Oracle® Student Learning

Programmer's Guide

Release 3.1.3

E20666-04

January 2012

ORACLE®

Oracle Student Learning Programmer's Guide, Release 3.1.3

E20666-04

Contents

B Content Integration Service Objects

List of Tables

List of Figures

Preface

This preface includes the following topics:

Audience

The Oracle Student Learning Programmer's Guide is intended for application developers who are interested in developing applications that interface with Oracle Student Learning (OSL).

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at
http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit
http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit
http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Documents

For more information, see the following documents in the Oracle Student Learning documentation set:

- *Oracle Student Learning (OSL) Installation and Deployment Guide*
- *Oracle Student Learning (OSL) Implementation Guide*
- *Oracle Student Learning (OSL) Learning Tool Admin User's Guide*
- *Oracle Student Learning (OSL) Learning Tool Customization Guide*
- *Oracle Student Learning (OSL) Learning Tool User's Guide*
- *Oracle Student Learning (OSL) Release Notes*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Part I

Data Loading Services

This part contains chapters about data loading services.

1 Overview

This chapter describes an overview of the data loading service.

The Data Loading Services are a collection of web services that provide operations to manage certain entities within OSL. These entities represent data that are typically loaded from external systems and can be broadly classified under the following categories:

- Calendar
- Classes & Enrollment
- Courses & Tags
- Framework
- Institution
- Offerings
- People & Relationship

1.1 Creation of Department

The Data Loading Services do not provide a service to create the Department. The Department is created during installation of OSL. The Data Loading Services provide a service to update the Department attributes.

1.2 Security

The Data Loading Services are secured services that are protected by the WebLogic Security mechanism. The default security policy configuration is defined by `oracle/wss_username_token_service_policy`. Any client access to these services is authenticated and authorized. The default security configuration can be overridden post deployment of the application.

1.2.1 Authentication

The `oracle/wss_username_token_service_policy` requires the client to pass in the credentials in the SOAP Header.

Below is an example of a SOAP message containing the credentials in the SOAP Header.

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:dat="http://www.oracle.com/osl/lt/service/dataloading"
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-sec
```

```
ext-1.0.xsd">
  <soap:Header>
    <wsse:Security>
      <wsse:UsernameToken>
        <wsse:Username>dataloading</wsse:Username>
        <wsse:Password>xxxxxxxxxxx</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    ---
  </soap:Body>
</soap:Envelope>
```

1.2.2 Authorization

All the services that are part of Data Loading are accessible to users that belong to the "DataLoadingGroup". The Data Loading Services expect a user with uid "dataloading" to be part of the "DataLoadingGroup". The authenticated user's uid should be "dataloading" and the user should be part of the "DataLoadingGroup".

1.3 Entity Identification Mechanism

At any point, a record in OSL can be uniquely identified either through the OSL Id or the combination of External Id and External System Id.

1.3.1 oslId

The oslId is the identifier generated by OSL for internal representation of data loaded through the Data Loading Services. The oslId is contained in the response to any successful create operation.

1.3.2 externalId and externalSystemId

OSL external systems require a mechanism to cross-reference individual records. The recommendation is to use an externalId and an externalSystemId to uniquely identify individual records. Any external system loading data into OSL can optionally provide these identifiers. When provided, OSL keeps track of them and external systems can cross-reference these records based on the externalId and externalSystemId.

1.4 Action List and List

To explain this concept, the PersonInfo data structure is used.

```
<xs:complexType name="personInfo">
  <xs:sequence>
    <xs:element name="academicYearLevel" type="xs:string" minOccurs="0"/>
    <xs:element name="addressActions" type="tns:addressActionInfoArray"
                minOccurs="0"/>
    <xs:element name="addresses" type="tns:addressInfo" minOccurs="0"
                maxOccurs="unbounded"/>
    <xs:element name="birthDate" type="xs:dateTime" minOccurs="0"/>
    <xs:element name="countryOfBirth" type="xs:string" minOccurs="0"/>
    <xs:element name="deathDate" type="xs:dateTime" minOccurs="0"/>
    <xs:element name="emailContactActions" type="tns:contactActionInfoArray"
                minOccurs="0"/>
    <xs:element name="emailContacts" type="tns:contactInfo" minOccurs="0"
```

```
                maxOccurs="unbounded"/>
      <xs:element name="firstName" type="xs:string" minOccurs="0"/>
      <xs:element name="gender" type="tns:personGender" minOccurs="0"/>
      <xs:element name="id" type="tns:idInfo" minOccurs="0"/>
      <xs:element name="languageActions" type="tns:languageActionInfoArray"
                minOccurs="0"/>
      <xs:element name="languages" type="tns:languageInfo" minOccurs="0"
                maxOccurs="unbounded"/>
      <xs:element name="lastName" type="xs:string" minOccurs="0"/>
      <xs:element name="loginId" type="xs:string" minOccurs="0"/>
      <xs:element name="maritalStatus" type="xs:string" minOccurs="0"/>
      <xs:element name="middleName" type="xs:string" minOccurs="0"/>
      <xs:element name="nameSuffix" type="xs:string" minOccurs="0"/>
      <xs:element name="phoneContactActions" type="tns:contactActionInfoArray"
                minOccurs="0"/>
      <xs:element name="phoneContacts" type="tns:contactInfo" minOccurs="0"
                maxOccurs="unbounded"/>
      <xs:element name="photograph" type="xs:base64Binary" minOccurs="0"/>
      <xs:element name="placeOfBirth" type="xs:string" minOccurs="0"/>
      <xs:element name="preferredName" type="xs:string" minOccurs="0"/>
      <xs:element name="relationshipActions"
                type="tns:relationshipActionInfoArray" minOccurs="0"/>
      <xs:element name="relationships" type="tns:relationshipInfo" minOccurs="0"
                maxOccurs="unbounded"/>
      <xs:element name="title" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
```

In the PersonInfo structure, phoneContactActions and relationshipActions are Action Lists. In the same structure, addresses, emailContacts, languages, phoneContacts and relationships are Lists.

Lists are returned by the service in response to certain operation and Lists are never sent in any request message. In other words, Lists are always constructed by the service and clients typically read the Lists.

The data structure of the Action Lists is similar to that of Lists except that an action flag on individual element indicates the action that is being performed on them. A client always sends an Action List to the service during create, createOrUpdate and update operation. The action flag can have the following permissible values:

```
 <xs:simpleType name="action">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Create"/>
      <xs:enumeration value="Update"/>
      <xs:enumeration value="Delete"/>
      <xs:enumeration value="CreateOrUpdate"/>
    </xs:restriction>
  </xs:simpleType>
```

During a create operation, the service assumes an implicit value of "Create" for the action flag. During an update and createOrUpdate operation, any of the permissible values can be specified and the service processes accordingly.

For instance, if a new address is to be added to a PersonInfo object, an action flag of "Create" should be passed in for that element during the updatePersons operation.

Similarly, if an existing address is to be deleted from a PersonInfo object, an action flag of "Delete" should be passed in for the element during the updatePersons operation, in addition to passing the identifiers that can uniquely identify the element.

If an action flag of "CreateOrUpdate" is provided for the address element during the updatePersons operation, if the address exists, it shall be updated. If it does not exist, it will be created and appended to the existing list of addresses.

1.5 Entity Operations

The Data Loading web services provide CRUD (Create, Read, Update, and Delete) operations on most of the entities. Delete operations are not supported on Person and Institution entities. This is to preserve data consistency as deleting a Person or an Institution could have adverse effect. The semantics of the various operations supported by Data Loading Services are summarized in the following sections.

1.5.1 create Operation

This operation supports the creation of an array of homogeneous entities as an atomic unit of work. The entire transaction is rolled back if any failure occurs during the creation process.

On successful creation, the response includes the oslId of each entity that has been created.

1.5.2 createOrUpdate Operation

This operation supports creating or updating an array of homogeneous entities as an atomic unit of work. The entire transaction is rolled back if any failure occurs during the create or update process.

If an oslId is specified, the service tries to locate the entity to update based on the oslId. If found, the entity is updated.

If no entity is retrieved for the specified oslId, the services check if the combination of externalId and externalSystemId is specified. If specified, the service tries to locate the entity based on the externalId & externalSystemId. If the entity is found, it is updated.

If the identification mechanism does not fetch an entity, a new record is created and the oslId pertaining to the record is returned to the caller in the response.

1.5.3 update Operation

This operation supports updating an array of homogeneous entities as an atomic unit of work. The entire transaction is rolled back if any failure occurs during the update process.

If an oslId is specified, the service tries to locate the entity to update based on the oslId. If found, the entity is updated.

If no entity is retrieved for the specified oslId, the service checks if the combination of externalId and externalSystemId is specified. If specified, the service tries to locate the entity based on the externalId & externalSystemId. If the entity is found, it is updated.

If the identification mechanism does not fetch an entity, an error is raised.

In an update operation, it is not required to send the entire object structure. In addition to the identification of the entity, it is sufficient to specify only the elements to be modified. The service updates the entity with only the modified elements. For instance, if the preferredName of a person with oslId 100 is modified to "John", it is sufficient to send the following information to the service.

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
```

```
xmlns:dat="http://www.oracle.com/osl/lt/service/dataloading"
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-sec
ext-1.0.xsd">
  <soap:Header>
    <wsse:Security>
      <wsse:UsernameToken>
        <wsse:Username>dataloading</wsse:Username>
        <wsse:Password>welcome1</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap:Header>
  <soap:Body>
    <dat:createPersons>
      <persons>
        <id>
          <oslId>100</oslId>
        </id>
        <preferredName>John</preferredName>
      </persons>
    </dat:createPersons>
  </soap:Body>
</soap:Envelope>
```

1.5.4 remove Operation

This operation supports deleting an array of homogeneous entities as an atomic unit of work. The entire transaction is rolled back if any failure occurs during the deletion process.

If an oslId is specified, the service tries to locate the entity to delete based on the oslId. If found, the entity is deleted.

If no entity is retrieved for the specified oslId, the service checks if the combination of externalId and externalSystemId is specified. If specified, the service tries to locate the entity based on the externalId & externalSystemId. If the entity is found, it is deleted.

Only the object identification mechanism must be specified for the remove operation. If other elements are specified, those are ignored by the service.

1.5.5 get Operation

This operation supports querying entities based on the identifiers.

If an oslId is specified, the service tries to locate the entity based on the oslId. If found, the entity is returned.

If no entity is retrieved for the specified oslId, the service checks if the combination of externalId and externalSystemId is specified. If specified, the service tries to locate the entity based on the externalId & externalSystemId. If the entity is found, it is returned.

Only the object identification mechanism must be specified for the get operation, if other elements are specified, those are ignored by the service.

1.6 Web Service Exceptions

During Data Loading Service operations, Web Service Exceptions might be encountered. These exceptions indicate possible errors or business rule violations in the input parameters. To help locate the root cause, error codes and error messages are contained within the exceptions. A summary of the Web Service Exceptions are provided below.

1.6.1 CreationException

A CreationException is thrown when the system cannot create the specified entity when attempting to save the data into the database.

1.6.2 DataConstraintException

A DataConstraintException is thrown when the modified data is in violation of the business rules. Common situations include updating the start date of a framework such that the start date is later than the end date or creating a framework with a non-unique name.

1.6.3 InvalidParameter

An InvalidParameter exception is thrown when the input parameters are invalid or required fields are missing.

1.6.4 RemoveException

A RemoveException is thrown when an entity cannot be removed. This occurs when an attempt is made to remove an entity that is in use. For example, an attempt is made to remove a context adoption when the adopted context is referenced in a learning item demonstration.

1.6.5 UpdateException

An UpdateException is thrown when the system cannot update the specified entity when attempting to save the data into the database.

1.6.6 ServiceException

A ServiceException is a general service exception that is thrown for an unspecified condition.

2

Web Services

This chapter describes the available web services and their descriptions.

2.1 DataLoadingPartyService

The DataLoadingPartyService is used to load party information. This includes institutions, institution groups, people and their relationships.

2.1.1 createInstitutions

This method creates new institutions. Note that the new institutions are created according to the order in which they are provided. For example, the first institution in the list is created followed by the second institution in the list.

All new institutions must have a parent institution. This parent institution must be an existing institution or created before it. In addition, under the same parent institution, the names of child institutions must be unique.

An institution should have only one primary address, email contact, and phone contact.

Parameter:

Table 2–1 createInstitutions Parameter Summary

Type	Description
InstitutionInfo[]	List of institutions to create

Return:

Table 2–2 createInstitutions Return Summary

Type	Description
InstitutionInfo[]	List of created institutions

Exception:

CreationException, DataConstraintException, InvalidParameters, ServiceException

2.1.2 createInstitutionGroups

This method creates new institution groups. The institution group name must be unique.

Parameter:

Table 2–3 createInstitutionGroups Parameter Summary

Type	Description
InstitutionGroupInfo[]	List of institution groups to create

Return:

Table 2–4 createInstitutionGroups Return Summary

Type	Description
InstitutionGroupInfo[]	List of created institution groups

Exception:

CreationException, DataConstraintException, InvalidParameters, ServiceException

2.1.3 createOrUpdateInstitutions

This method searches for the institutions using the IdInfo in the input InstitutionInfo and if the institution is found, it is updated; Otherwise, it is created. Note that the parent institution attribute cannot be updated.

Parameter:

Table 2–5 createOrUpdateInstitutions Parameter Summary

Type	Description
InstitutionInfo[]	List of institutions to create or update

Return:

Table 2–6 createOrUpdateInstitutions Return Summary

Type	Description
InstitutionInfo[]	List of institutions created or updated

Exception:

CreationException, DataConstraintException, InvalidParameters, ServiceException, UpdateException

2.1.4 createOrUpdateInstitutionGroups

This method searches for the institution groups using the IdInfo in the input InstitutionGroupInfo and if the institution group is found, it is updated; Otherwise, it is created.

Note: Only *Create* and *Delete* actions are allowed in the IdInfoActionInfo.

Parameter:

Table 2–7 createOrUpdateInstitutionGroups Parameter Summary

Type	Description
InstitutionGroupInfo[]	List of institution groups to create or update

Return:

Table 2–8 createOrUpdateInstitutionGroups Return Summary

Type	Description
InstitutionGroupInfo[]	List of institution groups created or updated

Exception:

CreationException, DataConstraintException, InvalidParameters, ServiceException, UpdateException

2.1.5 createOrUpdatePersons

This method searches for the persons using the IdInfo in the input PersonInfo and if the person is found, it is updated; Otherwise, it is created.

Parameter:

Table 2–9 createOrUpdatePersons Parameter Summary

Type	Description
PersonInfo[]	List of persons to create or update

Return:

Table 2–10 createOrUpdatePersons Return Summary

Type	Description
PersonInfo[]	List of persons created or updated

Exception:

CreationException, DataConstraintException, InvalidParameters, ServiceException, UpdateException

2.1.6 createPersons

This method creates new persons. A person should have only one primary address, email contact, and phone contact.

Parameter:

Table 2–11 createPersons Parameter Summary

Type	Description
PersonInfo[]	List of persons to create

Return:

Table 2–12 createPersons Return Summary

Type	Description
PersonInfo[]	List of created persons

Exception:

CreationException, DataConstraintException, InvalidParameters, ServiceException

2.1.7 getDepartment

This method returns the department.

Parameter:

None.

Return:

Table 2–13 getDepartment Return Summary

Type	Description
InstitutionInfo	Department

Exception:

ServiceException

2.1.8 getInstitutions

This method finds all the institutions using the IdInfo in the input InstitutionInfo and returns the complete InstitutionInfo including the ContactInfos and AddressInfos.

If the specified institution is not found in the system, no record is returned.

Parameter:

Table 2–14 getInstitutions Parameter Summary

Type	Description
InstitutionInfo[]	List of institutions to retrieve

Return:

Table 2–15 getInstitutions Return Summary

Type	Description
InstitutionInfo[]	List of institutions retrieved

Exception:

InvalidParameters, ServiceException

2.1.9 getInstitutionGroups

This method finds all the institution groups using the IdInfo in the input InstitutionGroupInfo and returns the complete InstitutionGroupInfo.

If the specified institution group is not found in the system, no record is returned.

Parameter:

Table 2–16 getInstitutionGroups Parameter Summary

Type	Description
InstitutionGroupInfo[]	List of institution groups to retrieve

Return:

Table 2–17 getInstitutionGroups Return Summary

Type	Description
InstitutionGroupInfo[]	List of institution groups retrieved

Exception:

InvalidParameters, ServiceException

2.1.10 getPersons

This method finds all the persons using the IdInfo in the input PersonInfo and returns the complete PersonInfo including the LanguageInfos, ContactInfos, and AddressInfos.

If the specified person is not found in the system, no record is returned.

Parameter:

Table 2–18 getPersons Parameter Summary

Type	Description
PersonInfo[]	List of persons to retrieve

Return:

Table 2–19 getPersons Return Summary

Type	Description
PersonInfo[]	List of persons retrieved

Exception:

InvalidParameters, ServiceException

2.1.11 updateInstitutions

This method searches for the institutions using the IdInfo in the input InstitutionInfo and updates the institutions with the specified attributes. Note that the parent institution cannot be updated.

Parameter:

Table 2–20 updateInstitutions Parameter Summary

Type	Description
InstitutionInfo[]	List of institutions to update

Return:

Table 2–21 updateInstitutions Return Summary

Type	Description
InstitutionInfo[]	List of updated institutions

Exception:

DataConstraintException, InvalidParameters, ServiceException, UpdateException

2.1.12 updateInstitutionGroups

This method searches for the institution groups using the IdInfo in the input InstitutionGroupInfo and updates the institution groups with the specified attributes.

> **Note:** Only *Create* and *Delete* actions are allowed in the IdInfoActionInfo.

Parameter:

Table 2–22 updateInstitutionGroups Parameter Summary

Type	Description
InstitutionGroupInfo[]	List of institution groups to update

Return:

Table 2–23 updateInstitutionGroups Return Summary

Type	Description
InstitutionGroupInfo[]	List of updated institution groups

Exception:

DataConstraintException, InvalidParameters, ServiceException, UpdateException

2.1.13 updatePersons

This method searches for the persons using the IdInfo in the input PersonInfo and updates the persons with the specified attributes.

