Web Services for Reflective Learning – Java SDK

Adam Marshall

Version of 2004-11-15

1 Introduction

We decided to use a similar approach to that employed by the CETIS implementation of the IMS Enterprise service[1].

2 Adopted Approach

A generalised factory was written to allow for easy implementations of the required interfaces.

The SDK comprises two parts:

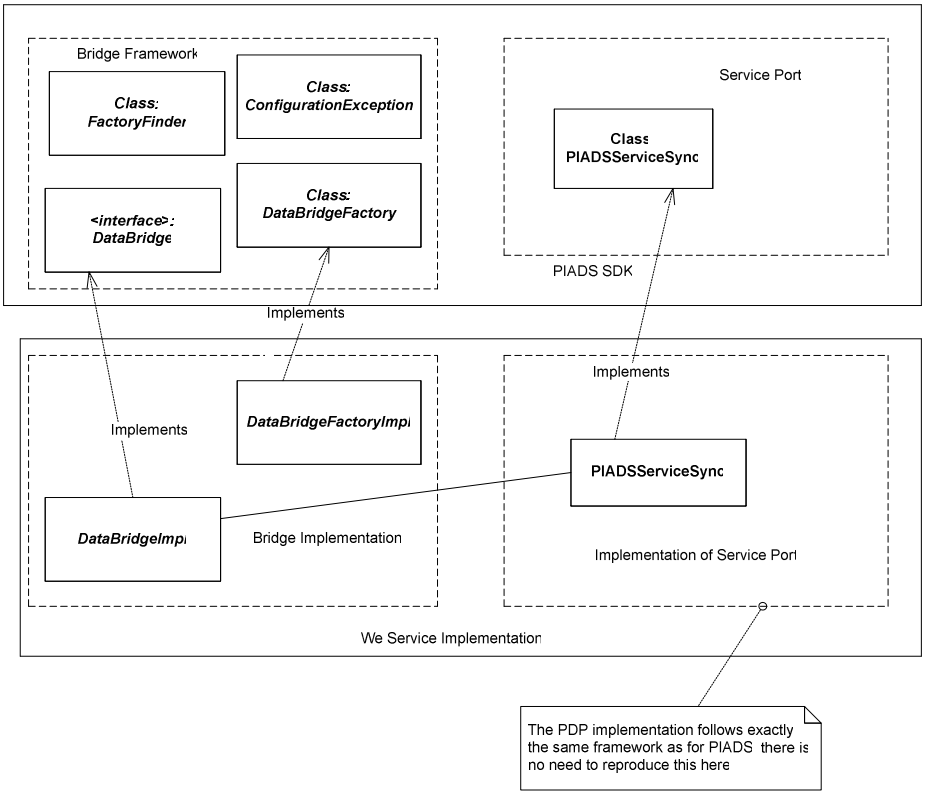
1. The Web Service core (service port)
2. The Data Bridge framework

It is organised in this fashion to make it easy to

1. add extra services to the core service definition
2. use different underlying PDP systems with the framework

The classes contained within the PDP WS SDK are open source and available for download. It is not intended that these be modified for local use, the idea being that the SDK interfaces be implemented in order to instantiate the service.

[1] https://sourceforge.net/projects/cetis-es/



The specific implementation in the above diagram is kept separate from the SDK and is only intended for use with the local PDP or PIADS system. The contents of the classes are detailed below.

3 Implementation Framework

It was deemed appropriate to base the work on the IMS Abstract Framework[2].

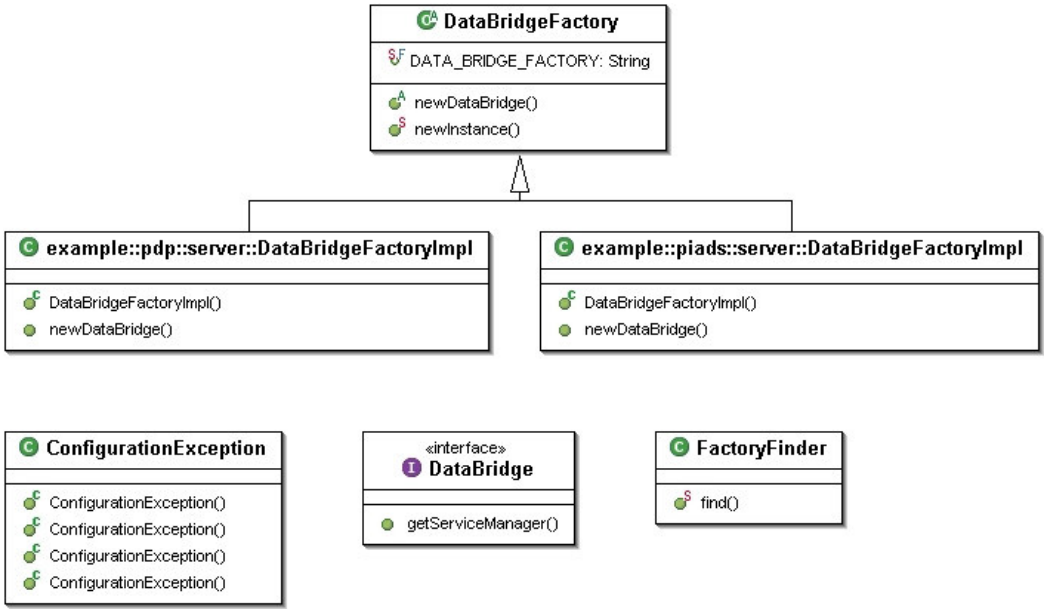
The rationale is that the framework is in the public domain and will be maintained and improved for the foreseeable future. Using this framework gives the work described here a certain amount of robustness and sustainability; it is also expected that developers will become familiar with this framework so should easily be able to use our outputs. We have also used some data types from the enterprise web service schemas (for example, 'Person' and 'UserId') and the common data definitions information model[3] (for example, 'Status' and 'LangString') for the same reasons.

