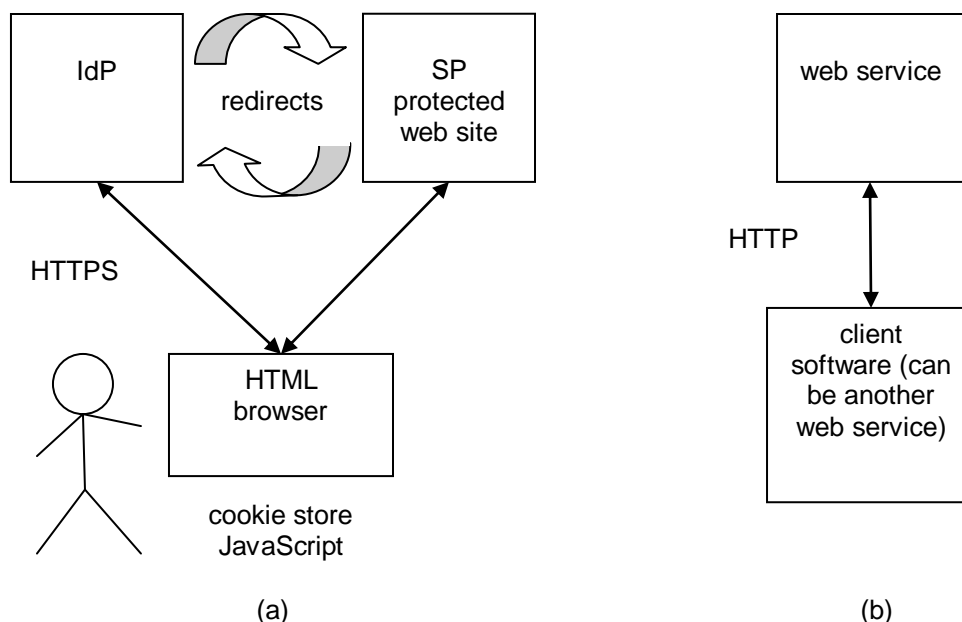


Figure 1

Web services though are usually detached processes invoked by plain HTTP requests from another web service or application, as shown in Figure 1(b). (SOAP and higher-level protocols may be layered on top of these requests). In general, web services are not directly invoked from a browser and may not have direct access to the end user. The arbitrary client software used will often not support SSL, redirection, cookies or JavaScript.

Even if it does emulate browser features such as redirection, a client without direct access to the user cannot handle being redirected to an IdP that expects the user to type a username and password into a web form, or select their home institution at a "Where Are You From?" (WAYF) page, as used in the UK federation.

However desirable it might be to use standard UK federation SP software to manage access to web services, these incompatibilities have ruled it out to date. Currently, the state of the art for the simple web services of interest to us remains uncontrolled access, or access controlled by IP-address checking or application-issued tokens. We are interested in Representational State Transfer (REST)-style web services; SOAP-based web services are not considered further here.

3.2.2 Aims and Objectives

Given the above context, the overall aim of the project was to produce software and documentation that would allow developers of web services to make use of federated access management, and in particular the UK Access Management Federation for Education and Research (the UK federation), to authenticate end users and obtain attributes about them for authorisation purposes.

Beneath this overarching aim were a number of more specific objectives:

1. The JISC-funded SEE-GEO¹ project at EDINA had developed a concept for combining federated authentication with web services by splitting access management into two parts. The first part is a SAML Service Provider component that a user can log in to using a standard SAML Identity Provider to obtain a session id. Only this part requires a human user at a browser, with support for cookies, HTTPS, etc. The second part is a "facade", which

¹ <http://edina.ac.uk/projects/seesaw/seegeo/>

forwards client requests containing a valid session id (either in the URL or in a cookie, if supported) to the protected web service and returns the responses. A firewall blocks attempts to bypass the facade. The session id links the two parts. Note that adding the facade can be done without modifying the underlying web service itself. The SEE-GEO implementation of this concept consisted of a monolithic facade written from scratch in Java, containing a SAML Service Provider, the forwarding mechanism and a database of information associated with the session ids. The SP component required a specific binding (the Artifact Binding) of the SAML browser SSO profiles and the request-forwarding mechanism was specific to a particular web service (the OGC Web Map Service), requiring detailed knowledge of its application-layer protocol. The first objective of the current project was to investigate whether a new implementation of this facade concept was possible using standard, open-source SAML Service Provider software that is being actively maintained (Shibboleth), without restricting the profiles and bindings that could be used beyond any restrictions imposed by a federation, and in a way that is not tied to a particular application protocol and could be easily deployed by web service developers to handle federated access management.

