
Technical Approach

HEI Course Database Web Services

External

VERSION:	2.0
LAST UPDATED:	April 2008

0.1 TABLE OF CONTENTS

0.1	TABLE OF CONTENTS	2
1	DOCUMENT OVERVIEW	3
1.1	INTRODUCTION	3
1.2	PURPOSE OF DOCUMENT	3
1.3	SCOPE.....	3
2	WEB SERVICE FUNCTIONS.....	4
2.1	TECHNOLOGY	4
2.1.1	Data Transmission.....	4
2.2	WEB SERVICE IMPLEMENTATION	4
2.3	SECURITY & AUTHENTICATION.....	4
2.4	WEB SERVICE END POINTS	5
2.5	DATA PROCESSING.....	6
2.5.1	getCourseCatalog	6
2.5.2	submitCourseCatalog	6
2.5.3	getCourse.....	8
2.5.4	getLocation.....	8
2.5.5	getInstitutionYear.....	8
2.5.6	getQualificationsList	8
2.5.7	General - Result Elements.....	8
2.5.8	Web Service Versioning.....	9
2.6	WSDL SPECIFICATION	9
2.7	DEPLOYMENT	9
2.8	PROCESSING	10
2.8.1	Parsing XML.....	10
2.8.2	Authentication	10
2.8.3	Authorisation.....	11
2.8.4	Data Validation	13
2.8.5	Optimistic Locking.....	13
2.8.6	Validation Exceptions Logging.....	14
2.8.7	Web Service Usage Logging.....	14
2.9	EXTERNAL TESTING OF WEB SERVICE	14
	APPENDIX A – EXAMPLE INTERACTION	15
	APPENDIX B – PRIMARY AND SECONDARY VALIDATION.....	16

1 Document Overview

1.1 Introduction

The Customer First Programme will deliver a new Student Finance Service for students starting from academic year 2009/10, implementing a customer focused and e-enabled student service vision. The HEI services work stream is a key enabling function of the Customer First Programme for ensuring that HEI-provided data required by the Student Finance Service is accurate, up-to-date and available at the correct time. Along with the benefits to the student's service design, HEI's should benefit from improved speed, accuracy and ease of use of SLC's interfaces.

The first of these components that SLC wish to implement is changes to the HEI Course Database. SLC is changing the way it receives course information from HEIs. The process is being migrated from legacy CD-ROM format to a new Web Service technology.

The change will enable simpler and faster communication of course information between the HEIs and SLC. A web-service will benefit HEIs because data can be passed over quicker and easier, and any validation issues will be identified earlier. Subsequent changes and additions to course information will also be easier to provide on an ad-hoc basis.

1.2 Purpose of document

This document details the technical solution for implementing the changes.

1.3 Scope

The scope of this document is limited to identify the technical solution to implement the HEI Courses Web Services.

2 Web Service Functions

The SOAP-based Web Services are to allow any HEI to update or insert course-related data into the SLC's HEI Courses System.

2.1 Technology

2.1.1 Data Transmission

SLC will use SOAP based JAX-RPC web services to facilitate data transfer between client machines and SLC systems. The format of the messages will be in SLC's bespoke course XML schema.

2.2 Web Service Implementation

SLC is using Apache Axis2 <http://ws.apache.org/axis2/index.html> to implement the SOAP messaging system. However the technology used is entirely at the discretion of all 3rd parties.

2.3 Security & Authentication

All transmissions between SLC and clients will be in HTTPS. This will be enforced from the content switch by SLC infrastructure.

Authentication will follow the WS Security standards, specifically the UsernameToken profile, and will be implemented in a manner consistent with the current HEI Services Portal security system.

2.4 Web Service End Points

The Web Service will provide six points of contact with SLC systems; these are known as Web Service End Points.

1. HEI Course Catalog download. This web service end point will provide details of HEI courses, in the format of the SLC courses XML Schema. Upon invocation, and authentication, the JAX-RPC servlet endpoint will produce a SOAP message response containing the data for all of an HEI's course details as they are currently held within SLC.

This end point will be used by clients to initially each year and potentially a number of times thereafter to synchronise the data at their end.

2. HEI Data Submissions. This web service end point will accept incoming data submissions for details which have been changed by the client application system. As per the XML Schema, a *catalog* and *institution* will always be present. There is no need for clients to supply all details for all courses during each submission; just details of courses which have been changed.

The data submission may consist of a number of inserts and updates to course data.

3. Fetch individual Course details. This endpoint will provide details of a single course. This endpoint may be used by a client to resynchronize their local application database if an update fails because the course being updated has been changed on SLC's database since the HEI downloaded it (possibly by the HEI via the HEI Portal). For a discussion of this, please see section 2.5.2.10.
4. Fetch individual Location details. This endpoint will provide details of a single location. This endpoint may be used by a client to resynchronize their local application database if an update fails because the location being updated has been changed on SLC's database since the HEI downloaded it (possibly by the HEI via the HEI Portal). For a discussion of this, please see section 2.5.2.10.
5. Fetch individual Institution Year details. This endpoint will provide details of a single institution year. This endpoint may be used by a client to resynchronize their local application database if an update fails because the institution year being updated has been changed on SLC's database since the HEI downloaded it (possibly by the HEI via the HEI Portal). For a discussion of this, please see section 2.5.2.10.
6. Fetch list of qualifications. This endpoint will provide details of all the currently active qualifications. It can be used to validate the *qualification* element of a *course*.

