

# CREE

## Contextual Resource Evaluation Environment

### Investigation of JSR 168 and WSRP standards

Part A: Initial investigation and assessment of the standards

*CREE Deliverable S2D1A*

Chris Awre, University of Hull

Jonathan Hunter, EDINA, University of Edinburgh

Tony Austin, Archaeology Data Service, University of York

Justin Tilton and Jim Farmer, instructional media + magic, inc.

September 2004

EDINA®

ads  
ARCHAEOLOGY  
DATA SERVICE



[www.hull.ac.uk/esig/cree](http://www.hull.ac.uk/esig/cree)



JISC

## **Contents**

1. Introduction	3
2. Partners	3
3. Analysis and development of JSR 168	3
4. Analysis and development of WSRP	4
5. Presentation within a portal	5
6. uPortal	6
7. Summary	7
8. References	7
Appendix A: EDINA assessment	8
Appendix B: Archaeology Data Service assessment	13
Appendix C: im+m assessment	19

## 1. Introduction

The CREE project aims to assess, test and document the user requirements of portal-embedded and non portal-embedded search interfaces in a broad range of user contexts. In order to facilitate the integration of these search interfaces within a portal environment, an investigation was initiated into adapting these search tools to be conformant with the WSRP and JSR 168 standards. The preliminary phase of this investigation ended in August 2004 and this report summarises these initial investigations. Additional investigation is ongoing and, it is intended, will result in the development of appropriate portlets for use in testing with users.

## 2. Partners

The partners within the technical development strand of CREE, and the search interfaces being adapted, are as follows:

Partner	Search interface
Archaeology Data Service, University of York	HEIRPORT
EDINA, University of Edinburgh	Xgrain and Balsa
industrial media + magic	Google
JAFER Team, University of Oxford	JAFER

## 3. Analysis and development of JSR 168

JSR 168<sup>1</sup> is a Java Community Process standard that specifies a set of APIs to enable components of a portal, or portlets, to be developed. The standard seeks to enable these portlets, once developed, to then be used within any JSR 168 compliant portlet container. The nature of the standard and its background implies a Java portlet container. However, many available portlet containers currently available and used within portals are Java-based, so wide interoperability and use is possible.

Structurally, a JSR 168 portlet is very similar to a Java servlet. Where the existing search interfaces being investigated within CREE are Java-based already, adaptation of existing servlets to JSR 168 portlets is not considered to be difficult. Subsequent experience has demonstrated this in the cases of both HEIRPORT and JAFER. The NetBeans IDE has been found valuable in this work. The EDINA interfaces are written in Perl; however, both were designed to support machine-to-machine interfaces and it has been considered feasible to develop JSR 168 portlets to act as clients to these m2m interfaces. im+m have been able to develop a JSR 168 portlet interface to the Google API, enabling access to this through a portal.

As such, the ability to present, or wrap, existing search interfaces as JSR 168 portlets has proved to be possible and reasonably straightforward. The level of functionality that can

be delivered through these portlets requires further work, though. In order for the portlets to work effectively within a portal, they need to allow input from the search request, pass the information to the search interface, and then receive and process the results once the search has been completed. The search protocol standard being used has an influence on this. So, for JAFER, implementation of a portlet to the JAFER Z39.50 client database has been successfully carried out. A similar portlet for the JAFER SRW client database carries out the first half of the process, sending the search off, but there are outstanding difficulties with retrieving and displaying the results. EDINA will also be using SRW, in the form of the OCLC SRW client reference implementation, in a two-stage process; SRW will be used to pass the search requests to Xgrain, which itself may use a number of protocols to search each target, according to what is available and configured for these targets. This ability to use different protocols is a flexible and highly beneficial part of the JSR 168 standard, which does not specify what should or should not be used.

Work is concentrating on single search requests at this stage. Looking ahead, the search interfaces being adapted do offer the ability to cross-search a number of targets, although how easy it will be to fully replicate all functionality is undetermined as yet. For example, it has been recognised that alerting the user to the status of the search at each target may be difficult as the portlet container is reliant on the associated Java servlet container, running underneath the portlet container, which can only send a response on the status of the searches on receipt of a request, and cannot update this information automatically.