2. We wanted to produce a demonstration OGC web service using this new implementation, accessible via the standard UK federation authentication mechanisms from unmodified client software. In the end though, it was decided that the highest-impact outcome for the widest audience would be obtained for the least effort and lowest risk by taking advantage of some similar work that had been done previously in another part of the EDINA geospatial team, which we discovered during the early investigation and familiarisation parts of the project. That work had used a very similar concept to provide OGC Web Map Service access to EDINA Digimap data for a small group of trial users. The main demonstration therefore consisted of modifying the production Digimap service to make the Web Services feature available to all registered users using that previous implementation, which is integrated into the Digimap code, rather than the implementation that was developed during this project. Our implementation was however demonstrated to the participants in a live webinar² organised by the EDINA geospatial team as part of the OGC Web Services Shibboleth Interoperability Experiment they were managing in November 2010.
3. Since the original SEE-GEO work was done, new developments in Shibboleth³ and the advent of a supporting library now allow a web application protected by one Shibboleth SP to invoke a web service protected by a second Shibboleth SP. The web service has access to attributes of the user who is logged in to the calling web application. This attribute information comes from the user's identity provider, not from the web application or the library, which merely relays a SAML assertion signed by the user's IdP. Therefore the web service need not trust the web application. The mechanism may be chained, with one web service invoking another in the same way to any depth in an "n-tier" scenario. This Shibboleth-based method is sufficiently new that no local experience of it was available. The objective here was therefore firstly to gain such experience by setting up a test environment, and secondly to determine whether the method, which was developed for a specific use case where the web application was a uPortal portal and the web services were portlets, would also work in the case of a simple Java web application and a web service that is not a portlet. Note that because the web service client must be modified to use the library, the existence of this new option will not eliminate the need for the facade method in all cases.

3.2.3 Outline of Work Undertaken

The project developed in the following phases:

1. In late 2009, before the formal start of the project in January 2010 and as part of EDINA's institutional contribution, desk research was done to inform the selection of the objectives discussed above and gain an initial level of understanding of the previous SEE-GEO work by reading papers and meeting its project manager, of the new Shibboleth delegation technology

² <http://edina.ac.uk/projects/wstieria/files/PR-webinar.pdf>

³ <https://spaces.internet2.edu/display/ShibuPortal/Configuring+Shibboleth+Delegation+for+a+Portal>

- by travelling to Zurich to meet with Chad La Joie from the Shibboleth Core Team, who agreed to act as a consultant, and of relevant efforts elsewhere (WRAP, MashSSL, Ping Identity).
2. Starting in January 2010, hands-on development work began. Initially the objective was to produce quickly a working prototype based on the facade concept from SEE-GEO, but using the standard Shibboleth SP in place of the previous bespoke SAML SP code. The intent at this stage was that the experience gained would feed in to ongoing improvements or re-implementations of this software as required by its initial users, who would be external organisations wishing to make web services publicly available via federated access management.
 3. During February-March 2010, analysis of the requirements for a prototype pointed to the strong similarities between the client-facing part of the “facade” and an HTTP proxy. The facade is essentially a proxy that only forwards requests containing a valid session id in the URL. We therefore started to investigate whether an off-the-shelf proxy could be made to perform the required validation, at least for a prototype, looking first at HTTP::Proxy (a perl module) and then at the proxy and URL-rewriting features of the Apache web server.
 4. Apache turned out to be flexible enough to do the job. To confirm that this approach would satisfy the objective of providing federated access management to an OGC web service from standard, unmodified client software, a free, open-source desktop GIS package (QGIS) was installed and used to test federated access via a facade to an OGC Web Map Service serving EuroGlobalMap data. Access to the underlying server is restricted by a firewall (IP-address protection) but the facade can provide wider access based on user attributes supplied by an identity provider.
 5. The emphasis of the project shifted somewhat during this period. Originally it was assumed that the new facade implementation would consist of software to be written by the project and developed based on feedback first from internal EDINA users and then from external users. However, a working prototype was now available that combined an off-the-shelf Shibboleth SP with an off-the-shelf web server (Apache) to give a functional system that, due to the use of industrial-strength components, could be expected to be both higher performance and more robust than a prototype written from scratch. With sufficient documentation this approach should be deployable by the external web service developers who were the original target user group. In practice, an approach based on standard software would likely be more acceptable to them than any alternative that would require installation of new software outputs from a research project.
 6. A decision was therefore made to produce a “how-to” document that would explain the facade concept and how to implement it by suitable configuration of Apache, using an OGC Web Map Service as an example, and to regard this document as being the required implementation of the facade concept. This document (Technical Note 1) was published on the project’s web site in April 2010. Readers interested in applying this method of using federated access management to their own services should consult that document.
 7. Consideration was then given to the exact form of the demonstrator envisaged in the project plan (an OGC web service). A meeting was held with staff from throughout EDINA at which this question was discussed. A number of interesting ideas were proposed, including access to Historic Digimap from mobile clients, integration of data from two separate protected web services (3D building models, LIDAR measurements), census XML data feeds and direct external access to the back-end web services behind Digimap.
 8. It was while discussing the last of these ideas that previous work by the Digimap team to make those web services externally available to a trial user group, using a technique similar to the facade idea, was considered more closely
 9. It was clear that, although the EDINA geospatial team had been (and remained) supportive of making the web services which form the back end of Digimap publicly available via federated access, the licensing conditions restricted access to individually registered users. This would require any facade to be closely integrated into Digimap’s existing user registration and logging system rather than separate as originally envisaged.
 10. It was therefore decided that the highest impact outcome would be obtained with the lowest effort and risk by taking the work previously done for the trial group and integrating it into the live production service. This work was done during the summer of 2010.
 11. Originally we had hoped to validate the external demonstrator by finding interested partners and getting them to provide feedback. However, although a team member with strong previous experience of setting up external collaborations arranged an exploratory meeting