2.5 Data Processing

2.5.1 *getCourseCatalog*

Each year, SLC staff will run a data population process to create the course data for the forthcoming academic year based on the current academic year's live data. Certain data items such as the course term dates will be nullified to prevent accidental non-update of the term dates in the data for the new academic year. This data will be created in the SLC Courses Staging Area henceforth called the *staging database*.

When a client requests the course data for a specific HEI code and academic year, the web service will check to see if that data is already available in the staging database.

- If not, an error will be returned to that effect. The most likely reason is that the client HEI has requested the data too soon, before the data population process has been run, or perhaps there is a problem.
- If so, the data that currently exists in the staging database, including any updates made by the HEI since the data was copied from the previous year's data, will be returned to the web service client.

The data will be returned as a *catalog* element which includes course, location and institution year data for the HEI code specified.

2.5.2 *submitCourseCatalog*

2.5.2.1 *Submission catalog and response catalog*

The *catalog* element sent to SLC to update course details can consist of course data updates for one or more HEI's. The web service will check that the user authentication details supplied are valid. If not a general authentication error will be returned.

If the authentication details are valid, they will still be checked to ensure that they are valid for updating the details of each HEI specified in the *catalog*. If not, an HEI-specific authentication error will be returned and all of the submissions for the invalid HEI will not be performed.

If the authentication details are valid for a particular HEI in the catalog, then each of the submissions for that HEI will be applied individually within its own success unit. I.e. if an individual insert or update transaction fails, only that transaction will fail and the other inserts and updates for that HEI will still be attempted.

The *courseCatalogSubmissionRequest* element contains an *academicYear* element which indicates which academic year the submission is meant to be for. The schema definition allows data for multiple academic years to be present in the catalog. However, within the context of the data submission, only data for this global academic year will be considered to be significant. ***Data for any other academic years will simply be ignored.*** No errors will be reported against such data.

The web service will return a response catalog (*catalogSubmissionResponse*) that shows the result of each insert or update. It will also show general and specific authentication failures. The response catalog will be structured similarly to the submission catalog. However, no assumption should be made that the order of individual HEI details or the individual inserts and updates within those details will be the same as the order of data in the submission catalog.

2.5.2.2 *Inserts vs. Updates*

For each type of submitted element – *course*, *institutionYear* and *course location* – a specific method will be used to determine if the submitted element is an insertion of a new element or an update to an existing one.

2.5.2.2.1 *course*

For *course* elements the distinguishing value for an update is the SLC course code - *slcCourseCode*. This value must correspond to a course at the HEI being updated otherwise a validation error will occur. If the SLC course code is null (or not provided), then this course will be assumed to be an insert.

2.5.2.2.2 *institutionYear*

For *institutionYear* elements the distinguishing feature is whether the *academicYear* value specified in the *institutionYear* already exists for the HEI to which it refers. If so, it will be an update. Otherwise it will be an insert.

2.5.2.2.3 *location*

For *location* elements, the distinguishing feature will be whether the combination of *campus* and *franchise* already exist for the HEI to which it refers. If so, it will be an update. Otherwise it will be an insert.

2.5.2.3 Course Updates

Each course update should contain all of the attributes of the course and will completely overwrite the values currently held in the staging database for that course. This includes any course years and terms within that course. I.e. if the update does not contain a course year or term that previously did exist in the staging database then that course year or term will be removed during the update. For each course update, a *courseSubmissionResponse* element will be included showing the *slcCourseCode* and *shortName* of the course and whether the update was successful.

2.5.2.4 Course Inserts

An SLC course code will be assigned automatically during the data insertion. This assigned value will be echoed back to the client in the *courseSubmissionResponse* along with the *shortName* of the course to allow the client application to store the newly assigned SLC course code with the course in its own database.

2.5.2.5 InstitutionYear Updates

An update to an institution year will completely overwrite any values previously held in the staging database for the specified institution year. An *institutionYearSubmissionResponse* element will be included in the *institutionSubmissionResponse* element indicating the success or otherwise of the update.

2.5.2.6 InstitutionYear Inserts

It is not possible for an HEI to insert an InstitutionYear entity. These will be set up by SLC along with the rest of the data for a new academic year.

2.5.2.7 Location Updates and Inserts

It is an integrity constraint of the course setup that only one course location is the primary location – i.e. has the *primary* element set to *Y*. This constraint will be checked after each insert / update of course locations and if not true the insert or update will fail. The primary location must also be active. Therefore, when changing the primary location, the update should be done by first marking the current primary location as not primary, then updating or inserting the new primary location.

2.5.2.8 Location Updates

An update to a location will completely overwrite any values previously held in the staging database for the specified location. A *locationSubmissionResponse* element will be included in the *institutionSubmissionResponse* element indicating the success or otherwise of the update.

