



Project Document Cover Sheet

Project Information			
Project Acronym			
Project Title	PIOP3 – Newcastle		
Start Date	February 2010	End Date	July 2010
Lead Institution	Newcastle University		
Project Manager & contact details	Paul Horner – paul.horner@ncl.ac.uk , 0191 246 4539		
Partner Institutions			
Project Web URL	http://www.eportfolios.ac.uk/Leap2A		
Programme Name (and number)			
Programme Manager	Lisa Gray		

Document Name			
Document Title	Project Report		
Author(s) & project role	Paul Horner, Project Manager		
Date	29 th July 2010	Filename	Newcastle_PIOP3_Report.docx
URL			
Access	<input type="checkbox"/> Project and JISC internal		<input type="checkbox"/> General dissemination

Document History		
Version	Date	Comments
V1	29 th July 2010	

Final Report

Extension & Development of LEAP2A (PIOP 3) – Newcastle University

Overview

Between February and July 2010, Newcastle University were involved in the JISC funded extension and development of Leap2A (PIOP3) Project. Our project aimed to extend the work from the Dynamic Learning Maps project¹ to produce and open and clearly documented web services allow other systems to asynchronously transfer e-portfolio data between systems using Leap2A.

During the project we provided this web service and documented it online at <http://learning-maps.ncl.ac.uk/Leap2A-api/>. We also enhanced our existing support for the Leap2A standard by updating our exports to comply with the 2010-07 version.

Key Deliverables

The key outcomes of this project were:

- Our web service is documented at <http://learning-maps.ncl.ac.uk/Leap2A-api/> (see below also)
- The Leap2A web service is currently being used by students through the Dynamic Learning Maps tool
- We have been in discussions with Nottingham University and Pebble Learning to determine a standard approach to web service discovery and authentication.
- We have contributed to the final 2010-07 version of the LEAP2A standard.
- Our work was demonstrated during the ‘interoperability challenge’ at the EIFE-L ePortfolio 2010 conference in London, and the presentation ‘interoperability in action’ by Paul Horner at the same conference specifically detailed the work conducted in this project.
- Sample data exports are available at <http://www.eportfolios.ac.uk/LEAP2A>. These include a zip file (containing uploaded files) and a static XML document; both are conformant to the 2010-07 version. The XML export is a valid Atom feed² and validates against the Leap2A validator³.
- Our data imports currently support the 2009-03 version, but we hope to move this to 2010-07 at some point (but this was outside of the scope of this project)
- Test accounts for ePET are available at <http://www.eportfolios.ac.uk/eportfolio>. To register for an account, you simply need to provide us with your email address. This demo version contains the same level of documentation provided to most users.

¹ <http://learning-maps.ncl.ac.uk>

² Validated at [http://validator.w3.org/feed-check.cgi?url=http%3A%2F%2Fwww.eportfolios.ac.uk%2FLEAP2A%2FLEAP2A_example.xml](http://validator.w3.org/feed-http%3A%2F%2Fwww.eportfolios.ac.uk%2FLEAP2A%2FLEAP2A_example.xml)

³ Leap2A validator - <http://www.niallbarr.me.uk/Leap2A/xslt/index.php>

- Documentation relating specifically to the ePET implementation has been updated at www.eportfolios.ac.uk/LEAP2A. This includes local category schemes.

The Web Service API

Documentation on the Web Service API is available online at <http://learning-maps.ncl.ac.uk/Leap2A-api/> and in Appendix 1.

The API uses a RESTful interface, so that data is transferred using HTTP only. This means that no complex protocols such as SOAP or XML-RPC are required. The main driver for using this mechanism was to keep the API as simple to implement as possible. The two systems we used for the API are built using Python web frameworks. Dynamic Learning Maps is built using Django, and ePet using Zope. In our experience Python support for HTTP is far superior to its support for other protocols, meaning that ultimately our development time would be reduced by using a RESTful interface.

The HTTP POST used to send ePortfolio data splits the Leap2A record into a number of smaller component parts which are used as parameters to send this data, so that there are parameters for the title, content, categories and type. It also contains a unique identifier (called source) that is used to link the data inside the ePortfolio with the page that created it in Dynamic Learning Maps.