- with MIMAS in March 2010 and contacted other organisations by phone, the conclusion was reached that in the absence of a formal partnership agreement with corresponding funding, it would not be possible to involve external partners productively at this time.
12. The demonstrator originally planned was to be an Open Geospatial Consortium (OGC) web service, due to the geospatial heritage of the previous SEE-GEO project and to maximise the chance of the method developed being picked up and used more widely within the OGC world. Discussions with the SDSS expert group suggested an additional demonstration that the technique was generally applicable, and not restricted to OGC web services, would be useful.
 13. We therefore decided to attempt to apply the facade method to WebDAV, which allows remote access to files in a directory tree over an HTTP-based web service protocol. The Apache web server provides a standard, free WebDAV server implementation, while common desktop operating systems, including Windows, MacOS and Linux, all have built-in WebDAV clients. Federated management of access to remote WebDAV file directories would be useful in itself and would also allow interested parties to try the facade mechanism on a real service without having to do a lot of software installation and setup.
 14. The attempt to produce an Apache-based facade for the WebDAV protocol was only partially successful (see the later section on lessons learned). If federated authentication for WebDAV had been a primary objective of the project it would have been possible to overcome the remaining obstacles by writing a dedicated WebDAV facade implementation rather than relying on Apache as the implementation. At this point though, the project had reached the half-way point and it was felt that moving on from the facade method to look at the Shibboleth delegation/n-tier method was desirable. Technical Note 2 describing the experience with WebDAV was therefore written up and published on the project's web site in June 2010.
 15. In August 2010, a start was made on the pre-requisites for working with Shibboleth delegation, setting up one physical server and two virtual machines (all three running CentOS5, for which Shibboleth service provider binary packages are available), installing a Shibboleth identity provider on one, a Shibboleth service provider and Tomcat web app container on the second, and on the third machine a second Shibboleth SP to protect a test web service. All three Shibboleth entities (the IdP and two SPs) were then registered in and configured for the UK federation, and tested independently in normal operation (without delegation).
 16. This was fairly well-trodden ground for the author, who was already familiar with setting up Shibboleth entities within the UK federation. The next step was to set up a Java development environment (Java JDK, Eclipse, Maven plug-in for Eclipse) for the test web application which would invoke the web service. This was "my first Java program" so required a bit of learning on the job.
 17. The first venture into new Shibboleth territory was installation of the delegation plug-in for the IdP. Initially this did not work. The problem was traced to an incompatibility between the delegation plug-in and version 2.2.0 of the IdP. This was reported to the plug-in developers via our consultant and worked round by installing an earlier version of the IdP (first 2.1.3, the version the plug-in was written against, and then 2.1.5, which also worked).
 18. The next stage was to write a dummy JSP web app and check that it correctly received attributes about the user logged in to the Shibboleth SP protecting it. This initially failed because receiving attributes as environment variables from Apache (mod_shib) in the normally recommended way does not work for Java web apps proxied via the AJP protocol using mod_proxy_ajp, which restricts the variable names allowed to names starting with "HTTP_". Once this was realised, the problem was avoided by taking attribute values directly from Shibboleth HTTP headers instead.
 19. The test web app needs to invoke a web service on a separate machine, basically initiating a GET and receiving the response. Java has built-in ways of doing this. However, the JASIG delegation library, which allows an application to access a Shibboleth-protected web resource, is implemented as a sub-class of the HttpClient class from the Apache Foundation's HTTP components project, which provides more powerful and flexible HTTP features than the basic Java ones. We therefore decided to begin by reading about the HttpClient class and using it to invoke the test web service.
 20. At this point, we could log in to the web application using our IdP with the delegation plug-in and see output containing both the user attributes presented to the web app and the output returned by the test web service. But that web service was being invoked by plain HttpClient

and not yet by the delegation-enhanced JASIG library version that would allow the web service to see the user's attributes. By now it was December 2010 though, and it was clear that it would not be possible to install and configure the JASIG library before the project end date. An extension was requested and granted by the JISC programme manager to March 2011.

21. During January and February of 2011, the JASIG delegation library was incorporated into the test JSP project build (using Maven) and the source code of the library inspected to determine what parts of the context provided by the uPortal environment for which it was originally developed would need to be emulated for it by the test JSP. This was crucial, because at the outset of the project we had no *a priori* assurance that it would even be possible to disentangle the library from uPortal in this way. In the event it turned out to be not only possible but relatively straightforward.
22. The rest of the work then consisted of re-configuring all three of the IdP, the web app (portal) SP and the web service SP as required to support delegation: the IdP must be told which SP entities are allowed to request delegation (the portal web app SP), and which entities may receive delegated assertions (the web service SP, in the uPortal case a portlet). The web app SP must be configured to make the public key of the IdP available to the delegation library. The whole of this configuration process is at present quite intricate and potentially error prone. However, the available documentation, although formidably dense and describing many manual steps, all of which must have been done correctly for operation to succeed, is both complete and accurate, which is something of a rarity in the author's experience, and to be commended. Although we had significant Shibboleth experience (which may have inadvertently filled in any documentation gaps) and made every endeavour to follow the recipe exactly, realising from the outset that any inaccuracies would result in hard-to-trace failures, still several debugging steps were required once everything was set up. This is not yet a plug-and-play technology and it is not claimed to be.
23. Once everything was set up through, the software functioned as expected, even in a live demonstration to the JISC programme manager on 25 February 2010. This was effectively the "hello, world" of delegation, with the user logging in to the test web app's Shibboleth SP and it then invoking the test web service on the user's behalf and displaying the output sent back. That output contained user attribute values obtained from the user's IdP without the user having to log in to the web service directly. The web service could use these attributes for making authorisation decisions. The interaction between the web app SP, the IdP and the web service SP was also observed directly, by inspection of Shibboleth log files.
24. Obviously it would be desirable to develop a more challenging, real-world application but that was not one of the project's objectives. In the event, simply assembling the components, understanding and configuring them, and verifying basic operation proved more than enough in the time available. On the other hand, we now have the experience we earlier lacked, which can potentially be passed on to others.