Parameter:

Table 2–24 updatePersons Parameter Summary

Type	Description
PersonInfo[]	List of persons to update

Return:

Table 2–25 updatePersons Return Summary

Type	Description
PersonInfo[]	List of updated persons

Exception:

DataConstraintException, InvalidParameters, ServiceException, UpdateException

2.2 DataLoadingCourseService

The DataLoadingCourseService is used to load course, course tags, offerings, and class information.

2.2.1 createClasses

This method creates new classes unique within the context of the associated offering. In addition, note the following:

- The students and teachers enrolled in the class must come from the same school as the class
- The start date of both the student enrollments and teacher enrollments must be on or after the start date of the class
- The end date of both the student enrollments and teacher enrollments must be on or before the end date of the class
- The start and end dates of the class must be within the start and end dates of the offering.
- The offering must belong to the school of the class.

Parameter:

Table 2–26 createClasses Parameter Summary

Type	Description
ClassInfo[]	List of classes to create

Return:

Table 2–27 createClasses Return Summary

Type	Description
ClassInfo[]	List of created classes

Exception:

CreationException, DataConstraintException, InvalidParameters, ServiceException

2.2.2 createCourses

This method creates new courses. Within a school, the course name and code must be unique. The tags attached to the course must also come from the same school.

Parameter:

Table 2–28 createCourses Parameter Summary

Type	Description
CourseInfo[]	List of courses to create

Return:

Table 2–29 createCourses Return Summary

Type	Description
CourseInfo[]	List of created courses

Exception:

CreationException, DataConstraintException, InvalidParameters, ServiceException

2.2.3 createOfferings

This method creates new offerings. Within a school, the offering name and code must be unique within the calendar. In addition, note the following:

- The framework specified must be adopted by the school.
- The course specified must belong to the same school as the offering.
- The calendar of the offering must be the calendar of the adopted framework or one of its sub-calendars.
- The start and end dates of the offering must be between the start and end dates of the calendar.
- There can only be one offering for a course, framework, and calendar combination and if the class is found.
- The calendar must be one of the adopted calendars of the school, or one of their sub-calendars.

Parameter:

Table 2–30 createOfferings Parameter Summary

Type	Description
OfferingInfo[]	List of offerings to create

Return:

Table 2–31 createOfferings Return Summary

Type	Description
OfferingInfo[]	List of created offerings

Exception:

CreationException, DataConstraintException, InvalidParameters, ServiceException

2.2.4 createOrUpdateClasses

This method searches for the classes using the IdInfo in the input ClassInfo and if class is found, it is updated; Otherwise, it is created. The restrictions in createClasses and updateClasses are also applicable here.

Parameter:

Table 2–32 createOrUpdateClasses Parameter Summary

Type	Description
ClassInfo[]	List of classes to create or update

Return:

Table 2–33 createOrUpdateClasses Return Summary

Type	Description
ClassInfo[]	List of classes created or updated

Exception:

CreationException, DataConstraintException, InvalidParameters, ServiceException, UpdateException

2.2.5 createOrUpdateCourses

This method searches for the courses using the IdInfo in the input CourseInfo and if the course is found, it is updated; Otherwise, it is created. Note that the institution of the course cannot be updated.

Parameter:

Table 2–34 createOrUpdateCourses Parameter Summary

Type	Description
CourseInfo[]	List of courses to create or update

Return:

Table 2–35 createOrUpdateCourses Return Summary

Type	Description
CourseInfo[]	List of courses created or updated

Exception:

CreationException, DataConstraintException, InvalidParameters, ServiceException, UpdateException

2.2.6 createOrUpdateOfferings

This method searches for the offerings using the IdInfo in the input OfferingInfo and if the offering is found, it is updated; Otherwise, it is created. The restrictions for createOfferings and updateOfferings are also applicable here.

Parameter:

Table 2–36 createOrUpdateOfferings Parameter Summary

Type	Description
OfferingInfo[]	List of offerings to create or update

Return:

Table 2–37 createOrUpdateOfferings Return Summary

Type	Description
OfferingInfo[]	List of offerings created or updated

Exception:

CreationException, DataConstraintException, InvalidParameters, ServiceException, UpdateException

2.2.7 createOrUpdateTags

This method searches for the tags using the IdInfo in the input TagInfo and if the tag is found, it is updated; Otherwise, it is created.

Parameter:

Table 2–38 createOrUpdateTags Parameter Summary

Type	Description
TagInfo[]	List of tags to create or update

Return:

Table 2–39 createOrUpdateTags Return Summary

Type	Description
TagInfo[]	List of tags created or updated

Exception:

CreationException, DataConstraintException, InvalidParameters, ServiceException, UpdateException

2.2.8 createTags

This method creates new tags. Within a school, the tag name must be unique.

Parameter:

Table 2–40 createTags Parameter Summary

Type	Description
TagInfo[]	List of tags to create

Return:

Table 2–41 createTags Return Summary

Type	Description
TagInfo[]	List of created tags

Exception:

CreationException, DataConstraintException, InvalidParameters, ServiceException

2.2.9 getClasses

This method finds all the classes using the IdInfo in the input ClassInfo and returns the complete ClassInfo.

If the specified class is not found in the system, no record is returned.

Parameter:

Table 2–42 getClasses Parameter Summary

Type	Description
ClassInfo[]	List of classes to retrieve

Return:

Table 2–43 getClasses Return Summary

Type	Description
ClassInfo[]	List of classes retrieved

Exception:

InvalidParameters, ServiceException

2.2.10 getCourses

This method finds all the courses using the IdInfo in the input CourseInfo and returns the complete CourseInfo.

If the specified course is not found in the system, no record is returned.

Parameter:

Table 2–44 getCourses Parameter Summary

Type	Description
CourseInfo[]	List of courses to retrieve

Return:

Table 2–45 getCourses Return Summary

Type	Description
CourseInfo[]	List of courses retrieved

Exception:

InvalidParameters, ServiceException

2.2.11 getOfferings

This method finds all the offerings using the IdInfo in the input OfferingInfo and returns the complete OfferingInfo.

If the specified offering is not found in the system, no record is returned.

Parameter:

Table 2–46 getOfferings Parameter Summary

Type	Description
OfferingInfo[]	List of offerings to retrieve

Return:

Table 2–47 getOfferings Return Summary

Type	Description
OfferingInfo[]	List of offerings retrieved

Exception:

InvalidParameters, ServiceException

2.2.12 getTags

This method finds all the tags using the IdInfo in the input TagInfo and returns the complete TagInfo.

If the specified tag is not found in the system, no record is returned.

Parameter:

Table 2–48 getTags Parameter Summary

Type	Description
TagInfo[]	List of tags to retrieve

Return:

Table 2–49 getTags Return Summary

Type	Description
TagInfo[]	List of tags retrieved

Exception:

InvalidParameters, ServiceException

2.2.13 updateClasses

This method searches for the classes using the IdInfo in the input ClassInfo and updates the classes with the specified attributes. The offering and school of the class cannot be updated.

In addition, enrollments cannot be changed if the class is end-dated. Removing teacher enrollments and student enrollments are also prohibited if there is one or more learning items created in the class lesson plan.

Parameter:

Table 2–50 updateClasses Parameter Summary

Type	Description
ClassInfo[]	List of classes to update

Return:

Table 2–51 updateClasses Return Summary

Type	Description
ClassInfo[]	List of updated classes

Exception:

DataConstraintException, InvalidParameters, ServiceException, UpdateException

2.2.14 updateCourses

This method searches for the courses using the IdInfo in the input CourseInfo and updates the course with the specified attributes. Note that the institution of the course cannot be updated.

Parameter:

Table 2–52 updateCourses Parameter Summary

Type	Description
CourseInfo[]	List of courses to update

Return:

Table 2–53 updateCourses Return Summary

Type	Description
CourseInfo[]	List of updated courses

Exception:

DataConstraintException, InvalidParameters, ServiceException, UpdateException

2.2.15 updateOfferings

This method searches for the offerings using the IdInfo in the input OfferingInfo and updates the offering with the specified attributes. Note that the calendar, course, framework, and school cannot be updated.

Parameter:

Table 2–54 updateOfferings Parameter Summary

Type	Description
OfferingInfo[]	List of offerings to update

Return:

Table 2–55 updateOfferings Return Summary

Type	Description
OfferingInfo[]	List of updated offerings

Exception:

DataConstraintException, InvalidParameters, ServiceException, UpdateException

2.2.16 updateTags

This method searches for the tags using the IdInfo in the input TagInfo and updates the tag with the specified attributes.

Parameter:

Table 2–56 updateTags Parameter Summary

Type	Description
TagInfo[]	List of tags to update

Return:

Table 2–57 updateTags Return Summary

Type	Description
TagInfo[]	List of updated tags

Exception:

DataConstraintException, InvalidParameters, ServiceException, UpdateException

2.2.17 removeClasses

This method finds all the classes using the IdInfo in the input ClassInfo and removes it. A class cannot be removed if there are enrollments in the class.

Parameter:

Table 2–58 removeClasses Parameter Summary

Type	Description
ClassInfo[]	List of classes to remove

Return:

Table 2–59 removeClasses Return Summary

Type	Description
ClassInfo[]	List of removed classes

Exception:

DataConstraintException, InvalidParameters, RemoveException, ServiceException

2.2.18 removeCourses

This method finds all the courses using the IdInfo in the input CourseInfo and removes them. If a course is referenced by an offering, then it cannot be removed.

Parameter:

Table 2–60 removeCourses Parameter Summary

Type	Description
CourseInfo[]	List of courses to remove

Return:

Table 2–61 removeCourses Return Summary

Type	Description
CourseInfo[]	List of removed courses

Exception:

DataConstraintException, InvalidParameters, RemoveException, ServiceException

2.2.19 removeOfferings

This method finds all the offerings using the IdInfo in the input OfferingInfo and removes them. An offering cannot be removed if there are any classes referencing it.

Parameter:

Table 2–62 removeOfferings Parameter Summary

Type	Description
OfferingInfo[]	List of offerings to remove

Return:

Table 2–63 removeOfferings Return Summary

Type	Description
OfferingInfo[]	List of removed offerings

Exception:

DataConstraintException, InvalidParameters, RemoveException, ServiceException

2.2.20 removeTags

This method finds all the tags using the IdInfo in the input TagInfo and removes them. If a tag is used in any course, it cannot be removed.

Parameter:

Table 2–64 removeTags Parameter Summary

Type	Description
TagInfo[]	List of tags to remove

Return:

Table 2–65 removeTags Return Summary

Type	Description
TagInfo[]	List of removed tags

Exception:

DataConstraintException, InvalidParameters, RemoveException, ServiceException

2.3 DataLoadingCalendarService

The DataLoadingCalendarService is used to load calendar information and also to adopt department calendars.

2.3.1 adoptCalendars

This method adopts a department calendar for use at a school. Only department calendars can be adopted. A calendar can only be adopted by a school once. Note that the parent calendar has to be adopted before the sub-calendars can be adopted.

Parameter:

Table 2–66 adoptCalendars Parameter Summary

Type	Description
AdoptCalendarInfo[]	List of department calendars to adopt

Return:

Table 2–67 adoptCalendars Return Summary

Type	Description
CalendarInfo[]	List of school calendars

Exception:

CreationException, DataConstraintException, InvalidParameters, ServiceException

2.3.2 createCalendars

This method creates new calendars. The calendar name must be unique within an institution. For a sub-calendar, the sub-calendar name must be unique under the parent calendar.

Note that the start and end date of a sub-calendar must not exceed the start and end date of the parent calendar.

Parameter:

Table 2–68 createCalendars Parameter Summary

Type	Description
CalendarInfo[]	List of calendars to create

Return:

Table 2–69 createCalendars Return Summary

Type	Description
CalendarInfo[]	List of created calendars

Exception:

CreationException, DataConstraintException, InvalidParameters, ServiceException

2.3.3 createOrUpdateCalendars

This method searches for the calendars using the IdInfo in the input CalendarInfo and if a calendar is found, it is updated; Otherwise, it is created. The restrictions in createCalendars and updateCalendars are also applicable here.

Parameter:

Table 2–70 createOrUpdateCalendars Parameter Summary

Type	Description
CalendarInfo[]	List of calendars to create or update

Return:

Table 2–71 createOrUpdateCalendars Return Summary

Type	Description
CalendarInfo[]	List of calendars created or updated

Exception:

CreationException, DataConstraintException, InvalidParameters, ServiceException, UpdateException

2.3.4 getCalendars

This method finds all the calendars using the IdInfo in the input CalendarInfo and returns the complete CalendarInfo.

If the specified calendar is not found in the system, no record is returned.

Parameter:

Table 2–72 getCalendars Parameter Summary

Type	Description
CalendarInfo[]	List of calendars to retrieve

Return:

Table 2–73 getCalendars Return Summary

Type	Description
CalendarInfo[]	List of calendars retrieved

Exception:

InvalidParameters, ServiceException

2.3.5 updateCalendars

This method searches for the calendars using the IdInfo in the input CalendarInfo and updates the calendars with the specified attributes.

Note that the parent calendar association cannot be updated.

Parameter:

Table 2–74 updateCalendars Parameter Summary

Type	Description
CalendarInfo[]	List of calendars to update

Return:

Table 2–75 updateCalendars Return Summary

Type	Description
CalendarInfo[]	List of updated calendars

Exception:

DataConstraintException, InvalidParameters, ServiceException, UpdateException

2.3.6 removeCalendars

This method finds all the calendars using the IdInfo in the input CalendarInfo and removes them. When a calendar is removed, all the sub-calendars under it are also removed. However, a school calendar cannot be removed if it is referenced in a framework adoption or an offering.

Parameter:

Table 2–76 removeCalendars Parameter Summary

Type	Description
CalendarInfo[]	List of calendars to remove

Return:

Table 2–77 removeCalendars Return Summary

Type	Description
CalendarInfo[]	List of removed calendars

Exception:

DataConstraintException, InvalidParameters, RemoveException, ServiceException

2.4 DataLoadingCurriculumService

The DataLoadingCurriculumService is used to load framework and framework-related information such as framework items, blueprint nodes, and proficiency sets.

2.4.1 createBlueprintHiearchies

This method creates a new blueprint node hierarchy. A blueprint node with no parent blueprint node is created as the root blueprint node. There can only be one root blueprint node for a framework. Framework node names must be unique within a framework. Framework node start and end dates must be between the start and end dates of both the parent framework node and the framework.

Note: All blueprint nodes directly or indirectly under the root blueprint node are created in the same framework as specified in the framework of the root blueprint node.

Parameter:

Table 2–78 createBlueprintHiearchies Parameter Summary

Type	Description
BlueprintNodeInfo[]	List of blueprint nodes to create

Return:

Table 2–79 createBlueprintHiearchies Return Summary

Type	Description
BlueprintNodeInfo[]	List of created blueprint nodes

Exception:

CreationException, DataConstraintException, InvalidParameters, ServiceException

2.4.2 createContexts

This method creates new contexts. Contexts with no parent context specified are created as department contexts. Contexts with parent context specified are created as school contexts. School contexts cannot have child contexts.

Parameter:

Table 2–80 createContexts Parameter Summary

Type	Description
ContextInfo[]	List of contexts to create

Return:

Table 2–81 createContexts Return Summary

Type	Description
ContextInfo[]	List of created contexts

Exception:

CreationException, DataConstraintException, InvalidParameters, ServiceException

2.4.3 createOrUpdateBlueprintHiearchies

This method searches for the blueprint nodes using the IdInfo in the input BlueprintNodeInfo and if the blueprint node is found, it is updated; Otherwise, it is created. The restrictions in createBlueprintHierarchies and updateBlueprintHierarchies are also applicable here.

Parameter:

Table 2–82 createOrUpdateBlueprintHiearchies Parameter Summary

Type	Description
BlueprintNodeInfo[]	List of blueprint nodes to create or update

Return:

Table 2–83 createOrUpdateBlueprintHiearchies Return Summary

Type	Description
BlueprintNodeInfo[]	List of blueprint nodes created or updated

Exception:

CreationException, DataConstraintException, InvalidParameters, ServiceException, UpdateException

2.4.4 createOrUpdateContexts

This method searches for the contexts using the IdInfo in the input ContextInfo and if a context is found, it is updated; Otherwise, it is created.

The restrictions in createContexts and updateContexts are applicable here.

Parameter:

Table 2–84 createOrUpdateContexts Parameter Summary

Type	Description
ContextInfo[]	List of contexts to create or update

Return:

Table 2–85 createOrUpdateContexts Return Summary

Type	Description
ContextInfo[]	List of contexts created or updated

Exception:

CreationException, DataConstraintException, InvalidParameters, ServiceException, UpdateException

2.4.5 createorUpdateOutcomeStatementDescriptorType

This method searches for the outcome statement descriptor types using the IdInfo in the input OutcomeStatementDescTypeInfo and if an outcome statement descriptor type is found, it is updated; Otherwise, it is created.

The restrictions in createOutcomeStatementDescTypes and updateOutcomeStatementDescTypes are applicable here.

Parameter:

Table 2–86 createorUpdateOutcomeStatementDescriptorType Parameter Summary

Type	Description
OutcomeStatementDescTypeInfo[]	List of outcome statement descriptor types to create or update

Return:

Table 2–87 createorUpdateOutcomeStatementDescriptorType Return Summary

Type	Description
OutcomeStatementDescTypeInfo[]	List of outcome statement descriptor types created or updated

Exception:

CreationException, DataConstraintException, InvalidParameters, ServiceException, UpdateException

2.4.6 createOrUpdateOutcomeStatements

This method searches for the outcome statements using the IdInfo in the input OutcomeStatementInfo and if the outcome statement is found, it is updated; Otherwise, it is created.

The restrictions in createOutcomeStatements and updateOutcomeStatements are also applicable here.

