e-Learning

SkillSoft
AHEAD OF THE LEARNING CURVE

OLSA Integration Guide

v1.0

www.skillsoft.com

Copyrights

SkillSoft Corporation
107 Northeastern Blvd.
Nashua, NH 03062
603-324-3000
87-SkillSoft (877-545-5763)
Information@SkillSoft.com

Trademarks

Table of Contents

Introduction

What is OLSA?

Open Learning Services Architecture (OLSA) is a comprehensive service oriented architecture initiative that is intended to simplify the effort required to integrate SkillSoft learning services with your Learner Management System(LMS) or portal of choice.

Why Integrate Using OLSA?

The following scenarios illustrate some of the benefits of using OLSA.

- **AICC Compliant LMS**
  Your AICC compliant LMS integrates with OLSA. The LMS uses the Asset Integration Service to automate the first time installation and periodic updating of SkillSoft hosted content that it is entitled to. The Asset Integration service provides AICC install files for SkillSoft hosted content, and the LMS uses standard AICC mechanisms to natively install, launch, and track SkillSoft hosted content. The LMS also uses the Search & Learn service to provide an enhanced search environment for content installed via the Asset Integration service. Your LMS users can natively access SkillSoft hosted content from their catalog as well as from search results returned by the Search-and-Learn service. Usage data in these situations are seamlessly sent to the LMS via AICC mechanisms.

- **Web Portal**
  Your web portal integrates with OLSA. The portal is not AICC compliant, but it can still use the Asset Integration Service to retrieve the asset ID list of SkillSoft hosted content that it is entitled to. The portal uses the SignOn service to launch SkillSoft hosted content by asset ID. The portal uses the Offline service to provide offline-play access to SkillSoft hosted content by asset ID. The portal can use the Search & Learn service as well, providing launch access to content using OLSA-based launch URLs embedded in the search results. Usage data in these situations are automatically managed by OLSA. The portal uses the Usage Data Synchronization service to present OLSA managed usage data to its users.

The term customer-application refers to either an LMS or portal as described above. In situations where a distinction is appropriate, this document will refer explicitly to either an LMS or a portal.

Technical Considerations

A customer-application using the OLSA Web Services has an associated customer-context defined in the OLSA Environment. This implies that the full-capabilities of the OLSA environment admin system are available for performing configuration functions not available through the OLSA Web Services.

The OLSA Web Services provides a single WSDL file that specifies all of its services (see the OLSA WSDL file for complete details on the APIs described in this guide).

- The OLSA Web Services is based on SOAP 1.2 bindings.
- The OLSA Web Services is available using either HTTP or HTTPS.
- The OLSA Web Services is only available initially as a SkillSoft hosted feature.
- This document assumes the reader has working knowledge of Web Services, AICC, and other web application technologies including SOAP, HTTP and HTTPS.

Security

The OLSA Web Services supports authentication using an OASIS specification named UsernameToken with the WSS: SOAP Message Security. This OASIS specification can be found at: *http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf.*

The OLSA Web Services optionally supports HTTPS to further secure the communication between the customer-application and the OLSA.

Additional References

- OLSA WSDL file: This is the precise description of the OLSA Web Service API specified in the Web Services Description Language (WSDL).
- OLSA Release Notes: Description of new features, changes to existing features, bug fixes and known issues with each OLSA release.
- OLSA Integration Kit Readme: The readme provides a description of Integration Kit components, system and software requirements, set-up instructions, and client code examples.

Web Services and Launch URLs

The OLSA Web services are for server-to-server communication only. Do not implement browser-based user interfaces that allow direct access to the OLSA Web service. The user interface should always send requests back to the customer-application, and the customer-application should issue the actual OLSA Web service call.

Some OLSA Web service calls return a launch URL. Launch URLs are for accessing training content or for accessing the OLSA environment. In either case, launch URLs may be embedded in user interfaces but not the OLSA Web service call that returns them. A launch URL is secured with an OLSA session id that is time-limited (usually several hours) and is user-specific. A launch URL should not be persisted for long-term storage it should be used as soon as it is received by the customer-application. A launch URL should be treated as an opaque value (its format and contents are subject to change from one release of OLSA to another).

The following example illustrates these points:

1. A user logs in to the customer-application via a Web browser client. The customer-application presents its UI in the Web browser client.

2. On the UI there is a link to access some feature in the OLSA environment. The user clicks the link.

3. The link turns into a JavaScript `openWindow` call to create a new browser window on the client that contains the result of the request. This first client request is sent to the customer-application.

4. The customer-application receives the first request from the client and makes the appropriate server-to-server OLSA Web service call. The call returns a launch URL that has an associated session id (time-limited and specific to the user).

5. The customer-application returns a response to the first client request (a JavaScript sequence that does a redirect to the launch URL).

6. The new window processes the response (the redirect JavaScript). The redirect JavaScript is executed on the client and processes the launch URL. The launch URL sends a second client request that is now sent to OLSA. OLSA returns the appropriate response (HTML in this case) and the response is now rendered in the new window.

SOAP Faults (Error conditions)

OLSA Web Service functions return SOAP faults when error conditions arise. All OLSA Web Service functions return the following SOAP fault types:

- **GeneralFault**: This is a catch-all fault type for any error condition that is not specifically outlined in the specification for a given function. This also includes authentication failures. The GeneralFault contains a detailed message on the nature of the error.

Synchronizing Server Clocks

OLSA implements the *Web Services Security Username Token Profile 1.0* described in the following document:

http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf

This security specification recommends that any web service provider reject any request whose creation time is older than five minutes. SkillSoft Hosting ensures that OLSA synchronizes with an atomic clock. The customer must also synchronize their server application environment with an atomic clock to avoid any time out issues.

If the server application environment is not synchronized with the web service the user would see the following exception:

```
System.Web.Services.Protocols.SoapException: WSDoAllReceiver: security processing
failed; nested exception is:

org.apache.ws.security.WSSecurityException: An error was discovered processing the
header. (WSSecurityEngine: Invalid timestamp The security semantics of message have
expired) at
System.Web.Services.Protocols.SoapHttpClientProtocol.ReadResponse(SoapClient Message
message, WebResponse response, Stream responseStream, Boolean asyncCall) at
System.Web.Services.Protocols.SoapHttpClientProtocol.Invoke(StringmethodName,
Object[] parameters)
```

Automatic User Registration or Update

Automatic user registration or update is supported by the OLSA Web Services to synchronize user information between itself and the customer-application. Several of the OLSA Web Service functions support this capability in addition to performing its primary function. This capability is often convenient for situations where making two Web service calls (one for registering or updating the user, and the other for performing the specific desired action) has unacceptable performance or integration-implementation issues.

The OLSA Web Services offer two options for automatic-user-registration-update. OLSA Web Service functions with this capability are detailed in this document and indicate which options they support.

All-user-attributes option

This option allows any and all user attributes to be specified for registration or update. This includes all (see below), plus the special argument `newUsername`.

On the first call for a given username, if the user with “username” does not exist then the user is created with the specified user attributes. On subsequent calls if the same arguments are specified then no changes are made to the user. If any argument values change then it is assumed that the caller wants the relevant attributes updated for the specified user. This allows registrations and updates of a user with a single call.

- `newUsername` is a special argument. It should only be specified if the caller wants an existing username changed to a different value. If this user does not exist yet then this value is ignored. This value should be kept unspecified if a change to the username is not desired.

- `groupCode` and `grouppath` are special arguments that allow the caller to specify in which user group to create the user. Only one or the other should be specified.

Simple-user-attributes option

This option allows a limited set of user attributes to be specified for registration only (not for update). These attributes are:

- `username`

- `password` (Optional. Ignored if user already is registered)

- `groupCode` (Optional. Ignored if user already is registered)

Using this option requires that all user groups in the OLSA environment for the customer are defined with a unique `groupCode`.

General User Attributes

There are several general user attributes that can be initialized and updated. Various APIs in this document specify these attributes as arguments. See OLSA WSDL for the complete list. Examples of general user attributes include:

- `username`
- `password`
- `firstname`
- `lastname`

Asset Integration Service (AI_)

The Asset Integration Service allows a customer-application to programmatically process assets and to natively install SkillSoft hosted content. The service provides capabilities for automating the management of the assets that a customer-application is entitled to. This includes the ability to:

- Get the initial list of 'entitled' assets to initialize the customer-application.
- Periodically get the list of newly 'entitled', 'modified', or 'not_entitled' assets to keep the customer-application up-to-date.
- Access standards compliant metadata (initially for AICC only) for entitled assets to install into the customer-application.

Assets installed via this service are described by standards compliant metadata with AICC launch URLs that point back to the OLSA environment. OLSA then mediates access to the SkillSoft hosted content.

There are several advantages to using the Asset Integration Service:

- The physical management of installed assets into the customer-application is simplified. The actual physical content is not transferred to the customer-application environment.
- Configuration of player settings and player versions is managed by SkillSoft and is transparent to the customer.

Supported Content

The Asset Integration Service supports the following SkillSoft asset types:

- CCA courses
- Business Skills courses
- SkillSimulations
- IT courses (e3, Classic)
- Express Guides
- Test Prep Exams
- Final Exams
- Mentoring Assets
- CCT course
- LOT
- Dialogue Recorded Sessions

Asset Integration service assets do not include Referenceware. Topics, Skillbriefs, and Jobaids are included implicitly with their parent courseware, but they are not individually installable via meta files generated by the Asset Integration Service.

Topic launch of natively installed Asset Integration assets

CCA, Business Skills and e3 courses support topic launch. A topic launch allows the end-user to randomly access a specific topic within the parent course. All usage-data tracking is generated under the context of the parent course. A topic launch can be performed on a natively installed course by appending a `topicid` argument to the usual AICC launch URL. Example of an AICC topic launch URL:

`http://launchURL?AICC_SID=sessionid&AICC_URL=lmsURL`**`&topicid=theTopicIDToLaunch`**

The `http://launchURL` is the URL to launch a given course as specified in the AICC .au file for the relevant course. The text in bold indicates the additional information necessary to launch the specified topic within the course.

Note: the AU.Web_Launch value would also have to be included if defined (see below).

