e-learning

OLSA Integration Guide 1.2 SR1

www.skillsoft.com

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

www.skillsoft.com http://www.skillsoft.com

Printed in the United States of America

Table of Contents

CHAPTER 1

Introduction

In This Chapter

WHAT IS OLSA?

Open Learning Services Architecture (OLSA) is a comprehensive service oriented architecture initiative 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 the LMS is entitled. 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. OLSA seamlessly sends the usage data in these situations 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. OLSA manages the usage data in these situations are automatically. 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.

CHAPTER 2

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*** **(**http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf \o http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf**).**

The OLSA Web Services optionally supports HTTPS, which increases the communication security 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.

OLSA Implicit Web Service Client-side Matrix

The following matrix specifies the combination of client side components supported for generating OLSA clients that invoke OLSA Web Services.

OLSA Client	JDK	AXIS	Visual Studio/WSE	.NET Framework
1.0	1.4 or 1.5	1.2	2003/2.0	1.1
1.1	1.4 1.5	1.2	2003/2.0 or 2005/3.0	1.1 2.0

In This Chapter

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, which is sent to OLSA. OLSA returns the appropriate response (HTML in this case) and renders the response 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 (http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf \o 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(SoapClien
t Message message, WebResponse response, Stream responseStream, Boolean
asyncCall) at
System.Web.Services.Protocols.SoapHttpClientProtocol.Invoke(StringmethodNam
e, 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 `newUsername`), 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

CHAPTER 3

Asset Integration Service (AI_)

The Asset Integration Service allows a customer-application to process programmatically assets and to install natively SkillSoft hosted content, including Referenceware, SkillView, Search & Learn, and Advanced Learning Structures. The service provides capabilities for automating the management of the assets that a customer-application is entitled. This includes the following abilities:

- 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 does not transfer to the customer-application environment.
- SkillSoft manages the configuration of player settings and player versions 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
- Dialogue Live
- Instructor Led Training
- JobAids
- Referenceware
- SkillView
- Search & Learn
- Advanced Learning Structures
- CCT course
- LOT
- Dialogue Recorded Sessions
- SCORM 1.2
- E3 Mix & Match
- Expert Sessions
- SkillBriefs

Topics, SkillBriefs, and Jobaids are included implicitly with their parent courseware, but you cannot individually install via Meta files generated by the Asset Integration Service.

Launching a Topic of Natively Installed Asset Integration Assets

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

http://launchURL?AICC_SID=sessionid&AICC_URL=ImsURL&topicid=theTopicIDToLaunch

The ***http://launchURL*** (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 AU.Web_Launch).

Launching A Section, Chapter Or Part Of A Natively Installed Books24x7 Assets

The Books24x7 asset supports the notion of a section, chapter, or part launch, all known as *chunks*. A chunk launch allows the end-user to access a specific place within the associated book. A book generates minimal tracking data; a chunk launch generates the same data for the associated book. You can perform a chunk launch on a natively installed book by appending the corresponding chunk ID argument to the usual AICC launch URL, using the same argument name topicid as described in the previous section). Example of an AICC chunk launches URL:

http://launchURL?AICC_SID=sessionid&AICC_URL=ImsURL&topicid=theChunkIDToLaunch

The ***http://launchURL*** *(*http://launchurl*)* would be the usual launch URL to launch a given course as specified in the AICC .au file for the relevant book. The text in bold above indicate the additional information to provide to launch within the specified chunk in the book.

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

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*** (http://www.section508.gov/)). To launch a natively installed asset with Web Accessibility enabled the following additional argument must be specified on the usual AICC launch URL. The following example shows an AICC Web Accessibility enabled launch URL, the &x508=1 indicates Web Accessibility enabled:

http://launchURL?AICC_SID=sessionid&AICC_URL=ImsURL&x508=1

Use the ***http://launchURL*** (http://launchurl) to launch a given asset as specified in the AICC .au file for the relevant course.

The following functions also support indicating the desired 508 mode for a user. These additional functions support an enable508 argument, which has the same semantics as the x508 argument in the RO launch URL:

- SL_FederatedSearch
- SL_DetailedSearch
- SL_RelatedSearch
- SL_PaginateSearch
- OF_GetDownloadAssetUrl
- SO_GetMultiActionOnSignOnUrl

There is an OLSA backend setting to enable 508 support company-wide. The table below illustrates the relationship between this company-wide setting and the x508 RO launch URL argument and the additional function's enable508 argument.

X508 argument on RO launch URL and additional functions' enable508 argument	Company-wide 508 setting	Notes
Explicitly set to 1	n/a	Company-wide setting ignored
Explicitly set to 0	n/a	Company-wide setting ignored
Omitted	0	Assumes the user is non-508
Omitted	1	Assumes the user is 508

Section, Chapter Or Part Launch Of A Natively Installed Books24x7 Asset

The Book asset supports the notion of a section, chapter, or part launch, all known as *chunks*. A chunk launch allows the end-user to access randomly a specific place within the associated book. A book generates minimal tracking data; a chunk launch will generate the same data for the associated book. You can perform a chunk launch on a natively installed book by appending the corresponding chunk ID argument to the usual AICC launch URL.

For example in AICC the chunk launch URL would look something like this:

`http://launchURL?AICC_SID=sessionid&AICC_URL=lmsURL&chunkid=theChunkIDToLaunch`

The ***http://launchURL*** *(*http://launchurl*)* would be the usual launch URL to launch a given course as specified in the AICC .au file for the relevant book. The text in bold above indicate the additional information to provide to launch within the specified chunk in the book.

Note: The AU.Web_Launch value would also have to be included if defined.

In This Chapter

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 - This refers to the sname provided by SkillSoft
- handle - returned by a AI_InitiateAssetMetaData call, optional

Outputs

- None

Additional Faults

- None

AI_ADDASSETTOGROUP

This function adds assets to any asset group. An asset can be in more than one asset group. See also ***AI_InitiateMakeChangesVisible*** (on page 37).

Inputs

- customerId - This refers to the sname provided by SkillSoft
- catalogPath - for relevant Asset Group
- assetId

Outputs

- None

Additional Faults

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

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, for example:

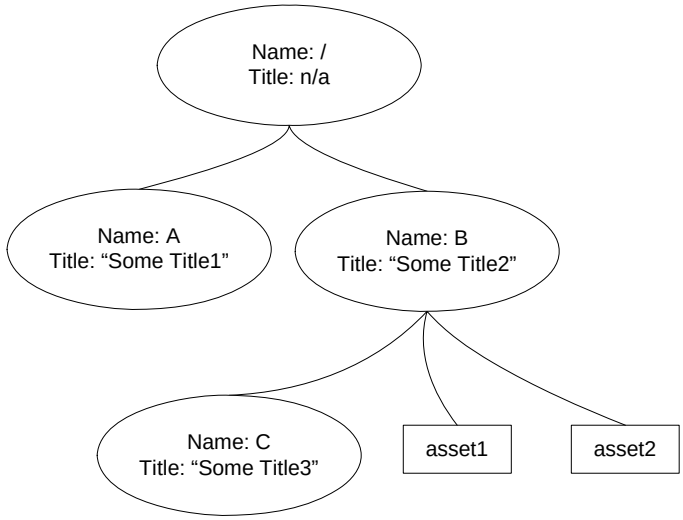


Figure 4: Catalog Group Structure

The top of the catalog structure always has the default super parent group /. There are 3 assets groups under /. Table 4. Definition of the Group Structure lists the properties of each asset group

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, you can use the following arguments:

- 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*** (on page 37).

Inputs

- customerId - This refers to the sname provided by SkillSoft
- parentCatalogPath
- assetGroupName
- assetGroupTitle

Outputs

- None

Additional Faults

- ObjectExistsFault

AI_GETCSVASSETMETADATA

The AI_GetCsvAssetMetaData function allows the customer to retrieve a SkillSoft proprietary comma-separated-value (CSV) formatted metadata on a single asset. The assetId argument specifies the asset to retrieve.

This function does not impact calls to AI_InitiateAssetMetaData, using the “all” or “delta” modes.

The caller should program itself to ignore all fields it does not want to use. OLSA could add new fields in the future, all new fields are always added to the end of the comma-separated-value list.

The following example shows a CSV result with each italicized name substituted with the actual value:

identifier, title, description, type, language, publisher, rights, isbn, creator, version, audience, duration, prerequisites, lesson, launchurl, launchType

Inputs

- customerId
- assetId

Outputs

- csv

AI_GETXMLASSETMETADATA

The AI_GetXmlAssetMetaData function allows the customer to retrieve a SkillSoft proprietary XML formatted metadata on a single asset. The assetId argument specifies the asset to retrieve.

This function will not have any impact on calls to AI_InitiateAssetMetaData using the "all" or "delta" modes.