Please note that although the *primary* element of a *location* is mandatory, it is not possible to change the value of this indicator on submission of an update to a location. The primary location is fixed when a new HEI is set up in SLC's core systems and cannot be changed. Thus, it is also not possible to submit a new location that has the primary indicator set to true as this would require the existing primary location to be updated to primary = false first.

2.5.2.9 Location Inserts

A successful insert of a location will create a new record in the staging database. A *locationSubmissionResponse* element will be included in the *institutionResponse* element indicating the success or otherwise of the insertion.

As already indicated in 2.5.2.8 above, it is not possible to insert a location that has the *primary* element set to true.

2.5.2.10 **Attempts to update out-of-date data**

Each significant entity in the data downloaded – *course*, *institutionYear* and *location* – has a timestamp (*lastUpdatedDateTime*) indicating when it was last updated in the staging database. These values should be passed back to the **submitCourseCatalog** web service endpoint when updates are being sent in. The timestamps are used by the web service to check that no other update has occurred to the staging data since the course catalog was downloaded by the HEI that would be overwritten by this submission.

In the event that an attempt is made to perform an update based on an out-of-date copy of an entity, then the web service will return an error message for that specific update and the update will not take place. The client may then use one of the other *get~* end points to fetch an up-to-date version of the entity to re-synchronize the client application and decide whether the update should be re-sent.

After a successful update or insertion, the timestamp will be updated in the staging database and the new timestamp passed back in the **result** element for each insert / update. The client application may then store each of these timestamps against their corresponding data so that subsequent updates can be performed. Alternatively, the course data can be re-fetched using either the **getCourseCatalog** or the individual fetch web service endpoints prior to performing further local updates and re-sending the changes.

When sending inserts, obviously there is no existing data timestamp so the *lastUpdatedDateTime* element can be omitted.

2.5.3 **getCourse**

This endpoint will return a single *course* element. As well as authentication, it requires the 4-character *heiCode* and the SLC course code (see section 2.5.2.2.1). It returns a single *course* element if one is found, otherwise it returns a **result** with an appropriate error.

2.5.4 **getLocation**

This endpoint will return a single *location* element. As well as authentication, it requires the 4-character *heiCode*, the *campus* identifier and the *franchise* flag. It returns a single location element if one is found, otherwise it returns a **result** with an appropriate error.

2.5.5 **getInstitutionYear**

This endpoint will return a single *institutionYear* element. As well as authentication, it requires the 4-character *heiCode* and the *academicYear* identifier. It returns a single *institutionYear* element if one is found, otherwise it returns a **result** with an appropriate error.

2.5.6 **getQualificationsList**

This endpoint will return a list of currently active qualifications that can be used in the *qualification* element of a course. Each qualification will consist of a code and description. As can be seen in the schema, it is the code that is referenced in the *course* element's *qualification* element. It is expected that this endpoint will be used in the following way.

During submission of a course catalog, the course qualification will be validated by SLC against the list of currently active qualifications. If it does not match any of them, the course update or insert will be accepted, but a task will be created to resolve the invalid qualification code. This task will involve discussion between SLC and the HEI to decide whether the invalid qualification code should be added to the list of qualifications, or changed to an existing one. The endpoint can be used by the HEI to all a check for potential alternative qualification codes, or even for validation within an HEI's own course maintenance application.

2.5.7 **General - Result Elements**

Each *~Response* type element contains a **result** element that shows whether the insert or update was successful or not. This includes the web service end point responses (**getCourseCatalogResponse**, etc. and the individual

courseResponse, **institutionYearResponse** or **locationResponse** in the **responseCatalog** of the **submitCourseCatalogResponse**.

If a request is entirely successful, the **resultNumber** will be 0. If the request is a **submitCourseCatalog** request and not all of the inserts or updates included in the submitted **catalog** were successful, then the **resultNumber** will be -1.

If a request is not successful, either partially or wholly, the **result** element will contain a result number, a result message and, in the case of successful inserts and updates, a **lastUpdatedDateTime** for the entity just inserted or updated.

A preliminary list of response numbers and messages is shown in section 2.8.3. Note that these are provisional.

2.5.8 Web Service Versioning

In order to ensure that clients of the web service are up to date with the latest version of the service, the request and response elements include a mandatory version element. The value of this element in submitted requests will be validated by the web service and any submitted request that does not provide a correct version number will be rejected as invalid XML. Similarly, each response will contain a version element that will allow the client to verify that the response message is of the form that they are expecting.

Each request and response version may be varied separately, though it is likely that for consistency, the version numbers of request / response pairs will be kept the same. Note that the version numbers will only be changed in the event that

- i) a change is made to the structure or meaning of the SOAP messages (structural changes would almost certainly fail validation anyway), or
- ii) a change is made to the protocol implemented by the web service and associated clients that would render previous clients invalid.

2.6 WSDL Specification

A separate Web Service Description Language (WSDL) specification will be produced for distribution, in conjunction with the XML Schema which will make up the format of the message. A copy of these specifications should be bundled with this document.

2.7 Deployment

The web service will be deployed on a container that implements HTTPS. Therefore, requests must use the HTTPS protocol which will ensure all authentication credentials are encrypted across the internet and not decrypted until they are within SLC's network boundary.