#### **4. Analysis and development of WSRP**

WSRP<sup>2</sup> is the OASIS standard Web Services for Remote Portlets. This web services standard seeks to enable ‘plug-n-play’ portals, facilitating the surfacing of access to search interfaces and many other areas of functionality within a portal. Unlike JSR 168 it is platform-agnostic, and services to be surfaced can originate from many different systems and environments.

WSRP uses SOAP as a means to communicate between a WSRP producer and a WSRP consumer. Both producer and consumer can be written in any language, although no Perl toolkits for WSRP have been identified as yet. It has not been felt appropriate within the CREE project to develop such a toolkit for use with the EDINA search interfaces. A toolkit does, however, exist for Java, WSRP4J, which has been developed as an Apache project. This allows both Java-based services and JSR 168 portlets to be exposed as WSRP portlets and surfaced within a WSRP compliant portal. Development of JSR 168 portlets for the EDINA search interfaces offers the potential to then use WSRP4J to expose these as WSRP portlets (see Fig 1.). It is recognised that WSRP4J may not be the long-term choice for such development due to its nature as a project; however, it is regarded as stable and usable at this time. JAFER have deployed this with the JAFER Z39.50 portlet and successfully run a search. There were remaining difficulties with rendering the results of this search within uPortal, but these have recently been resolved through co-operation with the uPortal developers.

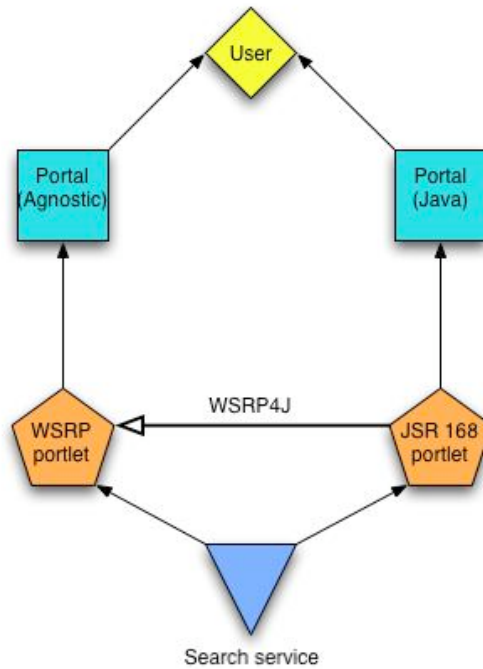


Fig 1. Portlet development paths

This use of WSRP4J has resulted in concentrated effort on the development of JSR 168 portlets amongst CREE partners, with a view to exploring WSRP in further detail at a later date.

General analysis of the WSRP standard has suggested that offering a portlet interface to an existing service may be affected by demands that the standard places on how the service should be developed (which, of course, an existing service may not necessarily adhere to). As such, existing services may find different levels of difficulty in exposing themselves as WSRP portlets. A good knowledge of a range of technologies is also required, including SOAP, XML/RPC, HTML and CSS, against Java for JSR 168.

## 5. Presentation within a portal

CSS is required when using WSRP because the WSRP producer, the part of the standard that exposes the service to be used, is itself responsible for how the portlet will look within the portal. As such, WSRP portlets do not necessarily take on the look and feel of the portal in which they are being surfaced, but can retain, and define, their own. JSR 168 portlets can be responsible for their own look and feel, or mark up, or they can rely on an external service, which may be the portal itself, determining how they will display.

There are pros and cons to both approaches. If the portlet relies on the portal determining its look and feel, then it will fit into the overall look and feel of the portal and a

standardised presentation can be achieved. If the portlet is responsible for its own look and feel, then greater control can be exercised over how the portlet and the information within it is presented. This latter approach facilitates the surfacing of the portlet within the portal, as the portal itself simply has to take what it is given and display it, albeit at the risk of lack of display consistency. im+m are investigating the use of XSL/XSLT with the Google JSR 168 portlet to produce a reference implementation of how a JSR 168 portlet can use XSL/XSLT to determine its own look and feel. Such transformations can combine with those of the portal to allow a variety of presentation styles to be made available, including an accessible text only one. EDINA's decision to use the OCLC SRW client to communicate with the m2m interfaces will also result in control of all the mark up being carried out by the JSR 168 portlet. Customisation can then take place to enable the portal to make use of this.