Required AICC fields for natively installed Asset Integration assets

To install and launch natively installed Asset Integration assets, the following AU file fields must be supported:

- AU.File_name: This is the launch URL for the asset. It must be specified as an absolute URL back to the OLSA environment.
- AU.Web_Launch: This lists the launch parameters. Specify additional arguments required for the launch here.

Enabling Web Accessibility for natively installed Asset Integration assets

Most course types support Web Accessibility (compliance with Section 508). To launch a natively installed asset with Web Accessibility enabled the following additional argument must be specified on the usual AICC launch URL. Example of an AICC Web Accessibility enabled launch URL:

`http://launchURL?AICC_SID=sessionid&AICC_URL=lmsURL`**`&x508=1`**

The `http://launchURL` is used to launch a given asset as specified in the AICC .au file for the relevant course. The text in bold indicates the additional information necessary to launch with Web Accessibility enabled.

AI_InitiateAssetMetaData

The AI_InitiateAssetMetaData function initiates a request to get entitlement status changes since the update-time and to get the metadata for relevant entitled assets.

Update-time

Entitlement status changes are determined from a given point in time known as the update-time. The update-time is managed by the OLSA. The update-time is determined by the following transaction sequence:

- AI_InitiateAssetMetaData (one call to open the transaction)
- AI_PollForAssetMetaData (one or more calls)
- AI_AcknowledgeAssetMetaData (one call to close the transaction)

If a transaction has never been completed the update-time is, effectively, the beginning of time. In this case all accessible assets are considered new and in the entitled state by AI_InitiateAssetMetaData.

If a transaction has been completed, the update-time is the time AI_InitiateAssetMetaData was called during the most recent completed transaction. All assets made accessible since that time are considered entitled. All assets modified since that time are considered modified. All assets made inaccessible since that time are considered not_entitled.

A candidate update-time is established every time AI_InitiateAssetMetaData is called. The candidate update-time is made the new update-time when the corresponding AI_AcknowledgeAssetMetaData is called.

All vs. Delta Modes

Specify the **all** mode in the call to AI_InitiateAssetMetaData to start with a “clean slate”. When this mode is specified then the following information is generated:

- Entitlement status: Identifies all assets currently accessible. These assets are marked with the status of entitled.
- Metadata: Generates metadata for all entitled assets.

To get any changes in entitlement (all assets whose entitlement status has changed since the update-time), specify the **delta** mode in the call to AI_InitiateAssetMetaData. When this mode is specified then the following information are generated:

- Entitlement status: All assets made accessible since the update-time are marked as entitled. All assets modified since the update-time are marked modified. All assets made inaccessible since the update-time will be marked not_entitled.
- Metadata: Metadata for all entitled or modified assets are generated.

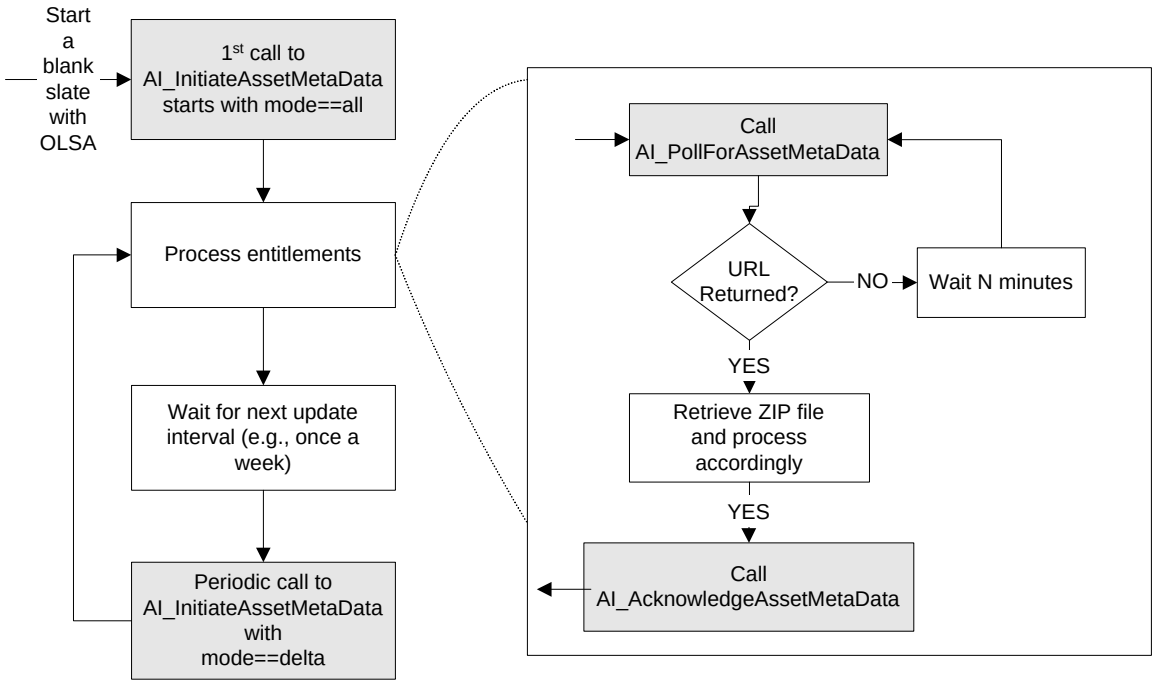

Figure 1: Transaction sequence for AI_InitiateAssetMetaData, AI_PollAssetMetaData, and AI_AcknowledgeAssetMetaData

The AI_InitiateAssetMetaData function returns a handle that can be used with the AI_PollForAssetMetaData function to poll the readiness status of the requested information. Note that calling AI_InitiateAssetMetaData only queues a request to generate the requested information. A call must be made to AI_PollForAssetMetaData to actually retrieve the requested information.

AICC File Generation for Books

Enabling Books

AICC file generation for books is controlled by a new skp_parameter record call called *enable_books_aicc_files*, which supports a value of 0 for disabled or 1 for enabled.

Book Level Launch

You can launch Books through the OLSA Sign-on service. The book level launch supports the launching of books at the book level and the chapter level using the following parameters:

```
Book ID -  ####

Chunk Launch - ####-####
```

AICC Filesets

The requested Asset metadata is packaged in a zip file. The zip file contains an Entitlement Status file at the top level. The zip file will contain an asset folder for each unique asset. Within each asset folder there are five AICC files using the naming conventions of assetid.crs, assetid.des, assetid.au, assetid.cst, and assetid.ort. The image below shows a sample set up:

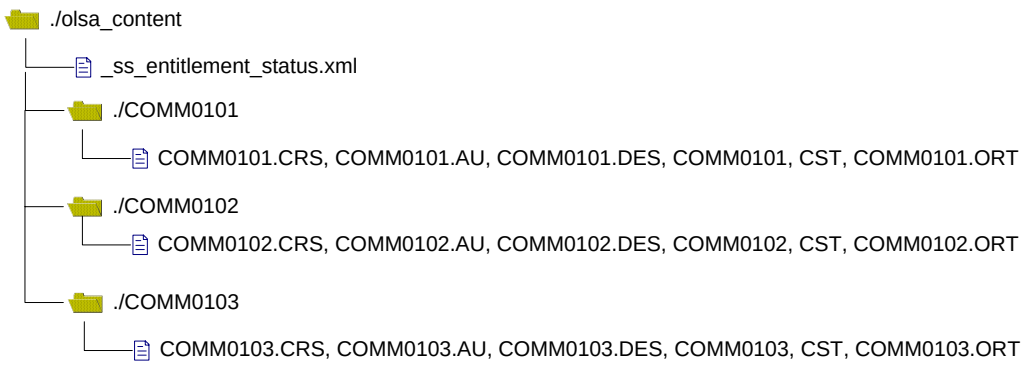

Figure 2: Sample AICC ZIP File

Entitlement Status File

The entitlement status file lists the entitlement status changes that have occurred since the update-time. Assets are sorted by ID in alphabetic order. The available entitlement status changes and recommended actions are listed below.

Entitlement status change	Recommended customer-application action
`entitled`	Install the asset
`not_entitled`	Uninstall the asset
`modified`	Asset metadata may have been modified. For example, the title of the asset may have changed. Take appropriate customer-application-specific actions to pick up potential meta data changes.

Table 1: Entitlement Status Changes

The format of the entitlement status will look like this:

```
<ENTITLMENT_STATUS>
    <ASSET ID='someID1' STATUS='entitled'/>
    <ASSET ID='someID2' STATUS='not_entitled'/>
    <ASSET ID='someID3' STATUS='modified'/>
  ... etc ...
</ENTITLEMENT_STATUS>
```

The customer-application can unzip and process the contents of the AICC files in any manner appropriate for its environment.

Entitlement Exceptions

OLSA saves the most recent change event related to a given asset (installed, uninstalled, deactivated, activated, or modified). In some circumstances, this can result in unexpected status values. This generally occurs when two or more change events occur between two transaction sequences. Use the table below as a guide on what actions to take if an unexpected status value is returned.

Current State of the Course in TPLMS	Returned Status Value	Action
Installed	entitled	Ignore or treat as modified
Not Installed	modified	Treat as entitled
Not installed	not_entitled	Ignore

Table 2. Entitlement Exceptions

Inputs

- customerId
- initiationMode (all or delta)
- metadataFormat (AICC)

Outputs

A handle that represents the initiated transaction. This handle is used with the AI_PollForAssetMetaData function to poll the readiness status of the requested information.

Additional Faults

- RequestAlreadyInProgressFault - A status indicating a AI_InitiateAssetMetaData/ AI_PollForAssetMetaData transaction is already in progress)

AI_PollForAssetMetaData

The AI_PollForAssetMetaData function polls the readiness status of the entitlement status and metadata requested using the handle returned by a prior call to AI_InitiateAssetMetaData.

If the metadata is not ready, wait for a reasonable time interval and call this function again with the same handle value. If the metadata is ready, then an HTTP-based URL to a zip file is returned. An additional HTTP request using the returned URL is necessary to access the metadata zip file.

Once a given zip file is ready, it remains available until AI_AcknowledgeAssetMetaData is called with its associated handle. This option is made available so a zip file package can be retrieved again if an error occurs during transit.

Inputs

- customerId
- handle (returned by a AI_InitiateAssetMetaData call)

Outputs

- URL to a zip file.