2.8 Processing

2.8.1 Parsing XML

The XML message will be validated based on the SLC bespoke schema. If it does not meet the standard specified then a SOAP response will be sent back containing an exception that will describe the error.

An error will be returned if the supplied XML document does not meet any of the following validation conditions

- XML is badly formed
- Supplied data is not an XML document according to the XML schema

The error returned will be dependent on the nature of the parse exception. Some examples are shown below along with their likely causes.

Error message	Cause	Corrective action
com.ctc.wstx.exc.WstxUnexpectedCharException: Unexpected character '?' (code ???) expected ...	Invalid syntactic token in XML	Ensure XML is valid
First Element must contain the local name, Envelope	First tag should be <soapenv:Envelope>	Ensure XML is valid SOAP message (SOAP 1.1 or 1.2)
java.lang.Exception: com.ctc.wstx.exc.WstxParsingException: Unexpected close tag </???.>; expected </???.>.	Unbalanced open and close tags	Correct XML document structure to ensure tags balance properly
Unexpected subelement duration	Missing <description> element. According to the XML Schema, the <description> element is mandatory. If it is not supplied, the parser reaches the next element (in this case <duration>) unexpectedly and throws an error.	Supply the mandatory element. Note that the <xs:sequence> tag requires its subelements to be ordered as per the specification.

Processing of the request will then stop.

N.B. Due to a feature of the Axis2 framework on which we are basing our web services, certain optional elements of type yesNo (course.ucasCourse and location.active) will default to FALSE if a pair of empty tags is specified for the element. i.e. <active></active> is the same as <active>>false</active>. However, we would not expect that a client would ever provide XML like this, but would always specify a value between the element opening and closing tags.

2.8.2 Authentication

Initial Authentication will involve validating that the supplied *username* & *password* parameters are valid.

Note that if an invalid password is supplied for a valid username then the Authentication System will register a failed login attempt. If three failed login attempts are registered then the user account will be locked, in a similar way to a user using the HEI Secure Website. If the web service account gets locked out, the HEI can unlock the account using the existing HEI Portal user administration functions. User account maintenance for the web

service user accounts can be performed by a user with the “HES Courses User Administrator” role. Note that User accounts for SLC’s HEI Portal have a username, password and secret answer. For web service usage only the username and password need to be supplied to authenticate the user. However, if the password is reset via the user administration function, both the password and secret answer will be reset to generated values. If the HEI staff want to change the password to some specific value then they can login to the portal using the web service account (username, password and secret answer) at which point they will be required to change the password and secret answer. Therefore, it may be desirable to keep track of the password and secret answer to allow further changes via the portal. If the secret answer is not known, the account can be reset again via User Administration.

In the event of failed initial authentication then a SOAP reply will be returned with an error message.

SOAP Message	Cause	Corrective action
<pre><soapenv:Envelope xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"> <soapenv:Header> <wsa:Action>http://www.w3.org/2005/08/addressing/soap/fault</wsa:Action> </soapenv:Header> <soapenv:Body> <soapenv:Fault> <faultcode>soapenv:Server</faultcode> <faultstring>WSDoAllReceiver: security processing failed</faultstring> </soapenv:Fault> </soapenv:Body> </soapenv:Envelope></pre>	Authentication failure	Ensure that the web service user account details are correct (e.g. by logging in to the HEI Portal using the web service user account details

Each of the response messages (*getCourseCatalogResponse*, etc.) has a **result** element (which contains **resultNumber** and **message** elements) in order to report request-level failures of this kind.

Processing of the request will then stop.

2.8.3 Authorisation

The secondary validation involves performing cross validation of the following information supplied with the request. Checking the now authenticated username supplied is valid to perform operations on the HEI(s) within the message.

A message may contain information on multiple HEI’s, for example where parent-child relationships exist between HEI’s.

In order to perform updates for a particular HEI Code, the supplied username must have been approved for update of that HEI. If this check fails then none of the updates for that **institution** will be attempted. Instead, an **institutionSubmissionResponse** will be returned with just the **heiCode** element and the **result** element with its **resultNumber** and **message** elements included with appropriate values.

Similarly, validation checks will be carried out on each course update to ensure that the course code being updated is a valid course code within the HEI corresponding to the **institution** element that it is within. This is to prevent accidental (or malicious) update of course information at an HEI for which the user is not approved. If this check fails, then the **courseResponse** element will contain only the **slcCourseCode** element and the **result** with appropriate **resultNumber** and **message** values.