## **6. uPortal**

In analysing the potential use of JSR 168 and WSRP, it is necessary to have a portal framework within which portlets described using these two standards can be surfaced and tested. The portal framework must be compliant with one or both of the two standards itself. A number of portal frameworks are currently available that offer such compliance, albeit most are commercial (including IBM, Oracle, BEA WebLogic, Plumtree, Sun, and SAP) and therefore unavailable to the CREE partners (although the University of Hull will test developed portlets in the Sun JES portal and against the Oracle portal verification service to test interoperability). Two open source portal frameworks presented themselves as possibilities: JetSpeed and uPortal. uPortal was chosen as the main testbed due to its production status and experience of its use at the University of Hull, where the testbed and ensuing demonstrators will be based. All CREE technical partners have also installed uPortal. Apart from offering experience of the environment, this also offers the Pluto portlet container. As an alternative, EDINA have installed just Pluto as a first step with the intention of moving onto uPortal later.

The choice of uPortal has partly affected the decisions made as part of the analysis carried out of the two standards. uPortal version 2.3 is JSR 168 compliant and is proven in supporting such portlets. This version currently uses a Portlet-to-Channel adapter to expose portlets via uPortal's native IChannel interface (channels being uPortal's method for displaying information prior to the JSR 168 standard being developed). From uPortal version 3, due in 2005, portlets will become the native interface and existing channels will use a Channel-to-Portlet adapter instead.

uPortal has had limited WSRP functionality since version 2.2. This is as a WSRP consumer, not as a producer, though. However, this functionality has proven to be problematic in practice and is being replaced with the WSRP4J toolkit in the recently released version 2.4, again just for the consumer side. Testing carried out on a pre-release of 2.4 by the JAFER Team has shown that there are still some unresolved difficulties with this as well, although these are currently being addressed.

Having a WSRP compliant consumer will allow CREE partners to develop WSRP portlets for testing within uPortal. In order to facilitate this development, CREE partner im+m will be dedicating effort to developing WSRP producer compliance within uPortal during autumn 2004 to complement the consumer functionality.

## **7. Summary**

The initial investigations into JSR 168 and WSRP have revealed much about the relative merits and possibilities of each in exposing existing search interfaces. It is clear that following the JSR 168 route first is regarded as the most practical route, even starting with Perl, especially as these JSR 168 portlets can then be exposed as WSRP portlets at a later date. The two are not, therefore, mutually exclusive, in terms of development effort or schedule. It remains to be seen through development of the portlets to what extent existing functionality can be re-produced within a portlet, but it is clear that exposing search functionality in general is possible, enabling this to be delivered to users within a wider portal environment.

## **8. References**

1. JSR 168, <http://developers.sun.com/prodtech/portalserver/reference/techart/jsr168/>  
(Confirmed 5 Oct 2004)
2. WSRP, [http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=wsrp](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrp)  
(Confirmed 5 Oct 2004)

**Appendix A**

**Jonathan Hunter**

**EDINA**

**13 August 2004**

**Notes on the Initial Investigation of the  
WSRP and JSR 168 standards with respect to  
EDINA's involvement in the CREE project**

## **Introduction**

Viewed from the perspective of a portal service provider, the JSR 168 and WSRP standards have a similar goal: they aim to make the aggregation of content into portals a simpler task. Assuming the portal is compliant with one of the standards, the portal service provider should be able to select any compliant portlet and configure their portal to deploy it. The APIs presented by a java portlet and a WSRP consumer to their respective containers are also very similar. Both standards require that portlets isolate the tasks of processing an action from rendering the results of that action. This is logical since both a portal will typically display more than one portlet at any instance. An action, instigated by a portal user on the portal window (composed of fragments of mark-up from each portlet), will only result in one portlet processing an action, but will require all the portlets currently display to be re-rendered as the user's browser window is refreshed. However, the means of achieving the plug and play interoperability that the standards try to promote is very different is the focus of this document.