Inputs

- customerId
- assetId

Outputs

- metadata

For more information about the MetaData format, refer to GetXmlAssetMetaDataResponse element in OLSA WSDL.

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*** (on page 37).

Inputs

- customerId - This refers to the sname provided by SkillSoft
- catalogPath - for Asset Group to delete

Outputs

- None

Additional Faults

- ObjectNotFoundFault

AI_EDITASSETGROUP

This function modifies the properties of an existing asset group. You can change the title or parent of an asset group with this command. See also ***AI_InitiateMakeChangesVisible*** (on page 37).

Inputs

- customerId - This refers to the sname provided by SkillSoft
- catalogPath - for Asset Group to Modify
- parentCatalogPath - new parent catalog path - optional
- assetGroupTitle - new group title - optional

Outputs

- None

Additional Faults

- ObjectNotFoundFault

AI_GETAICCASSETMETADATA

The AI_GetAiccAssetMetaData function allows the customer to retrieve AICC metadata on a single asset. The assetId argument specifies the asset to retrieve.

This function does not affect calls to AI_InitiateAssetMetaData, whether using the “all” or “delta” modes.

The XML returned has tags defined to describe the different AICC files such as CRS, CST, AU, DES and ORT. The following example located in Output shows the response returned from this function.

Inputs

- customerId - This refers to the sname provided by SkillSoft
- assetId

Outputs

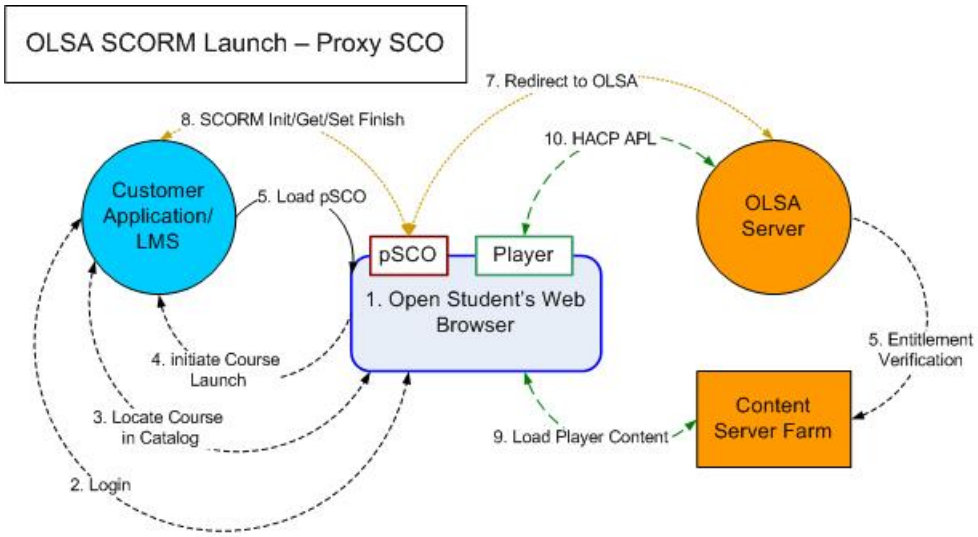
```
<aiccmetadata id='assetid'>
  <crs> <![CDATA[<crs file as string>]]></crs>
  <au>  <![CDATA[<au  file as string>]]></au>
  <des> <![CDATA[<des file as string>]]></des>
  <cst> <![CDATA[<cst file as string>]]></cst>
  <ort> <![CDATA[<ort file as string>]]></ort>
</aiccmetadata>
```

Additional Faults

- ObjectNotFoundFault - this indicates the asset does not exist.

AI_GETSCORMASSETMETADATA

This web service call generates the required manifest file and html file required in creating the PIF file. The proxy SCO performs LMSInitialize – LMSGetValue - LMSSetValue – LMSFinish calls to get appropriate data from the TPLMS and passes it to the redirect URL for further OLSA processing.



1. Open Browser.
2. Login into the Learning Management System.
3. Locate Course in the Learning Management System.
4. Initialize Course Launch in the Learning Management System.
5. Learning Management System presents the Proxy SCO, which is imported as a PIF.
6. Proxy SCO initializes the Learning Management System and then performs the Get/Set/Finish to close the session.
7. Proxy SCO uses the User ID from the "Get" and redirects the student to the OLSA platform.
8. OLSA performs the normal launch procedures, such as Entitlement.
9. Course player and content loads to the browser.
10. Player interfaces with the OLSA platform using Respective Learning Object API.

Inputs

- customerId
- assetId

Outputs

- manifest
- proxy sco
- format

For more information about the GetScormAssetMetaDataRequest and GetScormAssetMetaDataResponse, see the OLSA WSDL for further details.

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*** (on page 29) - one call to open the transaction
- ***AI_PollForAssetMetaData*** (on page 38) - one or more calls
- ***AI_AcknowledgeAssetMetaData*** (on page 22) - 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.

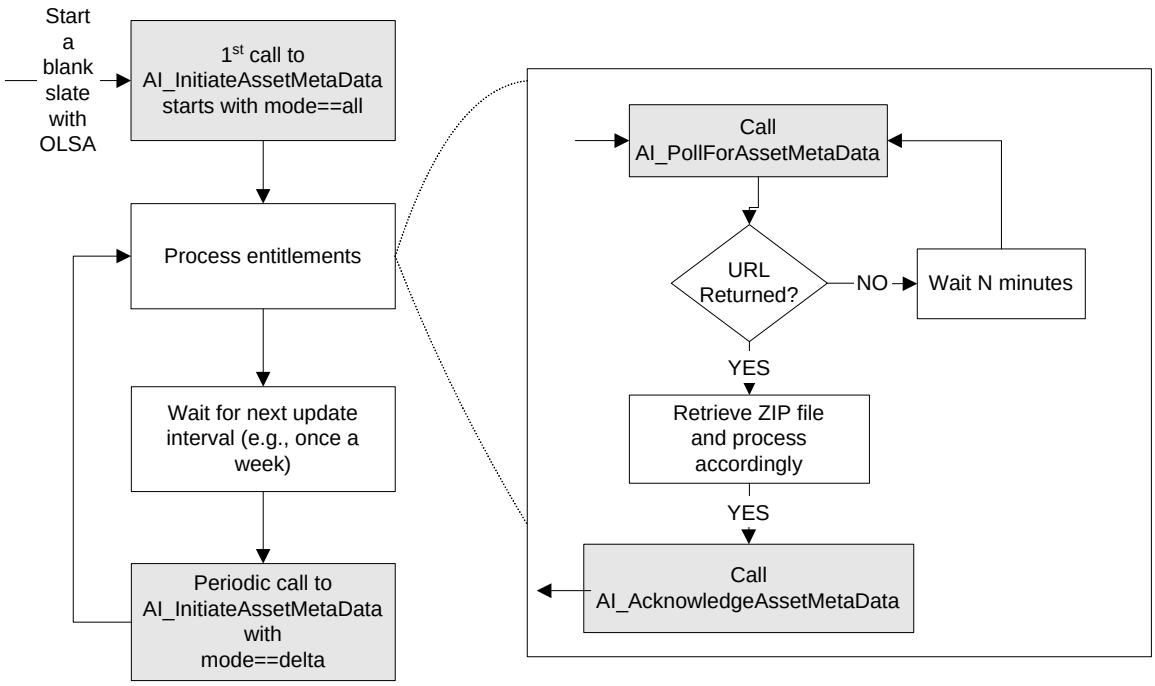

Start a blank slate with OLSA
1st call to AI_InitiateAssetMetaData starts with mode==all
Process entitlements
Wait for next update interval (e.g., once a week)
Periodic call to AI_InitiateAssetMetaData with mode==delta
Call AI_PollForAssetMetaData
URL Returned?
NO
Wait N minutes
YES
Retrieve ZIP file and process accordingly
YES
Call AI_AcknowledgeAssetMetaData

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

You can use the AI_InitiateAssetMetaData function, which returns a handle 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. You must make a call to AI_PollForAssetMetaData to retrieve the requested information.

AICC File Generation for Books24x7

Enabling Books24x7

SkillSoft has an OLSA setting specific to enable Books24x7 for AICC files. SkillSoft can set up and enable this action on the customer site if they have Books24x7 installed.