[2] http://www.imsglobal.org/af/index.cfm
[3] http://www.imsglobal.org/es/esv1p0/imscommon_infov1p0.html

4 The Software Development Kit (SDK)

There is an up to date guide to using the SDK on the project SourceForge site[4] which gives a practical demonstration of how to use the `DataBridgeFactory` and associated service definitions.

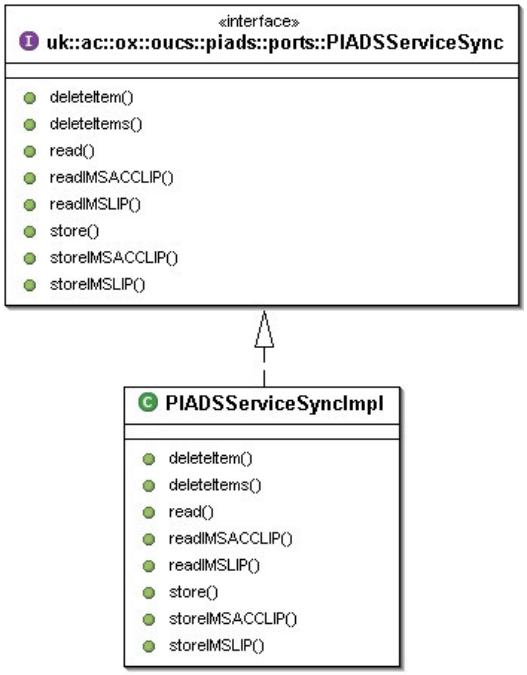
The following figure represents the data bridge and two example implementations (one for each service). The `DataBridgeFactoryImpl` class is written so that `new DataBridge` returns a copy of an implementation of the `DataBridge` interface. This implementation specifies which class implements the particular service port (PDP or PIADS).

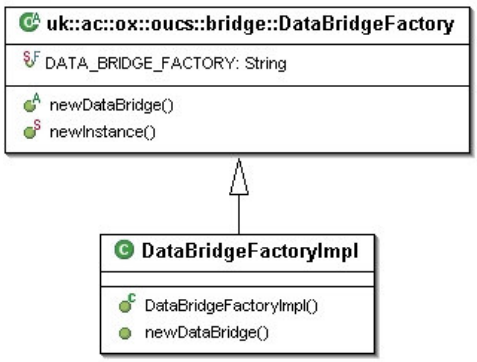


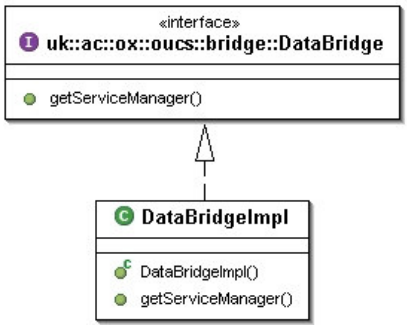
In the case of PIADS, an implementation of the `PIADSServiceSync` interface is written to perform the functions specified in the interface. The SDK contains an example implementation which returns 'unsupported' for each of its methods. This can be used as a basic for a more concrete (and useful) implementation.

The following diagram shows the three classes which need to be implemented to provide a PIADS service.