3.3 What did you learn?

Federated access to web services can be done with tools that are already in widespread use

Originally we supposed that a facade adding federated user authentication support to non-SOAP web services would need to be specially written software, as in previous work. Investigation showed, however, that the required functionality can be constructed in a simpler way using standard packages already well known to many system administrators: HTTP proxy and URL validation from Apache, SAML SP from Shibboleth, token issuance from local scripts (perl etc.) We have produced documentation to enable external users starting from scratch to apply the technique using this combination of standard tools.

But the facade technique falls down if endpoint URLs or paths are treated as data

An abstract conception of a web service and its clients leads to the expectation that the application-layer messages sent between them (requests and responses) would contain only application data, which would be independent of the location of the server or client involved. Such an architectural approach would be in keeping with the REST principle of a layered system that allows for intermediary

“connectors” (such as our authorisation facade) to be interposed transparently between client and server. However, in both of the web services that we have looked at in detail (OGC WMS and WebDAV), application-layer messages sometimes contain absolute references to URL paths at the server, forcing intermediaries to know the structure of the application protocol and rewrite URLs or paths embedded within the application data (similarly to products like EZproxy) to avoid being bypassed by the client. Such an intermediary is obviously not a general-purpose, application-independent component. The facade technique is therefore best suited to cases where the application protocol is familiar to the developer (or mutable) and this problem can be guaranteed not to occur.

When authentication is involved, in-house use cases are very different from external ones

During the planning of WSTIERIA, because EDINA Digimap is built using back-end OGC web service components it was believed that it would be relatively easy to identify the kind of tightly-coupled, in-house use cases where development could simply target in-house users' requirements as a good proxy for the requirements of the wider community. However, on closer inspection, the in-house use cases for these back-end OGC web services all presume prior user authentication at a “front door” web site (Digimap), obviating the need for the back end to perform authentication itself. Some prior work had been done within EDINA to investigate making the back-end web services directly available to a trial group of external users when used in concert with the standard Digimap web site to obtain an access link for use by a separate web service client. Further work was done within the current project to make this feature available beyond the trial group, to all registered Digimap users.

Shibboleth delegation is deployable at a test level by developers not in the Shibboleth Team

There is sufficient, and sufficiently accurate, online documentation available to enable developers with strong Shibboleth IdP, SP and Java experience to build a web application that uses the new delegation features to invoke Shibboleth-protected web services (and for developers with Shibboleth SP experience to construct such services). We managed to deploy a relatively simple configuration, as described previously, with only limited hand-holding from a consultant on the Shibboleth Core Team, much of which was related to Java and Maven rather than Shibboleth itself (due to the limited Java experience of the developer).

Production deployments would benefit from further software development

One exception where support from the consultant was required was in determining that the reason why the currently available IdP delegation plug-in would not work with version 2.2.0 of the IdP was because changes to the IdP's internal interfaces in 2.2.0 had made it incompatible with the plug-in. This was easily addressed in our test deployment by reverting to an earlier version of the IdP (2.1.5) but that approach is obviously less suitable in a production environment. It is not known whether a revised version of the plug-in will be produced before the planned incorporation of its functionality into a future version of the main IdP code (probably in version 3.0).

As discussed previously, the available documentation at present, while complete and accurate, would be quite fearsome for the average developer, especially one with little Shibboleth experience. This is mainly because it has to describe intricate manual re-configuration procedures involving many steps across three different software setups (IdP, portal, web service), likely on separate machines. In a presentation at the recent McShib conference in Edinburgh, Scott Cantor, the lead developer of the Shibboleth SP, described changes being considered which would migrate most of the SAML functionality of the JASIG delegation library (required by the web application on the “portal” system) out of the library and into the SP, leaving the library as a thin “shim”. In addition to simplifying the implementation, this would have the practical benefit of removing the need for the deployer to both explicitly configure the SP to extract the public key of the end user's IdP and to pass this same information into the delegation library by a method call.

Limitations on widespread cross-organisational deployment of Shibboleth delegation

It was known before the project started (so it is not a lesson learned but is worth stating here) that the delegation library depends on the end user's IdP, the web application (portal) SP, and the SPs

protecting the web services to be invoked by the application all supporting SAML2 and running relatively recent versions of the Shibboleth software (SP 2.2+). Measurements in February 2011 by Ian Young of the SDSS expert group show for the UK federation an “interoperability product” (of IdP and SP entities that will both support SAML2) of 30%. This number has been increasing over time (it was about 10% in May 2010) but is not yet at a level where an application intended for use by an unconstrained set of end users within the federation could use the technology on a production basis, because it would not be possible to restrict end users to those from institutions with IdPs that had already been upgraded to versions that support SAML2.