Response Code	Message	Description
0	OK	No errors occurred
-32001	User not authorised to access HEI	User is not authorised to perform this

		action for this HEI
-32003	One or more Institutions failed	There is a problem with one of the elements in the catalog
-32003	One or more institution component failures	There is a problem with one of the elements in the institution i.e. institution year, locations or courses
-32004	Cannot create location – associated institution cannot be found	The location cannot be associated to an HEI.
-32005	Cannot update – course academic year is live	Once course is live, the course cannot be amended for that academic year.
-32005	Cannot update – course is live	Once course is live, the course cannot be amended for that academic year.
-32005	Cannot update – course term is live	Once course is live, the course cannot be amended for that academic year.
-32005	Cannot update – course year is live	Once course is live, the course cannot be amended for that academic year.
-32005	Cannot update – institution year is live	Once institution year data is live, the data for that institution year cannot be amended.
-32005	Cannot update – location is live	Once location data is live, the location data cannot be amended for that.
-32006	Cannot update course – timestamps do not match	The timestamp of the course the HEI has is different from the timestamp of the course on Stage.
-32006	Cannot update institution year – timestamps do not match	The timestamp of the institution year the HEI has is different from the timestamp of the institution year on Stage.
-32006	Cannot update location – timestamps do not match	The timestamp of the location the HEI has is different from the timestamp of the location on Stage.
-32007	Cannot submit course without course academic year	Course must be associated to its course academic year
-32007	Insertion of Institution Years is not allowed	These cannot be created other than in the Capture Stage Data process, so cannot be inserted via the web service.
-32007	Cannot create primary location	These cannot be created other than in Class, so cannot be inserted via the web service.
-32007	Cannot change primary indicator	The Primary indicator that was set up when the Institution was created in Class cannot be changed.
-32008	Cannot process course – academic year does not match	The academic year of the course must match the academic year that is being processed
-32008	Cannot process institution year – academic year does not match	The academic year of the institution year must match the academic year that is

		being processed
-32009	Cannot update – course is queued	The course is pending promotion, so cannot be processed

2.8.4 Data Validation

2.8.4.1 Primary Data Validation

Once the initial XML parsing and user authentication checks have been performed the data is of the correct format to be inserted into the staging tables within HEI which will hold all newly submitted course information.

Once inserted into the database, the primary data validation take place. The primary validation consists of the kind of checks that one would expect if the course details where entered through the screens. I.e. the name contains characters only within the valid subset of alphanumeric characters, the dates are all valid dates, and so forth. The complete list of primary validation is shown in Appendix B.

2.8.4.2 Secondary Data Validation

Once the course submission has passed the primary data validation with no issues the secondary data validation module should process the information.

Secondary validations are the business the rules which are imposed on the course data, such as the OFFA fee cap and such. Again, this validation is shown in Appendix B.

2.8.4.3 Raise Exception Tasks

Errors from either of the validation routines will result in Validation Errors being raised against that HEI. These can be viewed via a web page within the SLC HEI Courses Portal and either corrected directly within the portal, or, more likely, corrected within the HEI's own system and resubmitted via the web service. It is anticipated that HEI's software will be written to minimise the possibility of validation failures on submission to SLC by replicating the SLC validation within their own systems. To facilitate this, the validation specifications are being included in the information pack available to HEI's.

2.8.4.4 Email Exception Summary

Upon completion of both validation modules, an email summary report will be sent to the HEI contact. This will contain the URL link to the new HEI Course Portal – Tasks and Errors screen, to address any issues arising

2.8.4.5 Promotion of Data to Live

If a course update or insertion is successfully validated, then the course and any associated course locations and the corresponding institution year will be immediately promoted to the "Live" data area. The staging data will have its status changed to Live and *no further update of the staging data will be possible*, either via the web service, or the portal web pages.

2.8.5 Optimistic Locking

The HEI Course Update screens within the SLC HEI Courses Portal will allow update to the staging data at any time up to the point where data for a course, institution year or location is promoted to Live. If such an update occurs, and the client system subsequently tries to post data in via the web service based on an earlier version of the data staging data, then the update will fail, as outlined in section 2.5.2.10.

2.8.6 Validation Exceptions Logging

All validation exceptions will be logged in the bespoke tables. The exceptions will be logged in the database. Reports and analysis on the type and frequency of validation failures can then be produced against this exception data.