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 following image shows an set up example:

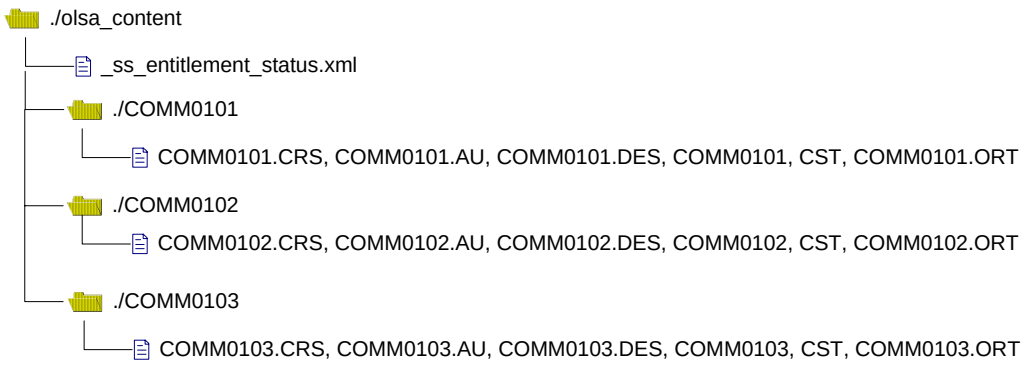


Figure 2: Sample AICC ZIP File

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. Table 1: Entitlement Status Changes lists the available entitlement status changes and recommended actions.

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 following is an example of the format of the entitlement status:

```
<ENTITLMENT_STATUS>
    <ASSET ID='someID1' STATUS='entitled'/>
    <ASSET ID='someID2' STATUS='not_entitled'/>
    <ASSET ID='someID3' STATUS='modified'/>
   .
   .
   .
</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. Table 2. Entitlement Exceptions lists 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

When using the OLSA Asset Integration Service for catalog synchronization, adhere to the following coding practice to handle specific exceptional circumstances.

Background

The following example illustrates at a high level how you should handle the OLSA Asset Integration Service Initiate-Poll-Acknowledge cycle by your LMS:

```
handle = AI_InitiateAssetMetaData (mode=all or delta)
while (true) {
   Sleep for N minutes
```

```
     AI_PollForAssetMetadata(handle)
     If (url-returned) {
        break;
     }
  }
  if (url_returned) {
    Retrieve zip file via URL
    // course-processing-loop
    while (there-are-courses-to-process-in-the-zip-file)
{
        // See the Recommendation below for what to put into this loop
   }
   AI_AcknowledgeAssetMetadata(handle)
  }
```

Exceptional Circumstance

There is a remote possibility that OLSA will return false positives for entitled, not_entitled, and modified status values for content under the following scenario:

During the AI_AcknowledgeAssetMetadata() call, OLSA crashes and cannot process the request. If this occurs then on the next iteration of the Initiate-Poll-Acknowledge cycle, the LMS will receive the same set of changes as in the previous cycle, plus additional changes that might have occurred in the interim.

Depending on the nature of the Acknowledge failure, the caller could see an OLSA GeneralFault or some sort of network exception. There really is no recourse for the LMS at this point. Nor should there be need for recourse since the LMS has successfully processed its side of this cycle.

Recommendation

SkillSoft recommends coding the course-processing-loop defensively to make this exceptional circumstance transparent to your LMS. The LMS should check if a course is installed or uninstalled before performing the respective install or uninstall action as illustrated in the following:

```
       // course-processing-loop-B
    while (there-are-courses-to-process-in-the-zip-file) {
                if (OLSA-course-status-is-entitled) {
                     if (course-is-uninstalled-in-LMS) install the course;
                }
                if (OLSA-course-status-is-not_entitled) {
                     if (course-is-installed-in-LMS) uninstall the course;
                }
                if (OLSA-course-status-is-modified) update the course;
   }
```

If an a priori check is not possible, then SkillSoft recommends wrapping the install and uninstall actions in an exception-handling block so that the LMS can ignore the relevant exception that may occur as shown below:

```
      // course-processing-loop-C
    while (there-are-courses-to-process-in-the-zip-file) {
```

```
            if (OLSA-course-status-is-entitled) {
                 try {
                   install the course;
                 } catch (AlreadyInstalledException) {
                    //ignore
                 }
            }
            if (OLSA-course-status-is-not_entitled) {
                 try {
                   uninstall the course;
                 } catch (NotInstalledException) {
                    //ignore
                 }
            }
            if (OLSA-course-status-is-modified) update the course;
   }
```

The update-action is an idempotent operation; the operation has no real impact, therefore you can update redundantly without any special checking or handling.

The only drawback of the recommendation is that if this circumstance should arise then the subsequent Initiate-Poll-Ack cycle will run a little longer than usual because the LMS will iterate through the data it has received before.However it will ensure the LMS and OLSA will remain in synchronization.

Inputs

- customerId - This refers to the sname provided by SkillSoft
- 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_INITIATEFULLBOOKSLISTINGREPORT

The AI_InitiateFullBooksListingReport function allows the customer to retrieve information about his referenceware entitlement. In summary mode, its returns the list of accessible Books24x7 assets. In detail mode, it also includes referenceware hierarchy information, which is where in the referenceware hierarchy a book asset resides in the OLSA environment. The userName argument allows the customer to specify the relevant entitlement or assignment (see Entitlement/Assignment). If the customer does not take advantage of pre-registering users (see ***User Management Service (UM_)*** (on page 61)) or making assignments (see ***Assignment Service (AS_)*** (on page 53)) into OLSA then he may omit the username argument, in which case all Books24x7 assets in his entitlement should be returned.

The report is in XML format.

Inputs

- customerId - This refers to the sname provided by SkillSoft
- reportFormat - XML)
- mode - summary or detail
- username - optional

Outputs

- A report ID handle. Use this value with the ***UTIL_PollForReport*** (on page 84) function to get the actual contents of the report.

Additional Faults

- None

AI_INITIATEFULLCOURSELISTINGREPORT

The AI_InitiateFullCourseListingReport function retrieves courseware 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 the function returns all assets in his entitlement.

The report output is available in HTML, CSV, or XML formats.

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

Course Title	Duration (Hrs:Min)	Course Number	Curriculum
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:

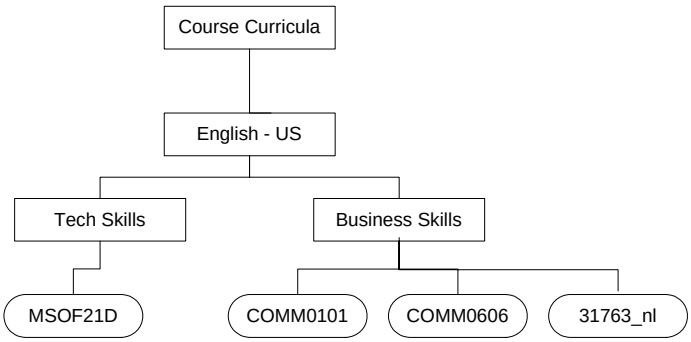


Figure 3: A defined catalog hierarchy

Inputs

- customerId - This refers to the sname provided by SkillSoft
- reportFormat - Accepts the following formats, HTML, CSV, or XML
- mode - Provides the report in summary or detail format
- username - This is an optional input

Outputs

- A report ID handle. Use this value with the ***UTIL_PollForReport*** (on page 84) function to get the actual contents of the report.

Additional Faults

- None

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 - This refers to the sname provided by SkillSoft

Outputs

- report ID handle - Use this value with the ***UTIL_PollForReport*** (on page 84) 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

AI_ISBOOKSHIERARCHYMODIFIED

This function indicates the modification status of the Books24x7 Hierarchy since the most recent AI_InitiateFullBooksListingReport invocation. For this function, the Books24x7 Hierarchy is the customer's complete referenceware entitlement returned by AI_InitiateFullBooksListingReport when the username argument is omitted. If no such events have occurred since the most recent AI_InitiateFullBooksListingReport invocation, then this function indicates that no modifications have occurred. If you call this function before running the AI_InitiateFullBooksListingReport it indicates that modifications have occurred.

Inputs

- customerId - This refers to the sname provided by SkillSoft

Outputs

- True - indicates the hierarchy has changed since the most recent AI_InitiateFullBooksListingReport invocation.
- False - Indicates the hierarchy has not changed

Additional Faults

- None

AI_ISCATALOGHIERARCHYMODIFIED

This function indicates the modification status of the Catalog Hierarchy since the most recent AI_InitiateFullCourseListingReport invocation. For this function, the Catalog Hierarchy is the customer's complete courseware entitlement returned by AI_InitiateFullCourseListingReport when the username argument is omitted. If no changes have occurred since the most recent AI_InitiateFullCourseListingReport invocation then this function indicates that no modifications have occurred. If you call this function before running AI_InitiateFullCourseListingReport it indicates that modifications have occurred.

Inputs

- customerId - This refers to the sname provided by SkillSoft

Outputs

- True - indicates the catalog hierarchy has changed since the most recent AI_InitiateFullCourseListingReport invocation
- False - Indicates the catalog hierarchy has not changed

Additional Faults

- None

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*** (on page 29).

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 - This refers to the sname provided by SkillSoft
- 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_REMOVEASSETFROMGROUP

This function removes an asset from any asset group. This function does not delete an asset. See also ***AI_InitiateMakeChangesVisible*** (on page 37).

Inputs

- customerId - This refers to the sname provided by SkillSoft
- catalogPath - for relevant Asset Group
- assetId

Outputs

- None

Additional Faults

- ObjectNotFoundFault - group or asset does not exist

CHAPTER 4

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, Jobaids, and Referenceware and 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. You can use the native attribute value for a topic 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, for example, 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

Asset Type	assetType	DTYPE
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	_ss_projects	Not yet assigned
Practice Labs	_ss_ practicelabs	Not yet assigned

Table 5. Asset TypesSearch Terminology

assetType - The kind of asset, for example, _ss_bs _ss_classic, _ss_book, and so forth. 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*** (see "Asset Integration Service (AI_)" on page 17). 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 following XML example shows the result set/bin/asset structure appears at a high level :