Additional Faults

- DataNotReadyYetFault (If the meta data is not ready yet)

AI_AcknowledgeAssetMetaData

The AI_AcknowledgeAssetMetaData function tells OLSA that the specified active AI_InitiateAssetMetaData/ AI_PollForAssetMetaData/ AI_AcknowledgeAssetMetaData transaction is completed. It also deletes the zip file created by AI_InitiateAssetMetaData.

This function can also be called by omitting the handle argument. This means it will invalidate any active AI_InitiateAssetMetaData/ AI_PollForAssetMetaData /AI_AcknowledgeAssetMetaData transaction. The update-time will not be modified. This effectively performs a reset function so the customer-application can restart a sequence from the most recently established update-time.

Inputs

- customerId
- handle (returned by a AI_InitiateAssetMetaData call, optional)

Outputs

- None

Additional Faults

- None

AI_InitiateFullCourseListingReport

The AI_InitiateFullCourseListingReport function retrieves entitlement information. In summary mode it returns the list of accessible assets. In detail mode it includes additional catalog hierarchy information such as where in the catalog hierarchy an asset resides. The `userName` argument is used to specify the relevant entitlement or assignment (see Entitlement/Assignment). If the customer does not take advantage of pre-registering users (see User Registration) or making assignments (see Assignment Types) into OLSA then he may omit the username argument, in which case all assets in his entitlement should be returned.

The report is returned in HTML or CSV format.

Course Title	Duration (Hrs:Min)	Course Number	Curriculum
Interviewing basics	2:00	31763_nl	Course Curricula/English – US/Business Skills
How to program in C++	2:00	MSOF21D	Course Curricula/English – US/Technical Skills
Managing cultural differences	2:30	COMM0606	Course Curricula/English – US/Business Skills
Communication skills	2:00	COMM0101	Course Curricula/English – US/Business Skills

Table 3: Sample full course listing report (detail mode and CSV format)

- Course title – the title of the asset
- Duration – the estimated duration to complete the course
- Course Number – alpha-numeric designation of the asset, also called the Asset Id
- Curriculum – location in the catalog hierarchy where the asset resides. Specified as a / delimited path of asset group titles.

For example, the report shown in Table 2 assumes the following catalog hierarchy is defined:

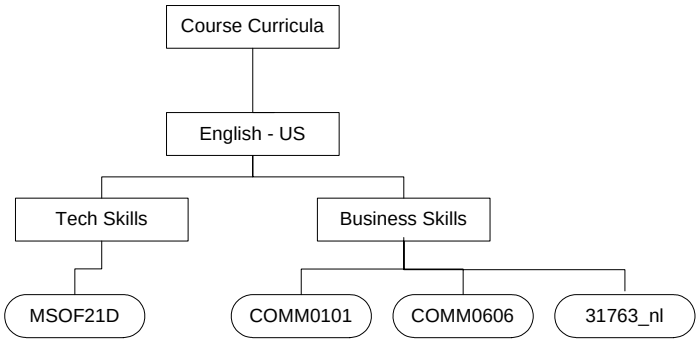

Figure 3: A defined catalog hierarchy

Inputs

- customerId
- reportFormat (HTML or CSV)
- mode (summary or detail)
- username (optional)

Outputs

- A report ID handle. This value should be used with the UTIL_PollForReport function to get the actual contents of the report.

Additional Faults

- None

AI_CreateAssetGroup

This function creates an asset group within the defined catalog structure. An asset group may contain zero or more assets or asset groups. An asset group may only have a single immediate parent group. Example:

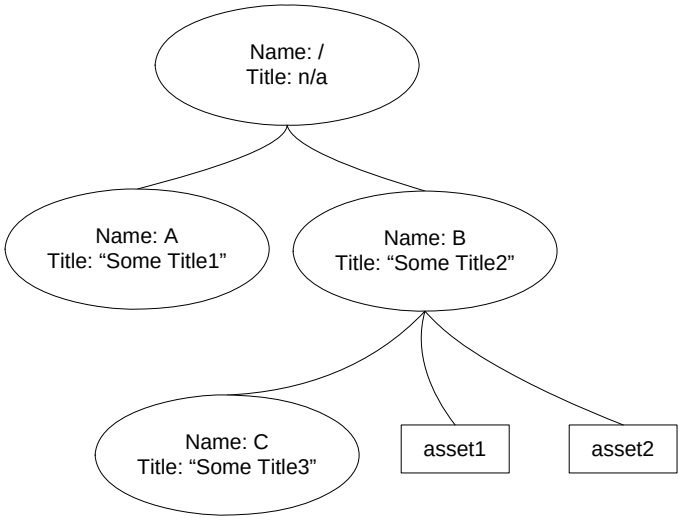

Figure 4: Catalog Group Structure

The top of the catalog structure always has the default super parent group /. There are 3 assets groups under /. The properties of each asset group are listed below:

Name	Its Catalog Path	Title	Parent Catalog Path
A	/A	Some Title1	/
B	/B	Some Title2	/
C	/B/C	Some Title3	/B

Table 4. Definition of the Group Structure

The catalog path to each asset is:

- /B/asset1
- /B/asset2

To create another asset group under C the following arguments can be used:

- Name: D
- Title: Some Title4
- Parent Catalog Path: /B/C

Asset Group names must be unique. **Characters in an asset group name are limited to a-z, A-Z, 0-9 and underscore (_)**. Asset Group names are case-sensitive. The Asset Group Title can be any arbitrary text used for display or reporting purposes. See also AI_InitiateMakeChangesVisible.

Inputs

- customerId
- parentCatalogPath
- assetGroupName
- assetGroupTitle

Outputs

- None

Additional Faults

- ObjectExistsFault

AI_EditAssetGroup

This function modifies the properties of an existing asset group. The title or parent of an asset group can be changed with this command. See also AI_InitiateMakeChangesVisible.

Inputs

- customerId
- catalogPath (for Asset Group to Modify)
- parentCatalogPath (new parent catalog path - optional)
- assetGroupTitle (new group title - optional)

Outputs

- None

Additional Faults

- ObjectNotFoundFault

AI_DeleteAssetGroup

This function deletes an existing asset group and its subordinate asset groups (if any). Any affected assets are left dangling if they have no remaining parent asset groups. See also AI_InitiateMakeChangesVisible.

Inputs

- customerId
- catalogPath (for Asset Group to delete)

Outputs

- None

Additional Faults

- ObjectNotFoundFault

AI_AddAssetToGroup

This function adds assets to any asset group. An asset can be in more than one asset group. See also AI_InitiateMakeChangesVisible.

Inputs

- customerId
- catalogPath (for relevant Asset Group)
- assetId

Outputs

- None

Additional Faults

- ObjectExistsFault (asset already in group)
- ObjectNotFoundFault (group or asset does not exist)

AI_RemoveAssetFromGroup

This function removes an asset from any asset group. This function does not delete an asset. See also AI_InitiateMakeChangesVisible.

Inputs

- customerId
- catalogPath (for relevant Asset Group)
- assetId

Outputs

- None

Additional Faults

- ObjectNotFoundFault (group or asset does not exist)

AI_InitiateMakeChangesVisible

This function will ensure that the effect of all asset group operations have been propagated to all parts of the OLSA environment. This is a resource-intensive operation and must be used with care. If you have a series of asset group operations to perform, it is recommended that you invoke this operation once after all other asset group operations in the series have been performed.

This function initiates the requested operation. The requested operation is not complete until an execution status report is completed. This function returns a reported handle to query for the execution status report.

Inputs

- customerId

Outputs

- A report ID handle. This value should be used with the UTIL_PollForReport function to get the actual contents of the executed status report for this operation.

Additional Faults

- RequestAlreadyInProgressFault (A status indicating an AI_InitiateMakeChangesVisible is already in progress)

Search & Learn Service (SL_)

This service allows a customer-application access to SkillSoft Search & Learn. Assets returned by Search & Learn include all courseware (including Topics, SkillBriefs and Jobaids) and Referenceware (including Chapters).

Key elements of Search & Learn configurations are entitlement and assignment (see Entitlement/Assignment definition). Both are used to control what assets a customer-application's users have access to via search results. The correct use of entitlement is required to adhere to any SkillSoft content license agreements (Referenceware in particular).

Search results contain OLSA launch URLs for all assets. A customer-application that chooses to launch assets using these URLs has all usage data automatically managed by the OLSA environment. Customer-applications using the Asset Integration Service can launch natively installed courseware assets. All usage data is automatically managed by the customer-application.

Search Results

Search results generate courseware hits at two levels: course level and topic level. Search results indicate the relevant course and topic IDs as native id attribute values. The native id attribute value for a course is the same as the asset id for a given course installed via the Asset Integration service. The native attribute value for a topic can be used as the topicid value for a topic launch of the parent course object. A native course launch URL is generated by following the customer-application's specific rules for creating a launch URL for installed course objects. A native topic launch can be created by following the same rules but appending the topicid=nativeIdForTopic to the URL (see Launch URLs).

Search result sets can be very large. The Search & Learn service supports iterating through search results in small increments to improve performance. The Search & Learn service also provides utility functions for getting and defining search configuration values (e.g., available languages, asset types, bins, and asset details).

Asset Types

Asset types serve as identification for the different types of content accessible via the OLSA Search & Learn service. The table below identifies the various asset types.

Asset Type	assetType	DTYPE
CCA	_ss_cca	course
Business Skills Course	_ss_bs	course
JobAids	_ss_ja	jobaid
Skill Briefs	_ss_sb	skillbrief
Generic LO	_ss_generic	custom
Classic Course	_ss_classic	course
e3 Course	_ss_e3	course
Simulation	_ss_sim	simulation
Mentoring	_ss_mentor	mentoring
TestPrep	_ss_tp	testprep
LOT	_ss_lot	custom
CCT Course	_ss_cct	custom
Legacy IT	_ss_legacy	course
ReferenceWare	_ss_book	book
Express Guides	_ss_eg	eguide
Instructor Led Training (ILT)	_ss_ilt	course
Custom Passive Content	_cust_pass	custom
Custom Courses	_cust_course	Custom
Project Centers		
Practice Labs		

Table 5. Asset Types

Search Terminology

assetType - The kind of asset, e.g., _ss_bs _ss_classic, _ss_book, etc. The Search Server uses this tag to properly categorize learning objects:

- _ss identifies the asset type as a proprietary SkillSoft learning object
- _cust identifies the asset type as a custom or third-party learning object

DTYPE - DTYPE identifies under which category a learning object will appear as the result of a search. For example, if a search finds two learning objects, one with assetType=_ss_bs and the other assetType=_ss_e3, both appear in the Course category.