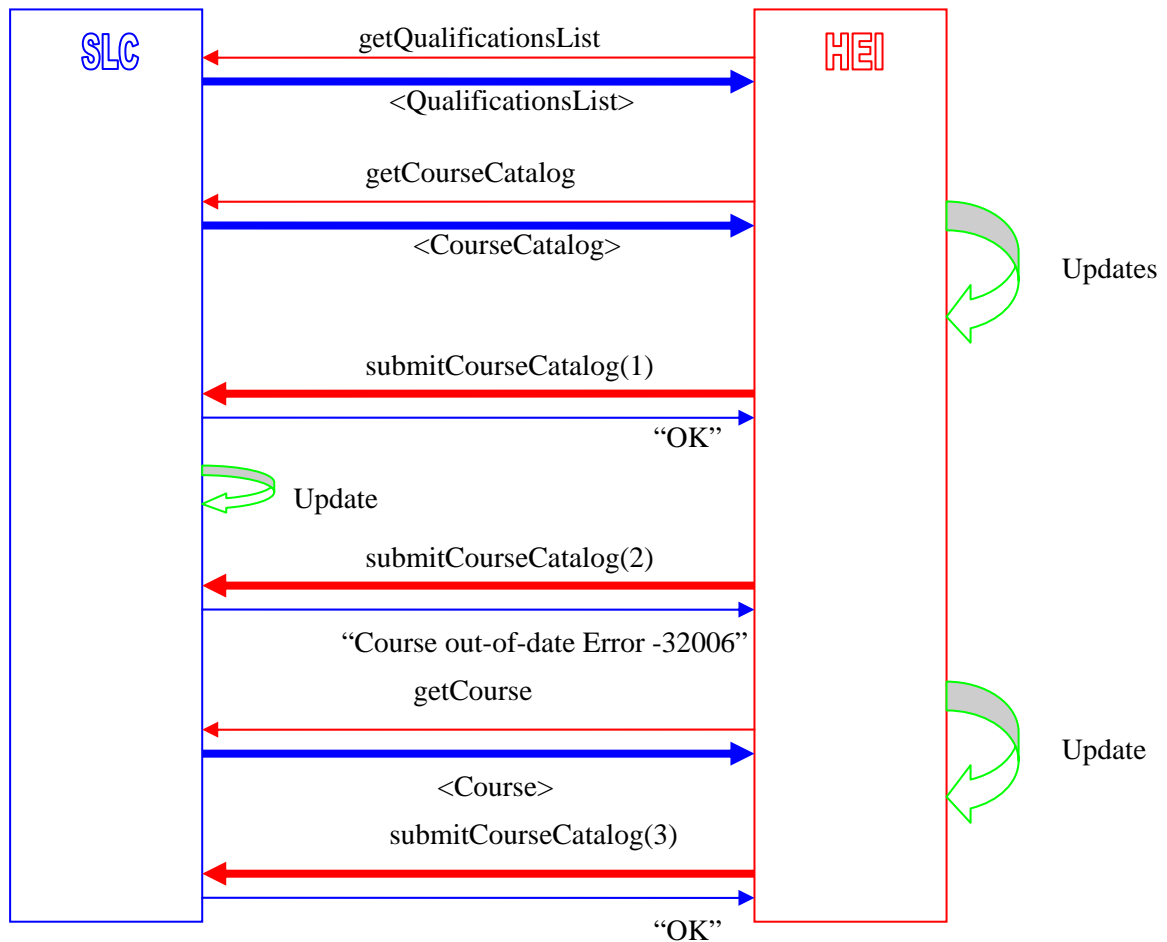
2.8.7 Web Service Usage Logging

All use of the web service endpoints will be logged within SLC's database for audit trail purposes.

2.9 External Testing of Web Service

Towards the end of the internal testing phase of the HEI Courses development project, provision will be made for external parties to test systems against a test instance of the web service. The exact details of the test environment to be used, including the URL and how testing shall be controlled has yet to be decided.

Appendix A – Example Interaction



This example is meant to demonstrate the likely usage of the web service. Firstly, `getQualificationsList` is called to provide the list of qualifications for validation within the HEI’s own system. This should only need done at the initial start-up of the service and subsequently if and when the list of valid qualifications is updated (by SLC).

Next, the course catalog is downloaded for the forthcoming academic year and the HEI staff update the details, providing course date information, etc. Some of the course data are then successfully sent back to SLC (`submitCourseCatalog(1)`). The second submission (`submitCourseCatalog(2)`) fails because one of the courses has already been updated in SLC’s course database between `getCourseCatalog` and `submitCourseCatalog(2)`. The HEI re-synchronises their data by re-fetching the single affected course using `getCourse`, re-applies the changes and re-submits the single course (`submitCourseCatalog(3)`).

Appendix B – Primary and Secondary Validation

B.1.1 Basic data Validation (Primary Validation) – Stage Institution Year

These are the Basic Data validation rules that will apply.

DATA ITEM	Type	User / System input	Mandatory		Format
			on Save	on Validate	
ACADEMIC YEAR	NUMBER (4)	User	Y	Y	<ul style="list-style-type: none"> Academic year entered by the user when logging in.
VARIABLE FEES INDICATOR	TEXT (1)	User	Y	Y	<ul style="list-style-type: none"> “Y” or “N”.
INSTITUTION FEE CAP	NUMBER (4)	User	N	Y	<ul style="list-style-type: none"> Must be in the format 9999. Leading zeros do not need to be input.
PREDICTED COURSES	NUMBER (4)	User	N	Y	<ul style="list-style-type: none"> Cannot be null or zero on Validate.

B.1.2 Business Rules Validation (Secondary Validation) – Stage Institution Year

These are the Business Rules validation that will apply.

DATA ITEM	Rules
ACADEMIC YEAR	<ul style="list-style-type: none"> Should not be a year that already exists in the Stage tables for that Institution. Must be greater than maximum Academic Year on COLLEGE on Live College/Course Data Model.
VARIABLE FEES INDICATOR	<ul style="list-style-type: none"> n/a
INSTITUTION FEE CAP	<ul style="list-style-type: none"> HEIs must have Institution fee cap information. Institution Fee Cap cannot be greater than XXSL_COLL_ACAD_YRS.OFFA FEE CAP for the Academic Year being processed for Institutions that are “PU”, i.e. where XXSL_HEI_TUITION_FEE_TYPES.HEI_TUITION_FEE_TYPE = ‘PU’. Institution Fee Cap cannot be greater than XXSL_HEI_TUITION_FEE_TYPES.HIGHER_TUITION_FEE_AMOUNT for the Academic Year being processed for Institutions that are not “PU”, i.e. where XXSL_HEI_TUITION_FEE_TYPES.HEI_TUITION_FEE_TYPE = ‘PV’ or ‘TR’. If the HEI is a non-VTF (variable tuition fee) participant, the Institution fee can not be greater than the standard tuition fee amount set annually (currently £1255 for Academic Year 2008)
PREDICTED COURSES	<ul style="list-style-type: none"> Cannot be null or zero.