[4] http://sourceforge.net/projects/pdpws/

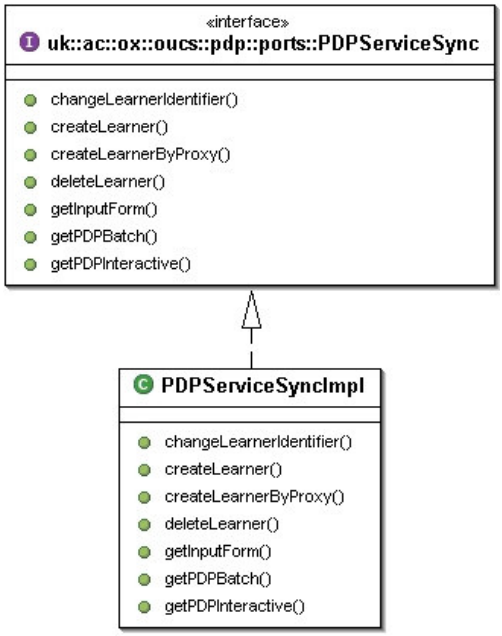


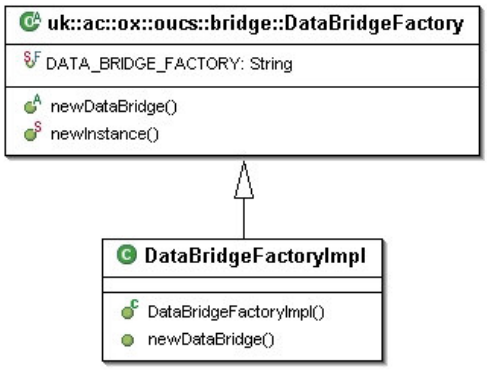


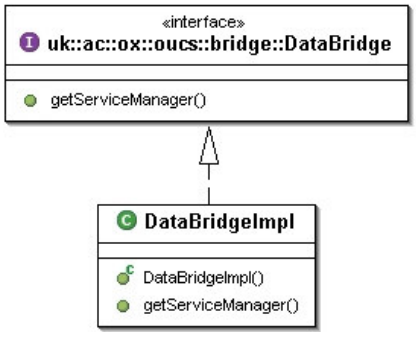


The corresponding classes for the PDP service are given below. There is a distinct similarity in architecture. To implement a PDP web service it is necessary to implement the

`PDPServiceSync` interface from the package `uk.ac.ox.oucs.pdp.ports`. An implementation of the bridge (and factory) will set up the service interface.







The source forge web site contains the following as 'released files'. The contents of the zip files should be obvious.

- `pdpws-piads-sdk-alpha-1.0.zip`
- `pdp-schemas-1.0.zip`
- `piads-schemas-1.0.zip`
- `pdp-javadocs-1.0.zip`
- `piads-javadocs-1.0.zip`

The main SDK contains documentation on how to use the SDK and gives an example implementation. The SDK also includes an Apache Ant[5] `build.xml` file for building the library and deploying the code.

5 PDP and PIADS WSDL and Schema files

The most up to date versions of these are to be found on the SourceForge web site, they are not included here are they are too verbose for inclusion.

The WSDLs employ a number of schemas notably:

- IMS Common schema (`imsCommonSchema_v1p0.xsd`[6])
- IMS Message Binding schema (`imsMessBindSchema_v1p0.xsd`[7])
- UK LIP schema (here we use the modified UK LIP variants which were developed for UK Progress files[8])
- IMS Enterprise person management data types schema (`imsPersonManDataSchema_v1p0.xsd`[9])

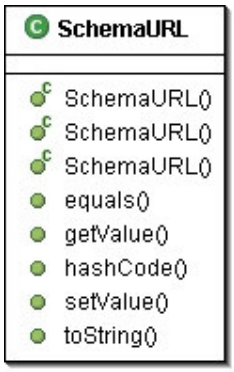
The latest versions of these schemas can be found on the IMS Global consortium web site, the versions used by the web service are on the project SourceForge site. The latest UK LIP schema are to be found on the CETIS web site[10].

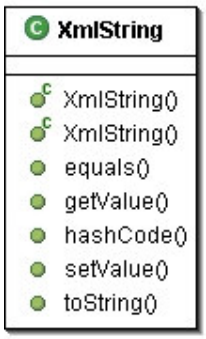
The WSDLs and schemas have been used to generate the PDP and PIADS classes that go to make up the SDK classes. This is done by using AXIS's WSDL2Java utility.

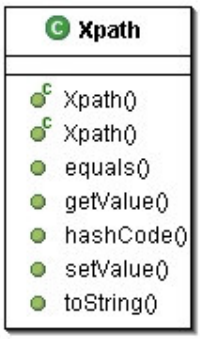
6 PIADS Web Service

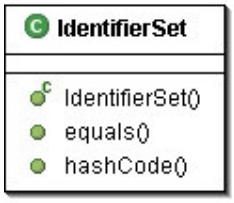
We present the UML of the objects used as parameters to the web service. The usage of these parameters is described after the diagrams.