Parameter:

Table 2–88 createOrUpdateOutcomeStatements Parameter Summary

Type	Description
OutcomeStatementInfo[]	List of outcome statements to create or update

Return:

Table 2–89 createOrUpdateOutcomeStatements Return Summary

Type	Description
OutcomeStatementInfo[]	List of outcome statements created or updated

Exception:

CreationException, DataConstraintException, InvalidParameters, ServiceException, UpdateException

2.4.7 createOutcomeStatements

This method creates new outcome statements. The proficiency level specified must be from the outcome proficiency set associated with the specified framework item. Also, this proficiency level must not be a proficiency sub-level (For example, a proficiency level with a parent proficiency level).

Outcome statement names must be unique within a framework item. Demonstrable outcome statements must have an associated proficiency level. Indicative outcome statements must not have an associated proficiency level. The start and end dates of the outcome statement must be between those of the framework item.

Parameter:

Table 2–90 createOutcomeStatements Parameter Summary

Type	Description
OutcomeStatementInfo[]	List of outcome statements to create

Return:

Table 2–91 createOutcomeStatements Return Summary

Type	Description
OutcomeStatementInfo[]	List of created outcome statements

Exception:

CreationException, DataConstraintException, InvalidParameters, ServiceException

2.4.8 createOutcomeStatementDescriptorType

This method creates new outcome statement descriptor types. Within a framework, the outcome statement descriptor type name must be unique.

Parameter:

Table 2–92 createOutcomeStatementDescriptorType Parameter Summary

Type	Description
OutcomeStatementDescTypeInfo[]	List of outcome statement descriptor types to create

Return:

Table 2–93 createOutcomeStatementDescriptorType Return Summary

Type	Description
OutcomeStatementDescTypeInfo[]	List of created outcome statement descriptor types

Exception:

CreationException, DataConstraintException, InvalidParameters, ServiceException

2.4.9 createFrameworks

This method creates new Frameworks. The name of each framework must be unique in the system.

Parameter:

Table 2–94 createFrameworks Parameter Summary

Type	Description
FrameworkInfo[]	List of frameworks to create

Return:

Table 2–95 createFrameworks Return Summary

Type	Description
FrameworkInfo[]	List of created frameworks

Exception:

CreationException, DataConstraintException, InvalidParameters, ServiceException

2.4.10 createFrameworkItemHiearchies

This method creates new Framework Item Hierarchies. A framework item with no parent framework item is created as the top level framework item. The blueprint node of top level framework item is set to the root blueprint node.

Note the following:

- The outcome proficiency set, if specified, must belong to the same framework as the framework item

- The specified blueprint node must belong to the same framework as the framework item
- The framework item must be aligned to the specified blueprint node. (For example, the specified blueprint node must be the child of the blueprint node of the parent framework item)
- Framework item codes must be unique within a framework.
- If a framework item's "Confirming Level" flag is set, then a proficiency set must be specified.
- A framework item's start and end dates must be between those of the blueprint node and the parent framework item.

Parameter:

Table 2–96 createFrameworkItemHiearchies Parameter Summary

Type	Description
FrameworkItemInfo[]	List of framework items to create

Return:

Table 2–97 createFrameworkItemHiearchies Return Summary

Type	Description
FrameworkItemInfo[]	List of created framework items

Exception:

CreationException, DataConstraintException, InvalidParameters, ServiceException

2.4.11 createOrUpdateFrameworkItemHiearchies

This method searches for the Framework item nodes using the IdInfo in the input FrameworkItemInfo and if a framework item is found, it is updated; Otherwise, it is created. The restrictions in createFrameworkItemHierarchies and updateFrameworkItemHierarchies are also applicable here.

Parameter:

Table 2–98 createOrUpdateFrameworkItemHiearchies Parameter Summary

Type	Description
FrameworkItemInfo[]	List of framework items to create or update

Return:

Table 2–99 createOrUpdateFrameworkItemHiearchies Return Summary

Type	Description
FrameworkItemInfo[]	List of framework items created or updated

Exception:

CreationException, DataConstraintException, InvalidParameters, ServiceException, UpdateException

2.4.12 createOrUpdateFrameworks

This method searches for the frameworks using the IdInfo in the input FrameworkInfo and if a framework is found, it is updated; Otherwise, it is created.

Parameter:

Table 2–100 createOrUpdateFrameworks Parameter Summary

Type	Description
FrameworkInfo[]	List of frameworks to create or update

Return:

Table 2–101 createOrUpdateFrameworks Return Summary

Type	Description
FrameworkInfo[]	List of frameworks created or updated

Exception:

CreationException, DataConstraintException, InvalidParameters, ServiceException, UpdateException

2.4.13 createOrUpdateProficiencySets

This method searches for the proficiency sets using the IdInfo in the input ProficiencySetInfo and if a proficiency set is found, it is updated; Otherwise, it is created. The restrictions in createProficiencySets and updateProficiencySets are also applicable here.

Parameter:

Table 2–102 createOrUpdateProficiencySets Parameter Summary

Type	Description
ProficiencySetInfo[]	List of proficiency sets to create or update

Return:

Table 2–103 createOrUpdateProficiencySets Return Summary

Type	Description
ProficiencySetInfo[]	List of proficiency sets created or updated

Exception:

CreationException, DataConstraintException, InvalidParameters, ServiceException, UpdateException

2.4.14 createProficiencySets

This method creates new outcome proficiency sets. Note that the outcome proficiency levels can have at most only one level of parent/child relationship. For example, an outcome proficiency level can have one or more proficiency sub-levels as its child, but the proficiency sub-levels cannot have subordinate proficiency levels.

In addition, within an outcome proficiency set, all proficiency level names must be unique. Every proficiency set with a framework must have a unique name. The start and end dates of a proficiency set must be between those of the framework.

Parameter:

Table 2–104 createProficiencySets Parameter Summary

Type	Description
ProficiencySetInfo[]	List of outcome proficiency sets to create

Return:

Table 2–105 createProficiencySets Return Summary

Type	Description
ProficiencySetInfo[]	List of created outcome proficiency sets

Exception:

CreationException, DataConstraintException, InvalidParameters, ServiceException

2.4.15 getBlueprintHiearархies

This method finds all the blueprint nodes using the IdInfo in the input BlueprintNodeInfo and returns the complete BlueprintNodeInfo including the child blueprint nodes.

If the specified blueprint node is not found in the system, no record is returned.

Parameter:

Table 2–106 getBlueprintHiearархies Parameter Summary

Type	Description
BlueprintNodeInfo[]	List of blueprint nodes to retrieve

Return:

Table 2–107 getBlueprintHiearархies Return Summary

Type	Description
BlueprintNodeInfo[]	List of blueprint nodes retrieved

Exception:

InvalidParameters, ServiceException

2.4.16 getContexts

This method finds all the contexts using the IdInfo in the input ContextInfo and returns the complete ContextInfo.

If the specified context is not found in the system, no record is returned.

Parameter:

Table 2–108 getContexts Parameter Summary

Type	Description
ContextInfo[]	List of contexts to retrieve

Return:

Table 2–109 getContexts Return Summary

Type	Description
ContextInfo[]	List of contexts retrieved

Exception:

InvalidParameters, ServiceException

2.4.17 getFrameworks

This method finds all the frameworks using the IdInfo in the input FrameworkInfo and returns the complete FrameworkInfo.

If the specified framework is not found in the system, no record is returned.

Parameter:

Table 2–110 getFrameworks Parameter Summary

Type	Description
FrameworkInfo[]	List of frameworks to retrieve

Return:

Table 2–111 getFrameworks Return Summary

Type	Description
FrameworkInfo[]	List of frameworks retrieved

Exception:

InvalidParameters, ServiceException

2.4.18 getFrameworkItemHiearchies

This method finds all the framework item nodes using the IdInfo in the input FrameworkItemInfo and returns the complete FrameworkItemInfo including child framework items.

If the specified framework item is not found in the system, no record is returned.

Parameter:

Table 2–112 getFrameworkItemHiearchies Parameter Summary

Type	Description
FrameworkItemInfo[]	List of framework items to retrieve

Return:

Table 2–113 getFrameworkItemHiearchies Return Summary

Type	Description
FrameworkItemInfo[]	List of framework items retrieved

Exception:

InvalidParameters, ServiceException

2.4.19 getOutcomeStatements

This method finds all the outcome statements using the IdInfo in the input OutcomeStatementInfo and returns the complete OutcomeStatementInfo.

If the specified outcome statement is not found in the system, no record is returned.

Parameter:

Table 2–114 getOutcomeStatements Parameter Summary

Type	Description
OutcomeStatementInfo[]	List of outcome statements to retrieve

Return:

Table 2–115 getOutcomeStatements Return Summary

Type	Description
OutcomeStatementInfo[]	List of outcome statements retrieved

Exception:

InvalidParameters, ServiceException

2.4.20 getOutcomeStatementDescriptorType

This method finds all the outcome statement descriptor types using the IdInfo in the input OutcomeStatementDescTypeInfo and returns the OutcomeStatementDescTypeInfo.

If the specified outcome statement descriptor type is not found in the system, no record is returned.

Parameter:

Table 2–116 getOutcomeStatementDescriptorType Parameter Summary

Type	Description
OutcomeStatementDescTypeInfo[]	List of outcome statement descriptor types to retrieve

Return:

Table 2–117 getOutcomeStatementDescriptorType Return Summary

Type	Description
OutcomeStatementDescTypeInfo[]	List of outcome statement descriptor types retrieved

Exception:

InvalidParameters, ServiceException

2.4.21 getProficiencySets

This method finds all the outcome proficiency sets using the IdInfo in the input ProficiencySetInfo and returns the complete ProficiencySetInfo.

If the specified outcome proficiency set is not found in the system, no record is returned.

Parameter:

Table 2–118 getProficiencySets Parameter Summary

Type	Description
ProficiencySetInfo[]	List of proficiency sets to retrieve

Return:

Table 2–119 getProficiencySets Return Summary

Type	Description
ProficiencySetInfo[]	List of proficiency sets retrieved

Exception:

InvalidParameters, ServiceException

2.4.22 loadFramework

This method loads the entire framework data. This includes:

- Create the framework and set up the institution and group availability
- Create the blueprint node hiearchy
- Create the proficiency sets
- Create the outcome statement descriptor types
- Create the framework item hiearchy

- Create the outcome statements
- Create the contexts
- If the framework is to be published, it is pubslihed

The restrictions in createFrameworks, createBlueprintHierarchies, createProficiencySets, createOutcomeStatementDescTypes, createFrameworkItemHierarchies, createOutcomeStatements, createContexts, and publishFrameworks are still applicable.

Parameter:

Table 2–120 loadFramework Parameter Summary

Type	Description
FrameworkLoadInfo	Framework to create

Return:

Table 2–121 loadFramework Return Summary

Type	Description
FrameworkLoadInfo	Created framework

Exception:

CreationException, DataConstraintException, InvalidParameters, ServiceException

2.4.23 removeBlueprintHiearchies

This method finds all the blueprint nodes using the IdInfo in the input BlueprintNodeInfo and removes them. Any subordinate nodes are also deleted. If the blueprint node is associated with a framework item, then it is not allowed to be deleted.

Parameter:

Table 2–122 removeBlueprintHiearchies Parameter Summary

Type	Description
BlueprintNodeInfo[]	List of blueprint nodes to remove

Return:

Table 2–123 removeBlueprintHiearchies Return Summary

Type	Description
BlueprintNodeInfo[]	List of removed blueprint nodes

Exception:

DataConstraintException, InvalidParameters, RemoveException, ServiceException

2.4.24 removeContexts

This method finds all the contexts using the IdInfo in the input ContextInfo and removes them. Any subordinate context are also deleted. However, a context cannot be deleted if the context has been adopted by any school.

Parameter:

Table 2–124 removeContexts Parameter Summary

Type	Description
ContextInfo[]	List of contexts to remove

Return:

Table 2–125 removeContexts Return Summary

Type	Description
ContextInfo[]	List of removed contexts

Exception:

DataConstraintException, InvalidParameters, RemoveException, ServiceException

2.4.25 removeFrameworks

This method finds all the frameworks using the IdInfo in the input FrameworkInfo and removes them and all framework-related entities such as outcome proficiency sets, blueprint nodes, framework items, outcome statements and contexts. A framework cannot be removed if it has been adopted by a school.

All restrictions in removeFrameworkItemHierarchies, removeBlueprintHierarchies, removeOutcomeStatementDescTypes, removeProficiencySets, removeOutcomeStatements, and removeContexts are still applicable.

Parameter:

Table 2–126 removeFrameworks Parameter Summary

Type	Description
FrameworkInfo[]	List of frameworks to remove

Return:

Table 2–127 removeFrameworks Return Summary

Type	Description
FrameworkInfo[]	List of removed frameworks

Exception:

DataConstraintException, InvalidParameters, RemoveException, ServiceException

2.4.26 removeFrameworkItemHiearchies

This method finds all the Framework item nodes using the IdInfo in the input FrameworkItemInfo and removes them. All subordinate framework items are also

deleted. A framework item cannot be deleted if it is associated with any learning item, demonstration or confirmation.

Parameter:

Table 2–128 removeFrameworkItemHiearchies Parameter Summary

Type	Description
FrameworkItemInfo[]	List of framework items to remove

Return:

Table 2–129 removeFrameworkItemHiearchies Return Summary

Type	Description
FrameworkItemInfo[]	List of removed framework items

Exception:

DataConstraintException, InvalidParameters, RemoveException, ServiceException

2.4.27 removeOutcomeStatements

This method finds all the outcome statements using the IdInfo in the input OutcomeStatementInfo and removes them. Any associated outcome statement descriptors are also deleted. If the outcome statement is associated with any learning item or assessments (confirmations or demonstrations), it cannot be deleted.

Parameter:

Table 2–130 removeOutcomeStatements Parameter Summary

Type	Description
OutcomeStatementInfo[]	List of outcome statements to remove

Return:

Table 2–131 removeOutcomeStatements Return Summary

Type	Description
OutcomeStatementInfo[]	List of removed outcome statements

Exception:

DataConstraintException, InvalidParameters, RemoveException, ServiceException

2.4.28 removeOutcomeStatementDescriptorType

This method finds all the outcome statements descriptor types using the IdInfo in the input OutcomeStatementDescTypeInfo and removes them. An outcome statement descriptor type cannot be deleted if it is being used in an outcome statement descriptor.

Parameter:

Table 2–132 removeOutcomeStatementDescriptorType Parameter Summary

Type	Description
OutcomeStatementDescTypeInfo[]	List of outcome statement descriptor types to remove

Return:

Table 2–133 removeOutcomeStatementDescriptorType Return Summary

Type	Description
OutcomeStatementDescTypeInfo[]	List of removed outcome statement descriptor types

Exception:

DataConstraintException, InvalidParameters, RemoveException, ServiceException

2.4.29 removeProficiencySets

This method finds all the proficiency sets using the IdInfo in the input ProficiencySetInfo and removes them. Any associated proficiency levels are also deleted. If the proficiency set is associated with a framework item, then it is not allowed to be deleted.

Parameter:

Table 2–134 removeProficiencySets Parameter Summary

Type	Description
ProficiencySetInfo[]	List of outcome proficiency sets to remove

Return:

Table 2–135 removeProficiencySets Return Summary

Type	Description
ProficiencySetInfo[]	List of removed outcome proficiency sets

Exception:

DataConstraintException, InvalidParameters, RemoveException, ServiceException

2.4.30 updateBlueprintHiearchies

This method searches for the blueprint nodes using the IdInfo in the input BlueprintNodeInfo and updates the blueprint nodes with the specified attributes. Note that the parent/child relationship of a blueprint node cannot be updated. In addition, the framework of a blueprint node cannot be updated.

Parameter:

Table 2–136 updateBlueprintHiearchies Parameter Summary

Type	Description
BlueprintNodeInfo[]	List of blueprint nodes to update

Return:

Table 2–137 updateBlueprintHiearchies Return Summary

Type	Description
BlueprintNodeInfo[]	List of updated blueprint nodes

Exception:

DataConstraintException, InvalidParameters, ServiceException, UpdateException

2.4.31 updateContexts

This method searches for the contexts using the IdInfo in the input ContextInfo and updates the contexts with the specified attributes. Note that the framework item cannot be updated.

Parameter:

Table 2–138 updateContexts Parameter Summary

Type	Description
ContextInfo[]	List of contexts to update

Return:

Table 2–139 updateContexts Return Summary

Type	Description
ContextInfo[]	List of updated contexts

Exception:

DataConstraintException, InvalidParameters, ServiceException, UpdateException

2.4.32 updateFrameworks

This method searches for the frameworks using the IdInfo in the input FrameworkInfo and updates the frameworks with the specified attributes.

Parameter:

Table 2–140 updateFrameworks Parameter Summary

Type	Description
FrameworkInfo[]	List of frameworks to update

Return:

Table 2–141 updateFrameworks Return Summary

Type	Description
FrameworkInfo[]	List of updated frameworks

Exception:

DataConstraintException, InvalidParameters, ServiceException, UpdateException

2.4.33 updateFrameworkItemHiearchies

This method searches for the nodes using the IdInfo in the input FrameworkItemInfo and updates the objects with the specified attributes. The framework, parent framework item, and blueprint node cannot be updated. When deleting a framework item, all subordinate framework items are also deleted. However, a framework item cannot be deleted if it is associated with any learning item.

Parameter:

Table 2–142 updateFrameworkItemHiearchies Parameter Summary

Type	Description
FrameworkItemInfo[]	List of framework items to update

Return:

Table 2–143 updateFrameworkItemHiearchies Return Summary

Type	Description
FrameworkItemInfo[]	List of updated framework items

Exception:

DataConstraintException, InvalidParameters, ServiceException, UpdateException

2.4.34 updateProficiencySets

This method searches for the outcome proficiency sets using the IdInfo in the input ProficiencySetInfo and updates the outcome proficiency sets with the specified attributes.

Parameter:

Table 2–144 updateProficiencySets Parameter Summary

Type	Description
ProficiencySetInfo[]	List of outcome proficiency sets to update

Return:

Table 2–145 updateProficiencySets Return Summary

Type	Description
ProficiencySetInfo[]	List of updated outcome proficiency sets

Exception:

DataConstraintException, InvalidParameters, ServiceException, UpdateException

2.4.35 updateOutcomeStatements

This method searches for the statements using the IdInfo in the input OutcomeStatementInfo and updates the outcome statements with the specified attributes. The framework item cannot be updated.