B.2.1 Basic data Validation (Primary Validation) – Stage Location

These are the Basic Data validation rules that will apply.

DATA ITEM	Type	User / System input	Mandatory		Format
			on Save	on Validate	
CAMPUS ID	TEXT (1)	User	Y	Y	<ul style="list-style-type: none"> • Upper case letters and numbers only. • Must be unique for that HEI across all locations, including Franchise locations.
FRANCHISE INDICATOR	TEXT (1)	User	Y	Y	<ul style="list-style-type: none"> • “Y” or “N”.
PRIMARY LOCATION INDICATOR	TEXT (1)	User	Y	Y	<ul style="list-style-type: none"> • “Y” or “N”.
NAME	TEXT (30)	User	Y	Y	Upper case only and characters “/”, “-”, “(”, “)”, “\”.
ADDRESS LINE 1	TEXT (60)	User	N	Y	<ul style="list-style-type: none"> • Alphanumeric • Address lines 1-3 must be populated on Validate.
ADDRESS LINE 2	TEXT (60)	User	N	Y	<ul style="list-style-type: none"> • As above
ADDRESS LINE 3	TEXT (60)	User	N	Y	<ul style="list-style-type: none"> • As above
ADDRESS LINE 4	TEXT (60)	User	N	N	<ul style="list-style-type: none"> • Alphanumeric if entered.
ADDRESS LINE 5	TEXT (60)	User	N	N	<ul style="list-style-type: none"> • Alphanumeric if entered.
POST CODE	TEXT (8)	User	N	Y	<ul style="list-style-type: none"> • Upper case only • Can include spaces • Must be in the standard slc postcode format.
PHONE NUMBER	TEXT (20)	User	N	Y (“N” if it is a Franchise)	<ul style="list-style-type: none"> • Follows standard slc telephone format. • Can include extension number, so may include letters, e.g. “e”, “x” or “t”.
FAX NUMBER	TEXT (60)	User	N	N	<ul style="list-style-type: none"> • Follows standard slc telephone format. • Can include extension number, so may include letters, e.g. “e”, “x” or “t”.
EMAIL ADDRESS	TEXT (60)	User	N	Y (“N” if it is a Franchise)	<ul style="list-style-type: none"> • Follows standard email format.
ACTIVE IND	TEXT (1)	User	N	Y	<ul style="list-style-type: none"> • “Y” or “N”.

B.2.2 Business Rules Validation (Secondary Validation) – Stage Location

These are the Business Rules validation that will apply.

DATA ITEM	Rules
FRANCHISE INDICATOR	<ul style="list-style-type: none"> • Cannot be “Y” if Primary Location is “Y”, as a Franchise location cannot be the Primary location. • A Franchise location will only have one set of contact details, however, if the Franchise location has more than one Campus, and has courses running at each of the campuses, then each location is treated as a separate franchise location.
PRIMARY LOCATION INDICATOR	<ul style="list-style-type: none"> • There must be one location set to the Primary location. • There can only be one Primary location for each Institution. • Cannot be “Y” if the Franchise indicator is “Y”, as the Primary location cannot be a Franchise. • The Primary location cannot be deleted.
NAME	<ul style="list-style-type: none"> • Must be unique across all locations and franchise locations for that HEI.
ADDRESS LINE 1	n/a
ADDRESS LINE 2	n/a
ADDRESS LINE 3	n/a
ADDRESS LINE 4	n/a
ADDRESS LINE 5	n/a
POST CODE	n/a
PHONE NUMBER	n/a
FAX NUMBER	n/a
EMAIL ADDRESS	n/a
ACTIVE IND	<ul style="list-style-type: none"> • A location cannot be marked inactive if it has current courses linked to that location.

B.3.1 Basic Data Validation (Primary Validation) – for course

DATA ITEM	Type	Mandatory		Format
		on Save	on Validate	
ACADEMIC YEAR	Text (2)	Y	Y	<ul style="list-style-type: none"> Numeric
SLC CODE	Text (6)	Y	Y	<ul style="list-style-type: none"> Numeric
UCAS COURSE?	Text (3)	N	Y	<ul style="list-style-type: none"> Alphanumeric Must be either Yes or No
UCAS COURSE CODE	Text (4)	N	N	<ul style="list-style-type: none"> Alphanumeric Must have exactly 4 characters.
CAMPUS ID	Text (2)	N	Y	<ul style="list-style-type: none"> Alphanumeric The campus should have already been created, i.e. should match one the campus ids in STAGE LOCATION.CAMPUS ID for all the locations of the STAGE INSTITUTION that belong to the HEI in question.
COURSE NAME	Text (41)	Y	Y	<ul style="list-style-type: none"> Alphanumeric Must be at least 3 characters in length. Must not start with the code of a qualification followed by a space Must not end with a space followed by the code of a qualification.
DURATION	Text (1)	Y	Y	<ul style="list-style-type: none"> Numeric In range 1 – 6.
METHOD OF ATTENDANCE	Text (2)	Y	Y	<ul style="list-style-type: none"> Alphabetic The value given should match one of the values in the METHOD OF ATTENDANCE Domain on CLASS. Where the METHOD OF ATTENDANCE is “PT” i.e. a part-time course, the only PT courses that will be valid are those where <ul style="list-style-type: none"> - the associated COURSE TYPE is either “other”, “franchised” or “PGCE” and - the associated QUALIFICATION is either PGCE, PGDE or CertEd. <p>In other words, for a part time to be valid it should succeed (course type = “other” or “franchised” or “PGCE”) and (qualification = “PGCE” or “PGDE” or “CertEd”)</p>
COURSE TYPE	Text (11)	Y	Y	<ul style="list-style-type: none"> Alphabetic The value given should