Search & Learn Asset- An Asset in the Search & Learn service is a superset of the definition used in the Asset Integration Service. Assets in this service include Topics, JobAids, SkillBriefs and Referenceware.

SearchAsset ID / Native ID -The search functions return results with attributes named Asset ID and Native ID. An Asset ID is this context is a combination of the <asset-type>:<native-id> which we call the SearchAssetID. The Native ID is exactly equivalent to the AssetID referred to everywhere else in this document.

Trackable asset - An asset that sends tracking data to the LMS.

Non-trackable asset -An asset that does not send any tracking data to the LMS.

Bin - A bin is a container. It can be defined to contain assets of one or more asset types.. Search results are grouped by bins. The search function should be called with a reasonable binsize value. There can be significant performance penalties throughout the system for using an excessively large binsize.

Search Parameter - This refers to a search configuration XML value. It includes items like a list of bins, with each bin specifying the asset types that are mapped to it. It also contains additional configuration values like the default bin sizes. It controls among other things what assets are searched for in a federated search, and how search results are organized.

Entitlement / Assignment- An entitlement is the set of all assets that the customer is allowed contractual access to. An assignment allows finer grain access to assets within an entitlement, down to the user-level within OLSA. Entitlement is managed via the Asset Integration service (see Entitlement). Assignment is managed through the Assignment service (see Assignment Types).

Language - Assets are filterable by a single language as well. The set of available languages will depend on the caller's entitlement.

Result Set - Also know as Search Results, the entire stream of the binned set of assets that match the specified search criteria. The result set/bin/asset structure appears at a high level like the XML fragment below.

```
<SearchResults>
   <bin binname='someBin1'>
      <asset assetid ='someAsset1'/>
      <asset assetid ='someAsset2'/>
      …
   </bin>
   <bin binname ='someBin2'>
   </bin>
         <bin name='someBin3'>
      <asset assetid='someAsset3'/>
      <asset assetid ='someAsset4'/>
      …
   </bin>
   …
</SearchResults>
```

SL_FederatedSearch

Given a search string, search results are returned in high-level, customizable categories (bin sets). During a federated search the search phrase is processed against assets that belong to all bins specified in the default search parameter.

This function initiates a search and returns the initial portion of the matching result set. The caller can page through the remainder of the result set by issuing a call to SL_PaginateSearch using the returned searchid value in the result set. The end of the result set is reached when all bins contain zero assets.

The set of assets returned are organized in bins. The assets within a bin are returned in decreasing relevancy ranking order.This function supports the Simple-user-attributes option of the automatic-user-registration-or-update capability (see User Resgistration).

Inputs:

- customerId
- searchPhrase
- languageCode
- userName
- groupCode (Optional. Ignored if user already is registered)
- password (Optional. Ignored if user already is registered)
- enable508

Outputs:

- A portion of the result set organized as a list of bins.
- Each bin will have its name and various meta data values.
- Each bin will list its associated assets in decreasing relevancy rank order.
- Each asset will include its ID, relevancy rank value and various meta data values.
- See the OLSA WSDL for complete details.

Additional Faults

- None

SL_DetailedSearch

Given a search string and a high-level category, search results are returned from a single high-level category (bin). In addition, more detailed information is included for each result. For example, a course hit might include its topic hits, while a book hit might include its chapter hits.

This function initiates a search and returns the initial portion of the matching result set. The caller can page through the remainder of the result set by issuing a call to SL_PaginateSearch using the returned searchid value in the result set. The end of the result set is reached when all bins contain zero assets.

The returned assets are organized in bins. The assets within a bin are returned in decreasing relevancy ranking order.This function supports the Simple-user-attributes option of the automatic-user-registration-or-update capability (see User Resgistration).

Inputs

- customerId
- searchPhrase
- binName
- count (a value between 5-10 is recommended)
- languageCode
- userName
- groupCode (Optional. Ignored if user already is registered)
- password (Optional. Ignored if user already is registered)
- enable508

Outputs

- A portion of the result set organized as a list of bins.
- Each bin will have its name and various metadata values.
- Each bin will list its associated assets in decreasing relevancy rank order.
- Each asset will include its ID, relevancy rank value and various metadata values.
- See the OLSA WSDL for complete details.

Additional Faults

- None

SL_RelatedSearch

This function performs a related search. A related search means internally generated keywords associated with the specified asset are used in the search to find related assets. Given an ID for an asset it will compute its associated keywords and issue a search using these keywords, returning any related assets. Associated keywords are automatically built from the metadata of the asset.

This function initiates a search and returns the initial portion of the matching result set. The caller can page through the remainder of the result set by issuing a call to PaginateSearch using the returned searchid value in the result set. The end of the result set is reached when all bins contain zero assets.

The set of assets returned are organized in bins. The assets within a bin will be returned in decreasing relevancy ranking order.This function supports the Simple-user-attributes option of the automatic-user-registration-or-update capability (see User Resgistration).

Inputs:

- customerId
- searchAssetId
- userName
- groupCode (Optional. Ignored if user already is registered)
- password (Optional. Ignored if user already is registered)
- enable508

Outputs:

- A portion of the result set organized as a list of bins.
- Each bin will have its name and various meta data values.
- Each bin will list its associated assets in decreasing relevancy rank order.
- Each asset will include its ID, relevancy rank value and various meta data values.
- See the OLSA WSDL for complete details.

Additional Faults

- None

SL_PaginateSearch

This function does not initiate a search but instead continues a search initiated by SL_FederatedSearch, SL_DetailedSearch or SL_RelatedSearch (specified by the searchId). Note you may only paginate through one bin at a time.

The set of assets returned are organized in bins. The assets within a bin will be returned in decreasing relevancy ranking order. The caller can page through the result set by issuing a call to PaginateSearch using the returned searchid value in the result set. The end of the result set is reached when all bins contain zero assets.

Inputs:

- customerId
- searchId
- binName
- start
- count (a value between 5-10 is recommended)
- enable508

Outputs:

- A portion of the result set organized as a list of bins.
- Each bin will have its name and various metadata values.
- Each bin will list its associated assets in decreasing relevancy rank order.
- Each asset will include its ID, relevancy rank value and various meta data values.
- See the OLSA WSDL for complete details.

Additional Faults

- None

SL_GetAttributes

This function returns a variety of attributes that can be used as criteria in various search functions. The attributes returned are scoped in a fashion similar to how assets returned in search results are scoped. The attributes are:

- A list of the language codes for all assets entitled to the specified user
- A list of ISO standard codes in the following format <languageCode><countryCode>

  <languageCode> is a two letter code in all lower-case.

  <countryCode> is a two letter code in all upper-case.
- A list of the asset types for all assets entitled to the specified user
- A list of asset types and their associated printable names (for display purposes)
- The bin to asset type mappings entitled to the user.

A list of bins is returned, with each bin specifying the asset types that are mapped to it. The list of bins can be used to determine what bins can be searched for a given user with the SL_DetailedSearch function. This function supports the Simple-user-attributes option of the automatic-user-registration-or-update capability (see User Resgistration).

Inputs:

- customerId
- username
- groupCode (Optional. Ignored if user already is registered)
- password (Optional. Ignored if user already is registered)

Outputs:

- See the OLSA WSDL for a complete description.

Additional Faults

- None

SL_GetSearchParameter

This function returns the default Search Parameter configuration for the specified customer.

Inputs:

- customerId

Outputs:

- See the OLSA WSDL for a complete description.

Additional Faults

- None

SL_SetSearchParameter

This function allows the caller to set the default Search Parameter configuration for the specified customer.

Inputs:

- customerId
- searchParameter (see the OLSA WSDL for a complete description)

Outputs

- None

Additional Faults

- None

SL_GetAssetDetail

This function provides the metadata for a specified asset plus the metadata for all its subordinate nested assets. For example, this allows the display of a book's table of contents.

Inputs:

- customerId
- assetId
- username
- searchId (optional)

Outputs:

- Metadata for the asset (see the OLSA WSDL for a complete description)

Additional Faults

- None

Assignment Service (AS_)

This service allows a customer-application to control the assignment of Referenceware and Courseware assets. Assignments control what assets are made visible to a given user or user group (see also Entitlement/Assignment definition). For example, if a search is issued using a given keyword OLSA does not return all assets that match the specified keyword. It scopes the results and returns only those assets that match the specified keyword and are part of the user's assignment. Assignment are also used to properly meet any SkillSoft content licensing agreements (for example, Referenceware).

Assignment Types

- **Collection assignment**: Assignment of Referenceware collections to users and user groups. A collection is a predefined grouping of multiple book assets (for example, IT Pro). Collections are a flat list of books; they are not hierarchical. Assignment of individual books is not supported.
- **Catalog assignment**: Assignment of asset groups or individual assets to users and user groups. An asset group can contain sub-asset groups to create a hierarchical tree structure.
- **Default assignment**: Users get the default assignment if not given a specific asset or collection assignment. The default assignment gives each user access to all Referenceware and Courseware assets that the customer is entitled to.

AS_GetSubscriptionData

This function retrieves information regarding the Referenceware subscription. The subscription is established outside of OLSA, typically during the initial set-up of OLSA with Referenceware. A customer with Referenceware access will have at least one subscription. The information returned will be zero, one, or more the one subscription descriptions.

Each subscription description includes:

- Subscription ID
- List of associated collection descriptions

Each collection description includes:

- Collection ID
- Display name for the collection

Inputs

- customerId

Outputs

- List of subscription descriptions, each containing a list of collection descriptions (see OLSA WSDL for complete details).

Additional Faults

- None

AS_SetCollectionAssignment

This function assigns collections to a user group. Any assignment to a user group is inherited by all users in the user group, including all of its sub groups. A direct assignment to a user group overrides any assignments inherited from any parent user group. To add a collection to a user group's current assignment, you must first get the list of assigned collections, add the collection to the list, and then assign the entire resulting new list.