A second constraint that was pointed out by our consultant at the outset of the project but which having done a test deployment we now understand at a much more visceral level, is that, to limit the potential for SPs to impersonate users to other SPs in an unconstrained way, the IdP requires explicit static configuration to know the entity IDs of both the portal web application and the web service SP(s). That means that if, for example, EDINA wanted to deploy a web application for use by UK federation users which made use of this technology to invoke back-end web services on behalf of the user, in the general case EDINA (or, even more improbably, the end users) would have to persuade the administrators of every IdP in the federation to reconfigure their mission-critical production IdP systems just to enable use of this particular application. The similar problem of service providers having to persuade all IdPs to release user attributes beyond the lowest common denominator has already proved hard to address in practice, so the omens are not good here.

Note that neither of these problems affects the use case for which the Shibboleth delegation software was initially built: a university portal web application invoking portlets that need user attribute information, where the portal, the portlets, the users and their IdP are all from the same organisation. Indeed, the author was approached at the McShib conference mentioned above by a developer from a UK university with a potential application very similar to this (the kind of scenario for which the work of the GRAND⁴ project, also funded by JISC within the AIM programme, would be relevant too).

Possible issue with self-signed certificates

One problem that surfaced when first attempting to run the test setup was an exception reported by the delegation library when attempting to forward the authentication request received from the target web service to the user’s IdP. This was a “peer not authenticated” exception, which caused puzzlement as the portal SP seemed to have been correctly configured to extract the IdP public key from the federation metadata and the test JSP code seemed to be calling the correct library method to pass this key in to the library (`samlSession.setIdPServerPublicKeys`). Investigation showed that the library appears to assume that the IdP trust-fabric certificate will be a commercial certificate issued by a Certification Authority (CA) from the standard Java list of trusted CAs. In our case though, the certificate for the test IdP was a self-signed certificate, so it was rejected at a lower level by the standard Java checks before even being seen by the library.

Unfortunately a self-signed certificate was not being used just because this was a test system. Originally the UK federation recommended that IdPs should use a trust-fabric certificate issued by a CA from a list of CAs qualified by the federation (such as the JANET Certificate Service or some commercial products). However, to decouple the federation from unpredictable and sometimes unannounced changes in commercial certificate hierarchies (which then had to be reflected in the federation metadata on an emergency basis), and to enable future inter-federation, and for several other reasons, this recommendation is changing. Now IdPs will be recommended to use a long-life, self-signed certificate as the trust-fabric certificate. (The federation itself has therefore taken on the identity-proving role previously performed partly by the CA). Other federations, including InCommon, are moving in a similar direction for similar reasons.

There is therefore potentially a growing incompatibility between the current implementation of the delegation library and federations, including the UK federation, that recommend the use of self-signed trust-fabric certificates for IdPs. For the test setup it was easy enough to work around this by making a Java keystore containing the IdP certificate and using an alternative keystore-based method

⁴ <http://www.jisc.ac.uk/whatwedo/programmes/aim/grand.aspx>

provided by the library, `setIdPClientTrustStore`, instead of `setIdPServerPublicKeys`⁵. This approach isn't suitable for general use though, since it would require the web application developer to make a keystore containing the public key for every IdP in the federation. That is why delegation library users are recommended to configure the SP to export the IdP public key, and pass that data to `setIdPServerPublicKeys`.

This issue was reported to Unicon, the commercial developer of the JASIG delegation library. (They were originally contracted to produce the library for the University of Chicago). As of the time of writing, acknowledgements had been received but not yet a substantive response, so whether this is genuinely a present limitation in the library or just a mis-understanding on our part is still an open question.

Delegation not dependent on uPortal framework

At the outset of the project it was not known whether delegation software, particularly the library, would be so entwined with uPortal that it would be either impossible or very difficult to use it outside that context.

Happily we can report that it has indeed been possible to make a test web application (a JSP) completely unrelated to uPortal and invoke a web service that was not a uPortal portlet. During planning of the project it was thought that it might be necessary to first deploy uPortal and become familiar with it, then attempt a delegation example within that framework, and finally try to construct a second test application outside that framework. As it turned out, reading the source code of the delegation library gave sufficient indication that it would work independently of uPortal that it was decided to skip the initial steps and go directly to a non-uPortal application, which was successful. It was, however, necessary to deduce (from the source code of the library and reading the documentation for `HttpClient`) how a non-uPortal caller should initialise any context depended on by the library. This turned out to be just construction of a suitable `ThreadSafeClientConnManager` object associated with `HttpClient`, although coming to that conclusion required a fair amount of work given that we were not previously familiar with either `HttpClient` or the delegation library.

3.4 Immediate Impact

3.4.1 Impact on host institution

The main immediate impact of this project on EDINA has been to provide a context within which previous work that made Digimap data available to a closed group of trial users via OGC web service interfaces was taken further and integrated into the live production service, broadening access to include any UK federation user who is a registered Digimap user.