Parameter:

Table 2–146 updateOutcomeStatements Parameter Summary

Type	Description
OutcomeStatementInfo[]	List of outcome statements to update

Return:

Table 2–147 updateOutcomeStatements Return Summary

Type	Description
OutcomeStatementInfo[]	List of updated outcome statements

Exception:

DataConstraintException, InvalidParameters, ServiceException, UpdateException

2.4.36 updateOutcomeStatementDescriptorType

This method searches for the outcome statement descriptor types using the IdInfo in the input OutcomeStatementDescTypeInfo and updates the outcome statement descriptor types. The framework cannot be updated.

Parameter:

Table 2–148 updateOutcomeStatementDescriptorType Parameter Summary

Type	Description
OutcomeStatementDescTypeInfo[]	List of outcome statement descriptor types to update

Return:

Table 2–149 updateOutcomeStatementDescriptorType Return Summary

Type	Description
OutcomeStatementDescTypeInfo[]	List of updated outcome statement descriptor types

Exception:

DataConstraintException, InvalidParameters, ServiceException, UpdateException

2.5 DataLoadingCurriculumAdoptionService

The DataLoadingCurriculumAdoptionService is used to adopt frameworks and contexts by schools.

2.5.1 adoptContexts

This method adopts contexts as part of the larger curriculum framework adoptions. The framework with which the context is associated must be adopted by the school.

Parameter:

Table 2–150 adoptContexts Parameter Summary

Type	Description
ContextAdoptionInfo[]	List of contexts to adopt

Return:

Table 2–151 adoptContexts Return Summary

Type	Description
ContextAdoptionInfo[]	List of adopted contexts

Exception:

CreationException, DataConstraintException, InvalidParameters, ServiceException

2.5.2 adoptFrameworks

This method adopts frameworks by schools for a school calendar. Before the framework can be adopted, it must be published. A school can only adopt a framework once and can only do so if the framework is made available to the school for adoption. Note that the specified calendar must belong to the school adopting the framework.

Parameter:

Table 2–152 adoptFrameworks Parameter Summary

Type	Description
FrameworkAdoptionInfo[]	List of frameworks to adopt

Return:

Table 2–153 adoptFrameworks Return Summary

Type	Description
FrameworkAdoptionInfo[]	List of adopted frameworks

Exception:

CreationException, DataConstraintException, InvalidParameters, ServiceException

2.5.3 getContextAdoptions

This method finds all the context adoptions using the IdInfo in the input ContextAdoptionInfo and returns the complete ContextAdoptionInfo.

If the specified context adoption is not found in the system, no record is returned.

Parameter:

Table 2–154 getContextAdoptions Parameter Summary

Type	Description
ContextAdoptionInfo[]	List of context adoptions to retrieve

Return:

Table 2–155 getContextAdoptions Return Summary

Type	Description
ContextAdoptionInfo[]	List of context adoptions retrieved

Exception:

InvalidParameters, ServiceException

2.5.4 getFrameworkAdoptions

This method finds all the framework adoptions using the IdInfo in the input FrameworkAdoptionInfo and returns the complete FrameworkAdoptionInfo.

If the specified framework adoption is not found in the system, no record is returned.

Parameter:

Table 2–156 getFrameworkAdoptions Parameter Summary

Type	Description
FrameworkAdoptionInfo[]	List of framework adoptions to retrieve

Return:

Table 2–157 getFrameworkAdoptions Return Summary

Type	Description
FrameworkAdoptionInfo[]	List of framework adoptions retrieved

Exception:

InvalidParameters, ServiceException

2.5.5 removeContextAdoptions

This method finds all the context adoption using the IdInfo in the input ContextAdoptionInfo and removes them. If a context adoption is removed, all subordinate context adoptions are also removed. Note that a context adoption cannot

be removed if the adopted context is used in any learning items, confirmations or demonstrations.

Parameter:

Table 2–158 removeContextAdoptions Parameter Summary

Type	Description
ContextAdoptionInfo[]	List of context adoptions to remove

Return:

Table 2–159 removeContextAdoptions Return Summary

Type	Description
ContextAdoptionInfo[]	List of removed context adoptions

Exception:

DataConstraintException, InvalidParameters, RemoveException, ServiceException

2.5.6 removeFrameworkAdoptions

This method finds all the framework adoptions using the IdInfo in the input FrameworkAdoptionInfo and removes them. A framework adoption can only be removed if there are no offerings created for the adopted framework.

Parameter:

Table 2–160 removeFrameworkAdoptions Parameter Summary

Type	Description
FrameworkAdoptionInfo[]	List of framework adoptions to remove

Return:

Table 2–161 removeFrameworkAdoptions Return Summary

Type	Description
FrameworkAdoptionInfo[]	List of removed framework adoptions

Exception:

DataConstraintException, InvalidParameters, RemoveException, ServiceException

2.6 DataLoadingGradeService

The DataLoadingGradeService is used to load graded proficiency set information. Adoption of department graded proficiency sets is also done through this service.

2.6.1 adoptProficiencySets

This method adopts a department graded proficiency set for use in a school. A school can only adopt a department graded proficiency set once.

Parameter:

Table 2–162 adoptProficiencySets Parameter Summary

Type	Description
AdoptProficiencySetInfo[]	List of graded proficiency sets to adopt

Return:

Table 2–163 adoptProficiencySets Return Summary

Type	Description
AdoptProficiencySetInfo[]	List of adopted graded proficiency sets

Exception:

CreationException, DataConstraintException, InvalidParameters, ServiceException

2.6.2 createGradedProficiencySets

This method creates new graded proficiency sets. A graded proficiency set must have a unique name within the school and have at least one proficiency level. Within a graded proficiency set, the proficiency levels must have unique names and codes and cannot have proficiency sub-levels. Collectively, the proficiency levels must cover the entire numeric range 0-100 and there must be no overlapping numeric ranges between the proficiency levels.

Parameter:

Table 2–164 createGradedProficiencySets Parameter Summary

Type	Description
ProficiencySetInfo[]	List of graded proficiency sets to retrieve

Return:

Table 2–165 createGradedProficiencySets Return Summary

Type	Description
ProficiencySetInfo[]	List of graded proficiency sets retrieved

Exception:

CreationException, DataConstraintException, InvalidParameters, ServiceException

2.6.3 createOrUpdateGradedProficiencySets

This method searches for the graded proficiency sets using the IdInfo in the input ProficiencySetInfo and if a graded proficiency set is found, it updates it; Otherwise, it creates it.

The restrictions in createGradedProficiencySets and updateGradedProficiencySets are applicable here.

Parameter:

Table 2–166 createOrUpdateGradedProficiencySets Parameter Summary

Type	Description
ProficiencySetInfo[]	List of graded proficiency sets to create or update

Return:

Table 2–167 createOrUpdateGradedProficiencySets Return Summary

Type	Description
ProficiencySetInfo[]	List of graded proficiency sets created or updated

Exception:

CreationException, DataConstraintException, InvalidParameters, ServiceException, UpdateException

2.6.4 getGradedProficiencySets

This method finds all the proficiency sets using the IdInfo in the input ProficiencySetInfo and returns the complete ProficiencySetInfo including the proficiency levels.

If the specified graded proficiency set is not found in the system, no record is returned

Parameter:

Table 2–168 getGradedProficiencySets Parameter Summary

Type	Description
ProficiencySetInfo[]	List of graded proficiency sets to retrieve

Return:

Table 2–169 getGradedProficiencySets Return Summary

Type	Description
ProficiencySetInfo[]	List of graded proficiency sets retrieved

Exception:

InvalidParameters, ServiceException

2.6.5 updateGradedProficiencySets

This method searches for the proficiency sets using the IdInfo in the input ProficiencySetInfo and updates the objects with the specified attributes. Note that the proficiency levels referenced in learning item assessments cannot be deleted.

Parameter:

Table 2–170 updateGradedProficiencySets Parameter Summary

Type	Description
ProficiencySetInfo[]	List of graded proficiency sets to update

Return:

Table 2–171 updateGradedProficiencySets Return Summary

Type	Description
ProficiencySetInfo[]	List of updated graded proficiency sets

Exception:

DataConstraintException, InvalidParameters, ServiceException, UpdateException

2.6.6 removeGradedProficiencySets

This method finds all the graded proficiency sets using the IdInfo in the input ProficiencySetInfo and removes them. The proficiency levels within the graded proficiency sets are also deleted.

A department graded proficiency set cannot be deleted if a school has adopted it. Furthermore, a graded proficiency set cannot be deleted if it is referenced in a learning item.

Parameter:

Table 2–172 removeGradedProficiencySets Parameter Summary

Type	Description
ContextAdoptionInfo[]	List of context adoptions to remove

Return:

Table 2–173 removeGradedProficiencySets Return Summary

Type	Description
ContextAdoptionInfo[]	List of removed context adoptions

Exception:

DataConstraintException, InvalidParameters, RemoveException, ServiceException

2.7 DataLoadingLookupValueService

The DataLoadingLookupValueService is used to update list of values information.

2.7.1 getValueTypes

This method finds all the value types using the IdInfo in the input LookupTypeInfo and returns the complete LookupTypeInfo.

If the specified list of values is not found in the system, no record is returned.

Parameter:

Table 2–174 getValueTypes Parameter Summary

Type	Description
LookupTypeInfo[]	List of value types to retrieve

Return:

Table 2–175 getValueTypes Return Summary

Type	Description
LookupTypeInfo[]	List of value types retrieved

Exception:

InvalidParameters, ServiceException

2.7.2 updateValueTypes

This method searches for the value types using the IdInfo in the input LookupTypeInfo and updates the objects with the specified attributes. Within the list of values, the value names must be unique.

Note that for values provisioned by OSL, only the external id, external system id, meaning, and description can be updated. Some of these OSL-provisioned values cannot be removed or end-dated.

Parameter:

Table 2–176 updateValueTypes Parameter Summary

Type	Description
LookupTypeInfo[]	List of value types to update

Return:

Table 2–177 updateValueTypes Return Summary

Type	Description
LookupTypeInfo[]	List of updated value types

Exception:

DataConstraintException, InvalidParameters, ServiceException, UpdateException

2.8 DataLoadingProfileService

The DataLoadingProfileService is used to load preference information.

2.8.1 createProfileOptions

This method creates new profileOptionValues. A profileOptionValue must have a PersonId and profileCode in the input ProfileOptionInfo.

Parameter:

Table 2–178 createProfileOption Parameter Summary

Type	Description
ProfileOptionInfo[]	List of profile options to create

Return:

Table 2–179 createProfileOption Return Summary

Type	Description
ProfileOptionInfo[]	List of created profile options

Exception:

CreationException, DataConstraintException, InvalidParameters, ServiceException

2.8.2 getProfileOptions

This method finds the profileOptionValue using the PersonId and profileCode in the input ProfileOptionInfo. If the PersonId is not specified or no profileOptionValue can be located for the specified PersonId and profileCode, the site-wide profileOptionValue for the specified profileCode is returned.

Parameter:

Table 2–180 getProfileOptions Parameter Summary

Type	Description
ProfileOptionInfo[]	List of profile options to retrieve

Return:

Table 2–181 getProfileOptions Return Summary

Type	Description
ProfileOptionInfo[]	List of profile options retrieved

Exception:

InvalidParameters, ServiceException

2.8.3 updateProfileOptions

This method searches for profileOptionValues using the PersonId and profileCode in the input ProfileOptionInfo and updates the profileOptionValue with the specified

attribute. If the PersonId is not specified, the site-wide profileOptionValue for the specified profileCode is updated.

Parameter:

Table 2–182 updateProfileOptions Parameter Summary

Type	Description
ProfileOptionInfo[]	List of profile options to update

Return:

Table 2–183 updateProfileOptions Return Summary

Type	Description
ProfileOptionInfo[]	List of updated profile options

Exception:

UpdateException, DataConstraintException, InvalidParameters, ServiceException

2.8.4 removeProfileOptions

This method finds profileOptionValue using the PersonId and profileCode in the input ProfileOptionInfo and removes them.

Parameter:

Table 2–184 removeProfileOptions Parameter Summary

Type	Description
ProfileOptionInfo[]	List of profile options to remove

Return:

Table 2–185 removeProfileOptions Return Summary

Type	Description
ProfileOptionInfo[]	List of removed profile options

Exception:

DataConstraintException, InvalidParameters, RemoveException, ServiceException

Part II

OSL Content Integration

This part contains chapters about content integration interfaces, services and descriptions.

3

Overview

This chapter provides an overview of the OSL content integration interfaces.

OSL uses an External Content Management System (ECMS) for the rich data functionality used in instructions, overviews, journal details, and so on. This functionality allows an OSL user to insert hyperlinks and images, attach files and audios to a rich data. In addition, exported learning items are stored in the ECMS.

The following block diagram briefly explains the architecture of OSL interactions with ECMS.

Figure 3–1 OSL Interactions with ECMS

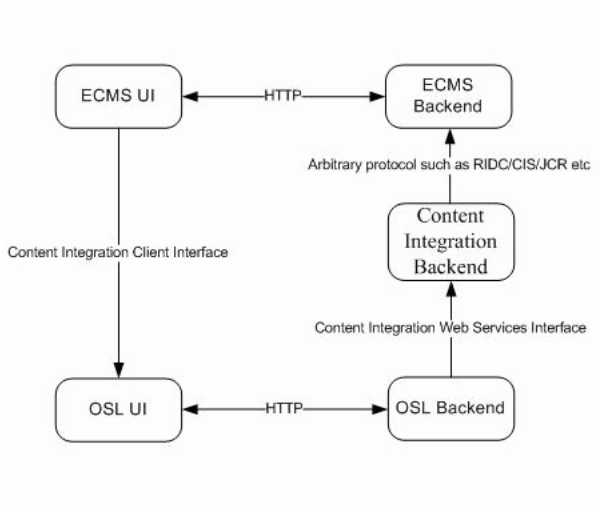

Figure 3–1 shows two interfaces of OSL Content Integration, one on the client-side (UI level integration) and the other on the back end (server-side integration).

These two interfaces are:

- Content Integration Client Interface (CICI)
- Content Integration Web Services Interface (CIWSI)

Table 3–1 list the terms used in OSL Content Integration.

Table 3–1 Definition of Terms

Term	Definition
Content Integration Backend	Implementation of CIWSI. OSL ships with a default implementation for Oracle Universal Content Manager (UCM), which uses RIDC to communicate with UCM back end.
ECMS UI	User interface rendered by the ECMS. There are two user interfaces that ECMS should render: ■ ECMS Search Results UI ■ ECMS Update Metadata UI OSL spawns these user interfaces as part of OSL functionality.
ECMS Update Metadata UI	User interface that is rendered by the ECMS for updating the metadata after publishing an exported learning item to ECMS.
ECMS Search Results UI	User interface that is rendered by the ECMS to list General Content.
Exported Learning Item	A package created after exporting learning item data. This package is in a ZIP format.
External Content Management System (ECMS)	Any Content Management System which supports versioning.
General Content	Content stored within ECMS and referenced within OSL and other applications such as portal or collaborative services. These are displayed in OSL as inserted hyperlinks and images in the rich data.
OSL Backend and ECMS Backend	The respective servers.
OSL Content	File attachment and audio that a user directly attaches to an OSL object. Such content is typically stored in the ECMS but partitioned separately from general content.
OSL Content User	Default back end implementation uses this UCM user to access OSL Content. This user should have RWD access to the Temporary and to the Permanent Accounts and OSL Storage Security Group.
OSL Context	Context in relation to the use of general content within OSL. For example, the class id, class name, and so on.
OSL Metadata User	The default back end implementation uses this UCM user to update the medata of General Content when associating context. This user should have RW access to all the General Content.
OSL Storage Security Group	The default back end implementation uses this UCM Security Group to store all the content that is checked in from OSL.
OSL UI	Teacher/Parent/Student Learning Tool user interface.
Permanent Space	Permanent space in the ECMS partitioned separately from General Content to store saved OSL Content.
Permanent Storage Account	The default back end implementation uses this UCM account to store permanently all OSL Content.

Table 3–1 (Cont.) Definition of Terms

Term	Definition
Rich Data	An OSL component that abstracts HTML content with inserted content references and images, file attachments, and audio recordings. This component is used for instructions, teacher overview, student overview, journal, submission, discussion, and observation details.
Search Filter	General Content listing filter. This filter could be used to filter out only the images, learning items, or any content type.

3.1 Content Integration Client Interface

The Content Integration Client Interface abstracts UI (browser) level interaction between the OSL UI and the ECMS UI. This service provides an interface for:

- Inserting General Content references to Rich Data
- Inserting an exported learning item into a lesson plan

The following URL abstracts the interface:

```
http://<HOST:PORT>/<APP-ROOT>/common/default/richdata/script.jsp
x?cid=<CON-ID>&cvid=<CON-VER-ID>&curl=<CON-URL>
```

It is expected that the ECMS UI should invoke this URL. Please refer to the *Oracle Student Learning Installation and Deployment Guide* for further details.

Table 3–2 List of Attributes

Name	Mandatory	Description
HOST	Yes	Host of OSL Web (LTWeb) deployment.
PORT	Yes	Port of OSL Web (LTWeb) deployment. For LT that is SSO enabled, the port number should be the SSO port number.
APP-ROOT	Yes	Application root of OSL Web (LTWeb) deployment.
CON-ID	Yes	Content Identifier of content to be inserted to rich data or lesson plan.
CON-VER-ID	No	Content Version Identifier of content to be inserted to rich data or lesson plan.
CON-URL	Yes	Content URL.

3.2 Content Integration Web Services Interface

The Content Integration Web Services Interface (CIWSI) abstracts all ECMS interactions with OSL at the back end.

It comprises of three service interfaces:

- General Content Service
- OSL Content Service

- Publish Learning Item Service

For more related information about these service interfaces, see Chapter 4, "Web Services".