Special Collection IDs

The following special collection IDs are also supported:

- _ALL_ the user group is assigned all collections associated with the subscription ID. The difference between ALL and using a fixed list of collections is that when list of collections within a subscription changes then ALL will automatically scope the affected user group accordingly.
- _NONE_ the user group does not get access to any collections regardless of the collection assignment of its parent user group.
- _DEASSIGN_ the user group has its assignment cleared. This results in the affected user group inheriting collection assignments from the parent user group (if any).

If a special collection ID is specified no other collection IDs may be specified in the same call. The command will override any previous SetCollectionAssignment. A GeneralFault is returned if the specified collection is not in the subscription associated with the user group.

Inputs

- customerId
- list of collectionIds (1 or more)
- groupCode (of user group to assign to)

Outputs

- None

Additional Faults

- None

AS_SetCollectionAssignmentByUser

This function assigns collections to an individual user. An individual user assignment override any inherited user group assignments. This command is the same in all respects as AS_SetCollectionAssignment, but it only affects an individual user. A GeneralFault is returned if the specified collection is not in the subscription associated with the user.

Inputs

- customerId
- list of collectionIds (1 or more)
- userName

Outputs

- None

Additional Faults

- None

AS_GetCollectionAssignment

This function queries the inherited and direct collection assignment of the specified user group.

Inputs

- customerId
- groupCode (of user group)

Outputs

- If the user group has no assignment (DEASSIGN) then a list with 0 collection elements is returned.

  ```
  <collections>

  </collections>
  ```

  If the user group has the special assignment NONE the following is returned:

  ```
  <collections>

              <collection><id>__NONE__</id></collection>

  </collections>
  ```

If the user group has the special assignment ALL the following is returned:

```
<collections>
            <collection><id>__ALL__</id></collection>
</collections>
```

If the user group has 1 or more collection assignments then the following is returned:

```
<collections>
            <collection id='someCollectionID1' inherited='1'/>
            <collection id='someCollectionID2'/>
            … any additional collection assignments…
</collections>
```

Note inherited collections assignments are indicated by the presence of the inherited attribute. See OLSA WSDL for complete description.

Additional Faults

- None

AS_GetCollectionAssignmentByUser

This function queries the inherited and direct collection assignment of the specified individual user. This command is the same in all respects as AS_GetCollectionAssignment, but it only applies to the specified user.

Inputs

- customerId
- userName

Outputs

- See AS_GetCollectionAssignment

Additional Faults

- None

AS_SetCatalogAssignment

This function assigns a catalog path to a user group. A catalog path is a special identifier that denotes a specific asset group within a courseware hierarchy. A catalog path identifies all courseware within the specified asset group and its sub asset groups recursively. Any assignment to a user group is inherited by all users in the user group (done recursively through all sub groups). An assignment to a user group overrides any assignments inherited from a parent user group.

To add a catalog path to a user group's current assignment, you first get the list of assigned catalog paths, add to this list, and then assign the entire resulting new list. This command overrides any previous AS_SetCatalogAssignment.

Setting catalogPaths to zero deassigns any assignment on the specified user group. A user group with no assignments inherits its assignment from the first parent user group with a direct catalog assignment.

Inputs

- customerId
- list of catalogPaths (0, 1 or more)
- groupCode (of user group)

Outputs

- None

Additional Faults

- None

AS_SetCatalogAssignmentByUser

This function assigns a catalog path to an individual user. This command is the same in all respects as AS_SetCatalogAssignment, but it only applies to the specified user.

Inputs

- customerId
- list of catalogPaths (0, 1 or more)
- userName

Outputs

- None

Additional Faults

- None

AS_GetCatalogAssignment

This function queries the inherited and direct catalog assignment of the specified user group.

Inputs

- customerId
- groupCode (of user group)

Outputs

- If the user group has no catalog assignment then a list with zero catalog elements is returned.

  ```
  <catalogs>
  </catalogs>
  ```

  If the user group has a direct catalog assignment then a list with 1 or more catalog elements is returned.

  ```
  <catalogs>
        <catalog id='someCatalogPathID1' inherited='1'/>
        <catalog id='someCatalogPathID2'/>
        ... any additional catalog assignments...
  </catalogs>
  ```

Note inherited collections assignments are indicated by the presence of the inherited attribute. See OLSA WSDL for complete description.

Additional Faults

- None

AS_GetCatalogAssignmentByUser

This function queries the inherited and direct catalog assignment of the specified user. This command is the same in all respects as AS_GetCatalogAssignment, but it only applies to the specified user.

Inputs

- customerId
- userName

Outputs

- See AS_GetCatalogAssignment

Additional Faults

- None

User Management Service (UM_)

This service allows a customer-application to perform various user management functions on the OLSA environment. These include the capability to:

- create, edit, and delete users
- create, edit, and delete user groups.

A user can be a member of multiple user groups. A user group can contain users and sub-groups. A user group can be a member of only a single parent user group.

Registering users and defining user groups may be necessary to properly establish catalog and collection assignments. Assignment can scope the contents of Search & Learn results for individual users. Assignment can also be used to properly meet any SkillSoft content licensing agreements (Referenceware in particular).

There are also some functions defined in the OLSA Web Services that support automatic user registration. This capability automatically creates a user (from provided user-credentials) if the user does not yet exist in the OLSA environment. If this is not sufficient to meet your needs then the User Management service provide additional capabilities.

UM_CreateUser

This function creates a new user in the OLSA environment. A user is created in a single group via this API.

Inputs

- customerId
- See General User Attributes for the remaining agreements

Outputs

- None

Additional Faults

- ObjectExistsFault (user already exists)

UM_EditUser

This function edits various attributes of an existing user in the OLSA environment. A user can only be moved to a single different group through this API. A user can be deactivated with this function by setting the active field to zero (0). This disables the user but retains any of the usage data managed by OLSA.

Inputs

- customerId
- newUserName
- See General User Attributes for the remaining arguments

Outputs

- None

Additional Faults

- ObjectNotFoundFault (user does not exist)

UM_DeleteUser

This function deletes an existing user from the OLSA environment. If the specified user is a member of multiple groups, that user is deleted from all user groups. Deleting a user also deletes any associated usage data managed by OLSA.

Inputs

- customerId
- userName

Outputs

- None

Additional Faults

- ObjectNotFoundFault (user does not exist)

UM_CreateUserGroup

This function creates a new user group in the OLSA environment. A user group can only be created in a single parent group.

Inputs

- customerId
- groupCode
- groupTitle
- parentGroupCode

Outputs

- None

Additional Faults

- ObjectExistsFaults (user group already exists)

UM_EditUserGroup

This function edits an existing user group in the OLSA environment. A user group can only be moved to a single different group.

Inputs

- customerId
- groupCode
- newGroupCode (internally this will be used as the group name as well) (optional)
- newParentGroupCode (optional)
- newTitle (optional)

Outputs

- None

Additional Faults

- ObjectNotFound (user group does not exist)

UM_DeleteUserGroup

This function deletes an existing user group in the OLSA environment. All sub-user groups (if any) are deleted recursively. Any affected users are deleted if they have no remaining parent user group. All usage data for the user is deleted as well.

Inputs

- customerId
- groupCode

Outputs

- None

Additional Faults

- ObjectNotFound (user group does not exist)

UM_AddUserToGroup

This function adds a user to an existing user group in the OLSA environment. A user can be added to multiple groups.

Inputs

- customerId
- userName
- List of groupCodes

Outputs

- None

Additional Faults

- ObjectExistsFault (user in already in group)
- ObjectNotFoundFault (user or group not found)

UM_RemoveUserFromGroup

This function removes a user from an existing user group(s) in the OLSA environment. A user must always be a member of at least one group. This command does not delete the user.

Inputs

- customerId
- userName
- List of groupCodes

Outputs

- None

Additional Faults

- ObjectNotFoundFault (user not in group, or group does not exist)

UM_ InitiateUserListingByGroupReport

This function retrieves information about the user population registered in the OLSA environment. The user population to list can be specified with the following options:

- allUsers (when true it means all top-level groups will be selected and subGroupName must be empty)
- subGroupName (when non-empty it means use the named subgroup, allUsers must be set to false)
- includeSubGroups (true means include all users in the selected group's subgroups)
- listBySubgroups (true means organize users by sub groups)

The report is returned in HTML or CSV format. Listed below is a sample user listing by group CSV report fragment with allUsers, includeSubGroups and listBySubgroups all set to true. Not all CSV fields are shown.

GroupTitle	GroupPath	groupCode	LoginName	LastName	FirstName
SkillSoft	/SkillSoft	SkillSoft	Admin	Admin	Admin
SkillSoft	/SkillSoft	SkillSoft	Smith1	Smith1	John
HR Team	/SkillSoft/HR Team	HR_TEAM	Jones1	Jones1	Fred
Books24x7	/Books24x7	Books24x7	Jones2	Jones2	Bill
Books24x7	/Books24x7	Books24x7	Smith2	Smith2	Sally

Table 6. Sample User Listing by Group Report

The above reports assumes the following user population that has been registered into the OLSA environment.

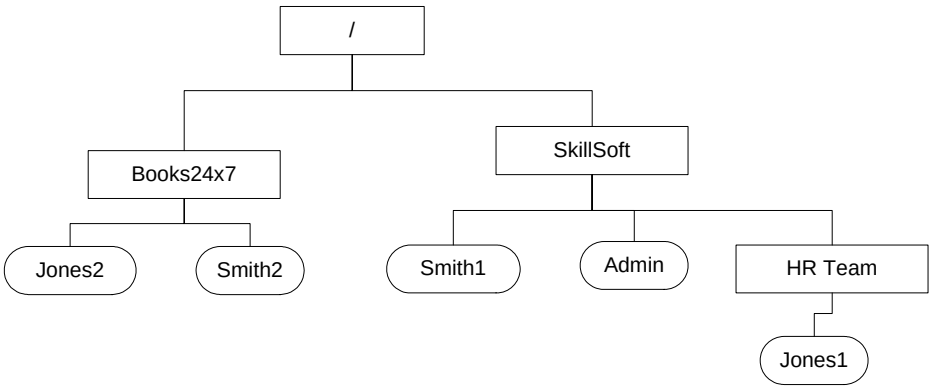

Figure 5. Sample User Hierarchy

Inputs

- customerId
- allUsers (true or false)
- subGroupName (non-empty only if allUsers==false)
- includeSubGroups (true or false)
- listBySubgroups (true or false)
- reportFormat (HTML or CSV)

Outputs

- A report ID handle. This value should be used with the UTIL_PollForReport function to get the actual contents of the report.