3.4.2 Wider community benefits

Making Digimap data available via web services in this way benefits Digimap users by allowing them to use desktop GIS software packages (which can act as OGC web service clients) to overlay Digimap maps with maps from other data sources (web or local) for display and analysis, without first having to download a local copy of a large Digimap data file. Digimap already allowed users to display maps online without downloading them, but not in combination with other data sources in this way. The fact that the closed trial user group was willing to register separately for this feature and provide feedback demonstrates its usefulness to some researchers.

3.4.3 Change in stakeholder attitudes

There was significant evolution in the point of view of the EDINA geospatial team as reflected by our liaison contact during the course of the project. At the outset and during the planning stages, WSTIERIA was seen as a way to take forward the previous work of the SEE-GEO project and further develop its facade concept for enabling web services (and specifically OGC web services) to use federated access management. The idea was to investigate whether this concept could be

⁵ Whether to talk about the library acting as an SSL client to the IdP or the IdP as the SSL server for the library depends on the point of view of the developer at the time the names were chosen!

generalised and the implementation improved, while also doing initial investigation of whether the new Shibboleth delegation/n-tier features would be usable by this community.

However, during Q1 of 2010 separate funding became available for a project initiated by a consultant (Andreas Mattheus) who had been closely involved in the SEE-GEO work. This Authentication Interoperability Experiment⁶ with a specific Shibboleth component was organised under the auspices of the OGC, involved contributions from the European ESDIN project and brought together again under its Shibboleth strand (which was led by EDINA) the same cast of characters who had previously worked on SEE-GEO, plus the main technical lead from the EDINA geospatial team. This work also replaced the bespoke SAML SP from SEE-GEO with a standard Shibboleth SP to protect the web service, but rather than continuing with the concept of a facade that could be used by unmodified client software, decided instead to investigate with a number of commercial vendors the possibility of modifying desktop GIS packages to support the SAML ECP profile directly, for a simpler user experience. This work took place as part of the OGC Web Service Shibboleth Interoperability Experiment in Q4 2010 and was initiated and led by the EDINA geospatial team. A reference implementation was produced by modifying the OpenJump client to provide a user interface for selecting an IdP (similar to an online WAYF but built in to the application) and to forward SAML authentication requests from a web service SP to the selected IdP, similarly to the JASIG delegation library (which is also based on ECP). This approach has been quite successful: a number of commercial vendors produced prototype variants of their GIS packages which can access protected web services via SAML ECP in this way. Interoperability between these prototypes was demonstrated at an OGC webinar. Two facade approaches were also demonstrated: WSTIERIA, where the facade is provided online by the operator of the protected web service, and another where facade software is installed locally at the client end, to proxy a protected external web service to <http://localhost> URLs. Either facade approach can be used by unmodified clients.

These changes in the external landscape, along with the realisation that making the back-end Digimap web services externally available using federated access management would necessarily require close integration with Digimap's existing user registration and logging system, combined to kill most of the internal impetus behind the facade method. In the absence of external "pull", this contributed to the decision to document the prototype facade implementation rather than re-implement it again, although as discussed previously that also made sense on its own merits as a way of adding federated access management to an existing web service using robust, standard tools that are familiar to most system administrators.

The EDINA geo team liaison has remained engaged in the project but the emphasis has shifted from the facade method to the Shibboleth delegation approach, which is seen as having the potential to enable orchestration of multiple layers of web services (true n-tier). The latter is likely to be of broad interest to the geospatial community as the inability to secure web services within "chains" is a recognised barrier to developing the Spatial Data Infrastructure agenda.

3.5 Future Impact

We hope to see future impact from this project in three areas:

1. Increasing availability of web services for UK HE/FE that leverage previous investment in federated access management to provide wider access rather than relying on inflexible IP-address protection (which can be ineffective for off-campus users). Sometimes such services have simply been "hidden" because the access management challenges have been too hard. This may come about either directly through deployment of the facade method documented by this project, or perhaps more likely, by web services developers becoming aware that federated access is indeed possible and adapting the technique directly into their own code, similarly to what we have done with Digimap.
2. Growing awareness and deployments of the Shibboleth delegation technology. Having demonstrated that this is mature enough to allow for at least test-level deployment within the

⁶ <http://www.opengeospatial.org/projects/initiatives/authie>

UK federation, we would expect others in the UK to have increased confidence in attempting to apply the technology, especially knowing that some level of relatively local deployment experience is now available.

3. Uptake of the results about Shibboleth delegation technology by elements within the geospatial community progressing the development of understanding how the web services that underpin Spatial Data Infrastructure may be securely combined. Across Europe, in association with the EU INSPIRE (Infrastructure for Spatial Information in Europe) directive, operational geospatial web services are currently being established by a wide range of different public authorities. Many of these services are protected and it may be hoped that WSTIERIA contributes to greater understanding of how they may be most effectively exploited.