This report gives a quick description of the services that we are trying to 'portalise', followed by the key points of the standards and then difficulties with the practical creation of 'toolkits' which conform to these standards.

## **GetRef and GetCopy: background information.**

GetRef is a fully functional cross-searching tool, that enables the user to present a simple search string to multiple services and receive matching results from each. Although it is referred to as a portal it is only a portal in the sense that it bring together multiple resources in the same subject area. It is a subject portal; it does not make use of a portal framework and the formal concept of a portlet has no meaning within the GetRef service. Although most users are only aware of the presentation layer seen in their browser window, it also offers a machine to machine interface (M2M). Work is curenly underway to create a SRW (Search and Retrieve for the Web) interface for GetRef. GetCopy is a location tool - it assists end users with the location of a copy of a journal, journal article or similar item which they have discovered.

The GetRef service is substantially more complex than GetCopy. For this reason, we have chosen to try to 'portalise' it first as we feel it will stretch our knowledge of the standards and expose any shortcomings. In particular the real time updates seen in the interface will be a particular challenge to implement in the portlet.

## **The pluses and minuses of the feasibility of using WSRP and JSR with the Xgrain and GetRef services**

The pluses and minuses of the standards are best discussed if the key aspects of the technologies are understood.

### **JSR 168: key points**

- JSR168 describes the aspects of the development of a portlet and its container in the Java language. The portlet will be a reusable software module (or toolkit) which should work in any JSR 168 compliant portlet container.
- The portlet container is necessary to provide a runtime environment in the which the portlet executes.
- A JSR 168 portlet is very similar to a Java servlet.
- Portlets must be initialised by the container before any action requests can be processed
- JSR 168 support in portal frameworks is proven. It is scheduled to replace the native iChannel interface in the next major release of the uPortal framework.
- The JSR specification does not require that there be a JSR 'producer'.
- The portlet may be standalone or it may communicate with an external service.
- If an external service is involved, the standard does not specify the protocol to be used.
- The portlet may be wholly responsible for the markup returned to the portal, or it may be reliant on an external service to provide the markup. The standard does not specify this.

### **WSRP: key points**

- WSRP - Web Service for Remote Portlets. WSRP specifies the interactions between a WSRP consumer and a WSRP producer. A portal framework which is WSRP compliant will provide a consumer for remote portlet services.
- The SOAP protocol is used for communication between the producer and the consumer.
- WSRP is language independent: the producer (and consumer) can be written in any language desired so long as the standard is adhered to.
- The WSRP producer is responsible for all the mark up. However, since the portal has to use a cascading style sheet (which is applied to all portlets on a portal page) the standard does make recommendations in this area.
- The WSRP consumer is implemented by the portal framework engineers. There is no need for a portlet provider to create a consumer as such - they only need supply configuration information to the portal service providers. This avoids the need for service providers to distribute and support remotely installed portlets.

### **JSR 168 and GetRef**

- The JSR portlet will act as a client to the M2M interface presented by the GetRef service.
- The portlet will make use of the OCLC SRW client reference implementation classes to query the M2M interface presented by the GetRef service.
- The decision that the portlet will use SRW to communicate with the M2M interface presented by GetRef has the consequence that ALL the markup will be generated by the portlet. The output returned to the portlet container can thus be customised to meet the needs of the cascading stylesheet of the portal.
- It will be difficult to inform users of the current status of the search of each of the multiple targets in the same way as the native GetRef user interface. This is because the portlet container is reliant on the servlet container, which will only send a response to a client request, when the response is complete.
- Achieving all the functionality seen in the service in the portlet will be difficult and will be a very substantial undertaking.
- The CportletAdapter class (which acts as an adapter between the native iChannel (in uPortal v2.3) and the portlet interface) is adequate for demonstrating the JSR portlets created by EDINA.
- This observation can be applied to both JSR and WSRP: it would be wise for the target service to generate XML output to which a different transformation can be applied (dependant on the intended recipient). When GetRef was conceived it was only intended that the consuming service would be a dedicated browser window.