Additional Faults

- None

Offline Integration Service (OF_)

This service allows a customer-application, e.g., a non AICC compliant portal, to manage access to the following launch-related capabilities:

- Perform the download of the SkillSoft Course Manager (SCM) application and an asset for offline play.
- Perform the upload of offline usage data to be automatically managed by the OLSA environment.

Offline usage data can only be sent back to the OLSA Environment. A customer that needs to keep usage data between the customer-application and the OLSA environment synchronized may use the Usage Data Synchronization service.

OF_GetDownloadAssetUrl

This function returns a launch URL that return client-side logic that activates the download of the specified asset for offline play. This client side logic includes SCM detection logic, if the SCM is not installed then it will initiate a SCM installation sequence on the client. Note the limitations on launch URLs.

Inputs

- customerId
- assetId
- username
- x508

Outputs

- A URL that will download the specified asset with the specified user as its session context.

Additional Faults

- DownloadNotEnabledFault: If the specified asset is not downloadable.

OF_GetUploadOfflineDataUrl

This function returns a launch URL that returns client-side logic that uploads usage data to the OLSA environment that was generated during offline play of downloaded assets. This client side logic includes SCM detection logic (if the SCM is not installed then no operation is performed). Note the limitations on launch URLs.

Inputs

- customerId
- userName

Outputs

- A URL that will perform an SCM upload using the specified user as its session context

Additional Faults

- None

Usage Data Synchronization Service (UD_)

The Usage Data Synchronization service allows a customer-application to access usage data from the OLSA environment for content managed via the Asset Integration Service. This allows a customer-application to keep the usage data for a given asset synchronized with the same asset in the OLSA environment.

Situations where a customer-application may find a need for this service are when the following value-add learning services are used and usage data is automatically managed by the OLSA environment:

- Offline usage data uploaded with the Offline Integration service
- Assets launched using Search & Learn launch URLs
- Assets launched from UI screens generated via the SignOn Service.

The capabilities defined in this service include:

- Accessing all usage data for a single user accessing a single asset
- Accessing a custom report for one or more users.

UD_GetAssetResults

This function returns all of the usage results for a user accessing a specific asset. If the asset is not specified the function returns results for all courses taken by the specified user. If summaryLevel=true only course level results are returned. If summaryLevel=false, course and lesson level results are returned (for courses that support lesson level results).

Inputs:

- customerId
- userName
- assetId (optional)
- summaryLevel (true or false)

Outputs:

- A list of result elements. See the OLSA WSDL for complete details.

Additional Faults

- NoResultsAvailableFault: If there are no results to return for the specified user.

UD_InitiateCustomReportByUsers

This function initiates the custom report to retrieve usage data for specified users. If username, firstname, and lastname are all omitted then all users are processed. If username, firstname or lastname are specified then all users that have a corresponding prefix-match (e.g., username of “smith” will match usernames “smith1” and “smith2”) are processed.

Usage data can be further filtered with the optional start date, end date and date modifier arguments.

Inputs:

- customerId
- Username (optional)
- Firstname (optional)
- Lastname (optional)
- Start date (optional)
- End date (optional)
- Date modifier (one of the following, ignored if start and end dates are both omitted)
    - “any” (any access date, this is the default)
    - “first” (first access date only)
    - “most” (most recent access date only)
    - “completion” (completion date only)
- listBy (one of the following)
    - “user” (details by user)
    - “asset” (details by asset)
- includeDeactivatedUsers (default is true)
- reportFormat (HTML or CSV)

Outputs:

- A report ID handle. This value should be used with the UTIL_PollForReport function to get the actual contents of the report.

Additional Faults

- None

UD_InitiateCustomReportByUserGroups

This function initiates the custom report to retrieve usage data by user group. Usage data can be further filtered with the optional start date, end date and date modifier arguments.

Inputs:

- CustomerId
- Group (string, if this starts with a "/" then it is assumed to be a user group path. Otherwise it will be interpreted as a group code)
- Start date (optional)
- End date (optional)
- Date modifier (one of the following, ignored if start and end dates are both omitted)
    - "any" (any access date, this is the default)
    - "first" (first access date only)
    - "most" (most recent access date only)
    - "completion" (completion date only)
- listBy (one of the following)
- user (details by user)
- asset (details by asset)
- includeDeactivatedUsers (default is true)
- includeSubgroups (default is true)
- reportFormat (HTML or CSV)

Outputs:

- A report ID handle. This value should be used with the UTIL_PollForReport function to get the actual contents of the report.

Additional Faults

- None

SignOn Service (SO_)

This service allows users direct access into the SkillPort platform environment. This includes the ability to seamlessly login and land on the SkillPort Home, Catalog or My Plan pages, as well as a specified course summary screen. Usage data generated from assets launched through SkillPort sessions are synchronized automatically with the OLSA Environment

SO_GetMultiActionOnSignOnURL

This function returns a launch URL that lands the specified user into the SkillPort environment's Home, Catalog, or My Plan screen. It can also land the user in a summary screen for a specified asset or launch the specified asset itself.

Note the limitations on launch URLs. This function supports the All-user-attributes option of the automatic-user-registration-or-update capability (see User Registration).

Inputs

- customerId
- actionType (home, myplan, catalog, summary, launch)
- assetId (asset id for a course, this must be non empty only for action=summary and action=launch)
- newUserName
- See General User Attributes for remaining arguments
- enable508 (default is false)

Outputs

One of the following:

- A URL that will land the user on the Home page of the SkillPort environment.
- A URL that will land the user on the My Plan page of the SkillPort environment.
- A URL that will land the user on the Catalog page of the SkillPort environment.
- A URL that will land the user on the specified Course Summary page of the SkillPort environment.
- A URL that will launch the specified asset.

Additional Faults

- None

Utility Service (UTIL_)

This section defines the Utility Service and its associated functions.

UTIL_PollForReport

This function retrieves the URL for the completed report given a report ID (for example returned by UD_InitiateCustomReportByUsers).

Inputs

- customerId
- reportId

Outputs

- URL to a report file.

Additional Faults

- DataNotReadyYetFault (If the report file is not ready yet)
- ReportDoesNotExistFault (If the report corresponding to the reportId does not exist)

UTIL_GetMentoringUrl

If an asset (for example a Business Skills course) has mentoring enabled, this function returns a launch URL that can be used to access the mentoring service using the given asset as the session context. This is different than a mentoring asset that can be installed and launched like any other asset via the Asset Integration Service. Note the limitations on launch URLs.

Inputs

- customerId
- usernName
- assetId (for the asset that has mentoring enabled)

Outputs

- A URL that will launch the mentoring screen using the specified asset and user at its session context.

Additional Faults

- MentoringNotEnabledFault: If the specified asset does not have mentoring enabled.

bUTIL_DeleteReport

This function deletes a completed report given a report ID (for example returned by UD_InitiateCustomReportByUsers).

Inputs

- customerId
- reportId

Outputs

- None

Additional Faults

- DataNotReadyYetFault (If the report file is not ready yet)
- ReportDoesNotExistFault (If the report corresponding to the reportId does not exist)

Configuration Service (CF_)

This service allows a customer-application to configure the behavior of content installed via the Asset Integration service.

CF_GetAiccSettings

This function queries the currently established AICC settings that are not implemented as player properties. This setting controls what value is generated into the Max_normal field of the appropriate AICC .crs file:

- max_normal: This controls the max_normal value for all content (integer: 1-99)

These settings control what values are generated in the specified fields for the putParam command for special content (e.g. Mentoring):

- Lesson_status: This controls the Lesson_status value (string: “not attempted”, “incomplete”, “completed”, “browsed”)
- Time: This controls the time value (any time value of the form hh:mm:ss)

This setting controls whether SkillSoft Course Player (SCP)-based content is launched using a signed applet or not:

- enable_signed_player_applets (0 or 1)

Inputs

- customerId

Outputs

- The response will be returned in the following format:

```
<aicc_settings>
      <file_crs>
            <max_normal>…</max_normal>
      </file_crs>
<putparam>
            <lesson_status>…</lesson_status>
            <time>…</time>
</putparam>
      < enable_signed_player_applets> .. </ enable_signed_player_applets>
</aicc_settings>
```

- See OLSA WSDL for complete description.

Additional Faults

- None

CF_SetAiccSettings

This function modifies the currently established AICC settings that are not implemented as player properties. To change a particular setting value, the caller must first invoke CF_GetAiccFileSettings. The caller must then update the appropriate setting value(s) in the XML response from CF_GetAiccFileSettings. This updated XML must be specified in its entirety as the aiccFileSettings argument below (changed as well as unchanged values) in the call to CF_SetAiccFileSettings.

See CF_GetAiccSettings for allowed types and ranges for the supported AICC settings. Additional tags cannot be specified in this call. Only those tags supported by CF_GetAiccSettings are allowed.

Inputs

- customerId
- aiccSettings (see CF_GetAiccSettings and OLSA WSDL for more details).

Outputs

- None

Additional Faults

- None

CF_GetPlayerProperties

This function queries the currently established SkillSoft Course Player (SCP) properties. These properties control what launch configuration values are transmitted by OLSA to the SCP for any SCP-based content (for example, Business Skills, IT) launched via the Asset Integration Referral Object launch URL mechanism.

See the SCP documentation for the description of available properties. Each property is described with a property tag. The property tag has as attributes:

- The name of the property (read-only)
- The type of the property (read-only)
- The enable status of the property. 1 means the property value will be transmitted by OLSA to the Player. 0 means the property value will not be transmitted.
- The tag value is the value of the property.

Inputs

- customerId

Outputs

- The response is returned in the following format:

```
<player_properties>
      <property name="…name of property…"
     type="text|integer|float|boolean"
     enabled="0|1">…value for property…</property>
      … more properties …
</player_properties>
```

See OLSA WSDL for complete description.

Additional Faults

- None

CF_SetPlayerProperties

This function modifies the currently established player properties. These properties control what launch configuration values are transmitted by OLSA to the SCP for any SCP-based content (e.g., Business Skills, IT) launched via the Asset Integration Referral Object launch URL mechanism. To get the list of available properties the caller must invoke CF_GetPlayerProperties.