4 Conclusions

The following general conclusions have been reached at the end of this project:

1. It is possible to overcome the difficulties and combine web services with federated access management using existing technologies.
2. An organisation offering a web service can add federated access management as an alternative method of access in addition to whatever method of external access (if any) is currently in place, without changing the web service at all, using the facade (proxy) method. The facade makes authorisation decisions on behalf of the protected service.
3. Unmodified client software can access the protected web service, at the cost of some user inconvenience and education (copying and pasting access URLs into client software).
4. It is possible to implement this using only standard system administration skills (Apache, scripting) and standard Shibboleth SP configuration.
5. Although the web service needs no modification, an understanding of its application-level protocol is necessary to ensure that any web service endpoint URLs transmitted as application data can be rewritten to refer to corresponding endpoints of the facade (proxy) instead.
6. Embedding web service endpoint URLs within application data sent to or from the service makes it harder to combine the service with a general-purpose intermediary component such as the facade. However, both services we looked at in detail (OGC WMS, WebDAV) do this.
7. The alternative Shibboleth delegation method is functional and can be applied without needing to understand the application protocol. It also enables a web service to invoke others to any depth (n-tier). This contrasts with the facade method, which cannot be chained (1-tier).
8. There are drawbacks though to Shibboleth delegation in its present form. It requires modification of client code to use the delegation library. Intricate, multi-step manual configuration, beyond the usual Shibboleth configuration, is required at the IdP, the web application's SP and at the SP(s) of the web service(s). At present, the IdP and all SPs involved must run Shibboleth: other implementations of federated access management may not support the Liberty protocols that are required in addition to SAML. All the entities involved must also support SAML2, which is still far from universally deployed within the UK federation.
9. Because reconfiguration of the IdP (and at present also the installation of a plug-in component) is necessary, it should be expected that initial deployments of the Shibboleth delegation features are likely to occur *within* organisations, where it will be possible to ensure that the IdP of interest to the user community is modified along with the SPs, rather than for general-purpose applications with users from many different organisations. The use case for

which the software was originally developed seems like a likely candidate: an institutional portal invoking protected back-end services on behalf of its users.

5 Recommendations

Recommendations for organisations providing web services, either internally or externally:

1. Consider whether a web service you provide might be useful to others. If it has not previously been made externally available because of the difficulty of authorising external users, it may be possible to use one or other of the methods described in this report to provide such access.

Recommendations for JISC:

1. Look at the service portfolio to determine whether existing web services could be made richer, or new functionality made available as web services, in light of the possibility of web services having access to user attribute information, such as institutional affiliation.

6 Implications for the future

For developers, the work described here offers the possibility of exposing web APIs for functionality that depends on knowing important attributes about the user. In particular such APIs can decide whether to grant access to users based on secure knowledge of their institutional affiliation. This should contribute towards the general trend towards componentisation of software, breaking functionality free from monolithic web applications where everything must be done through a browser interface because that is the only way users could be usefully authenticated. One of the techniques described goes further, pointing towards a future of deep networks of components, each of which will have secure access to user attribute information but without the user having to authenticate separately to each component.

For end users in the wider community, although the more adventurous may wish to experiment with “mashing” together the kinds of separated components discussed above, the main implication should be the increasing power of applications over time to be expected as the componentisation trend discussed above makes it ever more practical for end-user applications to combine existing components (services) from across the web.

As with all projects, one future implication that must be considered is sustainability. For the facade technique, the decision to document a procedure based on standard software (Apache, Shibboleth) rather than on custom software, removed the thorny issue of software sustainability, leaving simply some documents (Technical Notes 1 and 2) that will be preserved on the project web site which will continue to be maintained by EDINA. Because the demonstration was embedded into a production service (Digimap), it will be sustained as part of the ongoing development of that service. For Shibboleth delegation, sustainability of the Shibboleth components will be mostly the responsibility of the new Shibboleth Consortium. The exception is the Java delegation library used by the web (portal) application, which was developed and is presently being maintained by Unicon. There has been some public discussion of migrating part of this functionality into the Shibboleth SP.

Going beyond sustaining to further development, it should be clear from section 3.4.3 that within EDINA the greatest interest in future development is in the potential application of the Shibboleth delegation technique to the orchestration of multiple OGC web services on behalf of the end user. Colleagues have expressed interest in applying this technique to a variety of Spatial Data Infrastructure use cases, especially those currently being raised by the INSPIRE directive. Any follow-on here would have the additional advantage of acting to sustain the knowledge gained during the project about deploying Shibboleth delegation. Unlike the concrete outputs this would be harder to sustain without continued active use, especially as the software continues to evolve.

Looking more widely than the original project objectives, one possible alternative to the facade technique (but conceptually related to it) which was identified during the project’s initial landscape

review but was outside the scope of the project plan, was OAuth WRAP⁷. Originally contributed by Microsoft, Google and Facebook, this enables separate issuance of tokens for access to a web service using any authentication mechanism (therefore potentially including SAML/Shibboleth and the UK federation), with the web service validating these tokens. A system of separate short-term and long-term tokens offers flexibility and reduces the potential damage from token compromise. WRAP was taken up and has been incorporated into OAuth 2.0⁸, and a number of client and server implementations have since become available. Future work to identify the extent to which this mechanism can be directly applied to the UK federation may well be useful.

7 References

References have been included in-line throughout the document as footnotes containing URLs.