### **WSRP and GetRef**

- The WSRP standard requires a good grasp of many technologies including SOAP XML/RPC in addition to HTML and CSS. This is particularly the case with WSRP as there is currently no reference implementation for the WSRP producer. A reference implementation would free the programmer from the need to handle the low level implementation work.
- Undertaking a WSRP producer implementation would place development time on the underlying technology rather than portlet functionality specific to the GetRef service. There would be value in this work, but it might not be the best return on investment judged by the objectives of the CREE project.
- It is not a simple task to offer a portlet interface to an existing service. Implementors would be wise to study the specification ahead of the service implementation, because it makes specific demands on how a service must be developed (e.g. The operational change of state and the rendering of the HTML).

Clearly, the design of GetRef preceded the notion of WSRP. We think this two stage process is easier to handle in the JSR 168 portlet rather than the WSRP producer.

## **References**

WSRP v1.0 specification

[http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=wsrp](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrp)

JSR 168 portlet specification

<http://www.jcp.org/en/jsr/detail?id=168>

EDINA GetRef service information

<http://edina.ac.uk/getref/>

EDINA GetCopy service information

<http://edina.ac.uk/getcopy/>

Guide to the Perl Soap module

<http://guide.soaplite.com/>

Apache Axis, an implementation of the SOAP in Java

<http://ws.apache.org/axis/>

## Appendix B



Archaeology Data Service

Tony Austin (ADS) 12 August 2004

### Initial aim and objectives

“Investigate and document generic aspects of adapting a range of existing search tools and toolkits (JAFER toolkit, BALSAs, Heirport, Google APIs, cross-search) to be conformant with the WSRP and JSR 168 standards, thus facilitating their integration with any conformant national or institutional portal. Ensure that the results of this activity are disseminated effectively to both the HE/FE community and relevant standards bodies.”

(<http://www.hull.ac.uk/esig/cree/downloads/CREE-prop-public.pdf>)

### Identifiable tasks

- i. Literature search
  - a Standards to be used
  - b Implementations
  - c Portlet development
- ii. Software installation
- iii. Development cycle
- iv. Related development

#### **i.a) Literature search: Standards to be used**

The various standards to be employed were examined in detail

*JSR-000168 Portlet Specification V 1.0*

<http://www.jcp.org/aboutJava/communityprocess/review/jsr168/>

This part of the project looks the more straightforward. Structurally it is similar to that employed by servlet containers such as Apache Tomcat; technology which we already employ extensively. A JSR 168 compliant portlet container or API can exist under Tomcat. It is just a different set of class files extending core Java functionality.

*Introduction to JSR 168—The Java Portlet Specification*

[http://developers.sun.com/prodtech/portalserver/reference/techart/jsr168/pb\\_whitepaper.pdf](http://developers.sun.com/prodtech/portalserver/reference/techart/jsr168/pb_whitepaper.pdf)

This document was a very useful overview.

*Web Services for Remote Portlets (WSRP) Specification Version 1.0*

<http://www.oasis-open.org/committees/download.php/3343/oasis-200304-wsrp-specification-1.0.pdf>

*Web Services Description Language (WSDL) 1.1*

<http://www.w3.org/TR/wsd1>

WSRP provides a framework for consistently describing portlets created by PRODUCERS for remote portals or CONSUMERS. The latter harvest portlets and descriptive metadata and aggregate services for use by USERS. The descriptive metadata is written using WSDL. Some possible problem areas...

“Portlet URLs embedded in mark-up fragments often cannot (or should not be direct links to the Producer” i.e. should go via the Consumer.

There is a strict convention for non-ASCII characters in portlet URLs. Should be escaped by converting to format %HH where HH is the hexadecimal equivalent

When aggregating portlets possibility of naming conflicts – to avoid use XML namespace as prefix

Some HTML tags disallowed in mark-up fragments including <HEAD>, <BODY> and <TITLE>

### **i.b Literature search: Implementations**

We intend to use uPortal as a development environment as they are a part of the development team; however, a limited number of alternative portal implementations were examined in case an easier development path was available. The lead partner, University of Hull, will be undertaking in depth testing of toolkits with several portal implementations.