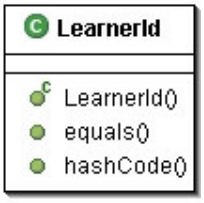
[5] http://ant.apache.org/
[6] http://www.imsglobal.org/services/common/imsCommonSchema_v1p0.xsd
[7] http://www.imsglobal.org/services/common/imsMessBindSchema_v1p0.xsd
[8] http://www.cetis.ac.uk/profiles/uklip/
[9] http://www.imsglobal.org/services/pms/xsd/imsPersonManDataSchema_v1p0.xsd
[10] http://www.cetis.ac.uk/profiles/uklip/

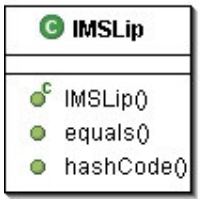


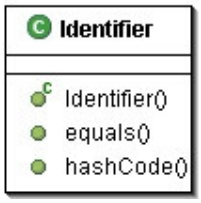












PIADS will manage the storage of records; it only makes sense to require it to implement, create, store and delete operations, behind the scenes the service will issue the relevant CRUD command to the target back-end database. The authentication of messages will be based upon a digital signature found in the SOAP header; if necessary, the message could also be encrypted; it is also understood that HTTPS should be used to transport the message.

There is a lot more information about the PIADS web service in the document "Web Services for Reflective Learning" found on the WS4RL project web site[11].

6.1 PIADS "read"

This will fetch data related to the stated learner (subject to view ability restrictions) from PIADS. It is implied that the PIADS will contact all the learner's data stores and (conceptually) collate all records into one document. This document will be processed in order to only return the specified structures.

6.1.1 Request Parameter(s)

- `Learner ID` – who the info is about
- `RequestingAuthorityId`. – who wants to info
- `LanguageSet` – language preferences
- `Namespace` – of xml format to be returned
- `SchemaLocation` - of xml format to be returned
- `Xpath` – subset of data to be returned

6.1.2 Response Parameter(s)

- `XmlString` – in appropriate format

6.2 PIADS "store"

This will store the submitted data in stores related to the stated learner subject to edit-ability restrictions; PIADS is responsible for ensuring that that the data is stored in the correct place.

6.2.1 Request Parameter(s)

- `Learner ID` – who the info is about
- `RequestingAuthorityId`. – who wants to info
- `Namespace` – of xml format to be returned
- `SchemaLocation` - of xml format to be returned
- `XmlString` - document specifying what information is to be deposited

6.2.2 Response Parameter(s)

- none

[11] http://www.cetis.ac.uk:8080/frameworks/learning_domain_services/personal_development_management/ws4rl/view

6.3 PIADS "deleteItem"

This can also be invoked by the learner or a third party; a third party may have supplied a (read-only) testimonial for the learner.

6.3.1 Request Parameter(s)

- `Learner ID` – who the info is about
- `RequestingAuthorityId`. – who wants to info
- `Namespace` – of xml format to be returned
- `SchemaLocation` - of xml format to be returned
- `Identifier` - ID of element to be deleted.

6.3.2 Response Parameter(s)

- none

6.4 PIADS "deleteItems"

As above but deletes multiple items.