```
<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>
```

In This Chapter

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 - This refers to the sname provided by SkillSoft
- 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 - Optional. Assume user is non-508

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_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.

The SkillSoft Search & Learn applies the following search result rank ranges to the specified icon gifs:

Rank Value Range	Icon.gif
<=25	h_lowest.gif ()
> 25 and <=50	h_low.gif ()
> 50 and <= 75	h_higher.gif ()
> 75	h_high.gif ()

This function supports the Simple-user-attributes option of the automatic-user-registration-or-update capability, see User Resgistration.

Inputs

- customerId - This refers to the sname provided by SkillSoft
- searchPhrase
- languageCode
- userName
- groupCode - Optional. Ignored if user already is registered
- password - Optional. Ignored if user already is registered
- enable508 - Optional. Assume user is non-508

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_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 - This refers to the sname provided by SkillSoft
- 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_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 - This refers to the sname provided by SkillSoft
- assetId
- username
- searchId - optional

Outputs

- Metadata for the asset, 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 - This refers to the sname provided by SkillSoft

Outputs

- See the OLSA WSDL for a complete description.

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 - This refers to the sname provided by SkillSoft
- searchId
- binName
- start
- count - a value between 5-10 is recommended
- enable508 - Optional. Assume user is non-508

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_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 - This refers to the sname provided by SkillSoft
- searchAssetId
- userName
- groupCode - Optional. Ignored if user already is registered
- password - Optional. Ignored if user already is registered
- enable508 - Optional. Assume user is non-508

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_SETSEARCHPARAMETER

Deprecated for OLSA 1.2.

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

Inputs

- customerId - This refers to the sname provided by SkillSoft
- searchParameter - see the OLSA WSDL for a complete description

Outputs

- None

Additional Faults

- None

CHAPTER 5

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 Entitlement/Assignment). For example, if you issue a search 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 pre-defined grouping of multiple book assets, for example, IT Pro. Collections are a flat list of books; they are not hierarchical. You cannot assign individual books.
- 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 entitlement.

In This Chapter

AS_GETCATALOGASSIGNMENT

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

Inputs

- customerId - This refers to the sname provided by SkillSoft
- 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 - This refers to the sname provided by SkillSoft
- userName

Outputs

- See AS_GetCatalogAssignment

Additional Faults

- None

AS_GETCOLLECTIONASSIGNMENT

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

Inputs

- customerId - This refers to the sname provided by SkillSoft
- 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 - This refers to the sname provided by SkillSoft
- userName

Outputs

- See AS_GetCollectionAssignment

Additional Faults

- None

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 - This refers to the sname provided by SkillSoft

Outputs

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

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 - This refers to the sname provided by SkillSoft
- 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 - This refers to the sname provided by SkillSoft
- list of catalogPaths - 0, 1 or more
- userName

Outputs

- None

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. If the specified collection is not in the subscription associated with the user group, the function returns a GeneralFault.

Inputs

- customerId - This refers to the sname provided by SkillSoft
- list of collectionIds - 1 or more
- groupCode - of user group to assign

Outputs

- None

Additional Faults

- None

AS_SETCOLLECTIONASSIGNMENTBYUSER

This function assigns collections to an individual user. An individual user assignment overrides 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 - This refers to the sname provided by SkillSoft
- list of collectionIds - 1 or more
- userName

Outputs

- None

Additional Faults

- None

CHAPTER 6

User Management Service (UM_)

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

- 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.

In This Chapter

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 - This refers to the sname provided by SkillSoft
- userName
- List of groupCodes

Outputs

- None

Additional Faults

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

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 - This refers to the sname provided by SkillSoft
- See General User Attributes for the remaining agreements

Outputs

- None

Additional Faults

- ObjectExistsFault - user already exists

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 - This refers to the sname provided by SkillSoft
- groupCode
- groupTitle
- parentGroupCode

Outputs

- None

Additional Faults

- ObjectExistsFaults - user group already exists

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 - This refers to the sname provided by SkillSoft
- userName

Outputs

- None

Additional Faults

- ObjectNotFoundFault - user 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 - This refers to the sname provided by SkillSoft
- groupCode

Outputs

- None

Additional Faults

- ObjectNotFound - user group does not exist

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 - This refers to the sname provided by SkillSoft
- newUserName
- See General User Attributes for the remaining arguments

Outputs

- None

Additional Faults

- ObjectNotFoundFault - user does not exist

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 - This refers to the sname provided by SkillSoft
- 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_INITIATEUSERLISTINGBYGROUPREPORT

This function retrieves information about the user population registered in the OLSA environment. You can specify the user population to list 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 function returns reports in HTML or CSV formats. HLINKR_6 lists a sample user listing by group CSV report fragment with allUsers, includeSubGroups and listBySubgroups all set to true. The table does not show all CSV fields.

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

GroupTitle	GroupPath	groupCode	LoginName	LastName	FirstName
Books24x7	/Books24x7	Books24x7	Smith2	Smith2	Sally

Table 6. Sample User Listing by Group Report

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

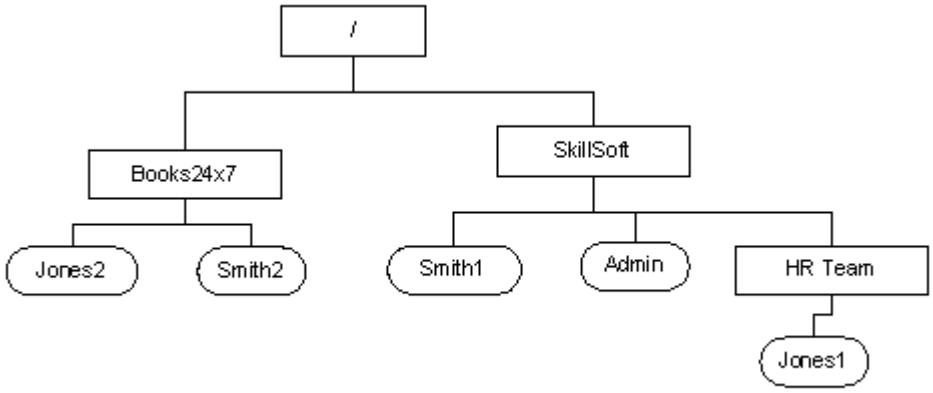


Figure 5. Sample User Hierarchy

Inputs

- customerId - This refers to the sname provided by SkillSoft
- 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. Use this value with the ***UTIL_PollForReport*** (on page 84) function to get the actual contents of the report.

Additional Faults

- None

UM_REMOVEUSERFROMGROUP

This function removes a user from an existing user group or groups 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 - This refers to the sname provided by SkillSoft
- userName
- List of groupCodes

Outputs

- None

Additional Faults

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

CHAPTER 7

Offline Integration Service (OF_)

This service allows a customer-application, for example, 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.

In This Chapter

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 - This refers to the sname provided by SkillSoft
- assetId
- username
- x508Optional - Assumes the user is non-508

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 - This refers to the sname provided by SkillSoft
- userName

Outputs

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

Additional Faults

- None

CHAPTER 8

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.
- Assets launched from UI screens generated from AddOns, and Advanced Learning Structures, such as KnowledgeCenter and Learning Program.

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
- Accessing a Learning Program report
- Accessing a KnowledgeCenter report

In This Chapter

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 - This refers to the sname provided by SkillSoft
- 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 (for example, 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 - This refers to the sname provided by SkillSoft
- 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. Use this value with the ***UTIL_PollForReport*** (on page 84) 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. You can filter the Usage data further with the optional start date, end date and date modifier arguments.

Inputs

- CustomerId - This refers to the sname provided by SkillSoft
- 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. Use this value with the ***UTIL_PollForReport*** (on page 84) function to get the actual contents of the report.

Additional Faults

- None

UD_INITIATEKNOWLEDGECENTERACTIVITYREPORT

This function will initiate the KnowledgeCenter Activity report. If group is omitted, then all users will be reported on.

Inputs

- customerId - This refers to the sname provided by SkillSoft
- group - This optional string is a group path that must start with "/"
- includeSubgroups - This is optional, the default value is true
- includeDeactivatedUsers - This is optional
- mode - Select summary or detail
- reportFormat - Select HTML, CSV or CSV16

Outputs

- A report ID handle. Use this value with the UTIL_PollForReport function to get the actual contents of the report.

Additional Faults

- None

UD_INITIATELEARNINGPROGRAMACTIVITYREPORT

This function initiates the Learning Program Activity report. If you omit username, firstname and lastname, then the function processes all users. If you specify the username, firstname or lastname, then the function process all users that have a corresponding prefix-match. For example, specifying the username of "smith" matched usernames "smith1" and "smith2" and processes those usernames. If you omit the listOfAssetIDs input, the function reports all assets including Learning Programs. If you enter a single assetID that does not exist, the function reports a GeneralFault stating the specific assets that do not exist. If you enter a list of assets, the function does not submit a report and sends a GeneralFault stating which assets do not exist.

Inputs

- customerId - This refers to the sname provided by SkillSoft
- Username - This is optional
- Firstname - This is optional
- Lastname - This is optional
- Group - If this optional string starts with a "/", the function assumes it is a user group path, otherwise, without the "/", interprets it as a group code
- includeSubgroups - This is optional, the default value is true
- breakOnGroups - This is optional, the default value is true
- singleLinePerUser - This is optional and only affects CSV reports
- listOfAssetIDs - This is optional and can be a comma separated list of assets [programs]
- includeDeactiatedUsers- This is optional
- includeLPsWithNoProgress - This is optional
- includeStartedLPs - This is optional, the default value is true
- includeCompletedLPs - This is optional, the default value is true
- mode - Select summary or detail
- listBy - Select one of the following:
- user - displays details by user
- asset - displays details by asset
- reportFormat - Select HTML, CSV or CSV16

Outputs

- The function returns the report ID handle. Use this value with the ***UTIL_PollForReport*** (on page 84) function to get the actual contents of the report.

Additional Faults

- None

CHAPTER 9

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

In This Chapter

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).

For the following types of assets, actionType=”summary” will land the user in the same location as actionType=”launch”:

- AddOns
- Knowledge Centers
- Learning Programs
- Books24x7

Inputs

- customerId - This refers to the sname provided by SkillSoft
- actionType - home, myplan, catalog, summary, launch
- assetId
  - Course asset ID - must be non empty only for *action*=summary and *action*=launch
  - Books24x7asset ID - *book*, *chapter*, and *language*

- newUserName
- See General User Attributes for remaining arguments
- enable508 - Optional. Assume user is non-508

Outputs

One of the following URL:

- 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

Books24x7Level Launch

You can launch Books24x7 through the OLSA Sign-on service. The book level launch supports the launching of Books24x7 at the book level using the following parameters:

Book ID - ####

Chunk ID - ####-####

For more information, refer to the following functions that also indicate the required 508 mode for a user. These additional functions support an enable508 argument, which has the same semantics as the x508 aregument in the RO launch URL:

- ***SL_DetailedSearch***
- SL_RelatedSearch
- SL_PaginatedSearch
- OF_GetDownloadAssetUrl
- SO_GetMultiActionOnSignOnUrl

There is an OLSA backend setting to enable 508 support company-wide. The following table illustrates the relationship between this company-wide setting and the x508 RO launch URL argument and the additional functions's enable508 argument.

x508 Argument on RO Launch URL and additional functions' enable508 argument	Company-wide 508 Setting	Notes
Explicitly set to 1	N/A	Company-wide setting ignored
Explicitly set to 0	N/A	Company-wide setting ignored
Omitted	0	Assumes the user is non-508
Omitted	1	Assumes the user is 508

For more information about a section, chapter, or part launch of a natively installed Books24x7 asset, see ***Section, Chapter, or Part launch of a newly Installed Books24x7 Asset***

CHAPTER 10

Utility Service (UTIL_)

This section defines the Utility Service and its associated functions.

In This Chapter

UTIL_DELETEREPORT

This function deletes a completed report given a report ID, for example, returned by ***UD_InitiateCustomReportByUsers*** (on page 73).

Inputs

- customerId - This refers to the sname provided by SkillSoft
- 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

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 - This refers to the sname provided by SkillSoft
- 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.

UTIL_POLLFORREPORT

This function retrieves the URL for the completed report given a report ID. For example, the URL returned by ***UD_InitiateCustomReportByUsers*** (on page 73).

Inputs

- customerId - This refers to the sname provided by SkillSoft
- 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

CHAPTER 11

Configuration Service (CF_)

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

In This Chapter

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, select an integer between 1 - 99

These settings control what values are generated in the specified fields for the putParam command for special content, for example, 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 - This refers to the sname provided by SkillSoft

Outputs

- The response returns 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_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 - This refers to the sname provided by SkillSoft

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_GETSCORMSETTINGS

This function allows the customer to retrieve current SCORM settings. The following SCORM settings include:

- Lesson_status - specifies the lesson status value to be set to the SCO for the TPLMS during launch. Valid values are not attempted, incomplete - default, and completed.
- Session_time - specifies the session time value set to the SCO for the TPLMS during launch. The default value is 00:00: 00.

Scorm_version - specifies the version of PIF file to be generated in initiate asset meta data AI functions. The supported values are scorm12 - default, and scorm24. The OLSA 1.2 release only accepts the value scorm12.

Inputs

- customerId

Outputs