*Apache Portals Project*

<http://portals.apache.org/>

Apache have their own portal implementations known as Jetspeed but have also produced reference implementations for both JSR 168 and WSRP in the form of Pluto and WSRPJ respectively. These are used by some portal implementations. For example, JSR 168 compliant versions of uPortal uses the Pluto portlet container.

*Oracle9iAS Portal Developer Kit (PDK): An Overview of Oracle's Support of OASIS/WSRP and JSR 168 Standards*

<http://portalstudio.oracle.com/pls/ops/docs/FOLDER/COMMUNITY/PDK/articles/OVERVIEW.WSRP.JSR168.HTML>

Oracle AS (Application Server) Portal adheres to the design principles of WSRP and JSR168. There are various free components such as their Portal Development Kit (PDK); however, an Oracle AS licence is required in order to use this as a development environment, which is not free. Another option is:

‘Oracle is offering a hosted portal verification service. This service allows vendors building WSRP producers to test that their implementations run in Oracle's environment. It provides an environment for registering your WSRP producer and adding its portlets to a portal page’.

*WebLogic Portal 8.1*

<http://dev2dev.bea.com/products/wlportal81/index.jsp>

WebLogic Portal 8.1 is described as adhering to the behavior defined by the JSR 168 specification in terms of portlet modes and portlet states. A free, non-expiring developer license for BEA WebLogic Workshop 8.1 and WebLogic Platform 8.1 is available. Thus this is a possible alternative although the statement on JSR 168 compliance appeared quite carefully worded.

*uPortal by JA-SIG*

<http://www.uportal.org/>

As noted already this is the preferred development environment. UPortal expected to have a JSR 168/WSRP compliant portal available from early within this project; however, this has now unlikely to happen until December 2004. There are versions that support JSR 168 or WSRP so this not a major problem beyond installing multiple instances of uPortal.

v. 2.2.1 WSRP producer and consumer

From uPortal FAQ “Web Services for Remote Portlets (WSRP) is supported by uPortal as of uPortal 2.2. The WSRP Proxy channel type replaces the Remote Channel that was introduced in uPortal 2.1. This WSRP Proxy acts as a WSRP consumer that can consume content from any WSRP producer, including another uPortal channel. uPortal 2.2 also implements WSRP as a producer. Note that as of uPortal 2.2, WSRP authentication is not supported.”

v. 2.3.3 JSR 168 portlet adapter and Pluto portlet container implementation

From uPortal FAQ “uPortal will support JSR-168 Portlets in uPortal 2.3 through a Portlet-to-Channel adapter. The Portlets will be managed using Apache's Pluto portlet container. Beginning in uPortal 3.0, Portlets will become uPortal's native content module, replacing the Channel. Support for IChannel's will continue though a Channel-to-Portlet adapter.”

Thus, there is a fudge in that portlets are converted to channels in this version of uPortal. On the surface at least this should not affect using uPortal as a development environment for JSR 168 compliant portlets.

### **i.c Portlet development**

*Introduction to JSR 168—The Java Portlet Specification*

[http://developers.sun.com/prodtech/portalserver/reference/techart/jsr168/pb\\_whitepaper.pdf](http://developers.sun.com/prodtech/portalserver/reference/techart/jsr168/pb_whitepaper.pdf)

Provides a detailed example of portlet construction.

*POST: Portlet Open Source Trading site*

<http://portlet-opensource.sourceforge.net/>

“Separate areas within POST exist for sharing JSR-168 and WSRP portlets. As with any open-source site on SourceForge.net, any registered organization can contribute portlets to POST, which become available to all other members of the open-source community.”

This site could be very useful in seeing how others have built JSR 168 portlets and WSRP portlet descriptions.

*Building JSR 168-Compliant Portlets with Sun Java Studio Enterprise*

<http://developers.sun.com/prodtech/portalserver/reference/techart/portlets.html>

This is proprietary but there is a 90-day free trial.

## ii. Software installation

A test installation of uPortal was undertaken. Downloaded and installed quickstart version of uPortal v. 2.3.1 for Windows. This was one of the easiest installs I have experienced. One minor hiccup uPortal packages need to append stuff onto any existing CLASSPATH variable of users environment. Paths are delimited by a semicolon. Something else installed on the machine has also added paths to CLASSPATH but had not set a final delimiting semicolon (which is normal practice). This crashed the install but was fairly easy to track down.