8 Appendices

⁷ <http://wiki.oauth.net/w/page/12238537/OAuth-WRAP>

⁸ <http://wiki.oauth.net/w/page/25236487/OAuth-2>

9 Appendix A

This is the code of the non-uPortal test web application (JSP) used with the Shibboleth delegation library, included here primarily as a record of how the connection manager required by the SAMLSession constructor provided by the delegation library may be initialised (using ThreadSafeClientConnManager).

```
<%@ page import="java.util.Enumeration" %>
<%@ page import="org.apache.http.client.HttpClient" %>
<%@ page import="org.apache.http.client.ResponseHandler" %>
<%@ page import="org.apache.http.client.methods.HttpGet" %>
<%@ page import="org.apache.http.conn.ClientConnectionManager" %>
<%@ page import="org.apache.http.conn.scheme.PlainSocketFactory" %>
<%@ page import="org.apache.http.conn.scheme.Scheme" %>
<%@ page import="org.apache.http.conn.scheme.SchemeRegistry" %>
<%@ page import="org.apache.http.conn.ssl.SSLSocketFactory" %>
<%@ page import="org.apache.http.impl.client.BasicResponseHandler" %>
<%@ page import="org.apache.http.impl.client.DefaultHttpClient" %>
<%@ page
import="org.apache.http.impl.conn.tsccm.ThreadSafeClientConnManager" %>
<%@ page import="org.apache.http.params.HttpParams" %>
<%@ page import="org.apache.http.params.BasicHttpParams" %>
<%@ page import="org.jasig.portal.security.provider.saml.SAMLSession" %>

<html>
<body>
<h2>Hello World!</h2>
<p>Client IP address: <%= request.getRemoteAddr() %></p>
<p>Test = <%=
org.apache.commons.lang.StringEscapeUtils.escapeHtml("<Test>") %></p>
<p>Parameter foo = <%= request.getParameter("foo") %></p>
<p>Attributes = <%
    Enumeration e = request.getAttributeNames();
    while (e.hasMoreElements()) {
        out.print(((String) (e.nextElement())) + "; ");
    }
%></p>
<p>Headers = <%
    Enumeration h = request.getHeaderNames();
    while (h.hasMoreElements()) {
        out.print(((String) (h.nextElement())) + "; ");
    }
%></p>
<p>
affiliation = <%= request.getHeader("affiliation") %> <br>
Meta-Signing-Keys = <%= request.getHeader("Meta-Signing-Keys") %> <br>
Shib-Assertion-Count = <%= request.getHeader("Shib-Assertion-Count") %>
<%
    HttpClient httpclnt = new DefaultHttpClient();
    ResponseHandler<String> respHandler = new BasicResponseHandler();
    int n = Integer.parseInt(request.getHeader("Shib-Assertion-Count"));
    String samlAssertion = null;
    for (int i = 1; i <= n; i++) {
        String cur = "Shib-Assertion-";
        if (i < 10) cur += '0';
        cur += i;
        String url = request.getHeader(cur);
        out.print("</p><p>" + cur + " = " + url);
        HttpGet httpGET = new HttpGet(url);
```

```
        String body = httpclient.execute(httpGET, respHandler);
        samlAssertion = body;
        body = org.apache.commons.lang.StringEscapeUtils.escapeHtml(body);
        out.print("<br>Body = " + body);
    }
    out.print("<br><br>length(samlAssertion) = " +
        String.valueOf(samlAssertion.length()));
%>
</p>
<p>Test: SAMLSession creation:</p>
<%
    HttpClient httpClient;

    SchemeRegistry sr = new SchemeRegistry();
    sr.register(new Scheme("http", PlainSocketFactory.getSocketFactory(),
80));
    sr.register(new Scheme("https", SSLSocketFactory.getSocketFactory(),
443));
    BasicHttpParams params = new BasicHttpParams();
    ThreadSafeClientConnManager cm =
        new ThreadSafeClientConnManager(params, sr);
    SAMLSession samlSession = new SAMLSession(
        samlAssertion, cm, params);

    out.print("Created SAMLSession<br>");
    samlSession.setPortalEntityID("https://culloch.ucs.ed.ac.uk/shib-sp");
    out.print("Done setPortalEntityID<br>");
    samlSession.setIdPClientPrivateKeyAndCert(
        "/etc/shibboleth/sp-key.tomcat",
        "/etc/shibboleth/sp-cert.pem");
    out.print("Done setIdPClientPrivateKeyAndCert<br>");
    String idpPublicKeys = request.getHeader("Meta-Signing-Keys");
    samlSession.setIdPServerPublicKeys(idpPublicKeys);
// out.print("idpPublicKeys = " + idpPublicKeys + "<br>");
// out.print("Done setIdPServerPublicKeys<br><br>");

    samlSession.setIdPClientTrustStore("/usr/share/tomcat5/webapps/wsidp.jks",
"xxxxxxxx");
    out.print("Done setIdPClientTrustStore<br>");
    samlSession.setSkipValidateIdp(true); // required if using DS

    httpClient = samlSession.getHttpClient();
    out.print("Done samlSession.getHttpClient<br><br>");

    HttpGet httpget = new HttpGet("https://dlib-sdss.ucs.ed.ac.uk/cgi-
bin/printenv");

    ResponseHandler<String> responseHandler = new BasicResponseHandler();
    String s = httpClient.execute(httpget, responseHandler);
    s = org.apache.commons.lang.StringEscapeUtils.escapeHtml(s);
    out.print(s);
%>
</body>
</html>
```