3.3 Security

CIWSI should be secured services and should use the following Security Policy:

- `oracle/wss_username_token_service_policy`

  This security policy requires username and password tokens in the SOAP Header. It is expected that OSL communicates with CIWSI using a single dedicated user. For related information about user configuration, see the *Oracle Student Learning Installation and Deployment Guide*.

3.4 WSDL and XSD files

The WSDL and the associated XSD files are available in the product install footprint. Navigate to `<OSL_INSTALLATION_DIR>/ContentIntegration/wsdl` to locate these files.

4

Web Services

This chapter describes the available web services.

4.1 GeneralContentService

The General Content Service abstracts OSL interactions with the ECMS in relation to General Content.

This service provides interfaces for:

- Associating and disassociating OSL context with General Content items
- Listing General Content items based on supplied OSL context

4.1.1 associateContent

OSL calls the associateContent method whenever a General Content item is associated with an OSL object's rich data. This can occur when a user creates or edits rich data, or shares or copies a learning item.

A ContentsAssociationInfo document (containing the OSL Context, OSL user information and a list of Content Ids) is passed as input to this method.

This method can be used to associate any or all aspects of the supplied context with the specified content items.

If this service throws any exception, OSL will not roll back its transaction but logs the error and continues with the business workflow.

Parameter:

Table 4–1 associateContent Parameter Summary

Type	Description
ContentsAssociationInfo	Information regarding OSL Context, Content Ids of General Content references, and OSL user's information

Exception:

ContentException shall be thrown if any problems or errors occur when communicating with ECMS.

InvalidParameters shall be thrown if any input parameter is invalid.

ServiceException shall be thrown if any general errors occur.

Behavior of the default implementation:

The default back end implementation maintains a map of OSL Context to Oracle UCM Metadata in the `osl_configuration.properties` file and updates the Metadata with the values from OSL Context using a delimiter. If a specific OSL Context is not mapped, then it is ignored. The update itself is done using the OSL Metadata User. For related information, see the *Oracle Student Learning Installation and Deployment Guide*.

4.1.2 disassociateContent

OSL calls the disassociateContent method whenever a General Content item is disassociated from an OSL object's rich data. This can occur when a user creates or edits rich data, or deletes a learning item.

A ContentsAssociationInfo document (containing the OSL Context, OSL user information and a list of Content Ids) is passed as input to this method.

This method can be used to disassociate any or all aspects of the supplied context from the specified content items.

If this service throws any exception, OSL will not roll back its transaction but logs the error and continues with the business workflow.

Parameter:

Table 4–2 disassociateContent Parameter Summary

Type	Description
ContentsAssociationInfo	Information regarding OSL Context, Content Ids of General Content references, and OSL user's information

Exception:

ContentException shall be thrown if any problems or errors occur when communicating with ECMS.

InvalidParameters shall be thrown if any input parameter is invalid.

ServiceException shall be thrown if any general errors occur.

Behaviour of the default implementation:

The default back end implementation does not implement this service.

4.1.3 getSearchURL

OSL calls the getSearchURL method to retrieve the URL that spawns the ECMS Search Results UI based on the supplied OSL Context and search filters.

A ContentSearchCriteriaInfo document (containing the OSL Context and Search Filters) is passed as input to this method.

OSL expects this method to return the HTTP URL.

If this service throws any exception, OSL reports the error to the OSL user.

Parameter:

Table 4–3 getSearchURL Parameter Summary

Type	Description
ContentSearchCriteriaInfo	Information regarding OSL Context and Search Filter

Return:

Table 4–4 getSearchURL Return Summary

Type	Description
String	HTTP URL of a resource, which lists all General Content associated with the input OSL Context and filtered based on the Search Filter

Exception:

ContentException shall be thrown if any problems or errors occur when communicating with ECMS.

InvalidParameters shall be thrown if any input parameter is invalid.

ServiceException shall be thrown if any general errors occur.

Behaviour of the default implementation:

The default back end implementation maintains a map of OSL Context to Oracle UCM Metadata based on the Search substring in `osl_configuration.properties` file and returns a `UCM GET_SEARCH_RESULTS` service URL using the Search substring as filter. If a specific OSL Context is not mapped, then it is ignored. For more related information, see the *Oracle Student Learning Installation and Deployment Guide*.

4.2 OSLContentService

The OSL Content Service abstracts interaction with the ECMS for managing the OSL Content.

This service provides interfaces for:

- Accessing the OSL Content
- Saving and deleting OSL Content from the ECMS.

4.2.1 deleteContent

OSL calls the deleteContent method to delete OSL Content items from the Permanent Space. This occurs when the user deletes an attachment or audio that was added during the edit session.

A ContentIdsInfo document (containing OSL user information and a list of Content Ids of OSL Content attachments to be deleted) is passed as input to this method.

If this service throws any exception, OSL will not roll back its transaction but logs the errors occur and continues with the business workflow.

Parameter:

Table 4–5 duplicateContent Parameter Summary

Type	Description
ContentIdsInfo	OSL user's information and a list of Content Ids of the OSL Content attachment to be removed

Exception:

ContentException shall be thrown if any problems or errors occur when communicating with ECMS.

InvalidParameters shall be thrown if any input parameter is invalid.

ServiceException shall be thrown if any general errors occur.

Behaviour of the default implementation:

The default back end implementation deletes the content using the OSL Content User.

4.2.2 getOSLContent

OSL calls the getOSLContent method to retrieve OSL content items. This occurs when the user attempts to open a file attachment or play recorded audio.

A ContentIdsInfo document (containing OSL user information and a list of Content Ids of OSL Content attachments) is passed as input to this method.

OSL expects this service to retrieve OSL Content data and return it.

If this service throws any exception, OSL reports the error to the OSL user.

Parameter:

Table 4–6 getOSLContent Parameter Summary

Type	Description
ContentIdsInfo	OSL user's information and a list of Content Ids of OSLContent attachments

Return:

Table 4–7 getOSLContent Return Summary

Type	Description
Array of ContentInfo	An Array of ContentInformation regarding the OSLContent attachment

4.2.3 saveContent

OSL calls the saveContent method to save attached files and audio to the ECMS.

A ContentsUploadInfo document (containing the OSL user information and a list of Upload Information for OSL Content attachments) is passed as input to this method.

OSL expects this method to upload the OSL Content item into the Permanent Space in the ECMS and return the details of the uploaded OSL Content.

If this service throws any exception, OSL reports the error to the OSL user.

Parameter:

Table 4–8 saveContent Parameter Summary

Type	Description
ContentsUploadInfo	ContentsUploadInfo OSL user's information and a list of UploadInformation for OSL Content attachments

Return:

Table 4–9 saveContent Return Summary

Type	Description
Array of ContentInfo	An Array of ContentInformation regarding the Uploaded OSL Content attachment.

Exception:

ContentException shall be thrown if any problems or errors occur when communicating with ECMS.

InvalidParameters shall be thrown if any input parameter is invalid.

ServiceException shall be thrown if any general errors occur.

Behaviour of the default implementation:

The default back end implementation uploads the content to the OSL Storage Security Group in UCM, using the Storage Account. This operation is done as the OSL Content User.

4.3 DeprecatedServices

The services listed below which are operations that involve temporary storage and duplication of content has been deprecated.

4.3.1 deleteTempContent

The methods listed below are operations that involve temporary storage and duplication of content. These methods have been deprecated since they are no longer called by OSL.

4.3.2 duplicateContent

4.3.3 getTempContent

4.3.4 saveTempContent

4.3.5 uploadTempContent

4.4 PublishLearningItemService

The Publish Learning Item Service abstracts interaction with the ECMS for importing and exporting learning items.

This service provides interfaces for:

- Accessing exported learning items
- Exporting a learning item to the ECMS
- Updating an exported learning item based on OSL Context

4.4.1 getMetadataUpdatePage

OSL calls the getMetadataUpdatePage method to retrieve the URL that spawns the ECMS Update Metadata UI for the recently exported learning item.

A ContentIdInfo document (containing Content identity information of the recently exported learning item) is passed as input to this method.

OSL expects this service method to return the HTTP URL of the exported learning item, which allows editing of the Metadata of the published Learning Item.

If this service throws any exception, OSL reports the error to the OSL user.

Parameter:

Table 4–10 getMetadataUpdate Parameter Summary

Type	Description
ContentIdInfo	Content identity information of the recently exported Learning Item

Return:

Table 4–11 getMetadataUpdate Return Summary

Type	Description
String	HTTP URL of a resource, which allows editing of the Metadata of the exported Learning Item

Exception:

ContentException shall be thrown if any problems or errors occur when communicating with ECMS.

InvalidParameters shall be thrown if any input parameter is invalid.

ServiceException shall be thrown if any general errors occur.

Behaviour of the default implementation:

The default back end implementation will show the Update Content Item UI for the exported learning Item. All the metadata that can be updated are displayed.

4.4.2 getPublishedLIContent

OSL calls the getPublishedLIContent method to retrieve the exported learning item from the ECMS to insert into a lesson plan.

A ContentIdsInfo document (containing the OSL user information and a list of the Content Ids of the exported learning items) is passed as input to this method.

OSL expects this method to retrieve exported learning item data and return it.

If this service throws any exception, OSL reports the error to the OSL user.

Parameter:

Table 4–12 getPublishedLIContent Parameter Summary

Type	Description
ContentIdInfo	Content identity information of the recently exported Learning Item

Return:

Table 4–13 getPublishedLIContent Return Summary

Type	Description
String	HTTP URL of a resource, which allows editing of the Metadata of the exported Learning Item

Exception:

ContentException shall be thrown if any problems or errors occur when communicating with ECMS.

InvalidParameters shall be thrown if any input parameter is invalid.

ServiceException shall be thrown if any general errors occur.

Behaviour of the default implementation:

The default back end implementation retrieves the exported Learning Item content as the OSL user. For more related information, see the *Oracle Student Learning Installation and Deployment Guide*.

4.4.3 publishLearningItem

OSL calls the publishLearningItem method when exporting learning items to the ECMS as General Content.

A LearningItemPublishInfo document (containing the OSL Context, OSL user's information, and a list of Upload Information for the exported Learning Items) is passed as input to this method.

OSL expects this method to upload the exported learning item to the ECMS, associate any or all of the supplied OSL Context with the uploaded content, and return back the Content Ids.

If this service throws any exception, OSL reports the error to the OSL user.

Parameter:

Table 4–14 publishLearningItem Parameter Summary

Type	Description
LearningItemPublishInfo	OSL Context, OSL user's information, and a list of Upload Information for the exported Learning Items

Return:

Table 4–15 publishLearningItem Return Summary

Type	Description
ContentIdsInfo	Content Identity information regarding the uploaded exported Learning Items

Exception:

ContentException shall be thrown if any problems or errors occur when communicating with ECMS.

InvalidParameters shall be thrown if any input parameter is invalid.

ServiceException shall be thrown if any general errors occur.

Behaviour of the default implementation:

The default back end implementation uploads the exported Learning Item content to the Public Security Group and Account in UCM, as the OSL user. Profile can also be set for the exported learning item. Account, Security Group and Profile are configurable.

5

Plugging in an Alternate Implementation

This chapter describes the process of how to plugin an alternate implementation of the OSL Content interfaces.

5.1 Plugging in an Alternate Backend Implementation

If required, an alternative implementation can be plugged in to OSL. This involves the following steps:

- Implementing CIWSI as expected by OSL and deploying it
- Configuring OSL to communicate with the alternate implementation
- Configuring security to access the alternate implementation

After configuration, OSL must be redeployed.

5.1.1 Configuring OSL to Connect to Alternate Backend Implementation

After implementing and deploying the alternate back end implementation, set the following properties in the OSL configuration file:

- osl.lt.service.content.contentProxytype: must be set to **WS**
- osl.lt.service.content.wsProxyGeneralContentServiceURL: service end point of the General Content Service Implementation
- osl.lt.service.content.wsProxyOSLContentServiceURL: service end point of the OSL Content Service Implementation
- osl.lt.service.content.wsProxyPublishServiceURL: service end point of the Publish Service Implementation

5.1.2 Configuring Security

OSL expects the alternate back end implementation to meet the security requirement as explained earlier. For related information, see the *Oracle Student Learning Installation and Deployment Guide*.

5.2 Plugging in with an Alternate ECMS UI

It is expected that the ECMS UI is customized according to the customer's specific needs. To help integrate the customized ECMS UI with OSL, a sample Reference Implementation for Oracle Universal Content Management (Oracle UCM) is provided with OSL. For further details, see the *Oracle Student Learning Installation and*

Deployment Guide. You can also refer to the Readme.txt provided in the Reference Implementation package.

A

Data Loading Service Objects

This appendix contains Data Loading service object references.

A.1 AddressActionInfo

AddressActionInfo captures the action to perform on an address and the address details relevant to the action.

Table A–1 list the attributes of addressActionInfo object.

Table A–1 AddressActionInfo Attributes

Name	Type	Description
action	action	Indicates the action to perform on the address. Example: A *Create* Action indicates that a new address would be created.
address	addressInfo	Address details.

Table A–2 lists the action type.

Table A–2 action

Name	Description
Create	Create the object.
CreateOrUpdate	Create or update the object.
Delete	Delete the object.
Update	Update the object.

A.2 AddressActionInfoArray

AddressActionInfoArray contains a list of AddressActionInfo objects.

Table A–3 list the attributes of addressActionInfoArray object.

Table A–3 AddressActionInfoArray Attributes

Name	Type	Description
item	addressActionInfo	The action to perform on an address and the address details relevant to the action.

A.3 AddressInfo

The address details are captured in the AddressInfo. An address should at a minimum, contain the usage type, street name, state, post code, and country code.

Table A–4 list the attributes of addressInfo object.

Table A–4 AddressInfo Attributes

Name	Type	Description
addressLine	string	The full address.
apartmentNumber	string	The apartment number.
apartmentNumberPrefix	string	Apartment prefix. Example: Level, Floor, and so on.
apartmentNumberSuffix	string	Apartment number suffix. Example: A
apartmentType	string	Type of apartment. Example: Suite
city	string	City.
countryCode	string	Country code. This attribute is mandatory for *createInstitutions* operation.
endDate	dateTime	Effective end date of the address.
id	idInfo	Identifier object that uniquely identifies the address.
postCode	string	Postal code of the address. This attribute is mandatory for *createInstitutions* operation.
startDate	dateTime	Effective start date of the address.
state	string	State. This attribute is mandatory for *createInstitutions* operation.
streetName	string	Name of Street. This attribute is mandatory for *createInstitutions* operation.
streetNumber	string	Street Number.
streetPrefix	string	Street Prefix. Example: Lot, Unit, POBox, and so on.
streetSuffix	string	Street suffix. Example: SE
streetType	string	Type of Street. Example: Lane, Road, and so on.
suburb	string	Suburb.
usageType	string	Indicates the functional use of the location. Example: Primary address, home, business. This attribute is mandatory for *createInstitutions* operation.

Table A–5 contactType

Name	Description
EMAIL	Email address.
PHONE	Phone number.

A.4 AdoptCalendarInfo

AdoptCalendarInfo captures the adoption of the department academic calendar by the school.

Table A–6 list the attributes of adoptCalendarInfo object.

Table A–6 AdoptCalendarInfo Attributes

Name	Type	Description
adoptedCalendarId	idInfo	Identifier object that uniquely identifies the adopted calendar.
adoptingSchoolId	idInfo	Identifier object that uniquely identifies the school adopting the department academic calendar. This a mandatory attribute for *adoptDepartmentCalendars* operation.
originalCalendarId	idInfo	Identifier object that uniquely identifies the department academic calendar to be adopted. This attribute is mandatory for *adoptDepartmentCalendars* operation.

A.5 BlueprintNodeActionInfo

BlueprintNodeActionInfo captures the action to perform on a blueprint node and the blueprint node details relevant to the action.

Table A–7 list the attributes of blueprintNodeActionInfo object.

Table A–7 BlueprintNodeActionInfo Attributes

Name	Type	Description
action	action	Indicates the action to perform on a blueprint node. Example: A *Create* Action indicates that a blueprint node would be created. This attribute is mandatory.
blueprintNode	blueprintNodeInfo	Blueprint node details. This attribute is mandatory.

Table A–8 lists the action type.

Table A–8 action

Name	Description
Create	Create the object.
CreateOrUpdate	Create or update the object.
Delete	Delete the object.
Update	Update the object.

A.6 BlueprintNodeActionInfoArray

BlueprintNodeActionInfoArray contains a list of blueprintNodeActionInfo objects.

Table A–9 list the attributes of BlueprintNodeActionInfoArray object.

Table A–9 BlueprintNodeActionInfoArray Attributes

Name	Type	Description
item	blueprintNodeActionInfo	The action to perform on a blueprint node and the blueprint node details relevant to the action.

A.7 BlueprintNodeInfo

Blueprint nodes are organized in a hierarchical structure. Each blueprint node can have one or more blueprint nodes under it. The details of each blueprint node are captured in a blueprintNodeInfo.

Table A–10 list the attributes of blueprintNodeInfo object.

Table A–10 BlueprintNodeInfo Attributes

Name	Type	Description
childrenLists	blueprintNodeInfo	List of blueprint nodes. This attribute is only populated in the response of the *createBlueprintNodes*, *getBlueprintNodes*, and *updateBlueprintNodes* operations.
childrenActions	blueprintNodeActionInfoArray	A list of actions to perform on the blueprint node. This is used in the *createBlueprintNodes*, *updateBlueprintNodes* operations when creating, updating, or deleting the blueprint nodes.
description	string	Blueprint node description.
endDate	dateTime	Effective end date of blueprint node.
frameworkId	idInfo	Identifier object that uniquely identifies a framework. This attribute is mandatory for *createBlueprintNodes* operation.

Table A–10 (Cont.) BlueprintNodeInfo Attributes

Name	Type	Description
id	idInfo	Identifier object that uniquely identifies a blueprint node.
name	string	Blueprint node name. This attribute is mandatory for *createBlueprintNodes* operation.
parentId	idInfo	Identifier object that uniquely identifies the parent blueprint node. This attribute is only set in the response of *createBlueprintNodes*, *getBlueprintNodes*, and *updateBlueprintNodes* operations.
startDate	dateTime	Effective start date of blueprint node. This attribute is mandatory for *createBlueprintNodes* operation.