6.4.1 Request Parameter(s)

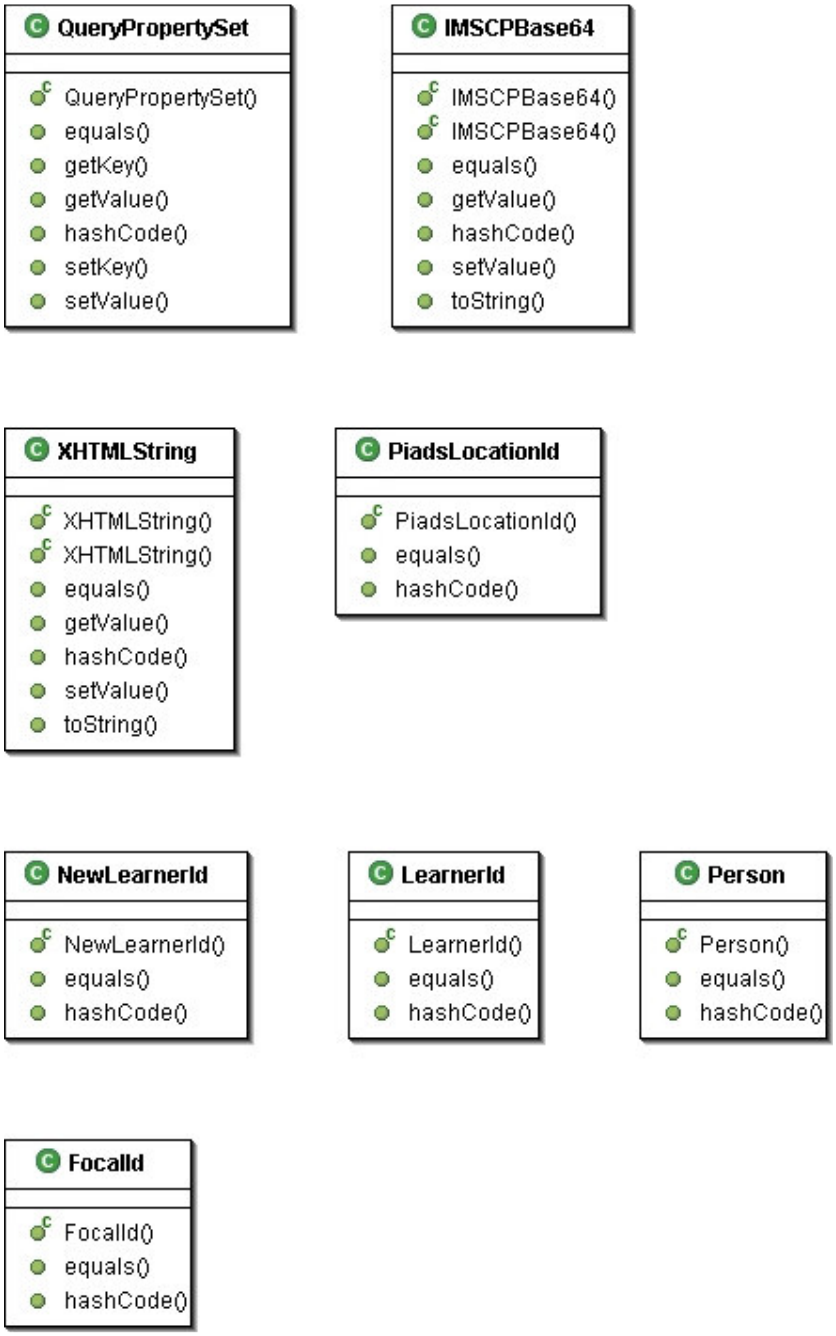
- `Learner` ID – who the info is about
- `RequestingAuthorityId`. – who wants to info
- `Namespace` – of xml format to be returned
- `SchemaLocation` - of xml format to be returned
- `IdentifierSet` - IDs of element to be deleted.

6.4.2 Response Parameter(s)

- none

7 PDP Web Service

We present the UML of the objects used as parameters to the web service. The usage of these parameters is described after the diagrams.



7.1 PDPWS calls

The service should include a facility to create and delete and manage accounts on the target PDP system (for example, "`createLearner`", "`deleteLearner`"). There is also a search service for the learning designer, plus services to return specified PDP processes (as IMS Learning Design and xHTML / xForms).

7.1.1 PDPWS "createLearnerByProxy"

This will be carried out in a similar fashion to that employed by the Person management facilities of the Enterprise Services model[12]. Our process is slightly different as we will be creating a learner account and will need to return account details to the client.

7.1.2 Request Parameter(s)

- `Person` – demographics of learner

7.1.3 Response Parameter(s)

- `LearnerId` – username (and password etc)

7.2 PDPWS "createLearner"

This is very similar to createLearnerByProxy except that here the UserId is supplied by the caller. See above.

7.2.1 Request Parameter(s)

- `LearnerId` - includes username (and password)
- `Person` – demographics of learner

7.2.2 Response Parameter(s)

- <none>

7.3 PDPWS "deleteLearner"

This is very similar to createLearner except in reverse!

7.3.1 Request Parameter(s)

- `LearnerId` - includes username (and password)

7.3.2 Response Parameter(s)

- <none>

7.4 PDPWS "changeLearnerIdentifier"

This is similar to createLearner except that here and existing (source) UserId is replaced by the (target) UserId supplied by the caller..

7.4.1 Request Parameter(s)

- `LearnerId` - includes username (and password)

[12] http://www.imsglobal.org/es/index.cfm

- `NewLearnerId` - includes username (and password)

7.4.2 Response Parameter(s)

- <none>

7.5 PDPWS "search service" (invoked by learning designer)

What is needed is a service to show learning designers the map of all possible "positions" relevant to that PDP service, along with some explanation of what each one is designed to be for. These may be structured hierarchically, or not. The learning designer needs to be able to select any permitted one. The search invocation will be relatively simple: something along the lines of "show me what PDP you offer".

In contrast to the learner-originated call described later, the designer-originated one will return the complete map of PDP "positions" for that PDPWS, along with metadata describing those positions. As the designers requesting this information are likely to be familiar with LOM, the obvious choice is to treat each PDP position as a learning object, and to base the metadata set to use in this context on the UK LOM Core (http://www.cetis.ac.uk/profiles/uklomcore). The UK LOM Core information model, along with many other similar metadata models, has a section (7, "Relation") dealing with relationships between leaning objects. This can be used to define the hierarchical structure, if appropriate for a particular PDPWS, using the Dublin Core relations "ispartof" and "haspart".