To set one or more available properties the caller should extract the desired subset of properties from the CF_GetPlayerProperties XML response. For each property the caller wants to "set", update the property's value and enabled status. This modified subset XML must then be specified as the playerProperties argument below in the call to CF_SetPlayerProperties.

A property must have its enabled attribute set to 1 for the value to be sent to the Player at launch time. The caller must not add properties beyond the available set. The caller must not change the name or type of an available property.

For example, if CF_GetPlayerProperties returns as a response:

```
<player_properties>
      <property name="a" type="text" enabled="0">item</property>
      <property name="b" type="integer" enabled="0">99</property>
      <property name="c" type="boolean" enabled="0">true</property>
</player_properties>
```

You can set the property “b” to the value 100 and enable it by sending this as the SCP properties value to CF_SetPlayerProperties:

```
<player_properties>
      <property name=”b” type=”integer” enabled=”1>100</property>
</player_properties>
```

Inputs

- customerId
- playerProperties (see CF_GetPlayerProperties and OLSA WSDL for more details).

Outputs

- None

Additional Faults

- None

CF_GetSkillSimProperties

This function queries the currently established SkillSim player properties. These properties control what launch configuration values are transmitted by OLSA to the SimPlayer for any SkillSim-based content launched via the Asset Integration Referral Object launch URL mechanism.

See the SkillSim documentation for the description of available properties. Each property is described with a property tag. The property tag has the following attributes:

- The name of the property (read-only)
- The type of the property (read-only)
- The enable status of the property. 1 means the property value will be transmitted by OLSA to the SkillSim Player. 0 means the property value will not be transmitted.
- The tag value is the value of the property.

Inputs

- customerId

Outputs

- The response will be returned in the following format:

```
<skillsim_properties>
   <property name="…name of property…"
     type="text| integer|float |boolean"
     enabled="0|1">…value for property…</property>
   … more properties …
</skillsim_properties>
```

See OLSA WSDL for complete description.

Additional Faults

- None

CF_SetSkillSimProperties

This function modifies the currently established SkillSim player properties. These properties control what launch configuration values are transmitted by OLSA to the SkillSim Player for any SkillSim-based content launched via the Asset Integration Referral Object launch URL mechanism. To get the list of available properties the caller must invoke CF_GetSkillSimProperties.

To set one or more available properties the caller should extract the desired subset of properties from the CF_GetSkillSimProperties XML response. For each property to "set", update the property's value and enabled status. This modified subset XML must then be specified as the skillsimProperties argument below in the call to CF_SetSkillSimProperties.

A property must have its enabled attribute set to 1 for the value to be sent to the SkillSim Player at launch time. The caller must not add properties beyond the available set. The caller must not change the name or type of an available property.

For example, if CF_GetSkillSimProperties returns as a response:

```
<skillsim_properties>
      <property name="a" type="text" enabled="0">item</property>
      <property name="b" type="integer" enabled="0">99</property>
      <property name="c" type="boolean" enabled="0">true</property>
</skillsim_properties>
```

You can set the property "b" to the value 100 and enable it by sending this as the playerProperties value to CF_SetSkillSimProperties:

```
<skillsim_properties>
      <property name="b" type="integer" enabled="1>100</property>
</skillsim_properties>
```

Inputs

- customerId
- skillsimProperties (see CF_GetSkillSimProperties and OLSA WSDL for more details).

Outputs

- None

Additional Faults

- None

Open Learning Services Portal (OLSP)

Overview

The Open Learning Services Portal (OLSP) is a front-end application that allows you to access a subset of the OLSA Web Services without interacting directly with the APIs. Use this portal to view and modify configuration settings specific to your OLSA implementation.

Login Page

Login validates that you have access to the OLSA system. Your User Name and Password are provided by SkillSoft.

Note: OLSP requires cookies to be enabled in your client browser.

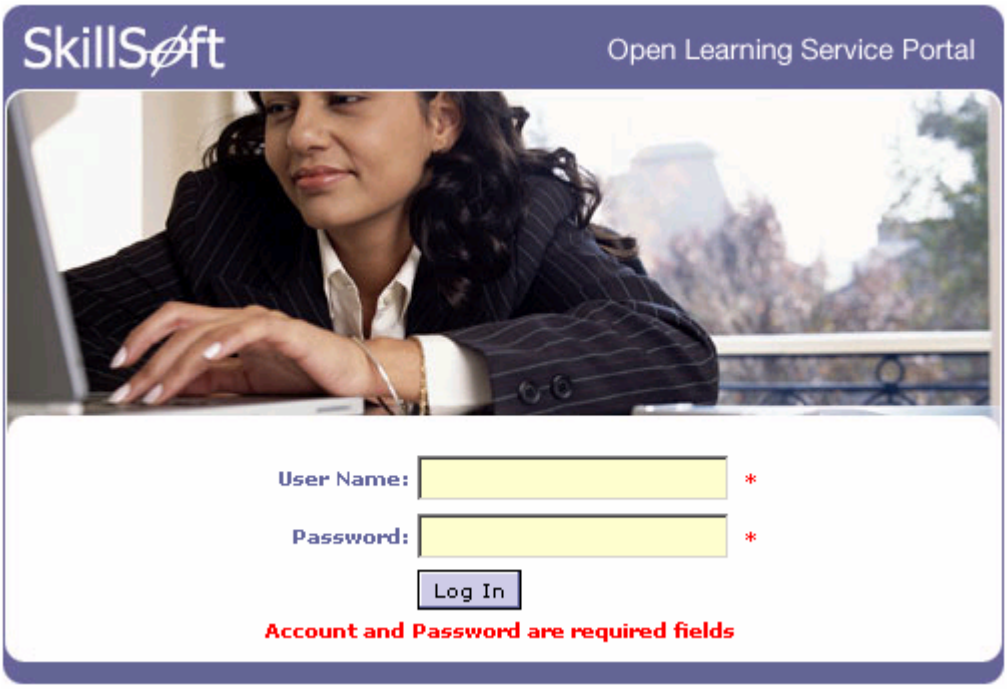

Figure 6: Login Page

AICC Configuration Options

The AICC Configurations Options page is the default page after login. Use the AICC Configuration Options page to:

- Set the Lesson Status and Session Duration for non-AICC conformant content.
- Set the `mastery_score` and `aicc_version` parameters in the generated AICC files.

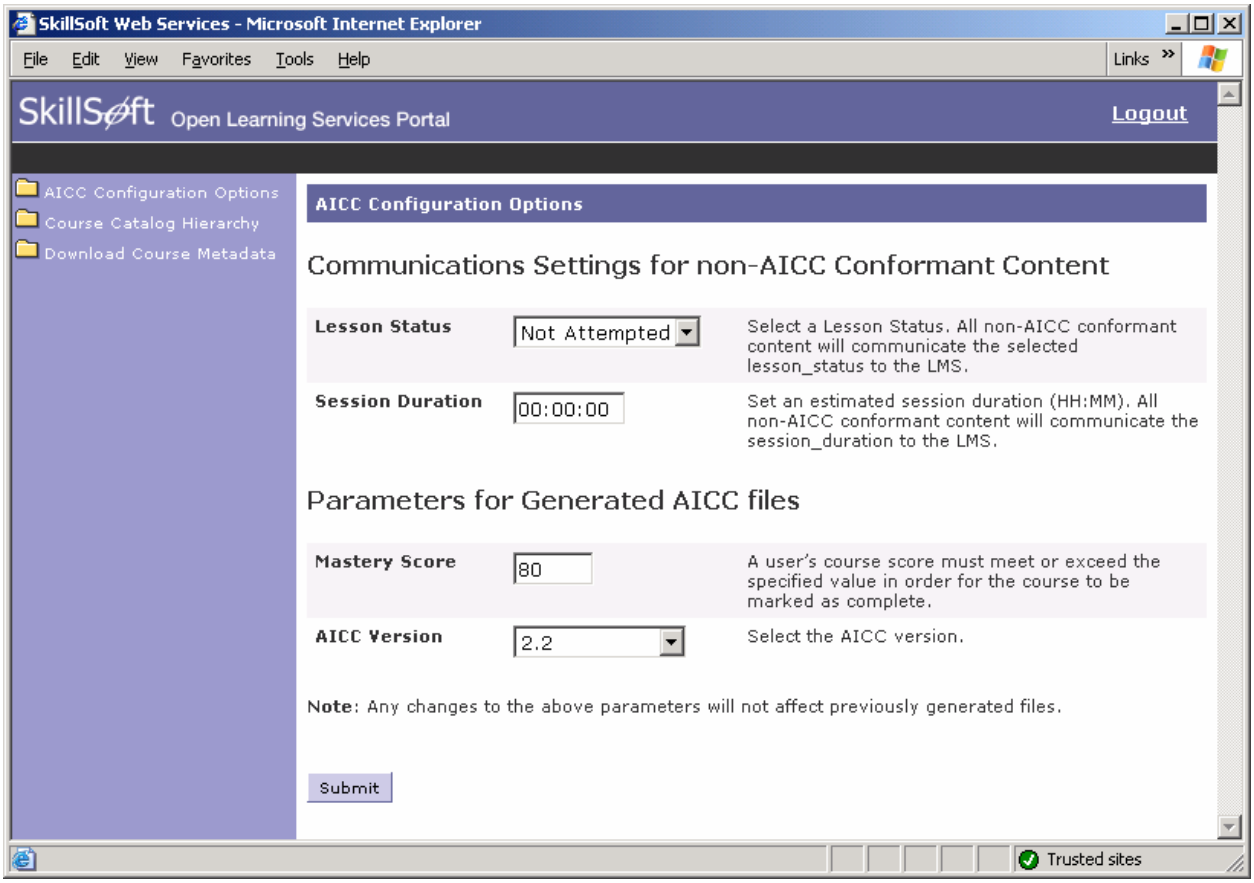

Figure 7: AICC Configuration Options

Lesson Status

All non-AICC conformant content communicates the selected Lesson Status to the LMS. This is a required value. Select one of the following options:

- Completed
- Not Attempted
- Browsed
- Incomplete

Session Duration

All non-AICC conformant content communicates the Lesson Status to the LMS. This is a required value that must be in HH:MM:SS format.

Mastery Score

The Mastery Score value is written into the generated AICC files. A user's score must meet or exceed the specified value in order for the course to be marked complete. This is a required value that can range from 0-100.