A.8 CalendarInfo

Typically, department and school would have academic calendars corresponding to the academic years. Under an academic calendar, there can be sub-calendars to represent shorter academic periods such as semesters.

The details of each of these academic calendars and academic sub-calendars are captured in a CalendarInfo.

Table A–11 list the attributes of calendarInfo object.

Table A–11 CalendarInfo Attributes

Name	Type	Description
endDate	dateTime	End date of the academic calendar. This attribute is mandatory for *createCalendars* operation.
id	idInfo	Identifier object that uniquely identifies the calendar.
institutionId	idInfo	Identifier object that uniquely identifies the department or school. This attribute is mandatory for *createCalendars* operation.
name	string	Calendar name. This attribute is mandatory for *createCalendars* operation.
parentId	idInfo	Identifier object that uniquely identifies the parent calendar.

Table A–11 (Cont.) CalendarInfo Attributes

Name	Type	Description
startDate	dateTime	Start date of the academic calendar. This attribute is mandatory for *createCalendars* operation.

A.9 ClassInfo

After an offering is created, multiple classes can be created against the offering. Students enrolled in these classes are taught based on the course of the offering. After a class is created, the offering and school cannot be updated.

The start date and end date of a class should not exceed the start date and end date of the offering.

Table A–12 list the attributes of classInfo object.

Table A–12 ClassInfo Attributes

Name	Type	Description
description	string	Class description.
endDate	dateTime	Effective end date of class.
enrollmentActions	enrollmentActionInfoArray	A list of actions to perform in the class enrollments. This is used in the *createClasses* and *updateClasses* operations when creating, updating, or deleting the class enrollments.
enrollmentsLists	enrollmentInfo	List of class enrollments. This attribute is only populated in the response of the *createClasses*, *getClasses*, and *updateClasses* operations.
id	idInfo	Identifier object that uniquely identifies a class.
institutionId	idInfo	Identifier object that uniquely identifies a school. This attribute is mandatory for *createClasses* operation.
name	string	Class name. This attribute is mandatory for *createClasses* operation.
offeringId	idInfo	Identifier object that uniquely identifies an offering. This attribute is mandatory for *createClasses* operation.
startDate	dateTime	Effective start date of class. This attribute is mandatory for *createClasses* operation.

A.10 ContactActionInfo

ContactActionInfo captures the action to perform on a phone contact or email contact and the contact details relevant to the action.

Table A–13 list the attributes of contactActionInfo object.

Table A–13 ContactActionInfo Attributes

Name	Type	Description
action	action	Indicates the action to perform on the address. Example: A *Create* Action indicates that a new address would be created. This attribute is mandatory.
contact	contactInfo	Email or phone contact details. This attribute is mandatory.

Table A–14 lists the action type.

Table A–14 action

Name	Description
Create	Create the object.
CreateOrUpdate	Create or update the object.
Delete	Delete the object.
Update	Update the object.

A.11 ContactActionInfoArray

ContactActionInfoArray contains a list of ContactActionInfo objects.

Table A–15 list the attributes of contactActionInfoArray object.

Table A–15 ContactActionInfoArray Attributes

Name	Type	Description
item	contactActionInfo	The action to perform on a phone or email contact and the contact details relevant to the action.

A.12 ContactInfo

Email contact details and phone contact details are captured in the contactInfo. A contactInfo represents either an email contact or a phone contact.

A phone contactInfo would have contactType *PHONE* and have the following attributes:

- areaCode*
- contactDetails
- contactEndDate
- contactPurpose*

- contactStartDate
- contactType*
- countryCode*
- id
- phoneExtension
- phoneNumber*
- phoneType*
- preferredContact*

An email contactInfo would have contactType *EMAIL* and have the following attributes:

- contactDetails
- contactEndDate
- contactPurpose*
- contactStartDate
- contactType*
- id
- emailAddress*
- emailFormat
- preferredContact*

Note: * indicates mandatory attributes for *create* operations.

Table A–16 list the attributes of contactInfo object.

Table A–16 ContactInfo Attributes

Name	Type	Description
areaCode	string	Area code for phone contact. This attribute is mandatory for *create* operation for phone contact.
contactDetails	string	Additional details for the contact.
contactEndDate	dateTime	Effective end date of the contact.
contactPurpose	string	Purpose of the contact. This attribute is mandatory for *create* operation.
contactStartDate	dateTime	Effective start date of the contact. This attribute is mandatory for *create* operation.
contactType	contactType	ContactType – *PHONE* or *EMAIL*. This attribute is mandatory for *create* operation.

Table A–16 (Cont.) ContactInfo Attributes

Name	Type	Description
countryCode	string	Country code for phone contact. This attribute is mandatory for *create* operation for phone contact.
emailAddress	string	Email address for email contact. This attribute is mandatory for *create* operation for email contact.
emailFormat	string	Email format for email contact. Example: plain text or HTML formatted text
id	idInfo	Identifier object that uniquely identifies the phone or email contact.
phoneExtension	string	Phone extension for phone contact.
phoneNumber	string	Phone number for phone contact. This attribute is mandatory for *create* operation for phone contact.
phoneType	string	Phone type for phone contact. This attribute is mandatory for *create* operation for phone contact.
preferredContact	yesNoType	Indicates whether the contact is the preferred point of contact. This attribute is mandatory for *create* operation.

Table A–17 lists the contactType type.

Table A–17 contactType

Name	Description
EMAIL	Email
PHONE	Phone

Table A–18 lists the yesNoType type.

Table A–18 yesNoType

Name	Description
Yes	Yes
No	No

A.13 ContextActionInfo

ContextActionInfo captures the action to perform on a context and the context details relevant to the action.

Table A–19 list the attributes of contextActionInfo object.

Table A–19 ContextActionInfo Attributes

Name	Type	Description
action	action	Indicates the action to perform on a context. Example: A *Create* Action indicates that a context would be created. This attribute is mandatory.
context	contextInfo	Context details. This attribute is mandatory.

Table A–20 lists the action type.

Table A–20 action

Name	Description
Create	Create the object
CreateOrUpdate	Create or update the object
Delete	Delete the object
Update	Update the object

A.14 ContextActionInfoArray

ContextActionInfoArray contains a list of contextActionInfo objects.

Table A–21 list the attributes of contextActionInfoArray object.

Table A–21 ContextActionInfoArray Attributes

Name	Type	Description
item	contextActionInfo	The action to perform on a context and the context details relevant to the action.

A.15 ContextAdoptionInfo

During a framework adoption, schools have a choice of adopting one or more contexts within the context. If a child context is adopted, the parent context is also automatically adopted. The details of a context adoption are captured in a contextAdoptionInfo.

Table A–22 list the attributes of contextAdoptionInfo object.

Table A–22 contextAdoptionInfo Attributes

Name	Type	Description
contextId	idInfo	Identifier object that uniquely identifies a context. This attribute is mandatory for *adoptContexts* operation.
frameworkAdoptionId	idInfo	Identifier object that uniquely identifies a framework adoption. This attribute is mandatory for *adoptContexts* operation.
id	idInfo	Identifier object that uniquely identifies a context adoption.

A.16 ContextInfo

Under a framework item, there can be one or more contexts. In addition, there can be one or more sub-contexts created under a context. The detail of each context is captured in a contextInfo.

Table A–23 list the attributes of contextInfo object.

Table A–23 ContextInfo Attributes

Name	Type	Description
childrenLists	contextInfo	List of contexts. This attribute is only populated in the response of the *createContexts*, *getContexts*, and *updateContexts* operations.
childrenActions	contextActionInfoArray	A list of actions to perform on the context. This is used in the *createContexts*, *updateContexts* operations when creating, updating, or deleting the contexts.
code	string	Code of the context.
description	string	Description of the context.
endDate	dateTime	Effective end date of the context.
frameworkItemId	idInfo	Identifier object that uniquely identifies a framework item. This attribute is mandatory for *createContexts* operation.
id	idInfo	Identifier object that uniquely identifies a context.
name	string	Context name. This attribute is mandatory for *createContexts* operation.
parentId	idInfo	Identifier object that uniquely identifies a parent context. This attribute is only set in the response of *createContexts*, *getContexts*, and *updateContexts* operations.

Table A–23 (Cont.) ContextInfo Attributes

Name	Type	Description
startDate	dateTime	Effective start date of the context. This attribute is mandatory for *createContexts* operation.

A.17 CourseInfo

A course is specific to a school and can be tagged to facilitate searching. The details of a course are captured in the courseInfo.

Table A–24 list the attributes of courseInfo object.

Table A–24 CourseInfo Attributes

Name	Type	Description
code	string	Course code. This attribute is mandatory for *createCourses* operation.
description	string	Course description.
endDate	dateTime	Effective end date of the course.
id	idInfo	Identifier object to uniquely identify the course.
institutionId	idInfo	Identifier object that uniquely identifies the school in which the course is taught. This attribute is mandatory for *createCourses* operation.
name	string	Course name. This attribute is mandatory for *createCourses* operation.
startDate	dateTime	Effective start date of the course. This attribute is mandatory for *createCourses* operation.
tagActions	idActionInfoArray	A list of actions to perform on the course's tags. This is used in the *createCourses* and *updateCourses* operations when creating, updating, or deleting the course's tags.
tagsLists	idInfo	List of tags of a course. This attribute is only populated in the response of the *createCourses*, *getCourses*, and *updateCourses* operations.

A.18 EnrollmentActionInfo

EnrollmentActionInfo captures the actions to be performed on the class enrollment and the enrollment details relevant to the action.

Table A–25 list the attributes of enrollmentActionInfo object.

Table A–25 EnrollmentActionInfo Attributes

Name	Type	Description
action	action	Indicates the action to perform on a class enrollment. Example: A *Create* Action indicates that a class enrollment would be created. This attribute is mandatory.
enrollment	enrollmentInfo	Class enrollment details. This attribute is mandatory.

Table A–26 lists the action type.

Table A–26 action

Name	Description
Create	Create the object
CreateOrUpdate	Create or update the object
Delete	Delete the object
Update	Update the object

A.19 EnrollmentActionInfoArray

EnrollmentActionInfoArray contains a list of enrollmentActionInfo objects.

Table A–27 list the attributes of enrollmentActionInfoArray object.

Table A–27 EnrollmentActionInfoArray Attributes

Name	Type	Description
item	enrollmentActionInfo	The action to perform on a class enrollment and the class enrollment details relevant to the action.

A.20 EnrollmentInfo

Both students and teachers can be enrolled in a class. Each student/teacher enrollment in a class is captured in an EnrollmentInfo.

After a learning item has been created in the class lesson plan, students and teachers in the class cannot be un-enrolled from the class. Instead, their enrollment can only be end-dated. An end-dated enrollment would indicate the end of a student/teacher's participation in the class.

Do note the start and end date of a class enrollment should not exceed the start and end date of the class.

Table A–28 list the attributes of enrollmentInfo object.

Table A–28 EnrollmentInfo Attributes

Name	Type	Description
endDate	dateTime	Effective end date of the class enrollment.

Table A–28 (Cont.) EnrollmentInfo Attributes

Name	Type	Description
id	idInfo	Identifier object that uniquely identifies the class enrollment.
personId	idInfo	Identifier object that uniquely identifies the person enrolled in the class. This attribute is mandatory for *create* operation.
startDate	dateTime	Effective start date of the class enrollment. This attribute is mandatory for *create* operation.
type	enrollmentType	The type of class enrollment. Example: `TEACHER_OF_CLASS` or `STUDENT_OF_CLASS`. This attribute is mandatory for *create* operation.

Table A–29 lists the enrollmentType type.

Table A–29 enrollmentType

Name	Description
STUDENT_OF_CLASS	Student enrollment to a class
TEACHER_OF_CLASS	Teacher enrollment to a class

A.21 FrameworkAdoptionInfo

After the framework is published, schools have to adopt the framework before they can use it. The framework adoption details are captured in the frameworkAdoptionInfo.

Table A–30 list the attributes of frameworkAdoptionInfo object.

Table A–30 FrameworkAdoptionInfo Attributes

Name	Type	Description
calendarId	idInfo	Identifier object that uniquely identifies a school academic calendar. This attribute is mandatory for *adoptFrameworks* operation.
frameworkId	idInfo	Identifier object that uniquely identifies a framework.
id	idInfo	Identifier object that uniquely identifies a framework adoption. This attribute is mandatory for *adoptFrameworks* operation.
institutionId	idInfo	Identifier object that uniquely identifies a school. This attribute is mandatory for *adoptFrameworks* operation.

Table A–30 (Cont.) FrameworkAdoptionInfo Attributes

Name	Type	Description
primary	yesNoType	Indicates whether the adopted framework is the primary framework. This attribute is mandatory for *adoptFrameworks* operation.

Table A–31 lists the enrollmentType type.

Table A–31 yesNoType

Name	Description
Yes	Yes
No	No

A.22 FrameworkInfo

The framework forms the basis on which the students' learning progress is assessed against. A framework has to be created before the blueprint node hierarchy, framework item hierarchy, and outcome proficiency sets can be created.

After a framework is created, it has to be made available to schools and institution groups before it can be adopted.

The basic details of the framework are captured in a frameworkInfo.

Table A–32 list the attributes of frameworkInfo object.

Table A–32 FrameworkInfo Attributes

Name	Type	Description
endDate	dateTime	Effective end date of the framework.
groupAvailableToActions	idActionInfoArray	A list of actions to perform on the institution group. This is used in the *createFrameworks*, *updateFrameworks* operations when creating, updating, or deleting the frameworks.
groupsAvailableToLists	idInfo	List of institution groups. This attribute is only populated in the response of the *createFrameworks*, *getFrameworks*, and *updateFrameworks* operations.
id	idInfo	Identifier object that uniquely identifies a framework.
institutionAvailableToLists	idInfo	List of institutions. This attribute is only populated in the response of the *createFrameworks*, *getFrameworks*, and *updateFrameworks* operations.

Table A–32 (Cont.) FrameworkInfo Attributes

Name	Type	Description
institutionAvailableToActions	idActionInfoArray	A list of actions to perform on the institution. This is used in the *createFrameworks*, *updateFrameworks* operations when creating, updating, or deleting the frameworks.
name	string	Framework name. This attribute is mandatory for *createFrameworks* operation.
publishedDate	dateTime	Date on which the date is published. This attribute is only set in the response of *getFrameworks* operation.
startDate	dateTime	Effective start date of the framework. This attribute is mandatory for *createFrameworks* operation.

A.23 FrameworkItemActionInfo

FrameworkItemActionInfo captures the action to perform on child framework items and the details of framework items relevant to the action.

Table A–33 list the attributes of frameworkItemActionInfo object.

Table A–33 FrameworkItemActionInfo Attributes

Name	Type	Description
action	action	Indicates the action to perform on a framework item. Example: A *Create* Action indicates that a framework item would be created. This attribute is mandatory.
frameworkItem	frameworkItemInfo	Framework item details. This attribute is mandatory.

Table A–34 lists the action type.

Table A–34 action

Name	Description
Create	Create the object
CreateOrUpdate	Create or update the object
Delete	Delete the object
Update	Update the object

A.24 FrameworkItemActionInfoArray

FrameworkItemActionInfoArray contains a list of frameworkItemActionInfo objects.

Table A–35 list the attributes of frameworkItemActionInfoArray object.

Table A–35 FrameworkItemActionInfoArray Attributes

Name	Type	Description
item	frameworkItemActionInfo	The action to perform on a framework item and the framework item details relevant to the action.

A.25 FrameworkItemInfo

Framework items are organized in a hierarchical structure. Each framework item can have one or more framework items under it. The details of each framework item are captured in a frameworkItemInfo.

Table A–36 list the attributes of frameworkItemInfo object.

Table A–36 FrameworkItemInfo Attributes

Name	Type	Description
blueprintNodeId	idInfo	Identifier object that uniquely identifies a blueprint node. This attribute is mandatory for *createFrameworkItems* operation.
childrenLists	frameworkItemInfo	List of child framework items. This attribute is only populated in the response of the *createFrameworkItems*, *getFrameworkItems*, and *updateFrameworkItems* operations.
childrenActions	frameworkItemActionInfoArray	A list of actions to perform in the child framework items. This is used in the *createFrameworkItems*, *updateFrameworkItems* operations when creating, updating, or deleting the framework items.
code	string	Code of the framework item. This attribute is mandatory for *createFrameworkItems* operation.
confirmingLevel	yesNoType	Indicates whether the confirmations can be made against the framework item. This attribute is mandatory for *createFrameworkItems* operation.
description	string	Framework item description.
endDate	dateTime	Effective end date of the framework item.
frameworkId	idInfo	Identifier object that uniquely identifies a framework. This attribute is mandatory for *createFrameworkItems* operation.
id	idInfo	Identifier object that uniquely identifies the framework item.

Table A–36 (Cont.) FrameworkItemInfo Attributes

Name	Type	Description
name	string	Framework item name.This attribute is mandatory for *createFrameworkItems* operation.
parentId	idInfo	Identifier object that uniquely identifies the parent framework item. This attribute is only populated in the response of the *createFrameworkItems*, *getFrameworkItems*, and *updateFrameworkItems* operations.
proficiencySetId	idInfo	Identifier object that uniquely identifies the proficiency set.
shortName	string	Short name for the framework item.
startDate	dateTime	Effective start date of the framework item. This attribute is mandatory for *createFrameworkItems* operation.

Table A–37 lists the yesNoType type.

Table A–37 yesNoType

Name	Description
Yes	Yes
No	No

A.26 FrameworkLoadInfo

Users can use FrameworkLoadInfo to create a complete framework with all the outcome proficiency sets, framework item hierarchy, blueprint node hierarchy, outcome statements created with the framework in one operation.

Table A–38 list the attributes of frameworkLoadInfo object.