- scormSettings - lesson_status, session_time, scorm_version

For more information about the GetScormSettingsRequest and GetScormSettingsResponse elements, see the OLSA WSDL.

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 - This refers to the sname provided by SkillSoft

Outputs

- The response returns 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_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 values 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 - This refers to the sname provided by SkillSoft
- aiccSettings - see CF_GetAiccSettings and OLSA WSDL for more details.

Outputs

- None

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, for example, 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 - This refers to the sname provided by SkillSoft
- playerProperties - see CF_GetPlayerProperties and OLSA WSDL for more details

Outputs

- None

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 - This refers to the sname provided by SkillSoft
- skillsimProperties - see CF_GetSkillSimProperties and OLSA WSDL for more details

Outputs

- None

Additional Faults

- None

CF_SETSCORMSETTINGS

This function allows customers to modify existing SCORM settings. To change a setting value, the caller must first invoke CF_GetScormSettings. The XML response returned can then be updated with appropriate values. This updated XML document must be as per SetScormSettingsRequest specified in the WSDL document.

Inputs

- customerId
- scormSettings- lesson_status, session_time, scorm_version

Outputs

- None - VoidResponse

For more information about the SetScormSettingsRequest element, see the OLSA WSDL.

CHAPTER 12

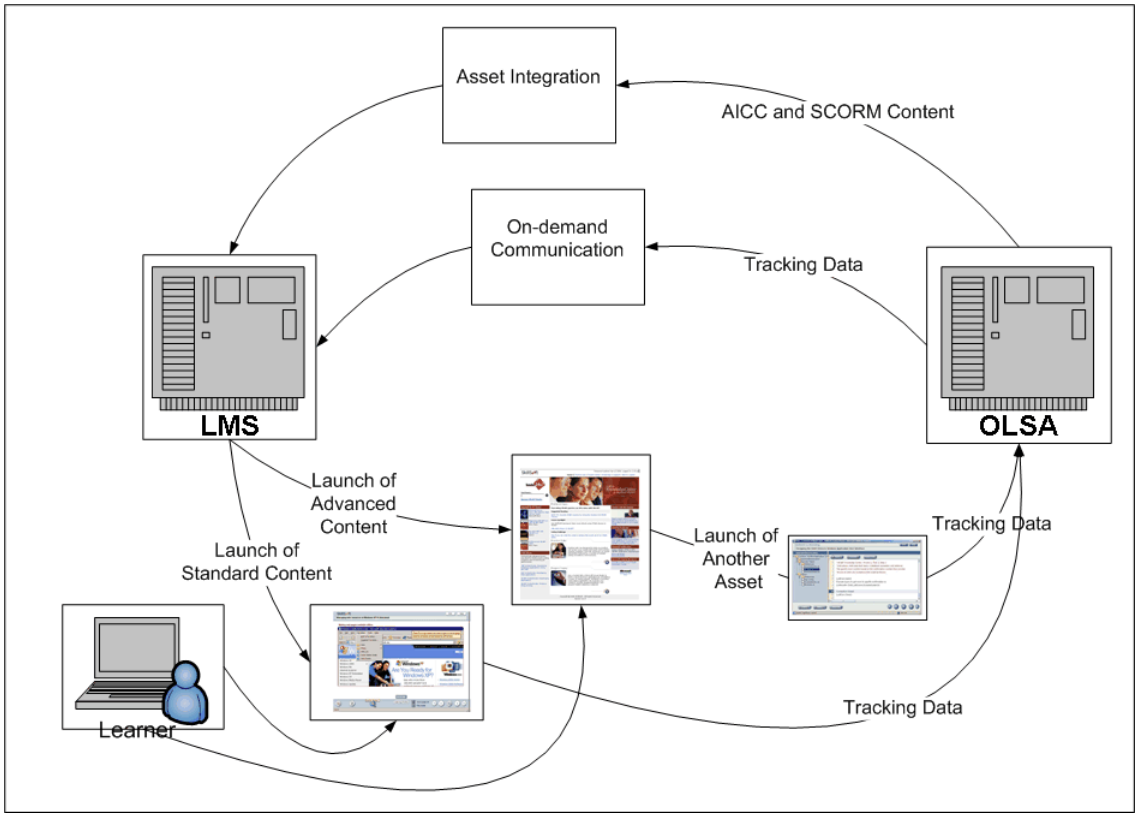
On-demand Communication (OC_)

On-demand Communication (OC) provides a near real time feed of tracking data to OLSA customers using the one-way-to-OLSA or one-way-to-OLSA-download modes.

The On-demand Communication (OC) web service is a mechanism for enhancing tracking data communication integration between a Content Publisher (OLSA) and a Content Consumer (LMS). On-demand Communication provides the following:

- Support for getting tracking data for assets launched from within Advanced content such as KnowledgeCenters and Learning Programs
- Support for getting tracking data for offline play
- Support for reconciled and merged tracking data generated from regular on-line access to standard content
- Support for getting tracking data for assets launched from Search & Learn results

The following diagram shows how On-demand Communication facilitates sending tracking data between the Learning Assets and the LMS. If a learner is accessing any Advanced content, the LMS does not have the ability to track usage data for any assets launched from inside the Advanced content. If you setup On-demand Communication the data for all asset accesses track back to OLSA, regardless of how it is launched.



When a learner completes a single session in a learning object, a raw tracking data is generated and transformed in to a neutral format, Tracking Data Record (TDR) and stored in to On-demand Communication system. On-demand Communication makes these TDRs available in chronological order to a given LMS. For more information about On-demand Communication, see the OLSA WSDL.

In This Chapter

OVERVIEW

The following includes the main actors in On-demand Communication: Figure 1 illustrates the relationship between these actors.

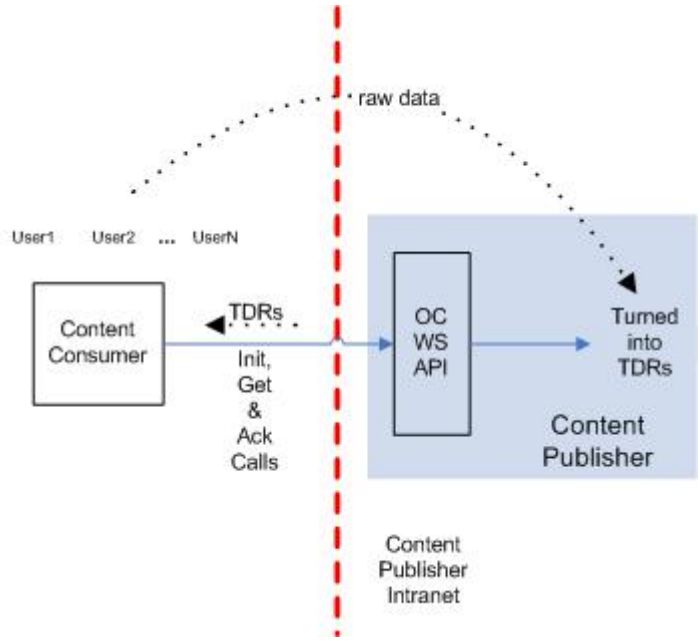


Figure 1: On-demand Communication Context Diagram

- The content consumer system, which serves its end-user population and will be a consumer of the near real-time feed of Tracking Data Records (TDR). It will make calls to the OC Web Service API to retrieve TDRs.
  A Tracking Data Record (TDR) is created from the raw tracking data generated by each end-user when he completes a single session with a learning object. As the content publisher (OLSA) receives raw tracking data from individual end-users, it puts TDRs into the OC system.

- The end users of a given content consumer system generates the raw tracking data as each end user completes a session with a learning object. Raw tracking data goes directly to content publisher.

- The content publisher system, which can serve many content-consumers, provides access to the OC Web Service API.

- The OC system itself, which collects and maintains TDRs generated by the end-users, and makes them available in near chronological order to a given content consumer. The content consumer uses the OC Web Service API (WS API) to initialize, get and acknowledge TDRs. The key functions include:

  - OC_InitializeTrackingData – the content consumer must invoke this function once each time it starts up before using either the Get or Acknowledge functions. This function provides the appropriate hook to allow the content consumer to reconcile TDRs if either the content consumer failed to send an Acknowledgement or the content publisher failed to receive an Acknowledgement

  - OC_GetTrackingData – this is used by the content consumer to get all unretrieved TDRs for all users or a specific user

  - OC_AcknowledgeTrackingData – this is used by the content consumer to indicate to the content publisher that the previously retrieved set of TDRs has been successfully processed

Simple protocol rules are specified to ensure that a given content consumer can uniquely filter duplicate data and reliably reconcile after system restarts, retrieve TDRs. For more information, see ***Pattern 1*** (see "Pattern 1 - Bulk Reporting" on page 103) and ***Pattern 2*** (see "Pattern 2 - End User Status" on page 104) approaches.

SYNCHRONIZE COURSE COMPLETION

For a direct launch of an asset, through a Referral Object, OLSA ensures that the TDR for this asset is available at the time of the actual end-user content completion event using the standard protocol primitives. For SCORM, the OLSA content synchronizes with the LMSFinish call. For AICC, OLSA content or the OLSA server itself synchronizes with the exitAU message. This means the TDR for the associated direct launch, through a Referral Object, will be ready for retrieval when the associated LMSFinish or exitAU is received by the LMS. This feature can be used to advantage when implementing the ***Pattern2*** (see "Pattern 2 - End User Status" on page 104) approach.

USES CASES

On-demand Communication supports the following use cases. These are the key use cases to support from the content consumer system perspective.

Use-case-01: Bulk Reporting

Description	In this use case, the content consumer system has a privileged level user that needs to run a report that contains results for a large number of end users. The number of end users and the volume of associated data can be extremely large. This use case typically occurs at a frequency of once a day, week, or month. A report does not have to be produced instantly. There is an expectation that the report will take about tens of minutes to generate.
Actors	Content consumer and content publisher
Assumptions	The content consumer has the necessary credentials to call the content publisher. Entitlements for the content consumer have been setup. 0 or more course launch sessions have been completed. The content consumer has done at least OC Pattern 1 integration with the content publisher.
Steps	▪ Privileged user logs into their content consumer system. ▪ User runs a report using features available from the native content consumer system. ▪ User accesses the completed report after N minutes. ▪ User logs out.
Variations	User can run more then one kind of report within his session.
Non-functional	
Issues	Depending on the time interval used by the content consumer to call the content publisher, the data in the report may be out of date by that same time interval.

Use-case-02: End-User-MyStatus-MyLearningPath

Description	In this use case the content consumer system has an end user that needs to perform a content consumer system action that needs a near-immediate response, for example, display the student's current status, or provide access to the next learning object in a content-consumer-implemented-composite object. The response will be based on the tracking data of any learning object the end-user has already accessed. In this use case the content consumer will need to retrieve any un-retrieved tracking data for a specific user as fast as possible from the content publisher. The number of end users can be extremely large. A high number of this use case may occur in bursts. The pattern will vary from content consumer systems. The data involved for a single user is relatively small, much smaller then a bulk report. The average response time in this use case must support the content consumer system's user interface's responsiveness requirements.
Actors	Content consumer and content publisher.
Assumptions	The content consumer has the necessary credentials to call the content publisher. Entitlements for the content consumer have been setup. 0 or more course launch sessions have been completed. The content consumer has done at least On-demand Pattern 2 integration with the content publisher.
Steps	▪ End user logs into their content consumer system. ▪ User performs an action such as displaying his current status, or getting access to the next learning object in some content-consumer-implemented-composite object. ▪ Action processes in N seconds. ▪ User continues with his session… ▪ User eventually logs out.
Variations	User can run one or more of this kind of action within his content consumer system login session.
Non-functional	
Issues	

Use-case-03: Reset Status of a Learning Object

Description	In this use case the content consumer system must be able to reset the state of a learning object, to the initialized state, for a particular user. This user may have already generated tracking data for this learning object. This user may have even already completed this learning object. One example of this is in a compliance setting where a particular learning object must be taken once a year, every year for some set of users for a given customer. Each time such a learning object is re-taken by the end-user, the learning object must be started with a blank slate. Any previous history collected for the end user must be preserved by the content publisher.
Actors	Content consumer and content publisher
Assumptions	The content consumer has the necessary credentials to call the content publisher. Entitlements for the content consumer have been setup. 0 or more course launch sessions have been completed. The content consumer has done either On-demand Pattern 1 or Pattern 2 integration with the content publisher.
Steps	▪ End user logs into their content consumer system. ▪ User navigates to a screen which requires him to re-take a learning object. ▪ User launches the learning object ▪ Learning object presents an initialized state, for example, unused, to the user. ▪ User takes the course. ▪ User exits course. ▪ User logs out.
Variations	User can run one or more of this kind of action during his content consumer system login session.
Non-functional	
Issues	

REAL TIME DELIVERY FACTORS

On-demand Communication Data delivery will be near real time. The factors that may impact the timeliness of data delivery to the content consumer include:

- The time interval between Get requests used by the content consumer
- The time it takes for the content consumer to process recently retrieved TDRs before issuing the next Get request
- The time it takes for the content publisher to convert raw tracking data into TDRs
- The retrieval algorithm used by OC to process TDRs for a particular Get request. To avoid timeouts, holding internal system threads too long, and so forth, the retrieval algorithm may only read a fixed maximum number of TDRs at a time
- The general load on the content publisher system

CHRONOLOGICAL ORDER OF DELIVERY FACTORS

The delivery of all TDRs will be in chronological order when the caller implements their integration using ***Pattern 1*** (see "Pattern 1 - Bulk Reporting" on page 103).

The delivery of some TDRs may not be in chronological order when the caller implements their integration using ***Pattern 2*** (see "Pattern 2 - End User Status" on page 104). Pattern 2 offers more near real time benefits but with the trade off potentially encountering some TDRs out of order. To mitigate this situation each TDR is marked with a unique ID and timestamp indicating when the content publisher received its associated data to allow the caller to reconcile out of order TDRs.

TRACKING DATA RECORD

A Tracking Data Record (TDR) represents the data generated from a single session of a given user interacting with a learning object. For example, the raw tracking data associated with the last AICC putParam for a given user or learning object session is what generates a TDR.

Tag Name	Type	Semantics
id	int	This ID is unique for all TDRs per content consumer system. This ID will be a monotonically increasing 32-bit integer that starts at 0.
timestamp	timestamp	The UTC timestamp of when the TDR is entered into the OC system.
userid	string	Equivalent to the AICC core student_id provided by the content consumer.
assetid	string	The content publisher's ID for the associated learning object. If data is received for an unknown asset then the content publisher may provide a capability to retrieve and install the metadata for the learning object on the fly.
reset	boolean	If the was previously sent to the content consumer but not acknowledged by the content consumer, then 1, otherwise 0.
format	string	The name of the format used in the data element. The currently supported format is "cmixml/1.0/ToLMS".
data	string	The actual data in "cmixml/1.0/ToLMS" format.
context	string	Reserved for future use.

Table 1: Tracking Data Record elements

A TDR can be generated in the following ways:

- Content consumer initiated launches – From the perspective of the consumer, also known as a direct launch. Learning objects launched in this manner are in the traditional model where the metadata, for example, AICC, SCORM, for the learning object is installed into the content-consumer system. At which point, such learning objects are then available to the end users of the content consumer system. An end user can now log into the content consumer system, navigate through its User Interface screens, and directly launch the learning object in question. Learning objects launched in this manner are initiated by the content consumer system.

- Content publisher initiated launches – From the perspective of the consumer, also known as an indirect launch. Learning objects launched in this manner use a new model. In this model, the content-consumer is unaware that its end users are accessing the learning objects in question. A learning object launched from within a learning object would be one example of this model. Learning objects launched via a content publisher's proprietary offline solution would be another example. Learning objects launched in this manner are initiated by the content publisher system.

In some cases, the learning object in question may be for content that is not natively yet installed or known in the content-consumer system. A content-consumer must be prepared for this contingency. A content consumer can choose to ignore such TDRs. In the case of the SkillSoft OLSA implementation, the content consumer has access to the Selective Asset Metadata Retrieval (SAMR) functions. These functions allow the content consumer to get AICC or SCORM metadata so the missing learning object can be installed as needed, if such a step is necessary to properly process such TDRs.

OC_INITIALIZETRACKINGDATA

This function informs the content publisher if it needs any Tracking Data Records (TDRs) that were sent to the caller but not acknowledged by the content publisher. You can retrieve these TDRs by using the OC_GetTrackingData calls.

Any resent TDRs will have it reset field set to 1. This is an indicator to the caller that such TDRs should be examined to ensure that the caller has not already processed or persisted them.

Use this function at system startup before issuing any OC_GetTrackingData or OC_AcknowledgeTrackingData calls. This allows the caller to sync back up with the content publisher if either party has terminated unexpectedly in the middle of a cycle of issuing OC_GetTrackingData and OC_AcknowledgeTrackingData calls.

The following pseudo code illustrates how to use this function.