After starting hsql and tomcat with ant the example uPortal instantiation was accessible on <http://localhost:8080>

**Note** this version of uPortal supports JSR 168 but not WSRP

## iii. Development cycle

The preceding has allowed a development plan to be formulated in terms of skills requirements and tasks. Initial development is going to focus on JSR 168 portlet development.

Work plan

- i) targeted reading (from above)
- ii) develop html code fragment which supports querying of one HEIRPORT target (familiarisation with what variables will need to be passed to the existing Zava (Java API underlying HEIRPORT targets) via existing Zavax portlet)
- iii) install uPortal 2.3.3 (PC initially)
- iv) implement simple example JSR 168 compliant portlets for use in uPortal (familiarisation with technology)
- v) implement JSR 168 portlet supporting the querying of a single HEIRPORT target
- vi) write documentation

Steps i) and iv) are ongoing and the use the NetBeans IDE with Pluto to create portlets is being investigated.

<http://www.netbeans.org/kb/articles/NBAndPortlets.html>

#### iv) Related development

In the wider world of Web Services

*Cross-Domain Resource Discovery Project ("Cheshire")*

‘The JISC/NSF funded “Cheshire” Project<sup>1</sup> has led to the development of a number of recently released toolkits known collectively as Cheshire 3<sup>2</sup> although there appear to be no implementations as yet. Toolkits so far developed are a CQL Parser (Common Query Language), SRW client and SRW server with support for a number metadata formats including DC, MarcXML, MODS, OAI MARC and EAD. They use SOAP or HTTP as communication or carrier protocols. A range of operating systems is supported, suggesting platform independence. The final report of the “Cheshire” project suggests that Cheshire 3 is likely to become a core part of the JISC information architecture.

Because of its newness and lack of existing implementations it is difficult to judge if it will be easier to set up than existing systems such as the Java APIs underlying HEIRPORT<sup>4</sup>. The documentation for Cheshire 3 notes that additional installs are required including ZSI and PyZ3950. It also notes non-trivial tasks dependent on existing set up. It is worth mentioning that other SRW toolkits are also becoming available, for example the YAZ toolkit from Index Data<sup>3</sup>

#### References

1 “Cheshire” Project <http://cheshire.lib.berkeley.edu/FINALDLI.htm>

2 Cheshire 3 <http://srw.cheshire3.org%20/>

3 Index Data <http://www.indexdata.dk/>

4 Austin, T., Pinto, F., Richards, J. and Ryan, N. 2002. 'Joined up writing: an Internet portal for research into the Historic Environment' in G. Burenhult (ed.) Archaeological Informatics: Pushing the Envelope CAA2001, BAR International Series 1016, 243-51.(also online at <http://www.cs.kent.ac.uk/pubs/2001/1261/content.pdf> )

## Appendix C

### Contextual Resource Evaluation Environment (CREE) Project Project Progress Report, instructional media + magic, inc. 16 August 2004

#### Workpackage All

im+m had suggested that it would serve as a contact with other organizations where that is appropriate considering geographical location or history. During this period, this effort included the following:

Z39.50 Maintenance Agency (U.S. Library of Congress) – im+m met with Library of Congress staff concerning the status of the SRW/SRU Web Services specification. The staff confirmed that the *February 2003 1.1 version* was intended as a production version. They said some software has been developed by library system vendors, generally to convert the Web Services-based SRW/SRU into Z39.50 requests. They said the U.S. Library of Congress has a production version that accesses the LOC catalog and a test version. They believed the British Library also had SRW/SRU service available. In the conversation they said Matthew Dovey, a major contributor to the specification, was developing some software. (This was confirmed at the CREE project meeting at the University of York).

JA-SIG uPortal Project – im+m has provided status of the JSR 168 and WSRP development efforts. im+m staff have provided general information to the JA-SIG community as part of their participation in uPortal development, and was available to the CREE partners. im+m continues to monitor and report on progress. Ken Weiner had expected to contribute a complete implementation of WSRP consumer and producer software for uPortal. He has been able to develop some of the consumer code, but because of other priorities has been and will be unable to continue his contribution. He has requested assistance in testing and completing and documenting his work on WSRP consumer, and to independently develop the WSRP producer code.