Table A–38 FrameworkLoadInfoAttributes

Name	Type	Description
blueprint	blueprintNodeInfo	Root blueprint node of the blueprint node hierarchy.
contextsLists	contextInfo	List of contexts.
framework	frameworkInfo	Framework basic details.
frameworkItemsLists	frameworkItemInfo	List of framework item hierarchies.
outcomeStatementDescTypesLists	outcomeStatementDescTypeInfo	List of outcome statement descriptor types.
outcomeStatementsLists	outcomeStatementInfo	List of outcome statements.
proficiencySetsLists	proficiencySetInfo	List of outcome proficiency sets.

Table A–38 (Cont.) FrameworkLoadInfoAttributes

Name	Type	Description
publishFramework	yesNoType	Indicates whether to publish the framework after the framework is created.

Table A–39 lists the yesNoType type.

Table A–39 yesNoType

Name	Description
Yes	Yes
No	No

A.27 IdActionInfo

IdActionInfo captures the action to perform given the id of an entity. Typical scenarios involve adding an entity as part of a larger entity. For example, adding institutions to an institution group.

Table A–40 list the attributes of idActionInfo object.

Table A–40 IdActionInfo Attributes

Name	Type	Description
action	action	Indicates the action to perform given the id of an entity. Example: A *Create* Action indicates that the entity would be added as part of a larger entity. This attribute is mandatory.
id	idInfo	Identifier object that uniquely identifies the entity. This attribute is mandatory.

Table A–41 lists the action type.

Table A–41 action

Name	Description
Create	Create the object
CreateOrUpdate	Create or update the object
Delete	Delete the object
Update	Update the object

A.28 IdInfo

Identifier object used to uniquely identify an entity such as an institution. An entity can be identified by either its OSL id or a pair of external system id and external id. The OSL id is generated by the OSL application.

The external id uniquely identifies the entity in another system that is external to OSL, while the external system id uniquely identifies this external system. If used, both the external id and external system id must be provided.

Table A–42 list the attributes of idInfo object.

Table A–42 IdInfo Attributes

Name	Type	Description
externalId	string	Id that is used to identify an entity in an external system. If present, must be used with the externalSystemId.
externalSystemId	string	Id that is used to identify an external system. If present, must be used with the externalId.
oslId	string	Number that is generated by OSL to uniquely identify an entity.

A.29 InstitutionGroupInfo

Institutions can be grouped into institution groups and are not constrained by their position in the institution hierarchy.

Table A–43 list the attributes of institutionGroupInfo object.

Table A–43 InstitutionGroupInfo Attributes

Name	Type	Description
description	string	Description of the institution group.
endDate	dateTime	Effective end date of the institution group.
id	idInfo	Identifier object that uniquely identifies the institution group.
institutionActions	idActionInfoArray	A list of actions to perform on the institutions in the institution group. This is used in the *createInstitutionGroups* and *updateInstitutionGroups* operations when adding or removing the institution from the institution group.
institutionsLists	idInfo	List of institutions in the institution group. This is only populated in the response of the *createInstitutionGroups*, *getInstitutionGroups*, *and updateInstitutionGroups* operations.
name	string	Institution group name. This attribute is mandatory for *createInstitutionGroups* operation.

Table A–43 (Cont.) InstitutionGroupInfo Attributes

Name	Type	Description
startDate	dateTime	Effective start date of the institution group. This attribute is mandatory for *createInstitutionGroups* operation.

A.30 InstitutionInfo

The department, schools, and other organizational entities such as regional or district level education entities are represented as institutions in OSL. All institutions are organized in a hierarchy with the department as the root of the institution hierarchy. The attributes of an institution is captured in the InstitutionInfo object.

Table A–44 list the attributes of institutionInfo object.

Table A–44 institutionInfo Attributes

Name	Type	Description
Sort Name	string	The sort order of party names when displayed in a list.
addressActions	addressActionInfoArray	A list of actions to perform on the institution's addresses. This is used in the *createInstitutions* and *updateInstitutions* operations when creating, updating, or deleting the institution's addresses.
addressesLists	addressInfo	List of addresses of the institution. This attribute is only populated in the response of the *createInstitutions*, *getInstitutions*, and *updateInstitutions* operations.
emailContactActions	contactActionInfoArray	A list of actions to perform on the institution's email addresses. This is used in the *createInstitutions* and *updateInstitutions* operations when creating, updating, or deleting the institution's email addresses.
emailContactsLists	contactInfo	List of email addresses of the institution. This attribute is only populated in the response of the *createInstitutions*, *getInstitutions*, and *updateInstitutions* operations.
id	idInfo	Identifier object that uniquely identifies an institution. This attribute is mandatory for *updateInstitutions* and *getInstitutions* operations.

Table A–44 (Cont.) institutionInfo Attributes

Name	Type	Description
institutionType	string	The type of the institution. Example: Department, Primary School, Non-School school. Refer to the list of value `OSL_Institution_Type` for a list of valid values. This attribute is mandatory for *createInstitutions* operation.
name	string	The name of the institution. This attribute is mandatory for *createInstitutions* operation.
parentInstitutionId	idInfo	Identifier object that uniquely identifies the parent institution. This attribute is mandatory for *createInstitutions* operation. This attribute cannot be updated.
phoneContactActions	contactActionInfoArray	A list of actions to perform on the institution's phone numbers. This is used in the *createInstitutions* and *updateInstitutions* operations when creating, updating, or deleting the institution's phone numbers.
phoneContactsLists	contactInfo	List of phone numbers of the institution. This attribute is only populated in the response of the *createInstitutions*, *getInstitutions*, and *updateInstitutions* operations.

A.31 LanguageActionInfo

LanguageActionInfo captures the action to perform on the language known by a person.

Table A–45 list the attributes of languageActionInfo object.

Table A–45 LanguageActionInfo Attributes

Name	Type	Description
action	action	Indicates the action to perform on the known language. Example: A *Create* Action indicates that a new known language would be created. This attribute is mandatory.
language	languageInfo	Language details. This attribute is mandatory.

Table A–46 lists the action type.

Table A–46 action

Name	Description
Create	Create the object
CreateOrUpdate	Create or update the object
Delete	Delete the object
Update	Update the object

A.32 LanguageActionInfoArray

LanguageActionInfoArray contains a list of LanguageActionInfo objects.

Table A–47 list the attributes of languageActionInfoArray object.

Table A–47 LanguageActionInfoArray Attributes

Name	Type	Description
item	languageActionInfo	The action to perform on a known language and the language details relevant to the action.

A.33 LanguageInfo

A person may be proficient in multiple languages. His/her proficiency in each of the language is captured in the LanguageInfo.

Table A–48 list the attributes of languageInfo object.

Table A–48 LanguageInfo Attributes

Name	Type	Description
comprehensionLevel	string	Level of proficiency in the language.
id	idInfo	Identifier object that uniquely identifies the known language.This attribute is mandatory for *createPersons* operation.
languageCode	string	Code representing the language. Example: GB represents British English, US represents American English. This attribute is mandatory for *createPersons* operation.
nativeLanguage	yesNoType	Indicates whether the language is the native language of a person. This attribute is mandatory for *createPersons* operation.
primaryLanguage	yesNoType	Indicates whether the language is the person's primary language for communications. This attribute is mandatory for *createPersons* operation.

Table A–49 lists the yesNoType type.

Table A–49 yesNoType

Name	Description
Yes	Yes
No	No

A.34 LookupValueActionInfo

LookupValueActionInfo captures the action to perform on a value and the details of the value relevant to the action.

Table A–50 list the attributes of lookupValueActionInfo object.

Table A–50 LookupValueActionInfo Attributes

Name	Type	Description
action	action	Indicates the action to perform on a value. Example: A *Create* Action indicates that the value would be added to the list of values. This attribute is mandatory.
lookupValue	lookupValueInfo	Value details. This attribute is mandatory.

Table A–51 lists the action type.

Table A–51 action

Name	Description
Create	Create the object
CreateOrUpdate	Create or update the object
Delete	Delete the object
Update	Update the object

A.35 LookupValueActionInfoArray

LookupValueActionInfoArray contains a list of lookupValueActionInfo objects.

Table A–52 list the attributes of lookupValueActionInfoArray object.

Table A–52 LookupValueActionInfoArray Attributes

Name	Type	Description
item	lookupValueActionInfo	The action to perform on a value and the value details relevant to the action.

A.36 LookupValueInfo

The details of the values contained in a list of values are captured in a lookupValueInfo.

Table A–53 list the attributes of lookupValueInfo object..

Table A–53 LookupValueInfo Attributes

Name	Type	Description
customizationLevel	customizationLevel	Level of customization that is allowed on the value.
description	string	Description of the value.
endDate	dateTime	Effective end date of the value.
id	idInfo	Identifier object that uniquely identifies a value.
lookupCode	string	Code representation of the value.
meaning	string	The meaning of the value.
startDate	dateTime	Effective start date of the value.

Table A–54 lists the customizationLevel type.

Table A–54 customizationLevel

Name	Description
APPL	Can be updated by Application
INSTN	Can be updated by Institution

A.37 LookupTypeInfo

The list of values used within OSL is captured using a lookupTypeInfo.

Table A–55 list the attributes of lookupTypeInfo object.

Table A–55 LookupTypeInfo Attributes

Name	Type	Description
customizationLevel	customizationLevel	Level of customization that is allowed on the list of values.
dataType	string	The data type allowed in the list of values.
description	string	A description of the list of values.
endDate	dateTime	Effective end date of the list of values.
id	idInfo	Identifier object that uniquely identifies a list of values.
lookupType	lookupType	Code representation of the list of values in OSL.
maxDataLength	string	The maximum length of the values that can allow in the list of values.
meaning	string	The meaning of the list of values.
startDate	dateTime	Effective start date of the list of values.

Table A–55 (Cont.) LookupTypeInfo Attributes

Name	Type	Description
valuesList	lookupValueInfo	List of the actual values contained in the list of values. This attribute is only populated in the response of the *updateValueTypes* and *getValueTypes* operations.
valuesActions	lookupValueActionInfoArray	A list of actions to perform on the values contained in the list of values. This is used in the *updateValueTypes* operation when updating the list of values.

Table A–56 lists the customizationLevel type.

Table A–56 customizationLevel

Name	Description
APPL	Can be updated by Application
INSTN	Can be updated by Institution

Table A–57 lists the lookupType type.

Table A–57 lookupType

Name	Description
OSL_ADDRESS_USAGE_TYPE	Usage purpose of address
OSL_APPROVAL_STATUS	Approval status of student to join the class
OSL_ASSESSED_OUTCOME_TYPE	Type of assessed outcome
OSL_ASSESSMENT_TYPE	Type of assessment
OSL_COLLECTION_CATEGORY	Type of student collection
OSL_COLLECTION_GROUP_RELATION	Relationship between collections
OSL_COLLECTION_RELATIONSHIP	Enrollment Type in a collection
OSL_CONTACT_POINT_PURPOSE	Purpose of the contact point
OSL_CONTACT_POINT_TYPE	Type of contact point
OSL_CONTENT_RESOURCE_TYPE	Type of the content resource
OSL_CUSTOMISATION_LEVEL	Level of customization of the lookup
OSL_DISCUSSION_TYPE	Type of discussion
OSL_EMAIL_FORMAT	Format of email

Table A–57 (Cont.) lookupType

Name	Description
OSL_EVENT_ACTION	Action Type captured in the event
OSL_EVENT_OBJECT_TYPE	Object captured in the event
OSL_EXTENSION_REASON	Reason for learning item extension
OSL_GENDER	Gender
OSL_INSTITUTION_TYPE	Type of institution
OSL_LANGUAGE	Language
OSL_LEARNING_ITEM_TYPE	Type of Learning Item
OSL_LOCK_TYPE	Type of lock applied on an object
OSL_NOTE_AUDIENCE	Target audience of an observation
OSL_NOTE_CATEGORY	Category of an observation
OSL_NOTE_TYPE	Type of an observation
OSL_OUTCOME_STATEMENT_TYPE	Outcome Statement Type
OSL_PARTY_TYPE	Type of party
OSL_PERSON_TITLE	Person title
OSL_PHONE_TYPE	Phone type
OSL_PROFICIENCY_LEVEL_TYPE	Type of proficiency level
OSL_PROFICIENCY_SET_TYPE	Type of proficiency type
OSL_RELATIONSHIP_TYPE	Party Relationship Type
OSL_SPOKEN_COMPREHENSION_LEVEL	Level of mastery in a language
OSL_WORKSPACE_TYPE	Workspace assigned to student
OSL_YES_NO	Yes or no

A.38 OfferingInfo

An offering is created when a course is offered by the school for an academic period. The details of the offering are captured in an OfferingInfo.

The start and end date of an offering should not exceed the start and end date of the academic calendar.

Table A–58 list the attributes of offeringInfo object.

Table A–58 OfferingInfo Attributes

Name	Type	Description
calendarId	idInfo	Identifier object that uniquely identifies the academic calendar in which the offering is made.
code	string	Offering code. This attribute is mandatory for *createOfferings* operation.
courseId	idInfo	Identifier object that uniquely identifies the course. This attribute is mandatory for *createOfferings* operation.
endDate	dateTime	Effective end date of the offering.
frameworkAdoptionId	idInfo	Identifier object that uniquely identifies the adopted framework. This attribute is mandatory for *createOfferings* operation.
id	idInfo	Identifier object that uniquely identifies the offering.
institutionId	idInfo	Identifier object that uniquely identifies the school. This attribute is mandatory for *createOfferings* operation.
name	string	Offering name. This attribute is mandatory for *createOfferings* operation.
startDate	dateTime	Effective start date of the offering. This attribute is mandatory for *createOfferings* operation.

A.39 OutcomeStatementDescActionInfo

OutcomeStatementDescActionInfo captures the action to perform on an outcome statement descriptor and the outcome statement descriptor details relevant to the action.

Table A–59 list the attributes of outcomeStatementDescActionInfo object.

Table A–59 OutcomeStatementDescActionInfo Attributes

Name	Type	Description
action	action	Indicates the action to perform on an outcome statement descriptor. Example: A *Create* Action indicates that an outcome statement descriptor would be created. This attribute is mandatory.
outcomeStatementDesc	outcomeStatementDescInfo	Outcome statement descriptor details. This attribute is mandatory.

Table A–60 lists the action type.

Table A–60 action

Name	Description
Create	Create the object
CreateOrUpdate	Create or update the object
Delete	Delete the object
Update	Update the object

A.40 OutcomeStatementDescActionInfoArray

OutcomeStatementDescActionInfoArray contains a list of outcomeStatementDescActionInfo objects.

Table A–61 list the attributes of outcomeStatementDescActionInfoArray object.

Table A–61 OutcomeStatementDescActionInfoArray Attributes

Name	Type	Description
item	outcomeStatementDescActionInfo	The action to perform on an outcome statement descriptor and the outcome statement descriptor details relevant to the action.

A.41 OutcomeStatementDescInfo

Outcome statement descriptor provides additional information on the outcome statements. The outcome statement descriptor details are captured in an outcomeStatementDescInfo.

Table A–62 list the attributes of outcomeStatementDescInfo object.

Table A–62 OutcomeStatementDescInfo Attributes

Name	Type	Description
descTypeId	idInfo	Identifier object that uniquely identifies the outcome statement descriptor type. This attribute is mandatory for *createOutcomeStatementDescs* operation.
description	string	Description of the outcome statement descriptor.
endDate	dateTime	Effective end date of the outcome statement descriptor.
id	idInfo	Identifier object that uniquely identifies the outcome statement descriptor.
name	string	Outcome statement descriptor name. This attribute is mandatory for *createOutcomeStatementDescs* operation.

Table A–62 (Cont.) OutcomeStatementDescInfo Attributes

Name	Type	Description
startDate	dateTime	Effective start date of the outcome statement descriptor. This attribute is mandatory for *createOutcomeStatementDescs* operation.

A.42 OutcomeStatementDescTypeInfo

A framework can have multiple types of outcome statement descriptors. The detail of an outcome statement descriptor type is captured in an outcomeStatementDescTypeInfo.

Table A–63 list the attributes of outcomeStatementDescTypeInfo object.

Table A–63 OutcomeStatementDescTypeInfo Attributes

Name	Type	Description
frameworkId	idInfo	Identifier object that uniquely identifies a framework. This attribute is mandatory for *createOutcomeStatementDescTypes* operation.
id	idInfo	Identifier object that uniquely identifies an outcome statement descriptor type.
name	string	Name of outcome statement descriptor type. This attribute is mandatory for *createOutcomeStatementDescTypes* operation.

A.43 OutcomeStatementInfo

Outcome statement describes the outcomes that can be achieved for a framework item.

There are three types of outcome statements:

1. An outcome statement with outcome statement type DEMONSTRABLE requires a proficiency level.
2. An outcome statement with outcome statement type ANNOTATIVE can have a proficiency level.
3. An outcome statement with outcome statement type INDICATIVE does not have a proficiency level.

The outcome statement details are captured in outcomeStatementInfo.

Table A–64 list the attributes of outcomeStatementInfo object.

Table A–64 OutcomeStatementInfo Attributes

Name	Type	Description
descriptorActions	outcomeStatementDescActionInfoArray	A list of actions to perform on the outcome statement descriptors. This is used in the *createOutcomeStatements*, *updateOutcomeStatements* operations when creating, updating, or deleting the outcome statements.
descriptorsLists	outcomeStatementDescInfo	List of outcome statement descriptors. This attribute is only populated in the response of the *createOutcomeStatements*, *getOutcomeStatements*, and *updateOutcomeStatements* operations.
endDate	dateTime	Effective end date of outcome statement.
frameworkItemId	idInfo	Identifier object that uniquely identifies a framework item. This attribute is mandatory for *createOutcomeStatements* operation.
id	idInfo	Identifier object that uniquely identifies an outcome statement.
name	string	Outcome statement name.This attribute is mandatory for *createOutcomeStatements* operation.
proficiencyLevelId	idInfo	Identifier object that uniquely identifies a proficiency level.
startDate	dateTime	Effective start date of outcome statement. This attribute is mandatory for *createOutcomeStatements* operation.
type	outcomeStatementType	The outcome statement type – *ANNOTATIVE*, *INDICATIVE*, *DEMONSTRABLE*. This attribute is mandatory for *createOutcomeStatements* operation.

Table A–65 lists the outcomeStatementType type.