```
// Somewhere in the content consumer system startup sequence…
try {
  OC_InitializeTrackingData()
  //
  // It is recommended to execute the first iteration
  // of the Pattern 1 code right here as well to synchronize
  // any missed TDRs.
  //
  // Ideally the remainder of the startup sequence should
```

```
    // wait until this first iteration in the Pattern 1 code is complete.
  } catch (GeneralFault) {
    // One possible error case is that the content publisher
    // system is not available.
    //
    // In this case the caller has the option to implement
    // a retry loop calling OC_InitializeTrackingData with a delay.
    // Or to skip this step.
    //
    // Skipping this step will result in any unacknowledged TDRs
    // not to be sent until the next time OC_InitializeTrackingData
    // is called.
  }
  // Continue with remainder of the startup sequence…
```

Inputs

- CustomerID

Outputs

- None

Additional Faults

- None

OC_GETTRACKINGDATA

Use this function to retrieve any Tracking Data Records that are available for the caller. A Tracking Data Record (TDR) represents the tracking data generated in a single session for a given user interacting with a learning object.

If one or more TDRs are returned, then the caller should process them first, and then invoke OC_AcknowledgeTrackingData to let the content publisher know that the records have been successfully processed. The call to OC_AcknowledgeTrackingData should specify the handle returned in the matching OC_GetTrackingData call.

Use OC_AcknowledgeTrackingData only if a non-empty result is returned by OC_GetTrackingData.

This section includes the following sections:

Pattern 1 - Bulk Reporting

Implement a single background thread that wakes up periodically to retrieve any available TDRs, this thread should omit the username and TdrType arguments when calling this function. This is the simplest level of integration. This approach will satisfy situations that can operate with data that is out of date by the interval used by the single background thread. This approach is designed to address the Use-case-01: Bulk Reporting.

The recommended interval for this approach is 10 minutes. The reason for an interval is to maintain optimal responsiveness and throughput of the content publisher system.

Then using only Pattern 1, for recovery purposes, the single background thread need only save the last TDR ID that was successfully processed. The content publisher guarantees that all TDR IDs will be unique per customer and monotonically increasing. The single background thread only needs to check:

```
last_processed_tdr_id < current_tdr_id
```

If true, then the TDR has not been seen before by the caller, and should be processed.

Pattern 2 - End User Status

Implement Pattern1, then periodically initiate calls to this function in select locations in its application where a delay in getting up to date data is not acceptable.

Note: In this pattern, this function must be called with the format that specifies a username argument. This pattern is designed to address the Use-case-02: End-User-MyStatus-MyLearningPath. Do not try to address the Use-case-01: Bulk Reporting with Pattern 2.

When using Pattern 2, *the recovery process must be more robust*. For recovery purposes, the caller should maintain the list of all TDR IDs that have been successfully processed. This is because multiple threads can make Get and Acknowledge calls, and any could fail for different reasons. In this case saving the last TDR ID for a less-than compare is not sufficient since thread A could successfully process TDR ID n+1 but thread B might have failed processing TDR ID n. Implement Pattern1. In addition, on an as-needed basis, the caller should sprinkle calls to this function in select locations in its application where a delay in getting up-to-date data is not acceptable.

Pseudocode Examples

The following pseudocode examples illustrate how you can implement the described patterns.

Pattern 1 - Bulk Reporting

```
// The following procedure illustrates Pattern 1 (see "Pattern 1 - Bulk
Reporting" on page 103) and should be
// executed only by a single background thread. During a given
// interval, it is designed to iterate calling the content publisher
// until NoResultsAvailableFault is detected or moreFlag is false.
//
proc Example_Pattern_1() {
  while (true) {
    boolean moreFlag = true
    DataResponse response = null
    while (moreFlag) {
      try {
        response = OC_GetTrackingData(“”, “”)
        Example_ProcessData(response.data)
        OC_AcknowledgeTrackingData(response.handle)
        moreFlag = response.moreFlag
      } catch (NoResultsAvailableFault) {
      // All up-to-date
        break
      } catch (GeneralFault) {
      // Error calling Get or Ack. No action required by caller.
      break
```