In summary, we are saying that a PDP system must support some sort of simple search service – the interface for this will be defined as part of the ELF common services layer (see http://www.cetis.ac.uk:8080/frameworks/common_services/search).

7.6 PDPWS "getPDPInteractive" (invoked by learner)

7.6.1 Request Parameter(s)

- `LearnerId`
- `PiadsLocationId` - URI
- `FocalId` – focus 'record'.
- `QueryPropertySet` - set of name-value pairs of other parameters relevant to that PDPWS, for instance: position in PDPWS (default: start); focal time or time span.

The focal record could be used by any PDPWS but there are no other common parameters. This is why this one has been abstracted from the property list (see below) and is an argument in its own right.

Initially, no focal record or other parameters need to be supplied, and the PDPWS returns a default menu for that learner, as designed and determined by the designers of the PDPWS. As discussed above, that default menu may have one or more of: a search box in which to enter keywords; the top level of a hierarchical menu system; or a fully-displayed structure of nested

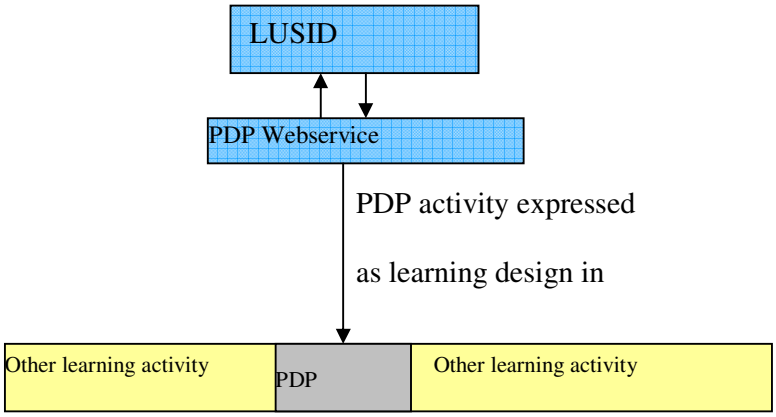
lists to choose from. Later parameters are provided by the PDPWS itself, during the course of usage.

A similar name-value pair approach is used by the OKI's OSIDs – here the caller and the WS must have an understanding of what the parameters are doing – this is a so called 'out of band' agreement[13]. It is necessary for both the consumer and provider to understand that they need to communicate specific proprietary information in order for the service to work correctly. Even though this does little for interoperability, we feel it is the only solution as all PDP systems will be diverse and require different information to work correctly.

7.6.2 Response Parameter(s)

- `IMSCPBase64` - ZIP file IMS Content Package containing IMS LD

The primary advantage of delivering PDP activities (Reflection, Action planning and Skills audit) as Learning Design via a web service would be so that educators could incorporate the activity directly into their learning activity flows and, for the more complex PDP activities, adapt and customise the activity to suit local conditions. For example they might want to employ a different skills framework suited to the subject area.



Unit of Learning created by remote learning designer

Note that the call to the web service could be made at two points: design time (of the learning activities containing PDP as part) or at run time in some kind of LD player. The call might also be made by an individual outside of any course demands (intrinsically motivated) from a Personal Learning Environment (PLE) for example. There are issues about digital rights to be addressed, perhaps most difficult in the design-time situation: How many copies, how many student uses, to what extent can changes be made?

[13] http://sakaiproject.org/faqs.html#What_are_the_Levels

7.7 PDPWS "getPDPBatch" (invoked by learner)

Almost exactly the same as above but for PLEs – this implies ALL data and data collection forms must be included in Content Package.

7.7.1 Request Parameter(s)

- See `getPDPInteractive` above.

7.7.2 Response Parameter(s)

- See `getPDPInteractive` above.

7.7.2.1 PDPWS "getInputForm"

This will be invoked from within the LD returned by the "getPDPInteractive" service call. To construct the PDR input form, the PDP system will have to retrieve the form template and then contact the learner's PIADS to retrieve the appropriate data which will be folded into the form before it is despatched to the user.

7.7.2.1.1 Request Parameter(s)

- `LearnerId`
- `PiadsLocationId` - URI
- `FocalId` – name of input form
- `QueryPropertySet` - set of name-value pairs of other parameters relevant to that PDPWS, for instance: position in PDPWS (default: start); focal time or time span

7.7.2.1.2 Response Parameter(s)

- `XHTMLString`

8 Development issues

8.1 LUSID

LUSID[14] was used as the web service server. There were a number of issues surrounding this. First of all a port to MS SQLServer had to be performed. The installation script for LUSID was broken. Once fixed, the DB port was fairly trivial. LUISID was built using the Unix `make` utility so this had to be replaced by an Apache Ant equivalent.