This is a lightweight version of Leap2A and it is not intended to provide the full functionality of the specification. Sending a record that contains two linked elements, for example, would be impossible. We chose this method for simplicity as we did not require any linked elements to be transferred. The next version of this API will replace the content, type and categories with XML markup of a valid Leap2A export containing the elements that we want to send. This would provide a much richer interface, but would make the importing much more complex.

The POST request returns the new ePortfolio record marked up as a Leap2A record. This allows the system that created it to know if it has been added correctly and to find out the system generated ePortfolio ID of that record.

The HTTP GET used to request data from the ePortfolio passes the source (unique identifier of the page) and username. This provides enough context to generate the Leap2A representation of everything created by that learner, linked to that particular source record. If the request is successfully received, the ePortfolio returns a Leap2A record containing details of the records that are attached to the value passed for the source (unique identifier). This Leap2A record conforms to the 2009-03 version but does not include anything that has been added or removed for 2010-07 so could be very easily updated to include the 2010-07 version and namespaces.

Authentication is handled in a less sophisticated way than we would have liked. An MD5 Hash of a string containing two variables passed in the request, alongside a secret that is known by the two systems is passed as a parameter in the request. The ePortfolio can reconstruct this key and will reject the import if the reconstructed key does not match the key included in the request.

At Newcastle we have four different ePortfolio systems, depending on the learner's programme and whether they are undergraduate or postgraduate students. As each programme only uses one of the ePortfolio systems, we can determine the ePortfolio URL relatively easily based on the logged in user's

programme code. To make the API calls as straightforward as possible, the location of each web service is at the same path on each ePortfolio. If other systems were using these APIs, we would have to adopt a more sophisticated approach to this.

Known Issues

Although the web service is currently being used successfully, we are aware of some small issues.

The first issue is that the POST request does not send a Leap2A document. Although this is simplistic and is based on the format of a Leap2A document, by splitting the parameters we have moved the web service a step away from Leap2A as a concept. This currently meets our requirements, but in order to meet the requirements of other users of Leap2A, we are aware that this may need to be rewritten so that it sends XML markup in replace of some of the Leap2A-based parameters.

A second issue is around sending the unique identifier of the page (the source element) to the ePortfolio. This is a necessary element that we have included to ensure that the importing ePortfolio can retrieve records that are attached to that source page. This is not included in Leap2A, and if this web service was ever to be utilised by other Leap2A partners, this may have to be changed and it is unlikely that most ePortfolio providers would be able to store this reference. However, it is included as a simple way of providing the link between the two components of the web service.

Collaboration

Other known issues have been addressed and discussed in collaboration with the University of Nottingham and PebblePad. Unfortunately the requirements of the three projects were quite different, which meant that it was not an appropriate time to implement an identical web service across the three institutions. Ultimately it would be our goal to merge the web services into one commonly used version, and for some of the more complex issues, namely authentication and web service discovery, we discussed a common approach.

Authentication

The issue of authentication was our biggest discussion point. Nottingham and PebblePad have opted for OAuth and have implemented that in their projects. Currently the Newcastle model uses an MD5 hash of a shared key, which is a step towards OAuth, but does not go to the same lengths as the OAuth protocol. Ultimately, for our web service to be used outside of Newcastle we will have to adopt a recognized authentication format, and OAuth would be our preferred mechanism for this.

The standard OAuth mechanism involves a large degree of user input, so that the end user always has to login to their ePortfolio to approve the data entry. Due to the nature of ePortfolios we felt that this was too much and would detract from the seamless integration that we had hoped for. As such we felt that 'authenticate once, authenticated always' was a more appropriate model. So that when a learner had said that a request was legitimate, we could assume that the request would always be legitimate. This would need us to store a token inside the requesting system, which is very similar to the existing (non-OAuth) model currently in place at Newcastle.

Python support for OAuth is relatively stable, and implementing this inside the Django-based Dynamic Learning maps system would be quite straightforward. However, when we have looked at

implementing this, we felt that it was not a trivial task to provide an oAuth layer to the Zope-based ePet system and for the purposes of Dynamic Learning Maps, the authentication method we have put in place is a relatively secure short term success.