```
        } catch (AnyNonOCError) {
          // Process the error…
          // If the error occurred in Example_ProcessData
       // then it is recommended that a call to
       // OC_InitializeTrackingData()
          // be made. This will ensure that any unacknowledged
       // and potentially unprocessed TDRs will be resent
       // on the next Get call.
       break
        }
      }
      sleep(RecommendedInterval)
    }
  }
```

Pattern 2 - End User Status

```
  // The following procedure should be called as-needed for
  // a specific user. It can be called simultaneously from many
  // different threads. Do not use this pattern to satisfy
  // the Use-case-01: Bulk Reporting.
  //
  proc Example_Pattern_2(username){
    boolean moreFlag = true
    DataResponse response
    while (moreFlag) {
      try {
        response = OC_GetTrackingData(“”, username)
        Example_ProcessData(response.data)
        OC_AcknowledgeTrackingData(response.handle)
        moreFlag = response.moreFlag
      } catch (NoResultsAvailableFault) {
     // All up-to-date
     break
      } catch (GeneralFault) {
     // Error calling Get or Ack. No action required by caller.
     break
      } catch (AnyNonOCError) {
        // Process the error…
        // Due to the multi-threaded nature of this
  // scenario, it is NOT recommended that a call to
```

```
    // OC_InitializeTrackingData()
       // be made here. This call should only be
// done by the Pattern 1 thread or the system
// startup thread.
break
    }
  }
}

// This code demonstrates how to implement a common
// processing module with a recommended recovery
// approach for Pattern 2 (see "Pattern 2 - End User Status" on page 104).
//
// How the setOfAlreadyProcessedIds functionality
// is implemented is internal and specific to each caller.
//
proc Example_ProcessData(TrackingDataRecord[] data) {
  foreach record in data {
    if (record.reset == 0) {
      // Safe to process, the caller has not seen this before.
      //
      // Also check timestamp to perform any reconciliation
      // if this record is older then any record already
      // processed for the same user. Out-of-order records
      // should not be an issue for different users.
      write record to DB, this is
      atomically adding record.id to setOfAlreadyProcessedIds
    } else {
      // May have seen this TDR before, check it. The
    // following TDR attributes are available for
    // reconciliation:
    // - id
    // - timestamp
      if (record.id in setOfAlreadyProcessedIds) {
        // skip this record we have already processed it before
      } else {
        // safe to process, the caller
        // may have seen it but was not
  // able to process and acknowledge it.
```

```
            write record to DB, this is
            atomically adding record.id to setOfAlreadyProcessedIds
          }
        }
      }
}
```

Function Details

This function does not return the same TDR more than once, even when called simultaneously from multiple threads. A given TDR will only be returned from one of the potential multiple calls to this function.

The only exception is if OC_InitializeTrackingData is called. If there are any TDRs that have been delivered to the caller but not acknowledged then they will be re-sent. Any re-sent TDRs will have its reset-field set to 1.

If the specified username does not exist in the content publisher system, then this function will respond in the same manner as if there were no TDRs available for an existing user.

Inputs

- CustomerID
- TdrType - all (default), direct, indirect
- Username - optional

Outputs

- Handle
- MoreFlag - true means there may be more TDRs to retrieve that match the specified criteria so the caller may choose to call this function again after processing the current array of TrackingDataRecords, otherwise false
- Array of TrackingDataRecords - matching the specified criteria
  - If array is non-empty, then Handle must be non-null. MoreFlag may be true or false.
  - If array is empty, then NoResultsAvailableFault will be thrown.

Additional Faults

- NoResultsAvailableFault - This fault is thrown if no data is available to be returned for the specified criteria.

OC_ACKNOWLEDGETRACKING DATA

Use this function to acknowledge the successful processing of Tracking Data Records from a prior OC_GetTrackingData call. Only use this call after the associated TDRs have been truly processed and persisted on the caller's side and if OC_GetTrackingData returned a non-empty result.

After using this call, any TDRs returned by the associated OC_GetTrackingData call cannot be re-retrieved from the content publisher. Any unacknowledged TDRs will be resent after OC_InitializeTrackingData is called. Some possible scenarios where unacknowledged TDRs may occur are:

- The caller's system terminates unexpectedly and fails to send the acknowledge
- The content publisher system terminates unexpectedly and fails to receive the acknowledge

Unacknowledged, resent TDRs will have its reset-field set to 1. It is the responsibility of the caller to filter out already-processed TDRs on its side.

A particular handle value may only be acknowledged once.

For more information, see OC_GetTrackingData.

Inputs

- CustomerID
- Handle - from a prior associated OC_GetTrackingData call

Outputs

- None

Additional Faults

- GeneralFault – Invalid Handle

Sorting Out Access Counts for Learning Objects

When using the On-demand Communication, tracking access counts and times for learning objects requires the following additional consideration:

- Learning objects installed into the content consumer using AICC or SCORM metadata files will only send minimum tracking data back to the content consumer via their respective launch and communication rules. For more information, see *Initialize Learning Object Launch Support*. Minimal tracking data does not contain the actual tracking data but it can be used to track access counts and times
- The same learning object types will also send Tracking Data Records via the OC to the content consumer
- Some learning objects, installed into the content consumer, will allow the user to launch other contained learning objects, which the content consumer may not be aware of. These contained learning objects will however generate TDRs.

Given these factors, the simplest way to properly track access counts and times for all learning objects is to do the following:

- Ignore the minimal default data events
- Use only the TDRs for access count or time purposes

START OVER FEATURE

This feature allows the third party LMS to provide "Start Over" parameter values during launch of assets for RO launches, when tracking to OLSA. When present, the “Start Over” condition triggers the OLSA tracking data to disregard any prior history, scores, bookmarks. and start fresh as if the asset had not been visited before. Since the results are in a "Start Over" state, the TDR generated is also flagged for ‘Start Over’.

This feature supports learning object launches for standard courseware content. It will ignore non-trackable learning objects. If a learning object is part of an object that contains other learning objects that you launch with this argument, it will not propagate to any contained learning objects.

One use of the Start Over feature is the ability to track new data without deleting any existing data, which is required for compliance tracking.

AICC Launch Enhancement

For AICC installed content, On-demand Communication supports a search argument that can be specified on the launch URL, to start the specified learning object with a fresh initialized state.

The optional search argument is called start_over and it can have two values 0 and 1; 0 is the default if the argument is omitted.

start_over=0 means launch the learning object using the tracking data state that was saved in the most recent previous launch for the specified user. If the learning object was never launched before then it will be launched in the initialized state. This is the default behavior.

start_over=1 means launch the learning object in the initialized state, regardless of how many times the learning object was previously launched by the specified user. Any previous historical data is saved in the content publisher system.

This feature is only supported for learning object launches for standard courseware content. It will be ignored for non-trackable learning objects.

SCORM Launch Enhancement

For SCORM installed content, OC supports a property specified in the response to LMSGetValue("cmi.launch_data").

The optional property is called start_over and it can have two values 0 and 1; is the default if the property is omitted.

start_over=0 means launch the learning object using the tracking data state that was saved in the most recent previous launch for the specified user. If the learning object was never launched before then it will be launched in the initialized state. This is the default behavior.

start_over=1 means launch the learning object in the initialized state, regardless of how many times the learning object was previously launched by the specified user. Any previous historical data is saved in the content publisher system.

This feature is only supported for learning object launches for standard courseware content. It will be ignored for non-trackable learning objects.

Side-effect of the Start Over Feature

The caller should be aware of the following side-effects that they may encounter with this feature:

1. End-user 'smith' launches a session with Asset A (with no &start_over=1 specified by the content consumer system).

2. End-user 'smith' completes the course and exits session (TDR1 is recorded for this session at time t0).

3. Later at time t1… End-user 'smith' launches another session with the same Asset A. This time it is launched with &start_over=1.

4. End-user 'smith' achieves 50% process in the Asset A and exits session (TDR2 is recorded for this session at time t2).

5. The content consumer makes a OC_GetTrackingData() call. This returns TDR1 and TDR2 in the correct chronological order. However, TDR1 specifies Asset A is complete and TDR2 specifies that Asset A is only 50% complete.

The content consumer must be able to manage this situation when using this feature. The TDRs will have an associated timestamp so that at a minimum, the content consumer will be able to determine the sequence of events at t0, t1 and t2.

Glossary of Terms

L

LMS
Learner Management System

O

OLSA
Open Learning Services Architecture

S

SAMR Service
Selective Asset Metadata Retrieval Service

T

TDR
Tracking Data Record

W

WSDL
Web Services Description Language

Index