AICC Version

The AICC version value is written into the generated AICC files. This is a required value. Select one of the following options:

- 2.2
- 3.5

OLSA Calls

When you click **Submit**, the OLSP makes the following OLSA calls:

- CF_Get_Aicc_Settings
- CF_Set Aicc_Settings
- CF_Get_Player_Properties
- CF_Set_Player_Properties
- CF_Get SkillSim_Properties
- CF_Set SkillSim_Properties

Course Catalog Hierarchy

Use the Course Catalog Hierarchy page to generate a FullCourseListing report that describes all of the courses and assets you are entitled to. Completed reports are copied to the OLSP server or a specified network file location.

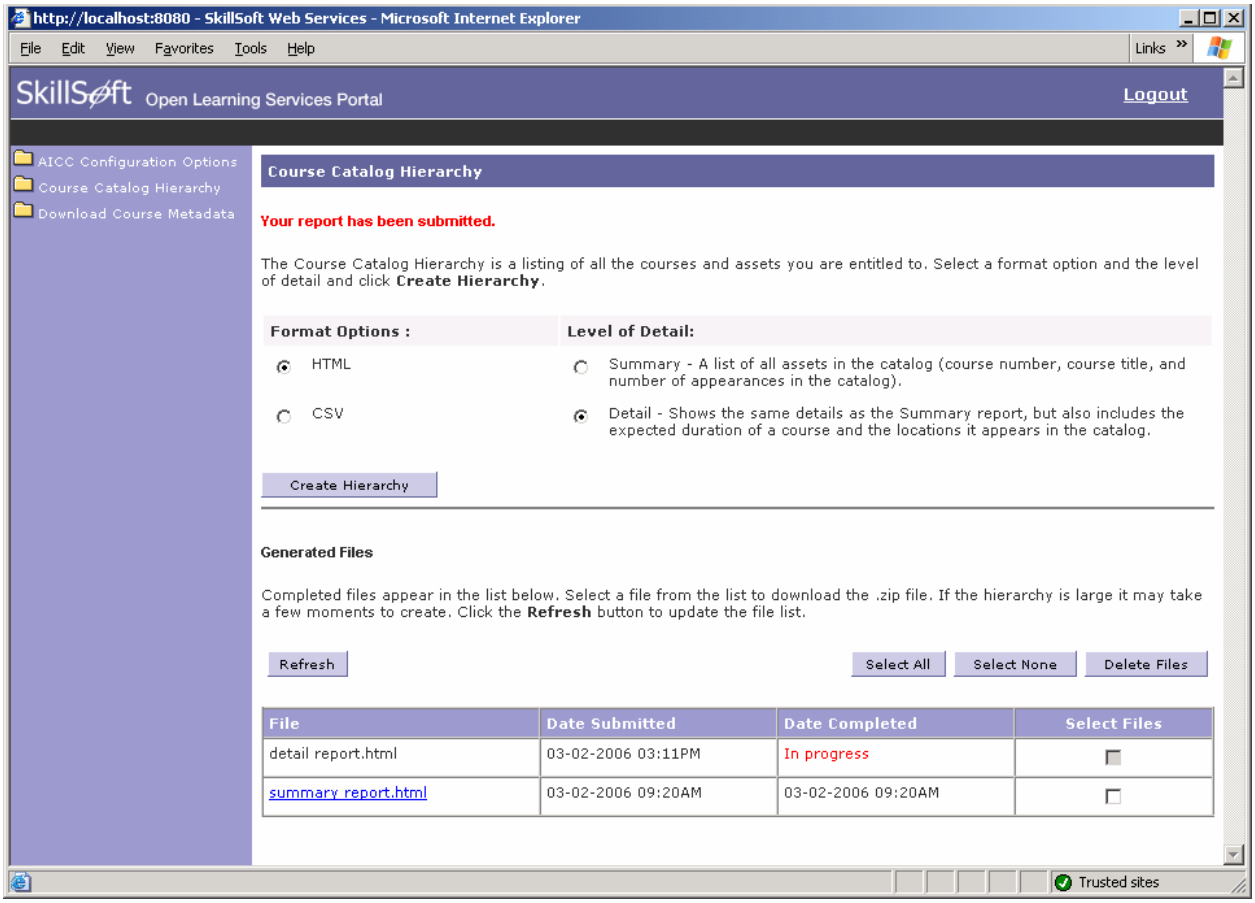

Figure 8: Course Catalog Hierarchy

Generate a Course Hierarchy Report

Select a **Format Option** (HTML or CSV16,UTF-16 format) and the **Level of Detail** (Summary or Detail) for the report and click **Create Hierarchy** to generate the report. The new report is added to the Generated Files list at the bottom of the page.

The filenames in the Generated Files list reflect the level of detail and format of the report (for example: detail report.html or summary report.csv). The Generated Files list also displays the date and time the report was submitted and when it was completed.

If a Report has not been finished, the date completed field will read `In progress.` Click **Refresh** to update the status of pending reports. A report is completed when the date and time appear in the Date Completed field.

To delete a report, click the **Selected Files** checkbox for that report. You can use the **Select All** button to select all files in the list or the **Select None** button to deselect all files in the list. Click **Delete Files** to remove the selected reports from the OLSP server or file location.

OLSA Calls

The OLSP makes a series of OLSA calls from this page.

- Click **Create Hierarchy**: AI_InitiateFullCourseListingReport
- Click **Refresh**: UTIL_PollForReport
- Click **Delete Files**: UTIL_DeleteReport

Download Course Metadata

Use the Download Course Metadata page to generate AICC metadata files. The metadata files for entitled assets are compressed into a zip file for download. There are two options for downloading metadata:

- Updated Metadata: downloads AICC filesets for newly released assets or for assets that have been updated since the last download.
- All Metadata: downloads all entitled AICC filesets.

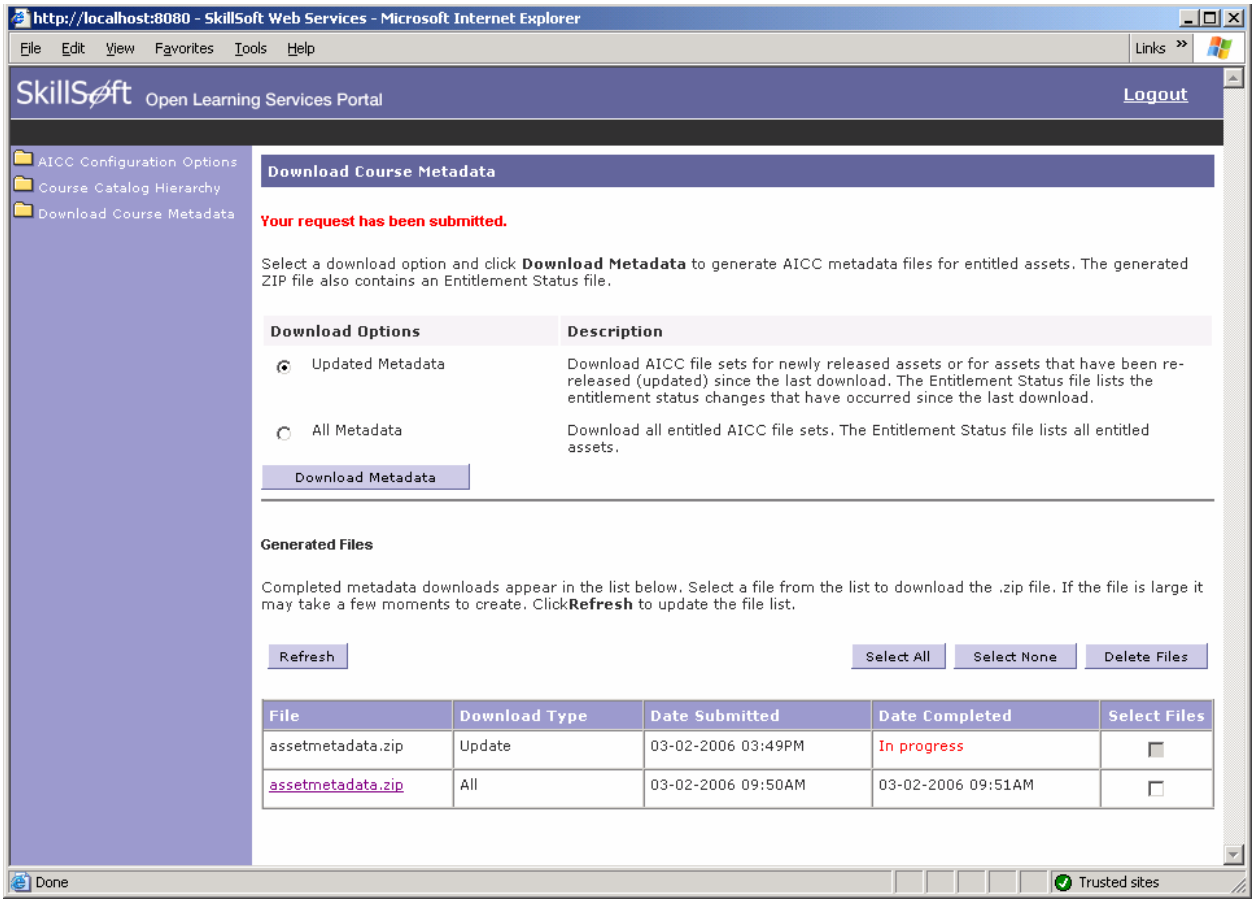

Figure 9: Download Course Metadata

Select a Download Option and click **Download Metadata.** The new zip file is added to the Generated Files list at the bottom of the page. The Download Type column indicates if the zip contains all metadata or updated metadata. The Generated Files list also displays the date and time the request was submitted and when it was completed.

If a metadata zip has not been finished, the date completed field will read `In progress.` Click **Refresh** to update the status of pending files. A file is completed when the date and time appear in the Date Completed field.

To delete a file, click the **Selected Files** checkbox for that file. You can use the **Select All** button to select all files in the list or the **Select None** button to deselect all files in the list. Click **Delete Files** to remove the selected files from the OLSP server or file location.

OLSA Calls

When you click **Download Metadata**, the OLSP makes the following OLSA calls to complete the transaction:

- AI_InitiateAssetMetaData
- AI_PollForAssetMetaData
- AI_AcknowledgeAssetMetaData