When obtained, LUSID was able to import an IMS LIP compliant data file, however, the vocabularies that were used by LUSID have now been modified[15] so this area also needed updating.

[14] http://lusid.liv.ac.uk/
[15] http://www.cetis.ac.uk/profiles/uklip/documentation

The decision was also made to use the new experimental UK LIP schemas (see below) so the XSLT template which LUSID uses was also updated. The rationale for this is that LUSID is a UK based PDP system and the UK LIP schemas were designed specifically for use with the UK Progress File; a large aspect of PDP is concerned with reflection and the IMS LIP version do not support the `<reflexion>` or `<assertion>` elements.

All these modification will be fed back to the LUSID team at Liverpool and will eventually appear in the open source version of the software when this appears in early 2005.

8.1.1 Work needed to provide a fully functioning PDP / PIADS web service

Putting a fully functioning PDPWS and PIADS service in front of LUSID would require a sizable amount of reprogramming of the system.

LUSID, at the moment, has aspects that behave like a VLE, aspects that behave like an e-portfolio system and aspects that are a PDP system. These different facets need factoring out – the VLE and e-portfolio features would be best served by dedicated systems, so the part of LUSID which is of interest here is the part dealing with PDP facilities.

The PDP facilities are defined almost solely by XML files: the skills framework[16], the field definitions (and their relationship to the UK Progress file LIP application profile), the guidance and explanations and, most of all, the page templates. There is also an area of the system that analyses user (LIP) data and presents guidance to the learner; this is a fairly small and underdeveloped part of the system. Lots more useful work could be done in this area.

For LUSID to be usable as a web service, a new processing engine would have to be produced to render the LUSID page templates as XForms plus IMS LIP model (or maybe to IMS QTI). (They are currently rendered in HTML 4.0.) Every LUSID page template would have to be modified to separate navigation from content. (At the moment, LUSID pages contain data entry input boxes plus a choice of up to 8 different 'save and move on' options - each with a different target page. The modification would be to abstract all the navigation to a separate navigation-only intermediate page - fairly trivial to do, but LUSID has about 200 pages - this would take a long time.)

In addition, somebody would have to author IMS Learning Design for the PDP processes that the service would return. (These would map onto the current LUSID workflow which is dictated by the aforementioned navigation.) As we have discovered in this project (see below), it is very difficult and time consuming to author LD ‘by hand’; also, a suitable LD player would have to be obtained, integrated with the VLE or PLE invoking the web services.

[16] A new skills framework based on IMS RDCEO and IMS VDEX is being developed as part of the JISC-funded Distributed e-learning strand project called SPWS (see http://www.jisc.ac.uk/index.cfm?name=deletspws). If time permits, LUSID will be modified to use this framework instead of the proprietary framework it currently uses.

LUSID would then have to be enhanced to produce IMS Content Packages from the above components.

The PIADS service will also require a lot of work. Moreover, it is not apparent weather this would be a sensible thing to do. A better solution would be to partition LUSID and hive off the data storage aspects from the PDP aspects so that the PIADS software is a separate stand-alone system which could be used by other systems such as e-portfolios. We are really talking about a major re-engineering of LUSID to support the web services. LUSID 3 perhaps?

As alluded to above, another major part of any PDP web service would be some sort of intelligent 'rule-based' system which could advise a learner what they should do next based on the learner's life-long learner record. LUSID does this in a very small way and would need to be greatly enhanced. This is very non-trivial and would make an excellent project in its own right.

In any case, as highlighted elsewhere, both IMS LD and Xforms need to support web service calls before any of the above will work. We are a fair way off a functioning PDP web service.

8.2 Axis

Developing with AXIS[17] has greatly simplified the process of deploying web services although the lack of documentation has been a problem. Towards the end of the project a book entitled Building Web Services with Java (Sams Publishing - ISBN 0-672-32641-8) was published which outlines the basics.

Axis has been used for

1. implementing the SOAP messaging
2. generating the java code (ports, types, messages etc) for inclusion in the SDK
3. effecting deployment (AdminClient)
4. monitoring SOAP traffic (SOAPMonitor)

AXIS has served us pretty well (apart from the odd problem detailed below), however, we have found the error messages to be particularly unenlightening.

8.3 Eclipse IDE

Eclipse[18] is an excellent environment for developing web services. Eclipse 3.x should be used as there are problems with memory leakage with Eclipse 2.1 when used with SOAP.