The Sakai Project – Sakai principals have met with im+m staff several times during this period. Because of the need to have an operational system for the fall term at the University of Michigan (beginning September 7) it has not been possible to develop and implement specifications—the Tools Portability Profile—on the preferred schedule. im+m has provided the CREE project with the status of both plans and development, primarily the details of implementation of JSR 168 and the use of Java Server Faces. There will be another meeting Friday August 20 between im+m CTO Mike Ivanov and Sakai architect Charles Severance.

OASIS Committees – The uPortal developers had maintained a dialog with representatives of the key technical committees developing the WSRP and Internationalization specifications. This dialogue continues.

[U.S.] Postsecondary Electronics Standards Council – im+m developed the agenda for a briefing to higher education on the status of e-Authentication. This brings together the sources of authentication—the SAML committee and Internet 2’s Shibboleth project—and a primary user—JSTOR. im+m will report on the status of these projects and the possibility of achieving consensus on the authorization attributes that would be exchanged. Yale University will be reporting on their project to add Shibboleth to their Central Authentication Services. (im+m is in direct communication with the London School of Economics on the implementation of Shibboleth in uPortal). This communication anticipates the requirement that some search channels, such as JSTOR, will require authorization.

### **Workpackage 1 (Months 1-6)**

A detailed investigation of the potential of JSR1 168 and WSRP for search integration involving Oxford, JAFER team, York (Heirport), Edinburgh (Xgrain/BALSA/Google API’s), Hull, and IM+M.

The search integration software currently available is being rewritten as JSR 168 portlets. Recognizing that the diverse users of these portlets may need different presentations, im+m has taken the Google API and developed a prototype that continues to use XSL style sheets and XSL transformation. Combined with this capability in the portal itself, some disabilities can be accommodated based on XSL/XSLT developed by the University of Hull in a prior project. The Google API is based on SOAP messaging; this also contributes a SRW/SRU implementation. Extension of the XSL/XSLT decision could be used to implement role-based presentations, presentations that were different because of user experience—i.e. beginning and expert, or even user preferences. This prototype can be viewed as a reference implementation for a JSR 168 portlet using the XSL/XSLT technology and a reference implementation for SOAP messaging.

Licensing for the Google API requires each person have a “key” from Google, Inc. and is limited to 1,000 searches each day. CREE Project Director Chris Awre has suggested that an institutional license is possible. The Google API does not carry advertising to the user and, if heavily used, could reduce Google’s potential advertising revenue.

im+m has been following Matthew Dovey’s development of SRW and expects to continue this work consistent with the Google referene design. Mr. Dovey has said he expects to complete his work in the next two weeks and will provide the code to im+m at that time.

The presentation of the portlets that im+m develops will serve as the basis of review and analysis and possibly user testing. As these presentations are refined, im+m can implement the new presentations using XSL/XSLT or these changes could be made by some other CREE partner. Similarly im+m can do similar work implementing designs for portlets be done by the CREE partners who originally developed the software.

When appropriate, im+m can document the reference implementation so the design guidance can be used by others.

Perhaps the work in this project could be used to develop a search capability for a local Website or the learning materials in local Website. im+m has done some preliminary review of the open source Lucene software from the Apache Software Foundation. Jakarta Lucene is described as: "... a high-performance, full-featured text search engine library written entirely in Java. It is a technology suitable for nearly any application that requires full-text search, especially cross-platform." It may be feasible to extend the search portlets to search against Lucene. This implies processing the query as contrasted to presenting the results and likely is outside the scope of this project. The intended use was for the colleges that did not have a search capability on their Website to make available a search portlet identical to those used for remote resources.

im+m is also developing a paper on the use of WSRP (Web Services Remote Portal). This paper briefly describes the purpose of WSRP and shows how it will be used by colleges and universities. After review and revision, im+m hopes this becomes a paper and presentation that can be used to explain when WSRP is appropriate technology and what are the benefits for colleges and universities.