Web Service Discovery

If our web service was to be commonly used, we discussed how we would know which systems supported it and the best way to manage this. One option would have been to ask the learner to enter the details of the ePortfolio they wished to connect to. However, we decided this would not be appropriate as it assumed more technical knowledge from the learner than we could reasonably expect. As such we decided that an online repository for compliant web services should be produced.

Effectively this repository could be called by a system to determine the right URL to use for any web service call to another compliant ePortfolio. At the moment, there are no two systems with the same web service, meaning that this repository is not currently required. However, should this become more widely used, Newcastle have volunteered to host this repository in the short term at a unique page at <http://www.eportfolios.ac.uk>. It was felt that ultimately this should be hosted at an independent website, possibly www.leapspecs.org.

Additional Work

Our project plan stipulated that we would concentrate on the web service and would not update our imports or exports. However, during the project we have taken steps to update the ePet export to comply with the latest version of Leap2A (2010-07). We have made sample data exports available at <http://www.eportfolios.ac.uk/LEAP2A>. These samples include a zip file (containing uploaded files) and a static XML document. The XML export is a valid Atom feed and validates against the Leap2A validator. We have also extended the scope of the export to include the new affiliations type. Our data imports have been updated to work slightly better with content from other providers and although it currently support the 2009-03 version, steps have been taken to stop the import from failing when importing newer versions of Leap2A and we hope to update our import routines to support 2010-07 at some point in the near future.

Evaluation of LEAP2A specification

We have a long history of using Leap2A, having been involved from its inception. We have also used many other interoperability standards, and Leap2A remains the most user-friendly and easy to work with specification available.

We do have some issues with some of the new developments in Leap2A, but as these are requirements of other systems and not specifically problems for the ePet implementation, it should be noted that these are ideological rather than technical issues.

The main concern is that the specification is moving away from its Atom-based roots. By definition, a file that conforms to the Atom standard should be recognised as a Leap2A file. However, under some of the recommendations made for 2010-07, this may not be the case. The deprecation of `<content type="media/type">` is of particular concern. Although we agree that this gives a second way of linking to attached content (alongside `<link type="enclosure">`), it is valid Atom, and therefore cannot be deprecated in Leap2A. It has also been suggested that we should all be trying to move

towards `<content type="xhtml" >` (rather than text or html) as this makes things easier for importing. However, many systems allow user input into these areas, and if users are adding their own markup we cannot guarantee that this will be xhtml (and in our experience it is more likely not to be). As such we are quite uncomfortable with any changes or recommendations that have been made that move Leap2A away from Atom.

It is also our opinion that the number of new elements is being less well managed as in previous versions. Previously the policy was to reuse wherever possible, but in 2010-07 that does not always seem to have been the case. Two new elements have been included for a role (the role name and id), and although it was suggested that these could have been merged (with the id as an attribute), the decision was taken to include two elements.

In the project report we produced for PIOP2 we made the recommendation to JISC that

The use of Atom has kept the specification very simple and straightforward, and this ethos of simplicity must be kept in place. The number of predicates must be kept to a minimum and the notion of this being a simple Atom feed cannot be overlooked even if it means that certain aspects of interoperability can't be achieved by the specification.

It is our opinion that this has not happened during PIOP3, and we would hope that going forward this recommendation is not overlooked.

Recommendations to JISC

During PIOP2 we expressed concern about the documentation of Leap2A, and unfortunately the current Leap2A website is still not particularly easy to use, and is no real improvement on the original website on the Cetus Wiki. This is no criticism of Simon Grant who has done an excellent (and unenviable) task of bringing this data together. However, our opinion remains that a wiki is not the correct type of website to describe a specification, and the lack of a real menu makes the site almost impossible to navigate. Each version of the specification should be written up and stored as static content, rather than as wiki pages.

We do feel that this web service is the next step for Leap2A. Conceptually, it will open up the use of ePortfolios to a wider audience as it will allow ePortfolio records to be created from inside other systems and allow ePortfolios to communicate so that a learner can have one interface to numerous portfolio repositories (or multiple interfaces to one repository). The existing web services developed by Newcastle, Nottingham and PebblePad need to be combined and evaluated to provide a usable mechanism for all ePortfolio systems to be able to transfer data without having to export and import huge repositories of data.