We used the Omondo (Eclipse UML) tools[19] for generating UML diagrams contained in this document – the free version supports class diagrams and works with both eclipse 2 and eclipse 3. The plug in seemed to work much better with the earlier version of eclipse – on many occasions with eclipse 3, totally blank 'diagrams' were produced with no explanation as to what had happened to the contents! (It could have been that I was doing something wrong

[17] http://ws.apache.org/axis/
[18] http://www.eclipse.org/
[19] http://www.omondo.com/

although doing the same thing with Eclipse 2 produced the diagrams which are rendered above!)

We had a bit of a problem with updating the CVS repository on Source Forge, we never really got to the bottom of the problem but the problem could have been caused by Eclipse 3's CVS client. In the end we used CVS from the command line (gcvs) and this worked fine. The symptoms were that everything seemed to work perfectly but the CVS repository was not actually updated!

8.4 Open 'Standards'

We have used a number of open standards at different points in the project.

8.4.1 UK LIP schema

These schemas were incorrect and out of date when the project started, we have corrected these errors and the new versions are available on the CETIS web site[20]. As noted on the site, these are experimental schemas.

The original IMS LIP 1.0 schema was first modified by Scott Wilson to make it work. This was also altered to include the new element "reflexion" and to allow dates and times properly. As part of the work for BS 8788 for BSI, Rob Tice created a new schema which included the old IMS LIP 1.0 schema, redefining things that had been clarified or changed as part of the BS 8788 drafting. The "reflexion" element was joined by "assertion".

What we have done is to look over the changes implied by the BS 8788 draft, and incorporated them (when they seemed important) not into a single modifications file redefining the IMS LIP 1.0 work (as did Rob Tice) but instead incorporating selected changes into original files themselves. As part of this, it made sense to change all the file names from ims_lip_... to uk_lip_...

We have also enhanced the UK Leap vocabularies[21] to take into account more reflective relationships between records.

8.4.2 IMS Enterprise

There is a bug in the new `imsCommonSchema_v1p0.xsd`[22] which prevents it form being used; this has been fixed and CETIS notified of the problem. To date no official response has been forthcoming and the IMS website still contains the faulty schema. A corrected version is available from the project's Source Forge site[23] - the bug concerns the definition of the `IMSextensionAnyDType`.

[20] http://www.cetis.ac.uk/profiles/uklip/schemas
[21] http://www.cetis.ac.uk/profiles/uklip/documentation/UKDLIPvocabs.xls
[22] http://www.imsglobal.org/services/common/imsCommonSchema_v1p0.xsd
[23] http://sourceforge.net/projects/pdpws/

8.4.3 IMS ACCLIP

We had originally intended to use AXIS (JAXB) to generate Java binding for this accessibility specification but there is a bug either in the spec or in AXIS which prevented this form happening. AXIS was sent into an infinite loop when trying to process the schema. In the end we achieved our goal in a different way.

8.4.4 IMS Learning Design

We had a lot of difficulty in writing IMS Learning Design to represent PDP processes. This situation is clearly temporary and is currently being addressed as more and more tools are being developed. We had to resort to doing the coding 'by hand'.

Current XML tools such as XMLSpy aren't able to support sophisticated schema constructs such as Include, Import and Redefine. Colin Tattersall of the Open University of The Netherlands has made available the "IMS Learning Design author schemas" which are re-implementations of the schema avoiding the troublesome constructs. This is very useful, but the instance XML documents need the original schema locations to run correctly in an LD player such as GoldLeaf, at which point they fail to validate in XMLSpy. Similar problems arise with the UK LIP specification. The XFORMation application is useful for creating xforms and can work directly from a schema to allow an author to construct a forms model that can produce an xml instance of the schema as the form output. Unfortunately the feature is not able to handle the UK LIP specification at present, so XFORM generation requires hand-coding.

Generally, the practical problems we had were to do with the schemas being too sophisticated for the tools and the early stage of LD player development for testing the stuff.

Any Learning Design that was created was passed to the SLEDGE[24] project for verification – the feedback given by them was invaluable (if not slightly confusing!). SLEDGE is based around improving the CopperCore engine[25] to make it more usable and robust. CopperCore gives validation errors about our "property-value" nodes, but the LD best practice guide and their examples seem indicate that we are doing it right. The latest content packages are available on the project web site[26].

We have also identified the need for LD to support embedded web service calls in the future.

IMS Content Packaging.

[24] http://www.jisc.ac.uk/index.cfm?name=sblds
[25] http://coppercore.sourceforge.net/
[26] http://www.cetis.ac.uk:8080/frameworks/learning_domain_services/personal_development_management/ws4rl/plan-an-essay.zip/view

8.4.5 XForms

We have identified the need for XForms to support web service calls, we understand that v1.1 of the specification will contain this functionality.