Table A–65 outcomeStatementType

Name	Description
ANNOTATIVE	Annotative Outcome Statement
DEMONSTRABLE	Demonstrable Outcome Statement
INDICATIVE	Indicative Outcome Statement

A.44 PersonInfo

Personal information such as the name, photograph, home addresses, emails, phone numbers, and the roles played by the person are captured as part of the PersonInfo.

Table A–66 list the attributes of personInfo object.

Table A–66 PersonInfo Attributes

Name	Type	Description
Display Name	string	The user name is specified by the **Display Name** attribute to distinguish between persons and institutions in the **osl_parties** table. This attribute is mandatory for **createPersons** and **updatePersons** operations. Missing this attribute will throw the exception `<Error> <oracle.osl.lt.service> <BEA-000000> <Throwing exception: oracle.osl.lt.service.ServiceException; Error code: 51; Message: Validation error, value is null, Party name>.`
Sort Name	string	The sort order of party names when displayed in a list.
academicYearLevel	string	The current academic level of a student.
addressActions	addressActionInfoArray	A list of actions to perform on the person's addresses. This is used in the *createPersons* and *updatePersons* operations when creating, updating, or deleting the person's addresses.
addressesLists	addressInfo	List of addresses of the person. This attribute is only populated in the response of the *createPersons*, *getPersons*, and *updatePersons* operations.
birthDate	dateTime	Date of birth.
countryOfBirth	string	Country of birth.
deathDate	dateTime	Death date.
emailContactActions	contactActionInfoArray	A list of actions to perform on the person's email addresses. This is used in the *createPersons* and *updatePersons* operations when creating, updating, or deleting the person's email addresses.
emailContactsLists	contactInfo	List of email addresses of the person. This attribute is only populated in the response of the *createPersons*, *getPersons*, and *updatePersons* operations.
firstName	string	Given name.

Table A–66 (Cont.) PersonInfo Attributes

Name	Type	Description
gender	gender	Gender.
id	idInfo	Identifier object that uniquely identifies the person.
languageActions	languageActionInfoArray	A list of actions to perform on the person's known languages. This is used in the *createPersons* and *updatePersons* operations when creating, updating, or deleting the person's known languages.
languagesLists	languageInfo	List of languages a person knows. This attribute is only populated in the response of the *createPersons*, *getPersons*, and *updatePersons* operations.
lastName	string	Family name.
loginId	string	Login id.
maritalStatus	string	Marital status.
middleName	string	Middle name.
nameSuffix	string	Name suffix. Example: Jr
phoneContactActions	contactActionInfoArray	A list of actions to perform on the person's phone numbers. This is used in the *createPersons* and *updatePersons* operations when creating, updating, or deleting the person's phone numbers.
phoneContactsLists	contactInfo	List of phone numbers of the person. This attribute is only populated in the response of the *createPersons*, *getPersons,* and *updatePersons* operations.
photograph	base64Binary	Photograph of the person.
placeOfBirth	string	Place of birth.
preferredName	string	Preferred name to address a person.
relationshipActions	relationshipActionInfoArray	A list of actions to perform in the person's relationships. This is used in the *createPersons* and *updatePersons* operations when creating, updating, or deleting the person's relationships.
relationshipsLists	relationshipInfo	List of relationships of a person. This attribute is only populated in the response of the *createPersons*, *getPersons*, and *updatePersons* operations.
title	string	Title.

Table A–67 lists the gender type.

Table A–67 gender

Name	Description
MALE	Male
FEMALE	Female
UNKNOWN	Unknown

A.45 ProficiencyLevelActionInfo

ProficiencyLevelActionInfo captures the action to perform on proficiency levels and the proficiency level details relevant to the action.

Table A–68 list the attributes of proficiencyLevelActionInfo object.

Table A–68 ProficiencyLevelActionInfo Attributes

Name	Type	Description
action	action	Indicates the action to perform on a proficiency level. Example: A *Create* Action indicates that a proficiency level would be created. This attribute is mandatory.
proficiencyLevel	proficiencyLevelInfo	Proficiency Level details. This attribute is mandatory.

Table A–69 lists the action type.

Table A–69 action

Name	Description
Create	Create the object
CreateOrUpdate	Create or update the object
Delete	Delete the object
Update	Update the object

A.46 ProficiencyLevelActionInfoArray

ProficiencyLevelActionInfoArray contains a list of proficiencyLevelActionInfo objects.

Table A–70 list the attributes of proficiencyLevelActionInfoArray object.

Table A–70 ProficiencyLevelActionInfoArray Attributes

Name	Type	Description
item	proficiencyLevelActionInfo	The action to perform on a proficiency level and the proficiency level details relevant to the action.

A.47 ProficiencyLevelInfo

Both outcome proficiency sets and graded proficiency sets can have proficiency levels.

In an outcome proficiency set, proficiency sub-levels can then be created under a proficiency level. Furthermore, the depth of the outcome proficiency levels can have maximumly two levels.

In a graded proficiency set, proficiency sub-levels cannot be created under a proficiency level. A graded proficiency levels cannot have sub-levels. Every graded proficiency level contains a range of values collectively. These ranges should not overlap and should cover the full range of values from 0 to 100.

The details of the outcome proficiency level, outcome proficiency sub-levels, or graded proficiency levels are captured in the proficiencyLevelInfo.

An outcome proficiency level has the following attributes:

- childrenLists
- childrenActions
- description
- id
- name*
- orderSequence*
- parentId
- type*

A graded proficiency level has the following attributes:

- childrenLists
- childrenActions
- code*
- description
- id
- lowerRange*
- lowerRangeInclusive*
- name*
- upperRange*
- upperRangeInclusive*

Note: * indicates mandatory attributes for *create* operations.

Table A–71 list the attributes of proficiencyLevelInfo object.

Table A–71 ProficiencyLevelInfo Attributes

Name	Type	Description
absoluteValue	string	Numeric value representation of the graded proficiency level. This attribute is mandatory for *create* operation for graded proficiency levels.

Table A–71 (Cont.) ProficiencyLevelInfo Attributes

Name	Type	Description
childrenLists	proficiencyLevelInfo	List of proficiency sub-levels. This attribute is only populated in the response of the *createProficiencySets*, *getProficiencySets*, and *updateProficiencySets* operations.
childrenActions	proficiencyLevelActionInfoArray	A list of actions to perform on the proficiency sub-levels. This is used in the *createProficiencySets*, *updateProficiencySets* operations when creating, updating, or deleting the proficiency sets.
code	string	Code of the proficiency level. This attribute is mandatory for *create* operation for outcome proficiency levels.
description	string	Description of the proficiency level.
id	idInfo	Identifier object that uniquely identifies the proficiency level.
lowerRange	string	Numeric value of the lower limit of the graded proficiency level. This attribute is mandatory for *create* operation for graded proficiency levels.
lowerRangeInclusive	yesNoType	Indicates whether the lower limit of a graded proficiency level should be included in the range. This attribute is mandatory for *create* operation for graded proficiency levels.
name	string	Name of the proficiency level. This attribute is mandatory for *create* operation.
orderSequence	string	Order of display of the outcome proficiency levels. This attribute is mandatory for *create* operation for outcome proficiency levels.
parentId	idInfo	Identifier object that uniquely identifies the parent outcome proficiency level. This attribute is only set in the response of *createProficiencySets*, *getProficiencySets*, and *updateProficiencySets* operations.
type	string	Proficiency level type – CORE or FOUNDATION. This attribute is mandatory for *create* operation for outcome proficiency levels.

Table A–71 (Cont.) ProficiencyLevelInfo Attributes

Name	Type	Description
upperRange	string	Numeric value of the upper limit of the graded proficiency level. This attribute is mandatory for *create* operation for graded proficiency levels.
upperRangeInclusive	yesNoType	Indicates whether the upper limit of a graded proficiency level should be included in the range. This attribute is mandatory for *create* operation for graded proficiency levels.

Table A–72 lists the yesNoType type.

Table A–72 yesNoType

Name	Description
Yes	Yes
No	No

A.48 ProficiencySetInfo

Both the details of an outcome proficiency set and graded proficiency set are captured using a proficiencySetInfo.

An outcome proficiency set would have type OUTCOME and the following attributes are:

- description
- endDate
- frameworkId*
- id
- levelsLists
- levelsActions
- name*
- startDate*
- type*

A graded proficiency set would have type GRADED and the following attributes are:

- description
- endDate
- id
- institutionId*
- levelsLists
- levelsActions
- name*

- startDate*

Note: * indicates mandatory attributes for *create* operations.

Table A–73 list the attributes of proficiencySetInfo object.

Table A–73 ProficiencySetInfo Attributes

Name	Type	Description
description	string	Proficiency set description.
endDate	dateTime	Effective end date of proficiency set.
frameworkId	idInfo	Identifier object that uniquely identifies the framework. This attribute is mandatory for *createProficiencySets* operation.
id	idInfo	Identifier object that uniquely identifies the proficiency set.
institutionId	idInfo	Identifier object that uniquely identifies the department or school. This attribute is mandatory for *createGradedProficiencySets* operation.
levelsLists	proficiencyLevelInfo	List of proficiency levels. This attribute is only populated in the response of the *createProficiencySets*, *getProficiencySets*, *updateProficiencySets*, *createGradedProficiencySets*, *getGradedProficiencySets*, and *updateGradedProficiencySets* operations.
levelsActions	proficiencyLevelActionInfoArray	A list of actions to perform on the proficiency levels. This is used in the *createProficiencySets*, *updateProficiencySets*, *createGradedProficiencySets*, and *updateGradedProficiencySets* operations when creating, updating, or deleting the proficiency sets.
name	string	Proficiency set name. This attribute is mandatory for *createProficiencySets* and *createGradedProficiencySets* operations.
startDate	dateTime	Effective start date of outcome proficiency set. This attribute is mandatory for *createProficiencySets* and *createGradedProficiencySets* operations.

Table A–73 (Cont.) ProficiencySetInfo Attributes

Name	Type	Description
type	proficiencySetType	Proficiency set type. This should be set to OUTCOME or GRADED for outcome proficiency sets and graded proficiency sets, respectively. This attribute is mandatory for *createProficiencySets* and *createGradedProficiencySets* operations.

Table A–74 lists the proficiencySetType type.

Table A–74 proficiencySetType

Name	Description
GRADED	Graded Proficiency Set
OUTCOME	Outcome Proficiency Set

A.49 RelationshipActionInfo

RelationshipActionInfoArray contains a list of relationshipActionInfo objects.

Table A–75 list the attributes of relationshipActionInfoArray object.

Table A–75 RelationshipActionInfo Attributes

Name	Type	Description
action	action	Indicates the action to perform on a person relationship. Example: A *Create* Action indicates that a new person relationship would be created. This attribute is mandatory.
relationship	relationshipInfo	Person relationship details. This attribute is mandatory.

Table A–76 lists the action type.

Table A–76 action

Name	Description
Create	Create the object
CreateOrUpdate	Create or update the object
Delete	Delete the object
Update	Update the object

A.50 RelationshipActionInfoArray

RelationshipActionInfoArray contains a list of relationshipActionInfo objects.

Table A–77 list the attributes of relationshipActionInfoArray object.

Table A–77 RelationshipActionInfoArray Attributes

Name	Type	Description
item	relationshipActionInfo	The action to perform on a person relationship and the relationship details relevant to the action.

A.51 RelationshipInfo

A person may have multiple relationships with multiple parties. For example, a person can be a teacher in a school and also be the parent of another person. Each of these relationships is captured by a relationshipInfo. The role of a person within a relationship is indicated by the relationshipType. The id of the institution or person that the current person is in a relationship with is stored in the targetPartyId.

Hence, person A is a parent of person B would be represented as such:

RelationshipType: PARENT_OF

TargetPartyId: Id of person B

Table A–78 list the attributes of relationshipInfo object.

Table A–78 RelationshipInfo Attributes

Name	Type	Description
endDate	dateTime	Effective end date of the person relationship.
id	idInfo	Identifier object that uniquely identifies a person relationship.
relationshipType	relationshipType	Denotes a person's role in the relationship. This attribute is mandatory for *create* operation.
startDate	dateTime	Effective start date of the person relationship. This attribute is mandatory for *create* operation.
targetPartyId	idInfo	Identifier object that uniquely identifies the institution or person that the current person is in a relationship with. This attribute is mandatory for *create* operation.

Table A–79 lists the relationshipType type.

Table A–79 relationshipType

Name	Description
DEPARTMENT_ADMIN_OF	Department Administrator
DEPARTMENT_CURRICULUM_ADMIN_OF	Department Curriculum Administrator
GUARDIAN_OF	Guardian
PARENT_OF	Parent

Table A–79 (Cont.) relationshipType

Name	Description
SCHOOL_ADMIN_OF	School Administrator
SCHOOL_CURRICULUM_ADMIN_OF	School Curriculum Administrator
STUDENT_OF	Student
TEACHER_OF	Teacher

A.52 TagInfo

Each school can create a series of tags. These tags are used to tag courses to facilitate search function within OSL.

Table A–80 list the attributes of tagInfo object.

Table A–80 TagInfo Attributes

Name	Type	Description
id	idInfo	Identifier object that uniquely identifies the tag.
institutionId	idInfo	Identifier object that uniquely identifies the school. This attribute is mandatory for *createTags* operation.
name	string	Tag name. This attribute is mandatory for *createTags* operation.

B

Content Integration Service Objects

This appendix contains OSL content integration service object references.

B.1 ContentsAssociationInfo

Represents Information about General Content references being associated with Rich Data.

Table B–1 list the attributes of ContentsAssociationInfo object.

Table B–1 ContentsAssociationInfo Attributes

Name	Type	Mandatory	Description
contentIds	ContentIdsInfo	Yes	General Content Identifiers.
context	ContentContextInfo	Yes	OSL Context.
userContext	UserContextInfo	Yes	OSL user making the CIWSI invocation.

B.2 ContentContextInfo

Represents List of OSL Context.

Table B–2 list the attributes of ContentContextInfo object.

Table B–2 ContentContextInfo Attributes

Name	Type	Mandatory	Description
contexts	context[]	Yes	List of OSL Context

Table B–3 lists the context type.

Table B–3 context

Name	Type	Mandatory	Description
contextType	contextType	Yes	Type of OSL Context
contextValue	string	Yes	Value of OSL Context

Table B–4 lists the contextType type.

Table B–4 contextType

Name	Type	Mandatory	Description
Enumeration	string	Yes	Legal Values are: UserName, UserId, SchoolName, SchoolId, CourseName, CourseId, ClassName, ClassId, LearningItemName, LearningItemId, CourseTagName, CourseTagId, FrameworkItemName, FrameworkItemId, OutcomeStatementName, OutcomeStatementId, StudentName, StudentId, SubmissionGroupName, SubmissionGroupId

B.3 ContentIdInfo

Represents Identity Information for Content in ECMS.

Table B–5 list the attributes of ContentIdInfo object.

Table B–5 ContentIdInfo Attributes

Name	Type	Mandatory	Description
contentId	string	Yes	Identifier of Content in ECMS.
contentVersionId	string	No	Version Identifier of Content in ECMS. If version identifier is not specified, then information contained in this structure represents the latest version.

B.4 ContentIdsInfo

Represents a list of Identity Informations for Content in ECMS.

Table B–6 list the attributes of ContentIdsInfo object.

Table B–6 ContentIdsInfo Attributes

Name	Type	Mandatory	Description
contentIds	ContentIdInfo[]	Yes	List of Identifiers of Contents in ECMS.
userContext	UserContextInfo	Yes	OSL user making the CIWSI invocation.

B.5 ContentInfo

Represents Content Information for Content in ECMS.

Table B–7 list the attributes of ContentInfo object.

Table B–7 ContentInfo Attributes

Name	Type	Mandatory	Description
contentId	ContentIdInfo	Yes	Identifier of Content in ECMS. If version identifier is not specified, then the content information should represent the latest version. Otherwise, it should be version specific.
contentURL	string	No	For ECMS supporting URL, this field should specify the URL of Content. If contentId specifies identifier for the latest version, then the URL should be for the latest version. Otherwise, the URL should point to the specific version.
contentTitle	string	Yes	Title of the Content.
contentFileName	string	No	File name of the Content in ECMS.
data	string	Yes	Raw data of Content.

B.6 ContentSearchCriteriaInfo

Represents Informations that corresponds search criteria to List General Content.

Table B–8 list the attributes of ContentSearchCriteriaInfo object.

Table B–8 ContentSearchCriteriaInfo Attributes

Name	Type	Mandatory	Description
filters	searchFilter[]	Yes	Filter.
context	ContentContextInfo	Yes	OSL Context.

Table B–9 list the searchFilter type.

Table B–9 searchFilter

Name	Type	Mandatory	Description
Enumeration	string	Yes	Legal Values are: Images, LearningItems, or any

B.7 ContentsUploadInfo

Represents Information to upload OSL Content.

Table B–10 list the attributes of ContentsUploadInfo object.

Table B–10 ContentsUploadInfo Attributes

Name	Type	Mandatory	Description
uploadInfos	uploadInfo[]	Yes	OSL Content Upload information.
userContext	UserContextInfo	Yes	OSL user making the CIWSI invocation.

Table B–11 lists the uploadInfo type.

Table B–11 uploadInfo

Name	Type	Mandatory	Description
data	base64Binary	Yes	Raw data of OSL Content to be uploaded.
title	string	Yes	Title of OSL Content.
fileName	string	Yes	File name of OSL Content.

B.8 LearningItemPublishInfo

Represents Information to upload Exported Learning Item. Inherits from ContentsUploadInfo.

Table B–12 list the attributes of LearningItemPublishInfo object.

Table B–12 LearningItemPublishInfo Attributes

Name	Type	Mandatory	Description
contextInfo	ContentContextInfo	Yes	OSL Context.

B.9 UserContextInfo

Represents OSL user making the CIWSI invocation.

Table B–13 list the attributes of UserContextInfo object.

Table B–13 UserContextInfo Attributes

Name	Type	Mandatory	Description
UserId	string	Yes	User Identifier (SSO user name) of OSL user making the invocation to CIWSI.