Appendix 1 – API Documentation

This information is correct up to 31st July 2010. For current details, please visit <http://learning-maps.ncl.ac.uk/Leap2A-api/>

The API uses a RESTful interface, so that data is transferred using HTTP only. This means that no complex protocols such as SOAP or XML-RPC are required. The main driver for using this mechanism was to keep the API as simple to implement as possible.

API Requests

There are two key parts to the API. The first is sending data from a system (Dynamic Learning Maps) into the ePortfolio (ePet), and the second is the reverse of this – i.e. retrieving data from the ePortfolio.

Learning Maps to ePet

To send information into the ePortfolio you send a simple HTTP POST. This post contains seven parameters –

1. **Username**
The username of the person entering the data. As Dynamic Learning Maps and ePet are both in use at the same institution this will be identical across the two systems. In cases where this is not the case, the username for the importing ePortfolio would need to be stored in the exporting system.
2. **Title**
The title of the entry (the same as Atom:title)
3. **Content**
The content of the entry (Atom:content)
4. **Type (optional)**
The type of entry (RDF:Type). Currently the Dynamic learning maps system only sends entry types.
5. **Category (optional)**
A comma separated list of categories (Atom:category). The category and type tells the importing system where to put the record.
6. **Source (optional)**
A unique identifier for the element inside the exporting system that the ePortfolio record is attached to. This allows the connection to be made between the importing and exporting systems. If not included, the entry is added as a standard entry with no link between the adding system and the ePortfolio
7. **Key**
An authentication key. This is an MD5 Hash of the string `username=USERNAME&title=TITLE&secret=SHAREDSECRET`, where the shared secret is known by the two systems. On import the ePortfolio can reconstruct this key and will reject the import if the reconstructed key does not match the key included in the HTTP POST. In the next version of the API we will be moving to oAuth for authentication, and while this is a step away from oAuth, it does provide security against malicious acts.

This is a very lightweight version of Leap2A and it clearly does not provide the full functionality of Leap2A. Sending a record that contains two linked elements, for example, would be impossible. We chose this method for simplicity as we did not require any linked elements to be transferred.

The next version of this API will replace the content, type and categories with XML markup of a valid Leap2A export containing the elements that we want to send. This would provide a much richer interface, but would make the importing much more complex.

The POST request returns the new ePortfolio record marked up as a Leap2A record. This allows the system that created it to know if it has been added correctly and to find out the system generated ePortfolio ID of that record.

ePet to Learning Maps

The web service request to retrieve data from the ePortfolio uses a HTTP GET request. The post contains three parameters –

1. **Username**
The username of the person who is requesting the data.
2. **Source**
The unique identifier for the element inside the exporting system that the ePortfolio record is attached to.
3. **Key**
An authentication key. This is an MD5 Hash of the string `username=USERNAME&source=SOURCE&secret=SHAREDSECRET`, where the shared secret is known by the two systems. The ePortfolio can reconstruct this key and will return an error page if the reconstructed key does not match the key included in the request. Again, this is a step away from the oAuth that will be available in the next version of the web service, but does provide security against malicious acts.

If the request is successfully received, the ePortfolio returns a Leap2A record containing details of the records that are attached to the value passed for the source (unique identifier). This Leap2A record conforms to the 2009-03 version but does not include anything that has been added or removed for 2010-07 so could be very easily updated to include the 2010-07 version and namespaces.

Web Service Discovery

Also known as where do I send requests to?

At Newcastle we have four different ePortfolio systems, depending on the learner's programme and whether they are undergraduate or postgraduate students. As each programme only uses one of the ePortfolio systems, we can determine the ePortfolio URL relatively easily based on the logged in user's programme code. To make the API calls as straightforward as possible, the location of each web service is at the same path on each ePortfolio. So to send a record to an ePortfolio, you would post to

```
http://PORTFOLIO_URL/eportfolio/Leap2A/send_eportfolio/
```

and to receive records, you send a get request to

```
http://PORTFOLIO_URL/eportfolio/Leap2A/get_eportfolio/
```

passing the required parameters to retrieve the desired records.