ICT Systems Development, Technical services

DATA ITEM	Type	Mandatory		Format
		on Save	on Validate	
				match one of the values in the COURSE TYPE Domain on CLASS
STATUS	Text (1)	Y	Y	<ul style="list-style-type: none"> Must be either "O" (Open) or "C" (Close) matching one of the values in the COURSE STATUS Domain on CLASS
QUALIFICATION	Text (11)	Y	Y	<ul style="list-style-type: none"> Alphanumeric The value given should match one of the values on QUALIFICATION

B.3.2 Business Rules Validation (Secondary Validation) – for course

DATA ITEM	Rules
UCAS CODE	<ul style="list-style-type: none"> The combination between UCAS CODE and CAMPUS ID creates uniqueness over a course for UCAS courses, i.e. there cannot be more than one course with the same UCAS CODE and CAMPUS ID. This validation then just covers courses which UCAS Course Indicator (UCAS COURSE? attribute in table above) is set to Y
DURATION	<ul style="list-style-type: none"> Must be equal to the number of course years.

B.4.1 Basic Data Validation (Primary Validation) – for course year

DATA ITEM	Type	Mandatory		Format
		On Save	On Validate	
FEE CAP AMOUNT	Text (4)	N	Y	<ul style="list-style-type: none"> • Must be in the format 9999 • Leading zeros do not need to be input. • The value entered for this attribute should be lower or equal than the OFFA Fee Cap set for the HEI in context for the academic year in question (i.e. the STAGE COURSE YEAR.FEE CAP AMOUNT should be lower or equal than the STAGE INSTITUTION YEAR.FEE CAP AMOUNT for the STAGE INSTITUTION YEAR.ACADEMIC YEAR in question)
COURSE YEAR	Text (1)	Y	Y	<ul style="list-style-type: none"> • Numeric • Between 0 and 6. • The combination between Academic Year and Course Year should be unique for the course in question; i.e. a course that takes place within an academic year can not have a duplicate entry for course year.

B.4.2 Business Rules Validation (Secondary Validation) – for course year

DATA ITEM	Rules
FEE CAP AMOUNT	<ul style="list-style-type: none"> • Non-Private HEIs must enter course fee information for each year of the selected course.
COURSE YEAR	<ul style="list-style-type: none"> • The correct year for foundation years is 0. • The course year number for a course which has a foundation course should always be lower in one unit than the duration of the associated course, i.e. STAGE COURSE YEAR.COURSE YEAR should be lower than its associated STAGE COURSE.DURATION -1. <p>On the other hand, the course year number for a course which does not have a foundation course should always be lower than the duration of the associated course, i.e. STAGE COURSE YEAR.COURSE YEAR should be lower than its associated STAGE COURSE.DURATION.</p> <p>The way to know if a course has a foundation course is based on the fact that any foundation course should be entered with course year zero (0).</p>

B.5.1 Basic Data Validation (Primary Validation) – for course term

DATA ITEM	Type	Mandatory		Format
		on Save	on Validate	
TERM NUMBER	Text (1)	Y	Y	<ul style="list-style-type: none"> Numeric Must be 1, 2 or 3. Each term must have a different term number to be unique for the course year in question.
START DATE	Date	N	Y	<ul style="list-style-type: none"> Must be at least 1 day before the corresponding End Date, i.e. minimum duration of each term is 1 day. Must be at least 1 day after any preceding End Date. Dates should be entered in the following format DD/MM/YYYY, where DD is Day, MM is Month, and YY or YYYY is Year.
END DATE	Date	N	Y	<ul style="list-style-type: none"> The last term's End Date must be less than a year from the first term's Start Date. (i.e. Academic course year should not be longer than a calendar year). Dates should be entered in the following format DD/MM/YYYY, where DD is Day, MM is Month, and YY or YYYY is Year.

B.5.2 Business Rules Validation (Secondary Validation) – for course term

DATA ITEM	Rules
START DATE	<ul style="list-style-type: none"> First Term's Start Date must be after 31st July of the academic year in question. For courses with the next qualification codes <ul style="list-style-type: none"> ○ MBCHB ○ MBBS ○ BMBS ○ BMED SC ○ BM ○ BMBCh <p>is permissible to have start dates for the first term after 1st of June within the academic year in question.</p>